ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών

Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών

Δικτύων Επικοινωνιών και Κατανεμημένων Συστημάτων,

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πρότυπα, τεχνολογίες και διεπαφές υποστήριξης για χρήστες με ιδιαιτερότητες

Κωνσταντίνος Π. Τσιμπερδώνης
Α.Μ. 20002

Εισηγητής:

Νικόλαος Ζάχαρης, Καθηγητής

(Κενό φύλλο)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πρότυπα, τεχνολογίες και διεπαφές υποστήριξης για χρήστες με ιδιαιτερότητες

Κωνσταντίνος Π. Τσιμπερδώνης
Α.Μ. 20002

Εισηγητής:

Νικόλαος Ζάχαρης, Καθηγητής

Digitally signed by Nikolaos Zacharis
Date: 2022.05.31 10:58:45 +03'00'

Εξεταστική Επιτροπή:

Αντώνιος Μπόγρης , Καθηγητής

Antonios Bogris
Digitally signed by Antonios Bogris
Date: 2022.05.31 21:48:21 +03'00'

Γεώργιος Πρεζεράκος , Καθηγητής

Georgios Prezerakos
Digitally signed by Georgios Prezerakos
Date: 2022.06.06 19:35:38 +03'00'

Ημερομηνία εξέτασης 31/05/2022

(Κενό φύλλο)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος/η Τσιμπερδώνης Κων/νος
του Πέτρου με αριθμό μητρώου 20002 φοιτητής/~~τρια~~ του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών του Τμήματος ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι …… και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Ο/Η Δηλών/~~ούσα~~

(Κενό φύλλο)

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της επεξεργασίας κειμένου. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένειά μου που θα ήθελε να τελειώσω τις σπουδές μου.

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με την προσβασιμότητα τόσο στο διαδίκτυο όσο και σε κινητές συσκευές, αφού δεν είναι δεδομένη για όλους τους χρήστες λόγω του ότι υπάρχουν κατασκευαστικοί και σχεδιαστικοί περιορισμοί στη διεπαφή της εφαρμογής που καθιστούν δύσκολη ή ακόμα και αδύνατη τη χρήση της από άτομα με αναπηρία (μόνιμη ή προσωρινή). Οι σκοποί της παρούσας διπλωματικής εργασίας, είναι : α) η παρουσίαση των σύγχρονων προδιαγραφών και σχεδιαστικών αρχών που προτείνουν οι διεθνείς οργανισμοί και θα πρέπει να ενσωματώνουν οι σελίδες των εφαρμογών για να είναι προσβάσιμες από όλους τους χρήστες και συσκευές. β) παρουσίαση και σύγκριση αυτοματοποιημένων μεθόδων/εργαλείων αξιολόγησης των δικτυακών τόπων στο αν πληρούν τα συγκεκριμένα κριτήρια-προδιαγραφές. γ) υλοποίηση ενός αυτοματοποιημένου εργαλείου για τον έλεγχο των δικτυακών τόπων ως προς τη συμμόρφωση τους στα διεθνή πρότυπα

ABSTRACT

The present thesis concerns the Accessibility to both the internet and mobile devices is not given to all users as there are construction and design limitations in the interface of the application that make it difficult or even impossible to use it by people with disabilities (permanent or temporary). The aims of this dissertation are: a) to present the modern specifications and design principles proposed by international organizations and should integrate the application pages to be accessible to all users and devices. b) presentation and comparison of automated methods / tools of evaluation of the websites in whether they meet the specific criteria-specifications. c) implementation of an automated tool for the control of websites in terms of their compliance with international standards.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

HTML: Hypertext Markup Language

CSS : Cascading Style Sheets

UN CRPD: United Nations Convention On The Rights Of Persons With Disabilities

XHMTL : Extensible Hypertext Markup Language

WAI : Wed Accessibility Iniative

W3C : World Wide Web

WCAG : Web Content Accessibility Guidelines

SC : Success Criterions

ATAG : Authoring Tools Accessibilty Guidelines

CMS : Content Management Systems

LMS : Learming Managmenst Systems

UAAG : User Agent Accessiblity Guidelines

WAI – ARIA : Web Accessibilty Initiative – Accessible Rich Internet Applications Suite

ACT : Accessibility Conformance Testing

TTML : Timed Text Markup Language

EARL : Evaluation and Report Language

WCAG-EM : Website Accessibilty Conformance Evaluation Methology

API : Application Programming Interface

WAD : Web Accessibilty Directive

SVG : Scalable Vendor Graphics

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο [1] αναλύεται το αντικείμενο της διπλωματικής εργασίας και γίνεται μια εισαγωγή στο διαδίκτυο και στις τεχνολογίες που χρησιμοποιούνται για την παρουσίαση της πληροφορίας.

1.1 Περιγραφή του αντικειμένου της διπλωματικής εργασίας

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι : α) η παρουσίαση των σύγχρονων προδιαγραφών και σχεδιαστικών αρχών που προτείνουν οι διεθνής οργανισμοί και θα πρέπει να ενσωματώνουν οι σελίδες των εφαρμογών για να είναι προσβάσιμες από όλους τους χρήστες και συσκευές. β) παρουσίαση και σύγκριση αυτοματοποιημένων μεθόδων/εργαλείων αξιολόγησης των δικτυακών τόπων στο αν πληρούν τα συγκεκριμένα κριτήρια-προδιαγραφές. γ) υλοποίηση ενός αυτοματοποιημένου εργαλείου για τον έλεγχο των δικτυακών τόπων ως προς τη συμμόρφωση τους στα διεθνή πρότυπα .

1.2 Γενικό πλαίσιο πρόσβασης

Το διαδίκτυο έχει επι της ουσίας σχεδιαστεί για να είναι προσβάσιμο από όλους τους ανθρώπους ανεξάρτητα από το εξοπλισμό και τα προγράμματα που διαθέτουν, την γλώσσα τους, την τοποθεσία τους και τις νοητικές ικανότητες τους. Συνεπώς η επίδραση της ανικανότητας αλλάζει ριζικά στο διαδίκτυο επειδή απομακρύνει τα εμπόδια της επικοινωνίας και της αλληλεπίδρασης που πολλοί άνθρωποι αντιμετωπίζουν στον φυσικό κόσμο. Όπως όταν οι ιστοσελίδες, εφαρμογές, τεχνολογίες ή εργαλεία είναι σχεδιασμένα με κακότεχνα, μπορούν να δημιουργήσουν εμπόδια και να αποκλείσουν ανθρώπους να τα χρησιμοποιήσουν με επιτυχία.

1.3 Τι είναι η προσβασιμότητα του διαδικτύου

Προσβασιμότητα στο διαδίκτυο σημαίνει ότι οι ιστοσελίδες,εργαλεία και τεχνολογίες είναι με τέτοιο τρόπο σχεδιασμένες και αναπτυγμένες ώστε άνθρωποι ακόμα και με δυσκολίες να μπορούν να τις χρησιμοποιήσουν με επιτυχία, και ειδικότερα να είναι σε θέση για να :

- Αντιλαμβάνονται , κατανοούν, προηγηθούν και αλληλοεπιδρούν με το διαδίκτυο
- Συνεισφέρουν στο διαδίκτυο.

Παρακάτω αναφέρονται μια σειρά από αναπηρίες που επηρεάζουν την προσβασιμότητα στο διαδίκτυο :

- Ακουστικές
- Γνωστικές / νοητικές
- Νευρολογικές
- Σωματικές
- Ομιλίας
- Οπτικές.

Η προσβασιμότητα στο διαδίκτυο έχει επιπλέον οφέλη και σε μερίδα ανθρώπων που δεν έχουν κάποιο είδος αναπηρίας όπως:

- Άτομα που χρησιμοποιούν κινητά τηλέφωνα, έξυπνες συσκευές (TVs και ρολόγια) και άλλες συσκευές με μικρές οθόνες, με διαφορετικές λειτουργίες εισόδου, κτλ.
- Ηλικιωμένα άτομα με μεταβαλλόμενες ικανότητες λόγω γήρανσης
- Άτομα με "προσωρινή δυσκολία" όπως ένα σπασμένο χέρι ή με χαμένα γυαλιά.
- Άτομα με "περιστασιακούς περιορισμούς" όπως αδυναμία έκθεσης σε έντονο ηλιακό φως ή σε περιβάλλον που δεν μπορούν να ακούσουν ήχο.
- Άτομα με αργή σύνδεση στο διαδίκτυο ή με περιορισμένο - ακριβό εύρος ζώνης

1.4 Η προσβασιμότητα είναι σημαντική για άτομα, επιχειρήσεις , κοινωνία.

Το διαδίκτυο είναι μια ολοένα και περισσότερο σημαντική λειτουργία/εργαλείο σε πολλούς τομείς της ζωής: Εκπαίδευση, απασχόληση, κυβέρνηση, εμπόριο, υγεία, φροντίδα, ψυχαγωγία. Είναι στοιχειώδες και κατανοητό πως το διαδίκτυο πρέπει να είναι προσβάσιμο για να παρέχει ισότιμη πρόσβαση και ευκαιρίες σε ανθρώπους με ποικιλόμορφες ικανότητες. Η απρόσκοπτη πρόσβαση στις τεχνολογικές επικοινωνίες, στις οποίες συμπεριλαμβάνεται και το διαδίκτυο , ορίζεται ως βασικό ανθρώπινο δικαίωμα από τον Οργανισμό Ηνωμένων Εθνών για τα Δικαιώματα των Προσώπων με Αναπηρία (UN CRPD).

Το διαδίκτυο προσφέρει μια πρωτοφανή πρόσβαση στην πληροφορία και στην αλληλεπίδραση για πολλούς ανθρώπους με αναπηρίες. Όπως πρόσβαση στα εμπόδια : εκτύπωσης , ακουστικών και οπτικών μέσων μπορούν πιο εύκολα να ξεπεραστούν μέσω των τεχνολογίες του.

Η προσβασιμότητα υποστηρίζει κοινωνικές εντάξεις για άτομα με διαφορετικές ανικανότητες, όπως:

- Γήρας
- Άτομα που σε αγροτικές περιοχές
- Άτομα που ζουν σε αναπτυσσόμενες χώρες

Επίσης από την προσβασιμότητα του διαδικτύου προσφέρονται καινοτόμες επιχειρηματικές δράσεις. Ο προσβάσιμος σχεδιασμός ενός ιστοτόπου βελτιώνει γενικότερα την εμπειρία και την ικανοποίηση του χρήστη , ειδικότερα σε ποικίλες καταστάσεις, διαφορετικών συσκευών και ηλικιωμένων χρηστών. Η προσβασιμότητα μπορεί να βελτιώσει την φίρμα, και να οδηγήσει σε καινοτομία καθώς και να επεκτείνει το πεδίο της αγοράς.

Η προσβασιμότητα στο διαδίκτυο απαιτείται από τον νόμο σε πολλές περιπτώσεις.

1.5 Τεχνολογίες που χρησιμοποιούνται για την παρουσίαση της πληροφορίας.

Παγκοσμίως είναι υπολογισμένο ότι τουλάχιστον 2,2 δισεκατομμύρια άνθρωποι στον κόσμο , έχουν κάποια δυσκολία στην όραση ή τύφλωση. Ο όρος Global Book Famine (Παγκόσμιος λοιμός βιβλίου) αναφέρεται στο γεγονός πως λιγότερο από το ποσοστό του 10% από τα δημοσιευμένα έργα, όπως βιβλία και εκπαιδευτικό υλικό είναι σε μορφή προσβάσιμη, όπως είναι η γραφή Braille, ευανάγνωστη εκτύπωση ή σε ηχητική μορφή. Το ποσοστό μειώνεται μόλις στο 1% στις αναπτυσσόμενες χώρες. Η έλλειψη στη πρόσβαση εκπαιδευτικών βιβλίων είναι ένα σοβαρό εμπόδιο στην προσπάθεια αυτών των ανθρώπων να έχουν ζώη γνώσης, μόρφωσης και δημιουργίας. Η προσβασιμότητα στο διαδίκτυο και ζητήματα πρόσβασης σε άτομα με οπτική δυσκολία δεν είναι νέες περιπτώσεις προβλημάτων για την κοινωνία άλλα συναφή. Για να είναι η πληροφορία που προσφέρεται από το διαδίκτυο προσβάσιμη σε χρήστες με δυσκολίες πρέπει να προσφέρεται με τις τεχνολογίες εκείνες που θα διευκολύνουν και θα ελκύουν τους χρήστες. Ό έλεγχος αυτών των χαρακτηριστικών λαμβάνει χώρα με την χρήση τεχνολογιών που σκοπό έχουν την όσο το δυνατόν πληρέστερη παρουσίαση της πληροφορίας.

Παρουσίαση τεχνολογιών για παρουσίαση της πληροφορίας:

- HTML: Η γλώσσα μορφοποίησης κειμένου(Hypertext Markup Language) είναι μια τεχνολογία δημιουργίας ιστοσελίδων. Παρέχει την βασική δομή μιας σελίδας μαζί με τα διαδικτυακά γραφικά (web graphics) και σενάρια (web scripting).

- Διαδικτυακά γραφικά (web graphics) είναι οπτικές αναπαραστάσεις σε μια ιστοσελίδα με σκοπό να βελτιώσουν ή να διευκολύνουν την απεικόνιση μιας ιδέας ή συναισθήματος. Επιπρόσθετος σκοπός είναι να προσελκύσει το ενδιαφέρον του χρήστη, να διασκεδάσει, να μορφώσει, ή και να διαφημίσει αυτά που απεικονίζονται στην διεπαφή.

- CSS : (Cascading Style Sheets)είναι μια γλώσσα υπολογιστή που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα μορφοποίησης κειμένου . Χρησιμοποιείται δηλαδή

για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

- Αναγνώστης οθόνης :Είναι μια τεχνολογία που βοηθά ανθρώπους που έχουν δυσκολίες όρασης να έχουν πρόσβαση και ανάδραση με ψηφιακό περιεχόμενο όπως ιστοσελίδες ή εφαρμογές μέσω ήχου ή επαφής. Η κύρια χρήση αυτών γίνεται από άτομα τυφλά ή χαμηλού επιπέδου όρασης

- Digital Braille: επιτέπει στους χρήστες να διαβάζουν κείμενο από μια φηψιακή οθόνη. Επι της ουσίας μια ανανεώσιμη οθόνη Braille είναι μια έξυπνη συσκευή που παράγει κώδικα Braille όταν συνδέεται με υπολογισή tablet ή smartphone.

- Λεζάντα (Caption) : Είναι γραμμές κειμένου που χρησιμοποιούνται για να περιγράψουν μια πολύπλοκη φωτογραφία ή ένα κείμενο. Η χρήση αυτής γίνεται συνήθως από άτομα χαμηλού επιπέδου όρασης.

- Απομαγνητοφωνημένο κείμενο (transcript): Είναι κείμενο το οποίο ενσωματώνεται μέσα σε video προκειμένου άτομα με χαμηλό επίπεδο ή καθόλου όραση να μπορούν να αντιληφθούν το νόημα του video.

- Voice Recognition software : Λογισμικό που εξυπηρετεί άτομα που δεν μπρούν να χρησιμοποιήσου. το ποντίκι ή το πληκτρολόγιο να επικοινωνήσουν με ψηφιακό περιεχόμενο.

- Selection Switches : εξυπηρετεί άτομα που δεν μπορούν να χρησιμοποιήσου. το ποντίκι ή το πληκτρολόγιο να επικοινωνήσουν με ψηφιακό περιεχόμενο.

- Μεγενθυντικό Φακός : Λογισμικό που παρέχει την δυνατότητα της μεγένθυνσης του κείμενου έτσι ώστε να γίνεται εύκολο στην ανάγνωση από άτομα με διαφόρους τύπους χαμηλής όρασης.

ΚΕΦΑΛΑΙΟ 2

Σε αυτό το κεφάλαιο [2] αναφέρονται τα πρότυπα και οι τεχνολογίες που υπάρχουν για να βοηθήσουν του χρήστες με προβλήματα - διαφόρων τύπων - για να έχουν πρόσβαση στη πληροφορία.

2.1 Πρότυπα

Για την παροχή προσβάσιμων ιστοτόπων στη καθημερινή ζωή των ανθρώπων, υπάρχουν οργανισμοί που δημιουργούν προδιαγραφές έτσι ώστε το περιεχόμενο και οι υπηρεσίες των ιστοτόπων να μπορούν να γίνουν αντιληπτές και να χρησιμοποιηθούν από όλους τους χρήστες και ειδικότερα από τους χρήστες με αναπηρία. Πρόσφατα η ευρωπαϊκή νομοθεσία (Ευρωπαϊκή νομοθεσία για την προσβασιμότητα -WAD- 2016/2102 σχετικά με την προσβασιμότητα των ιστοτόπων και των κινητών εφαρμογών σε φορείς του δημόσιου τομέα) έχει προωθήσει περαιτέρω το δικαίωμα σε ανθρώπους με αναπηρία να έχουν τη δυνατότητα της πρόσβασης σε δικτυακές δημόσιες πληροφορίες και υπηρεσίες. Επιπλέον η οδηγία WAD δίνει ιδιαίτερη έμφαση πάνω στη περιοδική παρακολούθηση των επιπέδων συμμόρφωσης της προσβασιμότητας των δημοσίων φορέων τονίζοντας πως τα αποτελέσματα αυτών πρέπει να συλλέγονται τακτικά για να δημιουργούνται συγκρίσιμα αποτελέσματα.

Προς αυτήν την κατεύθυνση και για την δημιουργία τέτοιου είδους εφαρμογών, η κοινοπραξία World Wide Web (W3C) δημιούργησε την διαδικτυακή πρωτοβουλία προσβασιμότητας (Web Accessibility Initiative - WAI), των οποίων οι στρατηγικές, πρότυπα και οι υποστηρικτικοί πόροι έχουν βαθιά επηρεάσει την εθνική και διεθνή νομοθεσία των οδηγιών στην προσβασιμότητα βασιζόμενοι σε συγκεκριμένα πρότυπα.

2.1.1 Πρότυπο WCAG (Web Content Accessibility Guidelines)

Για την δημιουργία ενός πιο προσβάσιμο περιεχόμενο ιστού το 1999 το W3C όρισε το πρότυπο - οδηγίες WCAG. Σύντομα διαπιστώθηκε ότι το συγκεκριμένο πρύτυπο ως προς το περιεχόμενο του διαπιστώθηκε δεν ήταν επαρκώς πλήρης και ακριβής στον ορισμό του. Έτσι το 2008 δημιουργήθηκε μια νέα έκδοση η WCAG 2.0 για την αντιμετώπιση των παραπάνω ζητημάτων.

Το πρότυπο WCAG είναι δομημένο σε τέσσερα επίπεδα και αναφέρεται στην πληροφορία που βρίσκεται σε μια ιστοσελίδα διαδικτύου ή σε μια εφαρμογή αυτού:

- Αρχές. Το περιεχόμενο του διαδικτύου θα πρέπει να είναι Αντιληπτό(Perceilable) , Λειτουργικό (Operable), Κατανοητό (Understable), Εύρωστο (Robust) , η ομαδοποίηση των οδηγιών στις τέσσερις αρχές έγινε με το αναθεωρημένο πρότυπο WCAG 2.0
- Κατευθυντήριες γραμμές. Για κάθε αρχή οι κατευθυντήριες γραμμές παρέχουν τους βασικούς στόχους προκειμένου να καταστεί το περιεχόμενο προσβάσιμο, σύμφωνα με την αρχή με την οποία σχετίζονται
- Κριτήρια επιτυχίας (SC). Για κάθε κατευθυντήρια γραμμή , τα SC παρέχονται ως δοκιμή που αντιπροσωπεύει τις απαιτήσεις για συμμόρφωση ως προ το πρότυπο. Στις περισσότερες περιπτώσεις ο έλεγχος πραγματοποιείται από το λογισμικό, ωστόσο υπάρχουν και περιπτώσεις που απαιτείται η χειροκίνητη παρέμβαση ειδικού στο αν κατά πόσο πληρεί το κριτήριο ή όχι. Το WCAG 2.0 παρουσίασε τρία επίπεδα της συμμόρφωσης που ταξινομούν τα κτιτήτια επιτυχίας με βάση τον αντίκτυπο τους στο συνολικό επίπεδο προσβασιμότητας του περιεχόμενου του διαδικτύου: Α(χαμηλό) , ΑΑ (μέτριο), ΑΑΑ(υψηλό).

- Επαρκείς και συμβουλευτικές τεχνικές. Κάθε κριτήριο επιτυχίας σχετίζεται με έναν αριθμό επίσημων τεχνικών που θα μπορούσαν να είναι επαρκής για την τήρηση της συμμόρφωσης ή απλώς συμβουλευτικές προτάσεις για το πως να βελτιωθεί η συμμόρφωση με το περιεχόμενο του ιστού σε σχέση με το κριτήριο επιτυχίας που εξετάστηκε. Οι τεχνικές αυτές χωρίζονται σε κατηγορίες, ανάλογα με την παρέμβαση που περιγράφουν και αυτή η κατηγορία υποδεικνύεται στο αρχικό γράμμα του ονόματος της τεχνικής.(π.χ. το γράμμα H σχετίζεται με την HTML, το γράμμα C με το CSS , το G είναι μια γενική τεχνική). Μεταξύ των τεχνικών υπάρχουν και οι αποκαλούμενες αποτυχίες (ξεκινούν με το γράμμα F). Το εργαλείο μας καθορίζει την επαλήθευση των οδηγιών μέσω ενός συνόλου σημείων ελέγχου. Το WCAG 2.1 έκδοση εισήγαγε 17 νέα κριτήρια επιπλέον της έκδοσης 2.0

Το WCAG πρωτίστως προορίζεται για

- προγραμματιστές περιεχομένου ιστού
- προγραμματιστές/γραφίστες με συγγραφικά εργαλεία ιστού

Για οποιαδήποτε άλλον ένα πρότυπο για πρόσβαση στον ιστό συμπεριλαμβανομένου και της πρόσβασης με κινητές συσκευές. Οι σχετικοί πόροι προορίζονται για να καλύψουν τις ανάγκες πολλών διαφορετικών ανθρώπων συμπεριλαμβομένων των ερευνητών, των διευθυντών για την λήψη αποφάσεων κ.α

2.1.2 Πρότυπο ATAG (Authoring Tools Accessibilty Guidelines)

Τα συγγραφικά εργαλεία (Authoring Tools) είναι λογισμικό και υπηρεσίες τα οποία οι "συγγραφείς" (web developers , designers, writters ,κτλ.) τα χρησιμοποιούν για να παράγουν περιεχόμενο ιστού. Για παράδειγμα, οι συντάκτες της HTML καθώς και τα συστήματα διαχείρισης περιεχομένου (CMS) δημιουργούν ιστότοπους για να επιτρέπουν στους χρήστες να προσθέσουν περιεχόμενο όπως ιστολόγια και κοινωνικοί ιστότοποι. Τα ATAG έγγραφα εξηγούν το πως:

- Τα προσβάσιμα συγγραφικά εργαλεία θα βοηθούν τα άτομα με ανικανότητες να μπορούν να δημιουργήσουν περιεχόμενο ιστοτόπου
- Να βοηθήσουν τους “συγγραφείς” στην δημιουργία περισσότερου προσβάσιμου περιεχόμενου ιστού. Συγκεκριμένα, να επιτρεπεί, να υποστηρίζει και να προάγει την παραγωγή του περιεχομένου που συμμορφώνεται με το Web Content Accessibility Guidelines (WCAG)

Το πρότυπο ATAG είναι πρωτίστως για προγραμματιστές συγγραφικών εργαλείων που περιλαμβάνουν τους παρακάτω τύπους:

- Συγγραφικά εργαλεία σελίδων ιστού όπως για παράδειγμα what-you-see-what-you-get (WYSIWYG) HTML συντάκτες.
- Λογισμικό για την δημιουργία ιστοσελίδων για παράδειγμα διαχείριση συστημάτων περιεχομένου (CMS) , συστήματα διαχείρισης μάθησης (LMS), εργαλεία μαθημάτων
- Λογισμικό το οποίο μετατρέπει τις τεχνολογίες περιεχομένου ιστού όπως επεξεργαστές κειμένου και άλλες εφαρμογές γραφείου.
- Συγγραφικά εργαλεία πολυμέσων
- Ιστότοποι οι οποίοι επιτρέπουν στους χρήστες να προσθέσουν περιεχόμενο όπως για παράδειγμα ιστότοποι.

Το ATAG και οι υποστηριζόμενοι πόροι έχουν πρόθεση να ικανοποιήσουν τις ανάγκες ανθρώπων που προέρχονται από διαφορετικά πεδία όπως:

- Άτομα που θέλουν να διαλέξουν συγγραφικό εργαλέιο το οποίο παράγει προσβάσιμο περιεχόμενο και μπορεί να χρησιμοποιήσει το πρότυπο ATAG.
- Άτομα κυρίως προγραμματιστές αλλά και εταιρίες ανάπτυξης και διανομής που θέλουν να ενισχύσουν τον υπάρχοντα κώδικα του εργαλείου συγγραφής και να βελτιώσουν την προσβασιμότητα σε μελλοντικές εκδόσεις.

Εκδόσεις ATAG :

- ATAG 1.0 : Εγκρίθηκε τον Φεβρουάριο του 2000 και εξακολουθεί να είναι ένα έγκυρο πρότυπο
- ATAG 2.0 : είναι το σημερινό πρότυπο. Είναι μια σύσταση του W3C. Αυτό σημαίνει ότι είναι ένα ολοκληρωμένο πρότυπο, έχοντας λάβει δημόσια σχόλια, έχει ολοκληρώσει δοκιμές υλοποίησης, καθώς επίσης έχει παρουσιάσει και παραδείγματα λογισμικού που το χρησιμοποιούν για να κάνουν τα προιόντα τους πιο προσιτά στο εμπόριο και έχει εγκριθεί από τα μέλη του W3C.

ATAG 2.0 περιέχει δύο κύρια μέρη:

- Μέρος Α περιέχει τη αυτοδημιουργία του ίδιου του συγγραφικού εργαλείου ως προσβάσιμο.
- Μέρος Β περιέχει το συγγραφικό εργαλείο ως βοήθεια στους συγγραφείς για να παράγουν προσβάσιμο περιεχόμενο.

ATAG 2.0 είναι οργανωμένο σε επίπεδα:

- Αρχές που παρέχουν υψηλού επιπέδου οργάνωση για τις κατευθυντήριες γραμμές
- Κατευθυντήριες γραμμές παρέχουν το πλαίσιο και τους σκοπούς για τα επιτυχημένα κριτήρια
- Επιτυχημένα κριτήρια είναι οι απαιτήσεις προσβασιμότητας οι οποίες γράφονται ως ελεγχόμενες δηλώσεις σε τρία επίπεδα Α , ΑΑ, ΑΑΑ.

2.1.3. Πρότυπο UAAG (User Agent Accessiblity Guidelines)

Το πρότυπο αυτό εξηγεί πως γίνονται προσβάσιμα σε άτομα με ειδικές ανάγκες τα προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης και άλλες εφαρμογές του ιστού. Ορισμένες ανάγκες για προσβασιμότητα καλύπτονται καλύτερα στο πρόγραμμα περιήγησης παρά στο περιεχόμενο ιστού όπως η προσαρμογή κειμένου, οι προτιμήσεις και η προσβασιμότητα της διεπαφής του χρήστη. Τηρώντας αυτό το πρότυπο, θα βελτιωθεί η προσβασιμότητα της διεπαφής του χρήστη και της ικανότητάς του να επικοινωνεί με άλλες τεχνολογίες συμπεριλαμβανομένου των υποστηρικτικών τεχνολογιών.

Το UAAG απευθύνεται κυρίως σε προγραμματιστές φυλλομετριτών ιστού , σε επεκτάσεις , προγράμματα ανάγνωσης και άλλες εφαρμογές που απαιτείται περιεχόμενο ιστού. Επίσης μπορεί να χρησιμοποιηθεί και από άτομα με διαφορετικό υπόβαθρο όπως:

- Άτομα τα οποία θέλουν να επιλέξουν προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης στο πιο είναι περισσότερο προσβάσιμο και τα συγκρίνουν.
- ‘Άτομα τα οποία επιθυμούν να βελτιώσουν προγραμματιστικά την πρόσβαση σε προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης.

Εκδόσεις UAAG :

- UAAG 1.0 : εγκρίθηκε τον Δεκέμβριο 2002
- UAAG 2.0: αναπτύχθηκε για να βοηθήσει τις μελλοντικές γενιές προγραμμάτων περιήγησης ιστού να γίνουν πιο προσιτές να παρέχει εναλλακτικές πληροφορίες με βάση την τεχνολογία και την πλατφόρμα των χρηστών. Το UAAG 2.0 παρέχει συγκεκριμένες πληροφορίες για προγράμματα περιήγησης και για επαγγελματίες που ασχολούνται με την προσβασιμότητα. Ενώ όμως αποδείχτηκε πως είναι δυνατή η εφαρμογή των κριτηρίων του προτύπου UAAG 2.0, το W3C δεν σχεδιάζει να το προωθήσει αλλά να το κάνει χρήση σε ομάδες εργασίας για μελλοντικές εργασίες προσβασιμότητας.

Το έργο των τρεχουσών ομάδων εργασίας για την προσβασιμότητα σε κινητά και την προσβασιμότητα χαμηλής όρασης δείχνει την σημασία της συνδυασμένης εξέτασης περιεχομένου διεπαφής χρήστη, επεκτάσεων και εφαρμογών. Ενώ πολλές δυνατότητες του προτύπου UAAG 2.0 υποστηρίζονται σε μεμονωμένα προγράμματα περιήγησης, υπάρχει η ανάγκη για πιο συνετή και αξιόπιστη υποστήριξη για τις λειτουργίες προσβασιμότητας σε όλα τα προγράμματα περιήγησης. Το UAAG 2.0 περιέχει συγκεκριμένες οδηγίες για προγραμματιστές προγράμματων περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης , που θέλουν να δημιουργήσουν μαι καλύτερη εμπειρία χρήστη για όλους.

2.2 Τεχνολογίες - Τεχνικές Προδιαγραφές

2.2.1 WAI – ARIA: (Web Accessibilty Initiative – Accessible Rich Internet Applications Suite) καθορίζει το τρόπο για να δημιουργηθεί περιεχόμενο και εφαρμογές διαδικτύου περισσότερο προσβάσιμες σε άτομα με δυσκολίες. Ειδικότερα εξυπηρετεί με δυναμικό περιεχόμενο και προηγμένα στοιχεία ελέγχου διεπαφής χρήστη με HTML, javascript και σχετικές τεχνολογίες. Χωρίς WAI – ARIA ορισμένες λειτουργίες που χρησιμοποιούνται σε ιστοχώρους δεν είναι διαθέσιμες σε ορισμένους χρήστες μς αναπηρίες, ειδικά σε άτομα που βασίζονται σε προγράμματα ανάγνωσης οθόνης και σε άτομα που δεν μπορούν να αναγνωρίσουν το ποντίκι. Το WAI – ARIA αντιμετωπίζει αυτές τις προκλήσεις προσβασιμότητας, για παράδειγμα, ορίζοντας τρόπους παροχής λειτουργικότητας στην υποστηρικτική τεχνολογία. Με το WAI – ARIA οι προγραμματιστές μπορούν να μετατρέψουν τις προηγμένες εφαρμοφές ιστού προσβάσιμες και χρησιμποιούμενες σε άτομα με ειδικές ανάγκες.

Τεχνικές Λύσεις : WAI – ARIA παρέχει ένα πλαίσιο προσθέτοντας χαρακτηριστικά για τον προσδιορισμό χαρακτηριστικών για την αλληλεπίδραση με τον χρήστη, πως σχετίζονται μεταξύ τους και την τρέχουσα καταστασή τους. Το WAI – ARIA περιγράφει τεχνικές πλοήγησεις για την επισήμανση περιοχών και κοινών δομών ιστού ως μενού, πρωτεύον και δευτερεύον περιεχόμενο , πληροφορίες banner και άλλους τύπους δομών διαδικτύου. Για παράδειγμα , με το WAI – ARIA, οι προγραμματιστές μπορούν να αναγνωρίσουν περιοχές των

σελίδων και να επιτρέψουν στους χρήστες το πληκτρολογίου να μετακινούνται εύκολα μεταξύ των περιοχών , αντί να χρειάζονται να πατήσουν το tab πολλές φορές. Το WAI – ARIA περιλαμβάνει επίσης τεχνολογίες τη χαρτογράφηση στοιχείων ελέγχου , περιοχές σε πραγματικό χρόνο και διεπαφών προγραμματισμού εφαρμογών προσβασιμότητας (API), συμπεριλαμβανομένων στοιχείων ελέγχου σε ακριβές διαδικτυακές εφαρμογές.

WAI – ARIA παρέχει στους συγγραφείς τα παρακάτω:

a. Ρόλους για να περιγράψουν τύπους γραφικών στοιχείων όπως "menu","slider" και “progressbar”.
b. Ρόλους για να περιγράψουν τύπους δομές μιας ιστοσελίδας όπως κεφαλίδες , περιοχές και πίνακες
c. Ρόλους για να περιγράψουν την κατάσταση από του τύπους γραφικών στοιχείων όπως "checked" σε ένα checked box ή "haspopup" για το “menu”.
d. Ιδιότητες για τον καθορισμό περιοχών σε πραγματικό χρόνο που πιθανόν να απαιτείται ενημέρωση (όπως τιμές μετοχών) καθώς και μια πολιτική διακοπής για αυτές τις ενημερώσεις όπως για παράδειγμα κρίσιμες ενημερώσεις ενδέχεται να παρουσιάζονται σε ένα παράθυρο διαλόγου ειδοποίησης.

Εκδόσεις WAI-ARIA :

- WAI-ARIA 1.0 : δημοσιεύτηκε σαν μια συμπληρωμένη σύσταση του W3C στις 20 Μαρτίου 2014.
- WAI-ARIA 1.1 : δημοσιεύτηκε σαν μια συμπληρωμένη σύσταση του W3C στις 14 Δεκεμβρίου 2017.
- WAI-ARIA 1.2 : είναι σε φάση ανάπτυξης

WAI-ARIA 1.2 :

Η έκδοση 1.2 θα επεκτείνει το WAI-ARIA 1.1 για να περέχει έναν μικρό αριθμό λειτουργιών για την ολοκλήρωση μοντέλου προσβασιμότητας HTML + ARIA.

Τα δημοσιευμένα προσχέδια του WAI-ARIA 1.2 είναι τα κάτωθι:

- WAI-ARIA 1.2 : παρέχει δυνατότητες για τον ορισμό προβάσιμων στοιχείων διεπαφής χρήστη και για την βελτίωση της προσβασιμότητας και της διαλειτουργικότητας του περιεχομένου και των εφαρμογών ιστού. Απευθύνεται κυρίως σε προγραμματιστές προγραμμάτων περιήγησης ιστού ,υποστηρικτικών τεχνολογιών και προγραμματιστών εργαλείων αξιολόγησης προσβασιμότητας.
- Core Accessibility API Mappings 1.2: περιγράφει πως τα προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης και άλλες εφαρμογές του ιστού πρέπει να εκθέτουν τη σημασιολογία των γλωσσών περιεχομένου σε προσβασιμότητα API. Η βασική ενότητα ορίζει την υποστήριξη που εφαρμόζεται σε πολλές τεχνολογίες περιεχομένου συμπεριλαμβανομένης της γενικής υποστήριξης πλοήγησης με πληκτρολόγιο και της χαρτογράφησης χαρακτηριστικών WAI-ARIA γενικής χρήσης
- Accessible Name and Description: Computation and API Mapping 1.2: περιγράφει πως τα προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης και άλλες εφαρμογές του ιστού καθορίζουν ονόματα και περιγραφές προσβάσιμων αντικειμένων από γλώσσες περιεχομένου ιστού και τα εκθέτουν σε API προσβασιμότητας. Αυτό επιτρέπει στις

υποστηρικτικές τεχνολογίες να συσχετίζουν και να αναμεταδίδουν το όνομα ή την περιγραφή των αντικειμένων στους χρήστες

- HTML Accessiblity API Mappings 1.0: επεκτείνει Core Accessibility API Mappings 1.2 και Accessible Name and Description: Computation and API Mapping 1.2 εξηγώντας πως τα προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης και άλλες εφαρμογές του ιστού αντιστοιχίζουν την HTML σε διεπαφές προγραμματισμού εφαρμογών προσβασιμότητας πλατφόρμας (API).
- SVG Accessiblity API Mappings 1.0 : επεκτείνει Core Accessibility API Mappings 1.2 και καθορίζει πως τα προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης και άλλες εφαρμογές του ιστού αντιστοιχίζουν SVG σε πλατφόρμα προσβασιμότητας APIs. Όταν υποστηρίζονται από προγράμματα περιήγησης, επεκτάσεις προγραμμάτων περιήγησης, προγράμματα παραγωγής πολυμέσων , προγράμματα ανάγνωσης και άλλες εφαρμογές του ιστού τότε επιτρέπουν στους συγγραφείς SVG να δημιουργήσουν ακριβές διαδικτυακές εφαρμογές.
- WAI-ARIA Authoring Practices 1.2: συνιστά προσεγγίσεις για να βοηθήσουν τους προγραμματιστές εφαρμογών ιστού να κάνουν προσβάσιμα γραφικά στοιχεία, πλοήγηση και συμπεριφορές χρησιμποιώντας ρόλους,καταστάσειςκαι ιδιότητες WAI-ARIA.

Μορφή τεχνικού εγγράφου:

Τα έγγραφα WAI-ARIA ακολουθούν την μορφή W3C για τεχνικές προδιαγραφές τα οποία περιλαμβάνουν πολλές ενότητες στην αρχή: συνδέσμους σε διαφορετικές εκδόσεις, συντάκτες ,πνευματικά δικαιώματα, περίληψη και σύνδεσμο για αποστολή τυχόν λάθη - παραλήψεις με e-mail προς τον συντάκτη.

- Audio and Video:
    - WebVTT: The Web Video Text Tracks Format : είναι μια μορφή για υπότιτλους ,περιγραφές video κειμένου και άλλα μεταδεδομένα που είναι χρονικά ευθυγραμμισμένα με περιεχόμενο ήχου ή video.

- Timed Text Markup Language (TTML) : προορίζεται για την δια κωδικοποίηση ή την ανταλλαγή χρονομετρημένων πληροφοριών κειμένου μεταξύ μορφών περιεχομένου διανομής παλαιού τύπου για υπότιτλους και λεζάντες

- Αξιολόγηση :

  Οι παρακάτω αναφορές υποστηρίζουν εργαλεία και μεθόδους για την αξιολόγηση των εφαρμογών της προσβασιμότητας.

  - Accessibility Conformance Testing (ACT) Overview: θεσπίζει και τεκμηριώνει κανόνες για τον έλεγχο της συμμόρφωσης του περιεχομένου του ιστού με τα πρότυπα της προσβασιμήτας.
  - Evaluation and Report Language (EARL) - Overview : αναγνώσιμη από μηχανή μορφή για αποτελέσματα δοκιμών
  - WCAG-EM Overview: Website Accessibilty Conformance Evaluation Methology : Είναι μια προσέγγιση για να καθορίζει πόσο καλά ένας ιστότοτος ακολουθεί το πρότυπο WCAG

- Εξατομίκευση:

  Η εξατομίκευση περιλαμβάνει την προσαρμογή της εμπειρίας του χρήστη ώστε να ανταποκρίνεται στις ανάγκες και στις προτιμήσεις του μεμονωμένου χρήστη. Οι συντάκτες περιεχομένου μπορούν να χρησιμοποιούν πρότυπα εξατομίκευσης για να παρέχουν έναν προεπιλεγμένο σχεδιασμό και να επιτρέψουν την εξατομίκευση του χρήστη με ελάχιστο εργασιακό κόστος

- Προφορά:

Αφορά προγράμματα ανάγνωσης οθόνης και σύνθεση κειμένου σε ομιλία προκειμένου να προφέρουν σωστά το περιεχόμενο

ΚΕΦΑΛΑΙΟ 3

Σε αυτό το κεφάλαιο [3] γίνεται μια παρουσίαση και σύγκριση αυτοματοποιημένων μεθόδων/εργαλείων αξιολόγησης των δικτυακών τόπων στο αν πληρούν τα συγκεκριμένα κριτήρια-προδιαγραφές.

Τα εργαλεία που επιλέχθηκαν να συγκριθούν είναι :

- Εργαλεία που μπορούν εύκολα να χρησιμοποιηθούν από οποιοδήποτε χρήστη και δεν απαιτούν μεγάλη και πολύπλοκη εγκατάσταση
- Εργαλεία για συγκεκριμένες εγκαταστάσεις λογισμικού και γνώσεις προγραμματισμού
- Εργαλεία που δεν προσεγγίζουν μια μόνο πτηχή της προσβασιμότητας όπως εικόνες ή χρωματική αντίθεση.

Τα εργαλεία αυτά είναι το MAUVE++ και το WAVE.

WAVE :

Το επιλεχθέν εργαλείο είναι σε θέση να επικυρώσει μια σελίδα ιστού εισάγοντας το URL στο προεπιλεγμένο πεδίο. Για κάθε σελιδα που αξιολογεί εμφανίζει τα σφάλματα σε προβολή αναφορικά με τον κώδικα της σελίδας όπου τα λάθη επισημαίνονται με εικονίδιο και στη γραμμή του κώδικα που εμφανίζονται. Επιπλέον το εργαλείο παρέχει μια οπτική αναπαράσταση των πτυχών της προσβασιμότητας μέσα στη σελίδα επισημαίνοντας με εικονίδια όχι μόνο τα σφάλματα προσβασιμότητας, αλλά τα δομικά στοιχεία μέσα στην σελίδα όπως και τα στοιχεία ARIA. Επιπλέον για να βοηθήσει στην πλοήγηση μεταξύ των σφαλμάτων το WAVE παρέχει ένα πλευρικό μενού που δείχνει όλες τις επισημασμένες πτυχές στη σελίδα και τις λεπτομέρειες τους.

MAUVE++ :

Το έτερο επιλεχθέν εργαλείο δίνει την δυνατότητα να επικυρώσει μόνο μια σελίδα ιστού ή ομάδες σελίδων ή ολόκληρους ιστότοπους. Είναι σε θέσει να αναλύσει σελίδες σύμφωνα με το πρότυπο WCAG 2.1 το οποίο παρέχει 17 επιπλέον κριτήρια (σε σχέση με το WCAG 2.0 το οποίο παρέχεται ως δυνατότητα επιλογής) επιτυχίας για την αντιμετώπιση των εμποδίων κινητής προσβασιμότητας και τις ανάγκες των

ατόμων με χαμηλή όραση και γνωστική ή μαθησιακής αναπηρίας. Παρέχει διπλή προβολή των αποτελεσμάτων προσβασιμότητας. Η μία προβολή είναι προσανατολισμένη στον κώδικα - όπου υπάρχουν τα σφάλματα και οι προειδοποιήσεις παρουσιάζοντας τον αριθμό γραμμής του αξιολογημένου κώδικα, η άλλη είναι μια γραφική προβολή για χρήστες χωρίς δεξιότητες προγραμματισμού ή οποία εμφανίζει τα σφάλματα και τις προειδοποιήσεις μέσω της βοήθειας γραφημάτων και πινάκων. Επιπλέον παρέχει μια ανασκόπηση των αποτελεσμάτων επικύρωσης που δείχνει τον αριθμό των σφαλμάτων, προειδοποιήσεων και επιτυχιών.

3.1 Σύγκριση των δύο εργαλείων

Για την εκτέλεση της σύγκρισης έχουν επιλεχθεί μια σειρά από δώδεκα γνωστές ιστοσελίδες που σχετίζονται με την καθημερινή ζωή των χρηστών σε τομείς (δύο ιστότοπου για κάθε τομέα): ηλεκτρονικό εμπόριο, δημόσιο σώμα , πολιτιστική κληρονομιά ,υγεία ,ταξίδια και εκπαίδευση. Για κάθε ένα από τους δώδεκα αυτούς ιστότοπους , τέσσερις υποχώροι έχουν επιλεχθεί : αρχική σελίδα , σελίδα με φόρμες, σελίδα με πολύ περιχόμενο κειμένου και σελίδα με πολλούς υπερσυνδέσμους. Οι ιστοσελίδες έχουν επικυρωθεί σύμφωνα με το πρότυπο WCAG 2.1 και με σεβασμό το επίπεδο συμμόρφωσης AA - που συνιστά η οδηγία WAD ως απαίτηση - και με τα δύο εργαλεία.

Για την διεξαγωγή της ανάλυσης εφαρμόστηκε η προσέγγιση του Brajnic [4] : Κάθε επιλεγμένη ιστοσελίδα επικυρώνεται από δύο εργαλεία και επιθεωρούνται χειροκίνητα από ένα εμπειρογνώμονα στην προσβασιμότητα άνθρωπο για να διαπιστωθεί η πληρότητα των αποτελεσμάτων των εργαλείων. Τα επίπεδα ελέγχου μπορούν να κατηγοριοποιηθούν σε:

- Πραγματικά θετικό (ΠΘ) : Αποτέλεσμα που ταξινομείται από το εργαλείο ως ΠΘ εάν μετά τον έλεγχο του εμπειρογνώμονα το θεωρεί σωστό και σχετικό
- Ψευδώς θετικό (ΨΘ) : Αποτέλεσμα που ταξινομείται από το εργαλείο ως ΨΘ εάν μετά τον έλεγχο του εμπειρογνώμονα το θεωρεί λάθος ή άσχετο

- Ψευδώς αρνητικό (ΨΑ) : Αποτέλεσμα που ταξινομείται από το εργαλείο ως ΨΑ εάν δεν μπορεί να εντοπιστεί τότε ο εμπειρογνώμονάς το θεωρεί σχετικό και λάθος

Εγκυρότητα : καλείται η μέτρηση η οποία δείχνει πόσο το εργαλείο εντοπίζει σωστά προβλήματα προσβασιμότητας δηλαδή πόσα από τα λάθη προσβασιμότητας ανιχνεύονται από το εργαλείο ΠΘ και δίνεται από τον τύπο :

$$Εγκυρότητα = \frac{ΠΘ}{ΠΘ + ΨΘ}$$

Πληρότητα: Καλείται η μέτρηση η οποία δείχνει πόσο το εργαλείο ανιχνεύει την πραγματική προσβασιμότητα δηλαδή πόσα λάθη ΠΘ ανιχνεύονται από το εργαλείο μεταξύ όλων των πραγματικών λαθών προσβασιμότητας και υπολογίζεται από τον τύπο:

$$Πληρότητα = \frac{ΠΘ}{ΠΘ + ΨΑ}$$

3.2 Αποτελέσματα Ελέγχου

Ο έλεγχος πραγματοποιήθηκε κατά την διάρκεια του Ιανουαρίου 2020 και με την τελευταία έκδοση και των δύο εργαλείων.

Εργαλείο	Εγκυρότητα	Πληρότητα
MAUVE++	90%	93,2%
WAVE	62,9%	12,9%

Πίνακας 1 Τιμές των εργαλείων ελέγχου

Από τα παραπάνω συμπεραίνουμε πως το εργαλαίο MAUVE++ είναι πιο πλήρης και πιο έγκυρο από το εργαλείο WAVE.

Διαπιστώθηκε πως δεν υπάρχουν ιδιαίτερες διαφορές στις επιδόσεις στο τύπο της αξιολογηθήσας ιστοσελίδας. Όπως φαίνεται και από τον πίνακα 2 ο οποίος αναφέρεται στην αξιολόγηση τεσσάρων ιστοσελίδων από τον ιστότοπο IKEA, η δομή της σελίδας δεν επηρεάζει τα συνολικά απότελέσματα ως προς την πληρότητα και την εγκυρότητα.

IKEA		Tot of detected Errors	FP	FN	TP
Home page	MAUVE++	803	30	0	773
	WAVE	1	1	773	0
Page with links	MAUVE++	812	20	1	792
	WAVE	1	1	793	0
Page with textual content	MAUVE++	757	12	1	745
	WAVE	1	1	747	0
Page with forms	MAUVE++	2	1	742	1
	WAVE	767	21	1	741

Πίνακας 2. Επικύρωση αποτελέσμάτων ιστοτόπου IKEA

Κατηγορία	Εγκυρότητα	
	MAUVE++	WAVE
Ηλεκτρονικό Εμπόριο	96,4	60
Δημόσιος φορέας	87,3	50
Πολιτιστική κληρονομιά	89	71,2
Υγεία	88,4	59,9
Ταξίδια	92	65,6
Εκπαίδευση	89,6	70,8

Πίνακας 3. Επικύρωση τιμών(%) των εργαλείων με ομαδοποίηση

Κατηγορία	Πληρότητα	
	MAUVE++	WAVE
Ηλεκτρονικό Εμπόριο	98,9	50
Δημόσιος φορέας	98,7	0,2
Πολιτιστική κληρονομιά	98,7	3,4
Υγεία	90,4	9
Ταξίδια	87,5	13,5
Εκπαίδευση	97	1,3

Πίνακας 4. Πληρότητα τιμών(%) των εργαλείων με ομαδοποίηση

Οι αναφερόμενες μέσες τιμές δείχνουν ότι το MAUVE++ ήταν ελαφρώς πιο πληρέστερο παρά έγκυρο ενώ το WAVE ήταν περισσότερο έγκυρο παρά πλήρης. Επιπλέον , υπήρξε σημαντική διαφορά στους όρους πληρότητας μεταξύ των δύο εργαλείων.

Το WAVE πραγματοποίησε την αξιολόγηση με την χαμηλότερη τιμή της πληρότητας (0.1%) στην αξιολόγηση στην δεύτερη ιστόσελιδα στην ομαδοποίηση «Δημόσιος φορέας». Αυτή η σελίδα είχε στην πραγματικότητα πολλούς υπερσυνδέσμους κατά τους οποίους το εργαλείο δεν εντόπισε τις παραβάσεις του SC 4.1.2 (missing label aria-label και aria-labelledby atrributes) . Την χαμηλότερη τιμή πληρότητας που παρατηρήθηκε (75%) αφορά στην αξιολόγηση της δεύτερης ιστοσελίδας του ιστοχώρου «Ταξίδια» η οποία παρουσίασε πολλά προβλήματα στην αντίθεση των χρωμάτων και το MAUVE++ δεν εντόπισε το σχετικό πρόβλημα. Για τιμή της εγκυρότητας οι χαμηλότερες καταγεγραμμένες τιμές ήταν:

- 20% για το WAVE που προκύπτει από την αξιολόγηση της πρώτης ιστοσελίδας στο τομέα του «Ελεκτρονικό Εμπόριο». Εδώ το WAVE δημιούργησε ψευδώς θετικά στην ανίχνευση σφαλμάτων αντίθεσης χρώματος.
- 77,3% για το MAUVE++ που προκύπτει από την αξιολόγηση της δεύτερης ιστοσελίδας στο τομέα του «Δημόσιου φορέα». Εδώ το MAUVE++ δημιούργησε ψευδώς θετικά εντοπίζοντας λάθη χρωματικών αντιθέσεων σε στοιχεία φορμών .

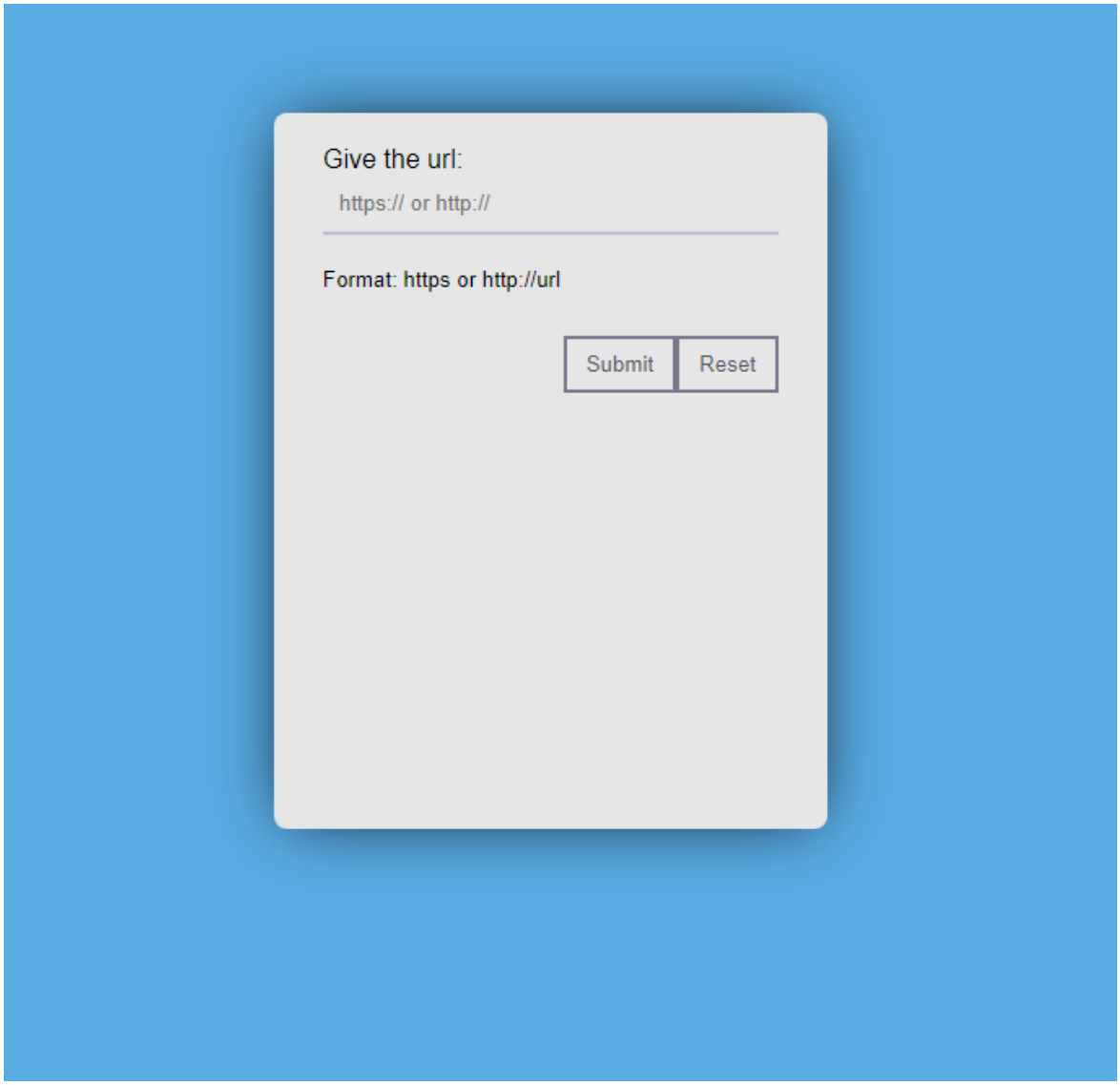
Όσον αφορά την συνέπεια εξετάστηκαν τα κριτήρια επιτυχίας που παραβιάστηκαν συχνότερα και από τα δύο εργαλεία. Αρχικά εξετάστηκαν στον αριθμό των επιτυχημένων κριτηρίων του WCAG2.1 με επίπεδο συμμόρφωσης AA , 50 συνολικά που και τα δύο εργαλεία μπορούσαν να ελέγξουν. Το εργαλείο MAUVE++ έλεγξε 33 από τα 50 κριτήρια ενώ το WAVE 25. Τα περισσότερα σφάλματα διαπιστώθηκαν σε ελλείψεις των χαρακτηριστικών ARIA, σε ετικέτες (labels) σε υπερσυνδέσμους καθώς και σε στοιχεία διεπαφής με υπερσύνδεσμο.
Ελέγχοντας περισσότερο τα αποτελέσματα από το εργαλείο MAUVE++, τα πιο ανιχνευμένα θέματα ΠΘ είχαν σχέση με το κριτήριο 4.1.2 (Name,Role,Value) με 5995 εμφανίσεις στις δοκιμές που πραγματοποιήθηκαν. Ακολούθησαν τα κριτήρια 1.4.10 με 433 εμφανίσεις και τέλος 2.4.4 με 113 εμφανίσεις.

Αντίθετα στο εργαλείο WAVE τα πιο ανιχνευμένα θέματα ΠΘ είχαν σχέση με το κριτήριο 2.4.4 (Link Purpose In Context)[2] με 118 εμφανίσεις στις δοκιμές που πραγματοποιήθηκαν.Ακολούθησε το κριτήριο 1.4.3 με 65 εμφανίσεις.
Αξίζει να σημειώσουμε πως το κριτήριο επιτυχίας 1.4.3(Contrast, Minimum) το οποίο είναι το δεύτερο σε αριθμό παραβιάσεων στο εργαλείο WAVE , στο εργαλείο WAUVE++ δεν βρέθηκε σε μεγάλο αριθμό παραβάσεων. Επιπλεόν το εργαλείο WAUVE++ δεν ανίχνευσε σε ικανοποιητικό βαθμο το κριτήριο επιτυχίας 1.4.3, ενώ στο εργαλείο WAVE δεν εντοπίστηκε στα αποτελέσματα αξιολόγησης το κριτήριο επιτυχίας 4.1.2

ΚΕΦΑΛΑΙΟ 4

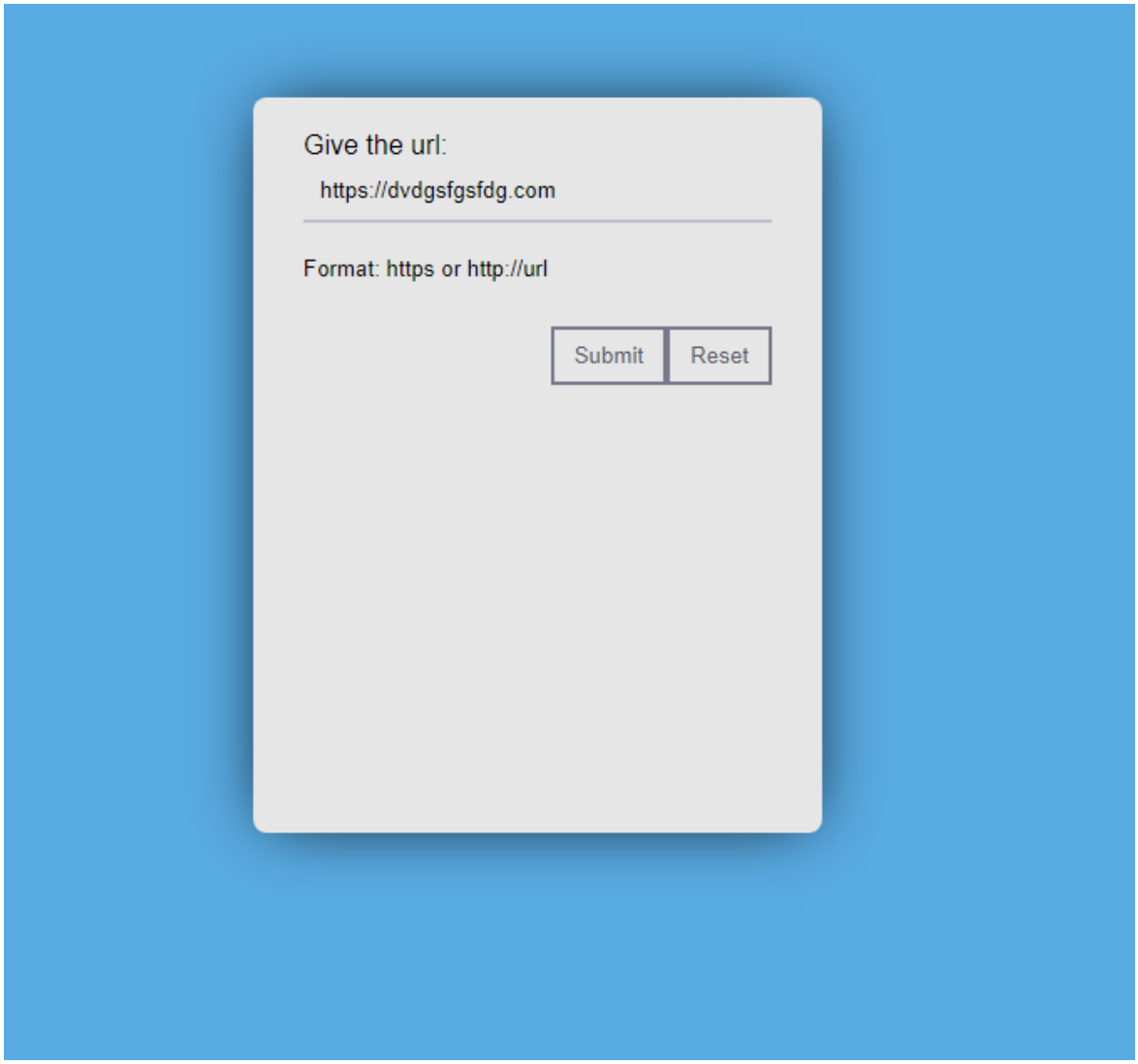
Στο κεφαλαιο αυτό παρουσιάζεται η υλοποίηση ενός αυτοματοιημένου εργαλείου που δοθείσας μιας ορθής ηλεκτρονικής διεύθυνσης , εμφανίζει τα αποτελέσματα κατά το πρότυπο WCAG 2.1.

Αρχική σελίδα όπου καταχωρείται το url



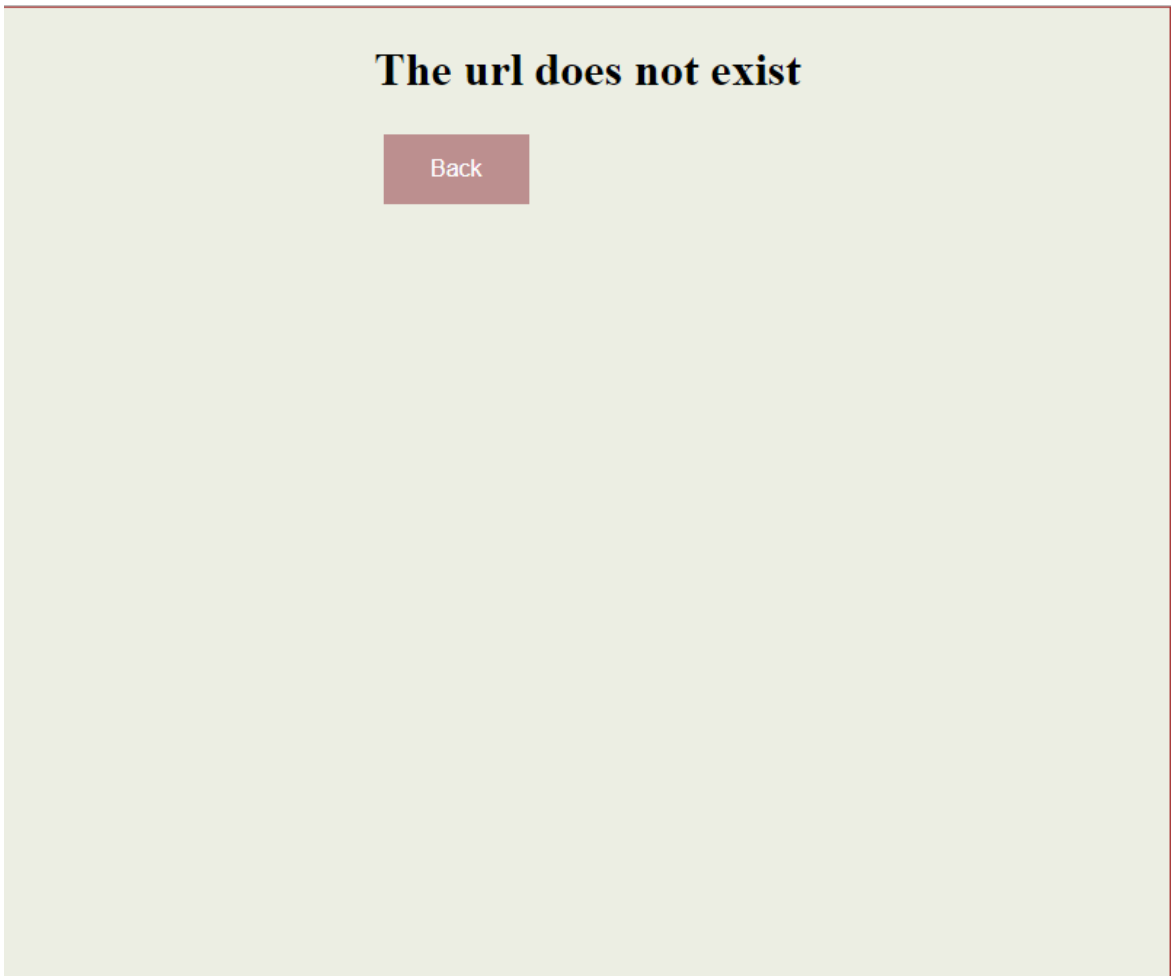
Σχημα 1

Σε περίπτωση όπου το url είναι ανύπαρκτο:



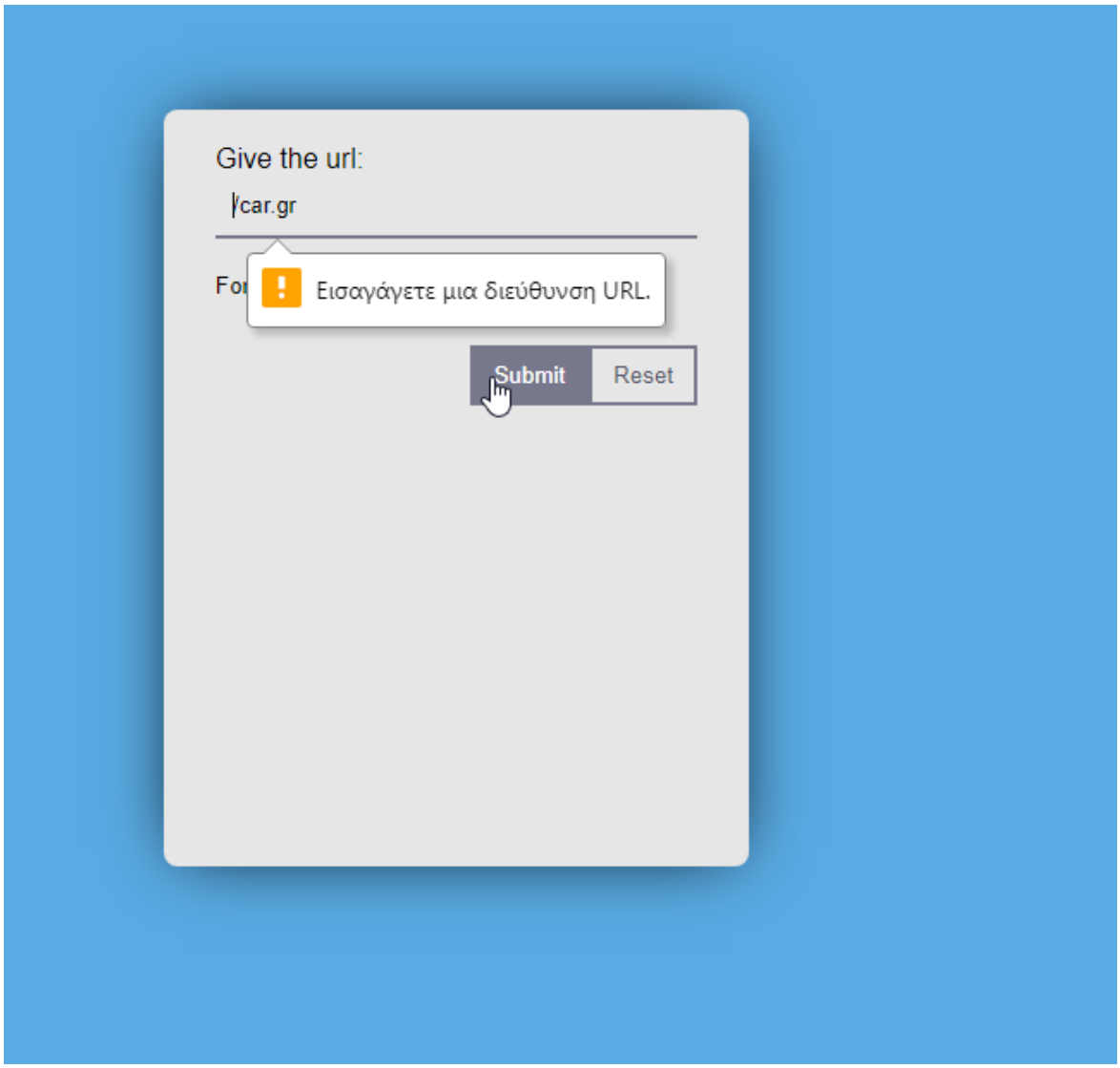
Σχήμα 2

Εμφανίζει την σελίδα και επιστρέφει στην αρχική με το button “back”:



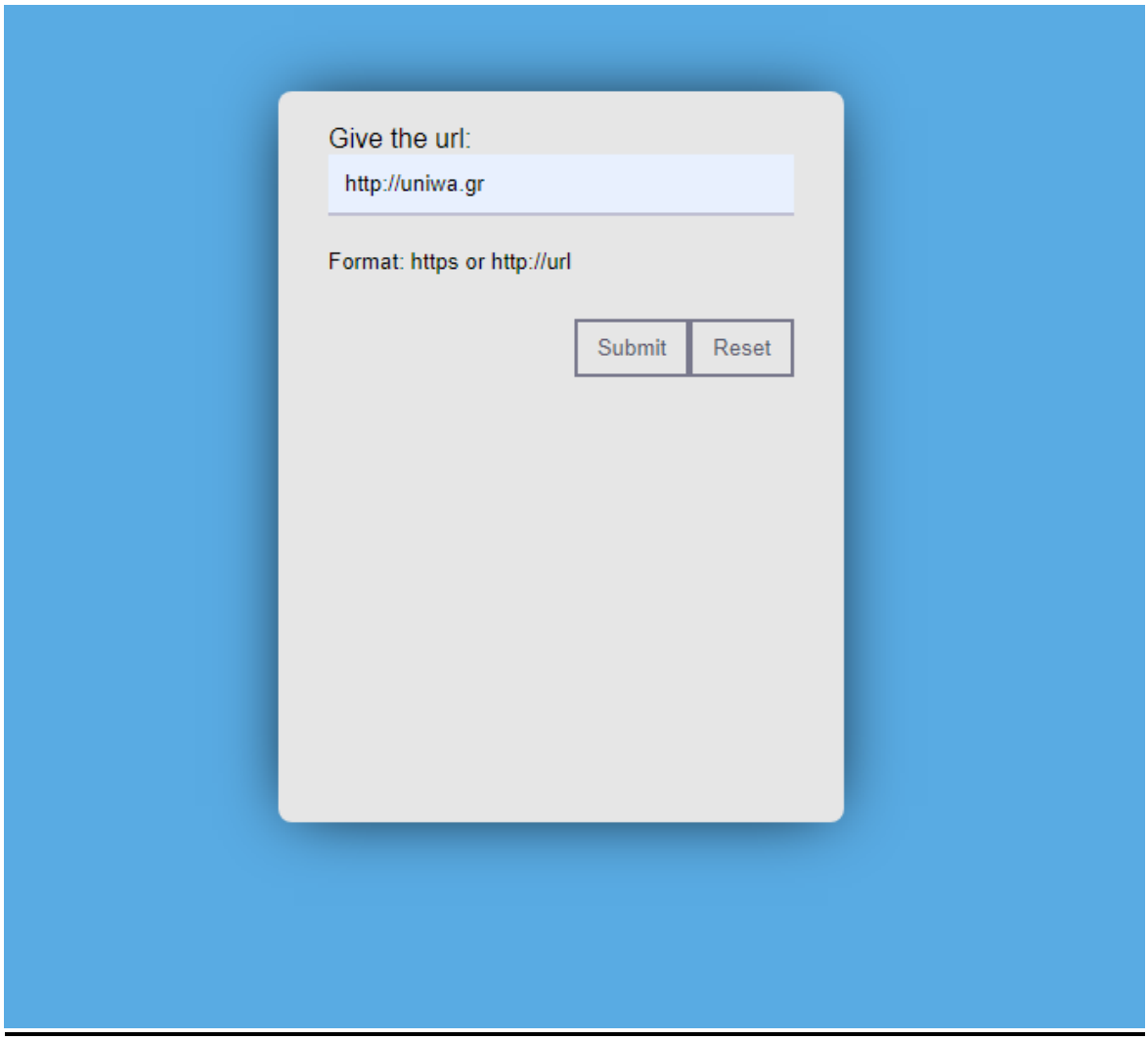
Σχήμα 3

Σε περίπτωση που το url δεν έχει έγκυρη μορφή:



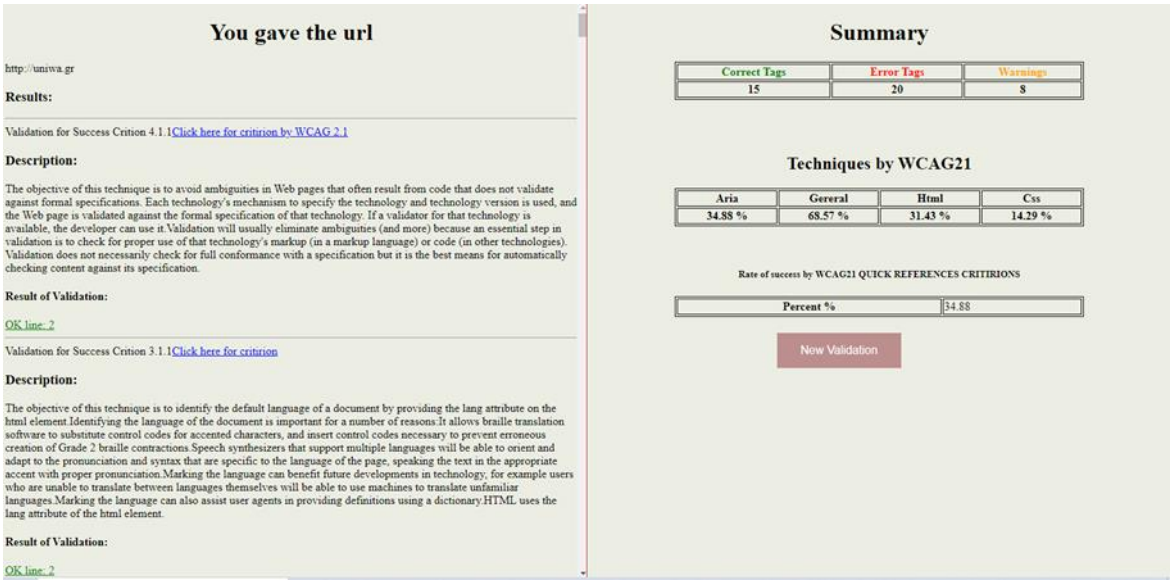
Σχήμα 4

Εισάγουμε ένα έγκυρο url πχ : https://uniwa.gr.



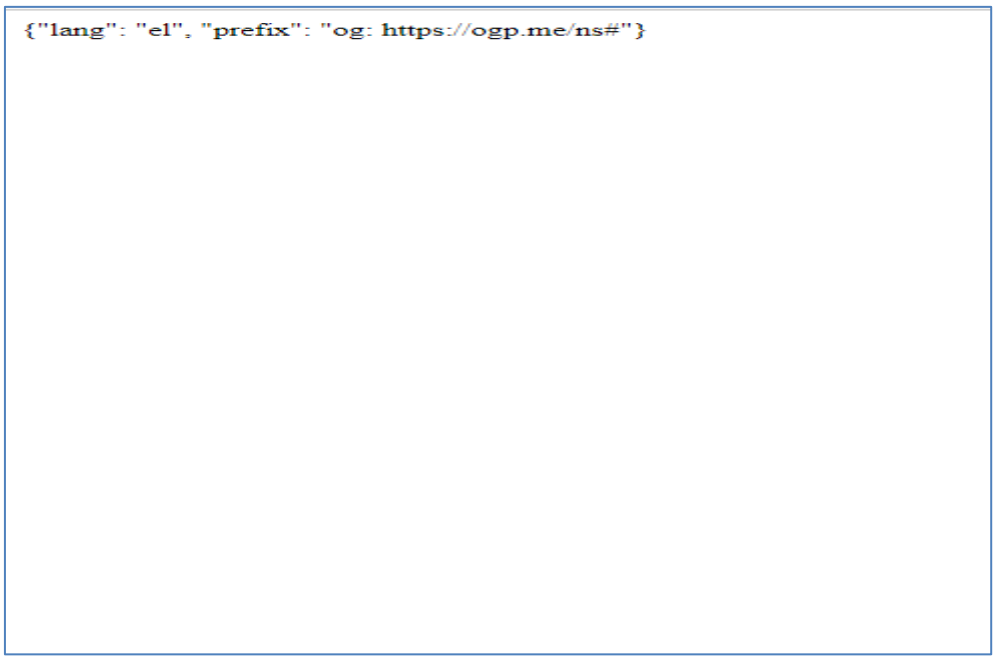
Σχήμα 5

Πατώντας «submit» εμφανίζει τα αποτελέσματα:



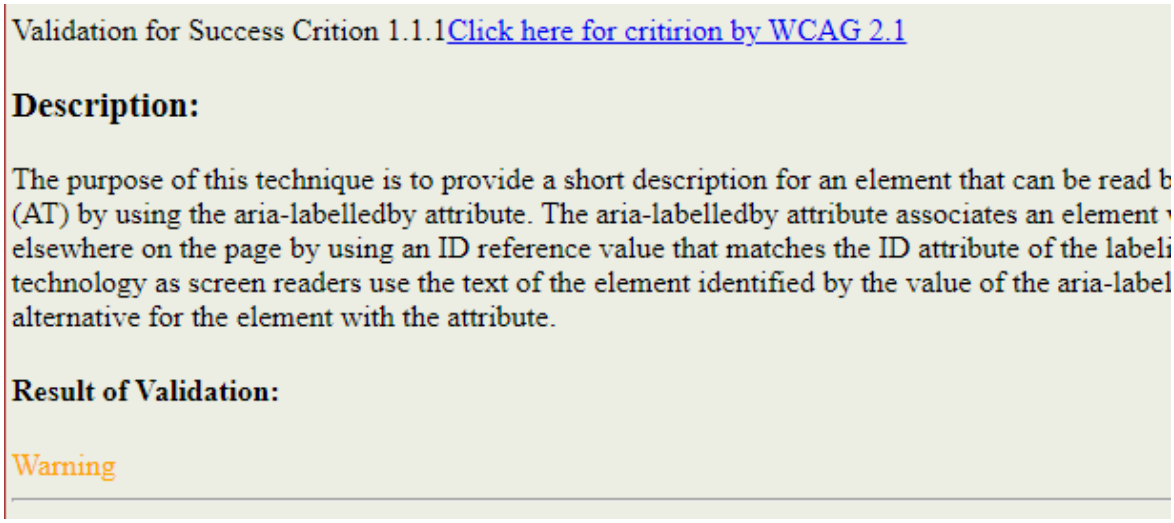
Σχήμα 6

Στο αριστερό μέρος εμφανίζεται το κάθε κριτήριο με το οποίο ελέγχεται η σελίδα. Με μπλε χρώμα εμφανίζεται το url που εξηγεί αναλυτικά το κρίτηριο από την επίσημη σελίδα του WCAG 2.1. Εν συνεχεία υπάρχει μια γρήγορη περιγραφή για το τι ακριβώς εξετάζει το κριτήριο. Αμέσως μετά έχουμε το αποτέλεσματα του ελεγχου. Εάν το κριτήριο υπάρχει στην εξεταζόμενη σελίδα έμφανίζεται με πράσινα γράμματα «οκ» και τον αριθμό γραμμής στον κώδικα της εξεταζόμενης σελίδας με link που πατώντας οδηγεί:



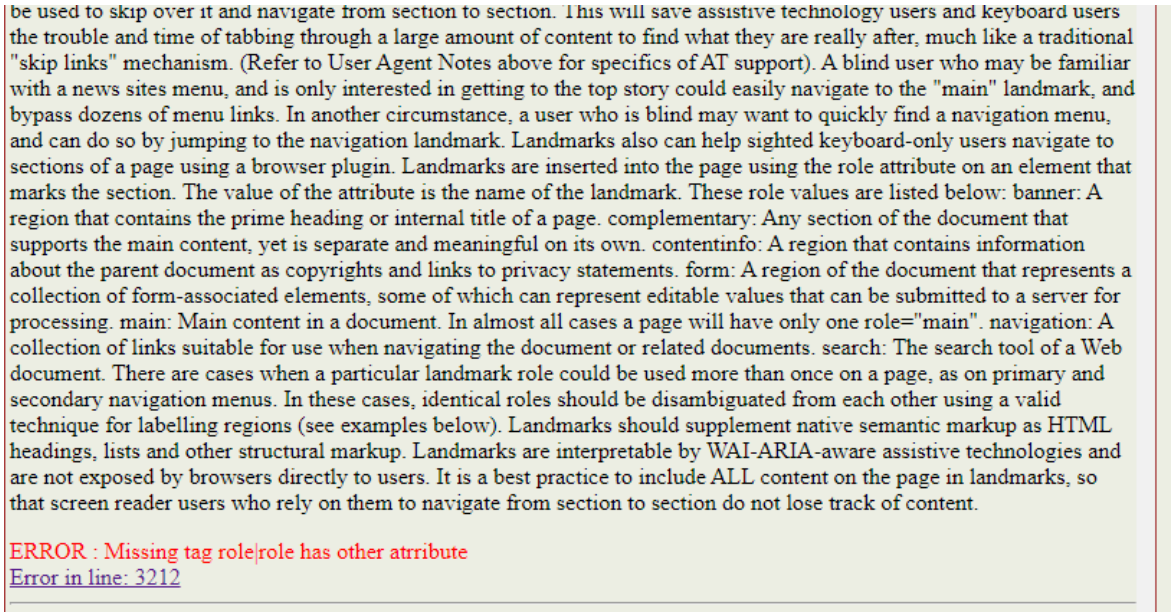
Σχήμα 7

Σε περίτωση που το κρίτηριο δεν υπάρχει καθόλου στην προς εξέταση σελίδα εμφανίζεται «warning» με πορτοκαλί χρώμα



Σχήμα 8

Σε περίπτωση που ο έλεγχος βρίσκει κρίτηριο με λάθος κατά το πρότυπο WCAG 2.1 εμφανίζει συνοπτικά το κάθος με κόκκινο χρώμα και ακριβώς από κάτω τον αριθμό της σελίδας που υπάρχει το λάθος και πατώντας το σύνδεσμο εμφανίζει το λάθος στον κωδικά της σελίδας.



Σχήμα 9

{"class": ["cli-modal"], "data-nosnippet": "true", "id": "cliSettingsPopup", "tabindex": "-1", "role": "dialog", "aria-labelledby": "cliSettingsPopup", "aria-hidden": "true"}

Σχήμα 10

ΚΕΦΑΛΑΙΟ 5

Συμπεράσματα - Προτάσεις

Στην εργασία αυτή έγινε προσπάθεια καταγραφής των δυσκολιών που αντιμετωπίζουν κατά την πρόσβαση τους στο διαδίκτυο τα άτομα με αναπηρίες. Αναφέρθηκε επίσης τι είναι η προσβασιμότητα στο διαδίκτυο, γιατί είναι σημαντική καθως και ποιες τεχνολογίες χρησιμοποιούνται για την διάδοση της πληροφορίας. Αρχικά καταγράφηκαν τα πρότυπα WCAG,ATAG και UAAG με τις αντίστοιχες εκδόσεις τους, που χρησιμοποιούνται για την ευκολία στην πλοήγηση και προσβασιμότητητα στην πληροφορία του παγκοσμίου ιστού.

Εν συνεχεία έγινε σύγκριση και παρουσίαση δύο αυτοματοποιημένων εργαλείων. Αξιολογήθηκαν τα εργαλεία MAUVE++, WAVE με τις εκδόσεις έως και Ιανουαρίου 2020 , στους τομείς εγκυρότητας και πληρότητας με το MAUVE++ να αποδεικνύεται πιο επαρκής.

Έπειτα έγινε προσπάθεια ανάπτυξης ενός παρομοίου εργαλείου βασισμένο στο πρότυπο WCAG 2.1 QCICK REFERENCES. Η ανάπτυξη του εν λόγω εργαλείου ολοκληρώθηκε για ανάλυση οθόνης από επιτραπέζιο υπολογιστή. Προτείνεται για μελλοντική εργασία η βελτίωση της έκδοσης και για χρήση από οθόνη κινητων τηλεφώνων. Ακόμη συνίσταται για μελλοντική εργασία η βελτίωση του εργαλείου ως προς την εμφάνιση των επιμέρους και συγκεντρωτικών αποτελεσμάτων και σε μορφή pdf.

Το εργαλείο διατίθεται ελεύθερο για χρήση στην διεύθυνση https://13.40.27.213

ΠΑΡΑΡΤΗΜΑ Α’

Στο παράρτημα αυτό παρατίθεται ο κώδικας ανάπτυξης της παρούσας εφαρμογής.

Αρχείο html :

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link type="text/css" rel="stylesheet" href="css2.css">

    <script type="text/javascript" src="script.js"></script>

    <title>Homepage</title>

</head>

<body>

    <form action = "http://localhost/codeMcs/submit_v1.py"method = "POST"
class ="form">

<labe for ="homepage">Give the url: </labe><br />

<input type = "url" name = "name" required  placeholder="https:// or http://"
pattern="https?://(+[0-9]|[A-Za-z]|[0-9])+[0-9]+.(gr|com|edu)"
autocomplete="on"><br><br>

<small>Format: https or http://url</H1></small><br><br>

<button type="reset">Reset</button>

<button type = "submit" >Submit </button>

</form>


</body>

</html>
```

Αρχείο python

```
# Libraries Declaration
import cgi
import cgitb
from copy import error
from ctypes.wintypes import PINT
from logging import exception
from pickle import TRUE
from queue import Empty
from urllib import response
from urllib.error import HTTPError
import bs4
from bs4 import SoupStrainer
from bs4 import BeautifulSoup, element
import sys , os , csv
cgitb.enable()
import re
import json
import urllib.request
import requests
import cssutils




print ("Content-type:text/html\r\n\r\n")
print ('<html>')
print ('<head>')
print('<meta charset="UTF-8">')
print('<meta http-equiv="X-UA-Compatible" content="IE=edge">')
print('<meta name="viewport" content="width=device-width, initial-scale=1.0">')
print ('<title> WCAG 2.1 QUICK REFERENCE CRITIRIONS </title>')
print('<meta name="viewport" content="width=device-width, initial-scale=1">')
print('<link type="text/css" rel="stylesheet" href="css1.css" />')
print('<script src="script.js">''</script>')
print ('</head>')
print ('<body>')

form = cgi.FieldStorage()
name= form.getvalue("name")

try:
        print('<div class="split left">')
```

```
        result = requests.get(name).text
        doc = BeautifulSoup(result, "html.parser")
        print ("<h1> You gave the url  </h1>" ,name)
        print("<h3>Results: </h3>")
        print("<hr>")

        # Counters Declaration
        #Declare three counters: A) for correct tag B) for error in tag and C)
for Warning

        counter_A=0
        counter_B=0
        counter_C=0
        counter1_A=0
        counter1_B=0
        counter1_C=0
        counter2_A=0
        counter2_B=0
        counter2_C=0
        counter3_A=0
        counter3_B=0
        counter3_C=0
        counter4_A=0
        counter4_B=0
        counter4_C=0
        counter5_A=0
        counter5_B=0
        counter5_C=0
        counter6_A=0
        counter6_B=0
        counter6_C=0
        counter7_A=0
        counter7_B=0
        counter7_C=0
        counter8_A=0
        counter8_B=0
        counter8_C=0
        counter9_A=0
        counter9_B=0
        counter9_C=0
        counter10_A=0
        counter10_B=0
        counter10_C=0
        counter11_A=0
        counter11_B=0
```

```
counter11_C=0
counter12_A=0
counter12_B=0
counter12_C=0
counter13_A=0
counter13_B=0
counter13_C=0
counter14_A=0
counter14_B=0
counter14_C=0
counter15_A=0
counter15_B=0
counter15_C=0
counter16_A=0
counter16_B=0
counter16_C=0
counter17_C=0
counter17_A=0
counter17_B=0
counter18_C=0
counter18_A=0
counter18_B=0
counter19_C=0
counter19_A=0
counter19_B=0
counter20_C=0
counter20_A=0
counter20_B=0
counter21_C=0
counter21_A=0
counter21_B=0
counter22_C=0
counter22_A=0
counter22_B=0
counter23_C=0
counter23_A=0
counter23_B=0
counter24_A=0
counter24_B=0
counter24_C=0
counter25_A=0
counter25_B=0
counter25_C=0
counter26_A=0
counter26_B=0
```

```
counter26_C=0
counter27_A=0
counter27_B=0
counter27_C=0
counter28_A=0
counter28_B=0
counter28_C=0
counter29_A=0
counter29_B=0
counter29_C=0
counter30_A=0
counter30_B=0
counter30_C=0
counter31_A=0
counter31_B=0
counter31_C=0
counter32_A=0
counter32_B=0
counter32_C=0
counter33_A=0
counter33_B=0
counter33_C=0
counter34_A=0
counter34_B=0
counter34_C=0
counter35_A=0
counter35_B=0
counter35_C=0
counter36_A=0
counter36_B=0
counter36_C=0
counter37_A=0
counter37_B=0
counter37_C=0
counter38_A=0
counter38_B=0
counter38_C=0
counter39_A=0
counter39_B=0
counter39_C=0
counter40_A=0
counter40_B=0
counter40_C=0
counter41_A=0
counter41_B=0
```

```
counter41_C=0
counter42_A=0
counter42_B=0
counter42_C=0
counter43_A=0
counter43_B=0
counter43_C=0
counter44_A=0
counter44_B=0
counter44_C=0
counter45_B=0
counter45_C=0
counter46_C=0
counter47_C=0
General_134 =0
General_159 =0
General_96=0
General_171 =0
General_174 =0
General_4 =0
General_105 =0
General_88 =0
General_131 =0
General_141 =0
GeneralA_141 =0
GeneralB_141 =0
GeneralC_141 =0
GeneralD_141 =0
GeneralE_141 =0
General_213 =0
General_107 =0
General_80 =0
General_197 =0
General_98 =0
General_57 =0
General_199 =0
General_91 =0
General_14 =0
General_17 =0
General_195 =0
General_215 =0
General_71 =0
html_91 =0
htmlA_91 =0
htmlB_91 =0
```

```
        htmlC_91 =0
        htmlD_91 =0
        htmlE_91 =0
        htmlF_91 =0
        htmlH_91 =0
        htmlJ_91 =0
        htmlG_91 =0
        html_4 =0
        htmlA_4=0
        html_62=0
        html_58 =0
        html_62=0
        html_57=0
        htmlA_58 =0
        Aria_8 =0
        Aria_21=0
        Aria_18=0
        Aria_10=0
        Aria_11=0
        Css_22 =0
        Css_32 =0
        Css_8=0
        Css_39=0
        Css_15 =0
        z=0

        # html page validation
        print("Validation for Success Crition 4.1.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/General/G134.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        version=doc.find_all( "html")
        print('<h3>'"Description:"'</h3>')
        print("The objective of this technique is to avoid ambiguities in Web
pages that often result from code that does not validate against formal
specifications. Each technology's mechanism to specify the technology and
technology version is used, and the Web page is validated against the formal
specification of that technology. If a validator for that technology is
available, the developer can use it.Validation will usually eliminate
ambiguities (and more) because an essential step in validation is to check for
proper use of that technology's markup (in a markup language) or code (in other
technologies). Validation does not necessarily check for full conformance with a
specification but it is the best means for automatically checking content
against its specification.","</br>")
        for tag1 in version:
           if doc.find_all("html"):
```

```
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error1.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error1.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter_A +=1
                General_134 +=1

           else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag html",'/br')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error1.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error1.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter_B +=1
                General_134 +=1
           break

        else:

            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning""</br>")
            print('</div>')
            counter1_C +=1
        print("<hr>")

        # html languague validation
```

```
        print("Validation for Success Crition 3.1.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H57.html">Click here for
critirion</a>','</br>')
        imgs_tag = doc.find_all("html" ,{"lang":True})
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print("The objective of this technique is to identify the default
language of a document by providing the lang attribute on the html
element.Identifying the language of the document is important for a number of
reasons:It allows braille translation software to substitute control codes for
accented characters, and insert control codes necessary to prevent erroneous
creation of Grade 2 braille contractions.Speech synthesizers that support
multiple languages will be able to orient and adapt to the pronunciation and
syntax that are specific to the language of the page, speaking the text in the
appropriate accent with proper pronunciation.Marking the language can benefit
future developments in technology, for example users who are unable to translate
between languages themselves will be able to use machines to translate
unfamiliar languages.Marking the language can also assist user agents in
providing definitions using a dictionary.HTML uses the lang attribute of the
html element.")
        print('</p>')
        for tag1 in imgs_tag:
            # Declare validation of tag
            if  doc.find_all("html" ,{"lang":True}):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error1.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error1.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                html_57 +=1
                counter1_A +=1

            # else inside if declares error in tag
            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing Tag Lang","</br>")
                print('</div>')
```

```
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error2.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error2.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break

                counter1_B +=1
                html_57 +=1
            break
        # else in for loop, declares Warning
        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter3_C +=1
        print("<hr>")

        # WGAG 2.1 / 1.Perceivable/ 1.1/Text Alternatives / Crtition 1.1.1
        print("Validation for Success Crition 1.1.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/Aria/Aria10.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        imgs_tag0 = doc.find_all("div",{'role':'img'})
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print("The purpose of this technique is to provide a short description
for an element that can be read by assistive technologies (AT) by using the
aria-labelledby attribute. The aria-labelledby attribute associates an element
with text that is visible elsewhere on the page by using an ID reference value
that matches the ID attribute of the labeling element. Assistive technology as
screen readers use the text of the element identified by the value of the aria-
labelledby attribute as the text alternative for the element with the
attribute.","</br>")
        print('</p>')
        for tag1 in imgs_tag0:
            if doc.find_all("div",{'role':'img','aria-labelledby':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id ="ok">')
```

```
                print('<a href = error3.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error3.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                Aria_10 +=1
                counter42_A +=1

            elif doc.find_all("div",{'role':'img'}):
                print('<div id="error">')
                print("ERROR : Missing tag aria-labellebdy","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error3.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error3.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter44_B +=1
                Aria_10 +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag role","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error3.html#error3>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error3.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter44_B +=1
                Aria_10 +=1
                break
        else:
```

```
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter4_C +=1
        print("<hr>")


        # WGAG 2.1 / 1.3 Adaptable // 1.3.1  Info and relatioships
        print("Validation for Success Crition 1.3.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/Aria/Aria11.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print("The purpose of this technique is to provide programmatic access
to sections of a web page. Landmark roles programmatically identify sections of
a page. Landmarks help assistive technology (AT) users orient themselves to a
page and help them navigate easily to various sections of a page.")
        print("They also provide an easy way for users of assistive technology
to skip over blocks of content that are repeated on multiple pages and notify
them of programmatic structure of a page. For instance, if there is a common
navigation menu found on every page, ")
        print('landmark roles (or "landmarks") can be used to skip over it and
navigate from section to section.')
        print('This will save assistive technology users and keyboard users the
trouble and time of tabbing through a large amount of content to find what they
are really after, much like a traditional "skip links" mechanism. (Refer to User
Agent Notes above for specifics of AT support).')
        print('A blind user who may be familiar with a news sites menu, and is
only interested in getting to the top story could easily navigate to the "main"
landmark, and bypass dozens of menu links. In another circumstance, a user who
is blind may want to quickly find a navigation menu, and can do so by jumping to
the navigation landmark.')
        print('Landmarks also can help sighted keyboard-only users navigate to
sections of a page using a browser plugin.')
        print('Landmarks are inserted into the page using the role attribute on
an element that marks the section. The value of the attribute is the name of the
landmark. These role values are listed below:')
        print('banner: A region that contains the prime heading or internal
title of a page.')
        print('complementary: Any section of the document that supports the main
content, yet is separate and meaningful on its own.')
        print('contentinfo: A region that contains information about the parent
document as copyrights and links to privacy statements.')
```

```
        print('form: A region of the document that represents a collection of
form-associated elements, some of which can represent editable values that can
be submitted to a server for processing.')
        print('main: Main content in a document. In almost all cases a page will
have only one role="main".')
        print('navigation: A collection of links suitable for use when
navigating the document or related documents.')
        print('search: The search tool of a Web document.')
        print('There are cases when a particular landmark role could be used
more than once on a page, as on primary and secondary navigation menus. In these
cases, identical roles should be disambiguated from each other using a valid
technique for labelling regions (see examples below).')
        print('Landmarks should supplement native semantic markup as HTML
headings, lists and other structural markup. Landmarks are interpretable by WAI-
ARIA-aware assistive technologies and are not exposed by browsers directly to
users.')
        print('It is a best practice to include ALL content on the page in
landmarks, so that screen reader users who rely on them to navigate from section
to section do not lose track of content.','</br>')
        print('</p>')
        imgs_tag5 = doc.find_all("div",{"role":True})
        for tag1 in imgs_tag5 :
            if (doc.find_all("div",{"role":'banner'})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter2_A +=1
                Aria_11 +=1
            elif(doc.find("div",{"role":'search'})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter2_A +=1
                Aria_11 +=1
            elif(doc.find_all("div",{"role":'navigaton',"aria-
labelledby":True})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                    counter2_A +=1
                    Aria_11 +=1
            elif(doc.find_all("div",{"role":'navigaton',"aria-label":True})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                    counter2_A +=1
                    Aria_11 +=1
            elif(doc.find("div",{"role":'main'})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
```

```
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                    counter2_A +=1
                    Aria_11 +=1
            elif(doc.find_all("div",{"role":'complementary'})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                    counter2_A +=1
            elif(doc.find_all("div",{"role":'contentinfo'})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                    counter2_A +=1
                    Aria_11 +=1
            elif(doc.find_all("div",{"role":'form'})):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error4.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                    counter2_A +=1
                    Aria_11 +=1

            else:
                    print('<div id="error">')
                    print("ERROR : Missing tag role|role has other
atrribute","</br>")
                    print('</div>')
                    x=(tag1.attrs)
                    y1=json.dumps(x)
                    print('<a href = error4.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                    with open('submit_v1.py','r') as firstfile ,
open('error4.html','w') as secondfile:
                         for line in firstfile:
                             secondfile.writelines(y1)
                             break
                    counter2_B +=1
                    Aria_11 +=1
            break
        else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="warning">')
                print("Warning","</br>")
                print('</div>')
                counter5_C +=1

        print("<hr>")

        # WGAG 2.1 / 1.2 Perceivable // 1.2 Time-based Media
        print("Validation for Success Crition 1.2"'<a
href="http://w3.org/WAI/WCAG21/Techniques/General/G159.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The purpose of this technique is to provide an accessible
alternative way of presenting the information in a video-only presentation.')
        print('In a video-only presentation, information is presented in a
variety of ways including animation, text or graphics,')
```

```
        print('the setting and background, the actions and expressions of
people, animals, etc. In order to present the same')
        print('information in accessible form, this technique involves creating
a document that tells the same story and presents the ')
        print('same information as the prerecorded video-only content. In this
technique, the document serves as a long description ')
        print('for the content and includes all of the important information as
well as descriptions of scenery, actions, expressions,')
        print('etc. that are part of the presentation.')
        print('If a screenplay for the video-only content was used to create the
content in the first place, this can be a good place to start. In production and
editing however, ')
        print('the final version often varies somewhat from the screenplay. To
use the screenplay, it would need to be corrected to match the final edited form
of the video-only presentation.')
        print('</p>')
        imgs_tag1 = doc.find_all("video",{"width":'"800" controls'})
        imgs_tag2
=doc.find_all("source",{"src":'command.mp4',"type":'video/mp4'})
        imgs_tag3 =
doc.find_all("track",{"src":'english.vtt',"kind":'subtitles',"scrlang":'en',"lab
el":'"English" default'})
        imgs_tag4 =
doc.find_all("track",{"src":'greek.vtt',"kind":'subtitles',"scrlang":'gr',"label
":'Greek'})
        for tag1 in doc.find_all("video",{"width":'"800" controls'}):
            if (doc.find_all("video",{"width":'"800" controls'})):
                for tag1 in
doc.find_all("source",{"src":'command.mp4',"type":'video/mp4'}):
                    if
(doc.find_all("source",{"src":'command.mp4',"type":'video/mp4'})):
                        for tag1 in
doc.find_all("track",{"src":'english.vtt',"kind":'subtitles',"scrlang":'en',"lab
el":'"English" default'}):
                            if
doc.find_all("track",{"src":'english.vtt',"kind":'subtitles',"scrlang":'en',"lab
el":'"English" default'}):
                                for tag1 in
doc.find_all("track",{"src":'greek.vtt',"kind":'subtitles',"scrlang":'gr',"label
":'Greek'}):
                                    if
doc.find_all("track",{"src":'greek.vtt',"kind":'subtitles',"scrlang":'gr',"label
":'Greek'}):
                                        print('<h4>'"Result of
Validation:"'</h4>')
```

```
                                    x=(tag1.attrs)
                                    y1=json.dumps(x)
                                    print('<div id = "ok">')
                                    print('<a href = error5.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                                    print('</div>')
                                    with open('submit_v1.py','r') as
firstfile , open('error5.html','w') as secondfile:
                                        for line in firstfile:
                                            secondfile.writelines(y1)
                                            break
                                        counter3_A +=1
                                        General_159 +=1
                                else:
                                    print('<h4>'"Result of
Validation:"'</h4>')
                                    print('<div id="error">')
                                    print("ERROR : Missing tag scr",'</br')
                                    print('</div>')
                                    x=(tag1.attrs)
                                    y1=json.dumps(x)
                                    print('<a href = error5.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                                    with open('submit_v1.py','r') as
firstfile , open('error5.html','w') as secondfile:
                                            for line in firstfile:
                                                secondfile.writelines(y1)
                                                break
                                    counter3_B +=1
                                    General_159 +=1
                                break
                            else:
                                print('<h4>'"Result of Validation:"'</h4>')
                                print('<div id="warning">')
                                print("Warning","</br>")
                                print('</div>')
                                counter6_C +=1

                        else:
                                print('<h4>'"Result of Validation:"'</h4>')
                                print('<div id="error">')
                                print("ERROR Missing tag track","</br>")
                                print('</div>')
                                x=(tag1.attrs)
                                y1=json.dumps(x)
```

```
                                    print('<a href = error5.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                                    with open('submit_v1.py','r') as firstfile ,
open('error5.html','w') as secondfile:
                                        for line in firstfile:
                                            secondfile.writelines(y1)
                                            break
                                    counter4_B +=1
                                    General_159 +=1
                                    break
                            else:
                                    print('<h4>'"Result of Validation:"'</h4>')
                                    print('<div id="warning">')
                                    print("Warning","</br>")
                                    print('</div>')
                                    counter7_C +=1

                        else:
                                    print('<h4>'"Result of Validation:"'</h4>')
                                    print('<div id="error">')
                                    print("ERROR Missing tag source","</br>")
                                    print('</div>')
                                    x=(tag1.attrs)
                                    y1=json.dumps(x)
                                    print('<a href = error5.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                                    with open('submit_v1.py','r') as firstfile ,
open('error5.html','w') as secondfile:
                                         for line in firstfile:
                                            secondfile.writelines(y1)
                                            break
                                    counter5_B +=1
                                    General_159 +=1
                                    break
                    else:
                                    print('<h4>'"Result of Validation:"'</h4>')
                                    print('<div id="warning">')
                                    print("Warning","</br>")
                                    print('</div>')
                                    counter8_C +=1

                else:
                                    print('<h4>'"Result of Validation:"'</h4>')
                                    print('<div id="error">')
                                    print("ERROR Missing tag video","</br>")
```

```
                                    print('</div>')
                                    x=(tag1.attrs)
                                    y1=json.dumps(x)
                                    print('<a href = error5.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                                    with open('submit_v1.py','r') as firstfile ,
open('error5.html','w') as secondfile:
                                          for line in firstfile:
                                            secondfile.writelines(y1)
                                            break
                                    General_159 +=1
                                    counter6_B +=1
                                    break
        else:
                                    print('<h4>'"Result of Validation:"'</h4>')
                                    print('<div id="warning">')
                                    print("Warning","</br>")
                                    print('</div>')
                                    counter9_C +=1

        print("<hr>")
        #WGAG 2.1/ 1.3 Adaptable // Not right CSS use// Ctiriria:
1.3.2,1.4.1,1.4.5,1.4.9,1,4,10,1.4.11,1.4.12,2.3.3,2.5.4
        print("Validation for Success Critirions:
1.3.2,1.4.1,1.4.5,1.4.9,1,4,10,1.4.11,1.4.12,2.3.3,2.5.4"'<a href=#
onclick="urls2()">Click here for critirions by WCAG 2.1''</a>','</br>')
        print("Critirion : 1.3.2")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that the order of
content presented to assistive technologies allows the user to make sense of the
content. Some techniques permit the content to be rendered visually in a
meaningful sequence ')
        print('even if this is different from the order in which the content is
encoded in the underlying source file.')
        print('For example, when mixing languages with different directionality
in HTML, the bidirectional algorithm may position punctuation in the wrong
location in the visual rendering. The visual rendering problem could be
corrected')
        print('by moving the punctuation in the content stream so that the
bidirectional algorithm positions it as desired,')
        print('but this would expose the incorrect content order to assistive
technology. The content is both rendered in ')
        print('the correct order visually and exposed to assistive technology in
the correct order by using markup to override the bidirectional algorithm.')
```

```
        print('When rendered visually, white space characters as space or tab
may not appear to be part of the content. However, when inserted into the
content to')
        print('control visual formatting, they may interfere with the meaning of
the content.')
        print('At a larger granularity, controlling the placement of blocks of
content in an HTML document using layout tables may produce a rendering in which
related information is positioned together visually, but separated in the
content stream. Since layout tables are read row by row, if the caption of an
illustration is placed in the row following the illustration, it may be
impossible to associate the caption with the image.',"</br>")
        print('</p>')
        print("-----------------------------------------------------------------
","</br>")
        print("Critirion : 1.4.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that when color
differences are used to convey information, as required form fields,')
        print('the information conveyed by the color differences are also
conveyed explicitly in text.','</br>')
        print('</p>')
        print("-----------------------------------------------------------------
","</br>")
        print("Critirion : 1.4.5 , 1.4.9")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to demonstrate how CSS can be
used to control the visual presentation of text.')
        print('This will allow users to modify, via the user agent, the visual
characteristics of the text to meet their requirement.')
        print('The text characteristics include aspects as size, color, font
family and relative placement.')
        print('CSS benefits accessibility primarily by separating document
structure from presentation. Style sheets were designed')
        print('to allow precise control - outside of markup - of character
spacing, text alignment, object position on the page, audio and speech output,
font characteristics, etc. By separating ')
        print('style from markup, authors can simplify and clean up the markup
in their content, making it more accessible at the same time.')
        print('Text within images has several accessibility problems, including
the inability to:','</br>')
        print('</p>')
        print('<ul>')
        print('<li>')
```

```
        print('be scaled according to settings in the browser')
        print('</li>')
        print('<li>')
        print('be displayed in colors specified by settings in the browser or
rules in user-defined style sheets')
        print('</li>')
        print('<li>')
        print('honor operating system settings, as high contrast')
        print('</li>')
        print("</ul>")
        print('<p>')
        print('It is better to use real text for the text portion of these
elements, and a combination of semantic markup and style sheets to create the
appropriate visual presentation. For this to work effectively, choose fonts that
are likely to be available on the user system and define fallback fonts for
users who may not have the first font that is specified. Newer machines and user
agents often smooth or anti-alias all text, so it is likely that your headings
and buttons will look nice on these systems without resorting to images of
text.')
        print('The following CSS properties are useful to style text and avoid
the need for text in images:','</br>')
        print('</p>')
        print('<ul>')
        print('<li>')
        print('The font-family property is used to display the code aspect in a
monospace font family.')
        print('</li>')
        print('<li>')
        print('The text-align property is used to display the text to the right
of the viewport.')
        print('</li>')
        print('<li>')
        print('The font-size property is used to display the text in a larger
size.')
        print('</li>')
        print('<li>')
        print('The font-style property is used to display text in italics.')
        print('</li>')
        print('<li>')
        print('The font-weight property is used to set how thick or thin
characters in text should be displayed.')
        print('</li>')
        print('<li>')
        print('The color property is used to display the color of text or text
containers.')
```

```
        print('</li>')
        print('<li>')
        print('The line-height property is used to display the line height for a
block of text.')
        print('</li>')
        print('<li>')
        print('The text-transform property is used to control the case of
letters in text.')
        print('</li>')
        print('<li>')
        print('The letter-spacing property is used to control the spacing of
letters in text.')
        print('</li>')
        print('<li>')
        print('The background-image property can be used to display text on a
non-text background.')
        print('</li>')
        print('<li>')
        print('The first-line pseudo class can be used to modify the
presentation of the first line in a block of text.')
        print('</li>')
        print('<li>')
        print('The :first-letter pseudo class can be used to modify the
presentation of the first letter in a block of text.')
        print('</li>')
        print('<li>')
        print('The :before and :after pseudo classes can be used to insert
decorative non-text content before or after blocks of text.')
        print('</li>')
        print('</ul>','</br>')
        print("----------------------------------------------------------------
","</br>")
        print("Critirion : 1.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to be able to present content
without introducing a horizontal scroll bar at a width ')
        print('equivalent to 320 CSS pixels, or a vertical scroll bar at a
height equivalent to 256 CSS pixels for text intended to scroll horizontally.')
        print('This is done by using layout techniques that adapt to the
available viewport space.')
        print('Grid layouts define layout regions that reflow as needed to
display the region on the screen. Although the exact')
        print('layout therefore varies, the relationship of elements and the
reading order remains the same when done right. ')
```

```
        print('This is an effective way to create designs that present well on
different devices and for users with different content-size preferences.')
        print('</p>')
        print('The basic principles of grid layouts are to:')
        print('<ol>')
        print('<li>')
        print('Define the size of layout regions using grid properties and media
queries for specific viewport sizes,')
        print('so they enlarge, shrink or wrap in the available space and
respond to zoom levels;')
        print('</li>')
        print('<li>')
        print('Position the layout regions in the grid container as a row of
adjacent grid items, which may wrap to new rows as')
        print('needed in much the same way as words in a paragraph wrap.')
        print('</li>')
        print('</ol>')
        print("-----------------------------------------------------------------
","</br>")
        print("Critirion : 1.4.11")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is enhance the focus indicator in
the browser, by creating a highly visible one in')
        print('the content. The default focus indicator in many browsers is a
thin, dotted, black line. It can be difficult to see')
        print('the line when it is around a form element which already has an
outline, when the focused element is inside a table')
        print('cell, when the focused element is very small, or when the
background of the page is a dark color. ')
        print('Some browsers use a pale blue outline, which can be difficult to
see on some backgrounds.')
        print('In this technique, when the user places focus on an element,
using the mouse, tab key, arrow keys, keyboard shortcuts, or any other method,')
        print('the application makes that focus more visible, using a
combination of a highly contrasting color, a thick line, and other visual
indicators as a glow.')
        print('</p>')
        print("-----------------------------------------------------------------
","</br>")
        print("Critirion : 1.4.12")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
```

```
        print('The objective of this technique is to demonstrate how the visual
appearance of spacing in text may be enhanced via style sheets while still
maintaining meaningful text sequencing. ')
        print('The CSS letter-spacing property helps developers control the
amount of white space between characters. This is recommended over adding blank
characters to control the spacing,')
        print('since the blank characters can change the meaning and
pronunciation of the word.')
        print('</p>')
        print("-------------------------------------------------------------------
","</br>")
        print("Critirion : 2.3.3")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print("The objective of this technique is to allow users to prevent
animation from being displayed on Web pages, via the use of of the 'prefers-
reduced-motion' CSS Media Query.")
        print('Some users experience distraction or nausea from animated
content. For example, if scrolling a page causes elements to move (other than
the essential movement associated with scrolling')
        print('it can trigger vestibular disorders. Enclosing the CSS that
creates the animations in a media query allows people to prevent those
symptoms.')
        print("A typical example is 'parallax scrolling', where backgrounds move
at different rates. The movement due to scrolling the page is essential (and
under the users control), but additional")
        print('movement triggered by the scrolling can also trigger vestibular
symptoms.')
        print('The understanding document for Motion Actuation includes links
for changing the reduce motion setting.')
        print('</p>')
        print("-------------------------------------------------------------------
","</br>")
        print("Critirion : 2.5.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that:')
        print('</p>')
        print('<ul>')
        print('<li>')
        print('People who use mounted devices or who cannot precisely manipulate
a device are able to provide inputs through more conventional user interface
components, and')
        print('</li>')
        print('<li>')
```

```
        print('People who may accidentally activate sensors due to tremors or
other motor impairments have the ability to turn off motion actuation to prevent
accidental triggering of functions.')
        print('</li>')
        print('</ul>')
        print('<p>')
        print('When a device sensor as an accelerometer or gyroscope is used to
gather input:')
        print('</p>')
        print('<ul>')
        print('<li>')
        print('A button, link, or other conventional control is provided that
does not rely on sensor input, and')
        print('</li>')
        print('<li>')
        print('A setting is provided to allow the user to turn off the sensor
data.')
        print('</li>')
        print('</ul>')
        print('</br>')
        print('<h5>'"Summary for critirions
1.3.2,1.4.1,(1.4.5,1.4.9,1.4.12),1,4,10,1.4.11,2.3.3,2.5.4"'<h5>',"</br>")
        sheets=[] # Declares an array for  CSS buffering
        imgs_tag6 = doc.find_all("style", type="text/css")
        for tag1 in imgs_tag6:
                if (not tag1.string):
                    continue
                sheets.append(cssutils.parseStyle(tag1.string))
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error6.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error6.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                break
        else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="warning">')
                print("Warning","</br>")
```

```
                print('</div>')
                counter4_A +=1
        counter4_A1 = (counter4_A +4)
        General_57 +=1
        General_14 +=1
        General_17 +=1
        General_199 +=1
        General_215 +=1
        Css_22 +=1
        Css_32 +=1
        Css_8 +=1
        Css_39 +=1
        Css_15 +=1



        print("<hr>")

        # WGAG 2.1 / 1.3. Adaptable / crtition 1.3.3
        print("Validation for Success Crition 1.3.3"'<a
href="http://w3.org/WAI/WCAG21/Techniques/General/G96.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 1.3.3")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that items within a
Web page are referenced in the content not only by shape, size, sound or
location, but also in ways that do not depend on that sensory perception. For
example, a reference may also describe the function of the item or its label.')
        print('</p>')
        imgs_tag7 = doc.find_all('form')
        for tag1 in imgs_tag7:
            if doc.find_all("form",{'value':'submit','value':'go'}):
                print('<h4>'"Result of Validation:"'</h4>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<div id = "ok">')
                print('<a href = error7.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
```

```
                counter5_A +=1
                General_96 +=1
            elif doc.find_all("form",{'value':'submit'}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag value:submit",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter7_B +=1
                General_96 +=1

            else:

                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag value:go",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter7_B +=1
                General_96 +=1
                break
        else:

            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter8_C +=1
```

```
        print("<hr>")

        # WGAG 2.1 / 1.4 Distinguishable/ crtition 1.4.2.
        print("Validation for Success Crition 1.4.2"'<a
href="http://w3.org/WAI/WCAG21/Techniques/General/G171.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 1.4.2")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print("The intent of this technique is to allow a user to control the
use of sounds in Web content. Someone that uses a screen reader may find it very
distracting and difficult to listen to their screen reader if there are also
sounds coming from Web content. Providing a way to play sounds only upon request
will give a user the control needed to listen to any sounds or other audio
without interfering with the output from a screen reader.")
        print('</p>')
        imgs_tag9 = doc.find_all('a',attrs={'href':(".mp3$")or(".mp4$")})
        for tag1 in imgs_tag9:
            if (doc.find_all('a' ,attrs={'href':(".mp3$")or(".mp4$")})):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error8.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error8.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter6_A +=1
                General_171 +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag a ","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error8.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error8.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
```

```
                    break
                counter8_B +=1
                General_171 +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter9_C +=1
        print("<hr>")

        # WGAG 2.1 / 1.4 Distinguishable/ crtition 1.4.4.
        print("Validation for Success Crition 1.4.4"'<a
href="http://w3.org/WAI/WCAG21/Techniques/General/G174.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 1.4.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('When the contrast between the text and its background for some
portion of the page has not been designed to meet the contrast level for Success
Criterion 1.4.3 or 1.4.6, it is possible to meet these guidelines using the
"Alternate Version" clause in the conformance requirements (Conformance
Requirement 1). A link or control on the page can either change the page so that
all aspects conform, or it could take the viewer to a new version of the page
that does conform at the desired level. Placing the link or control prominently
on the page will assist users in accessing the conforming content readily.')
        print('</p>')
        imgs_tag10=doc.find_all("strong")
        for tag1 in doc.find_all("strong") :
            if doc.find_all("strong"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error9.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error9.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                General_174 +=1
                counter7_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>')
```

```
                print('<div id="error">')
                print("ERROR : Missig tag strong",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error9.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error9.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter9_B +=1
                General_174 +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter10_C +=1


        print("<hr>")

        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        print("Validation for Success Crition 2.1.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
```

```
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        imgs_tag11 = doc.find_all('input')
        for tag1 in imgs_tag11 :
            if doc.find_all('input',{'type':'image','alt':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error10.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter8_A +=1
                html_91 +=1
            elif doc.find_all('input',{'type':'image'}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag alt",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter10_B +=1
                html_91 +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag img",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter10_B +=1
                html_91 +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter11_C +=1

        print("<hr>")

        #WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 , 2.4.4 keyboard
        print("Validation for Success Crition 2.1.1 , 2.4.4"'<a href=#"
class="urls3">Click here for critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        print("Critirion : 2.4.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
```

```
        print('The objective of this technique is to describe the purpose of a
link in the text of the link. The description lets a user distinguish this link
from links in the Web page that lead to other destinations and helps the user
determine whether to follow the link. The URI of the destination is generally
not sufficiently descriptive.')
        print('</p>')
        imgs_tag12 = doc.find_all("input")
        for tag1 in imgs_tag12:
            if doc.find_all("input",{'alt':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error11.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error11.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlA_91 +=1
                General_91 +=1
                counter9_A +=1
            elif doc.find_all("input"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR Missing tag alt",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error11.html>''Error in
line:',tag1.sourceline,'</a>''<id="refresh()">','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error11.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlA_91 +=1
                General_91 +=1
                counter11_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag input",'</br>')
                print('</div>')
                print(tag1.attrs)
```

```
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error11.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error11.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlA_91 +=1
                General_91 +=1
                counter11_B +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter12_C +=1




        print("<hr>")

        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
```

```
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        imgs_tag13 = doc.find_all("input")
        for tag1 in imgs_tag13:
            if doc.find_all("input",{'title':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error2.html#error2>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error6.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter10_A +=1
                htmlB_91 +=1
            elif doc.find_all("input"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag title",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error9.html#error9>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error9.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter12_B +=1
                htmlB_91 +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag input",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
```

```
                print('<a href = error12.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error12.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter12_B +=1
                htmlB_91 +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter13_C +=1

        print("<hr>")



        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
```

```
        print('</p>')
        imgs_tag14 = doc.find_all("label")
        for tag1 in imgs_tag14:
            if doc.find_all("label",{'for':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error2.html#error2>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error6.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlC_91 =+1
                counter11_A +=1
            elif doc.find_all("label"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag for","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#error10>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter13_B +=1
                htmlC_91 =+1


            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag label","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#error10>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
```

```
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter13_B +=1
                htmlC_91 =+1

            break
        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter14_C +=1
        print("<hr>")

        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')

        imgs_tag15 = doc.find_all("input")
        for tag1 in imgs_tag15:
            if doc.find_all("input",{'id':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
```

```
                print('<a href = error13.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error13.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlD_91 =+1
                counter12_A +=1
            elif doc.find_all("input"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag id","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error13.html''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error13.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter14_B +=1
                htmlD_91 =+1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR Missing tag input","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error13.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error13.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter14_B +=1
                htmlD_91 =+1

            break
```

```
        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter15_C +=1


        print("<hr>")


        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        imgs_tag16 = doc.find_all("input")
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        for tag1 in (imgs_tag13 and imgs_tag14 and imgs_tag15 and imgs_tag16):
            if
(doc.find_all("label",{'for':True})==doc.find_all("input",{'id':True})and
doc.find_all("input",{'checked':("checked")}) is True):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error14.html>''OK
line:',tag1.sourceline,'</a>','</br>')
```

```
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error14.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter13_A +=1
                htmlE_91 =+1
            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR Missing tags: for|id|checked)","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error14.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error14.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter15_B +=1
                htmlE_91 =+1
            break

        else:

            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter16_C +=1


        print("<hr>")


        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        imgs_tag17 = doc.find_all("input")
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
```

```
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        for tag1 in imgs_tag17:
            if  doc.find_all("input",{'type':'radio','checked':'checked'}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error15.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error15.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter14_A +=1
                htmlF_91 =+1
            elif doc.find_all("input"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tags type|checked","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error15.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error15.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
```

```
                        break
                counter16_B +=1
                htmlF_91 =+1
            elif doc.find_all("input",{'type':'radio'}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tags checked","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error15.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error15.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter16_B +=1
                htmlF_91 =+1
            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR Missing tag input","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error15.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error15.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter16_B +=1
                htmlF_91 =+1
            break

        else:

            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter17_C +=1
```

```
        print("<hr>")


        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        imgs_tag18 = doc.find_all("select")
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        for tag1 in imgs_tag18:
            if  doc.find_all("select",{'title':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error16.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error16.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter15_A +=1
                htmlG_91 =+1

            elif doc.find_all("select" ):
```

```
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag title","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error16.html#error9>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error16.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter17_B +=1
                htmlG_91 =+1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag select","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error16.html#error9>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error16.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter17_B +=1
                htmlG_91 =+1

            break

        else:

            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter18_C +=1

        print("<hr>")
```

```
        # WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        imgs_tag19= doc.find_all("textarea")
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        for tag1 in imgs_tag19:
            if  doc.find_all("textarea",{'title':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error17.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error17.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlH_91 =+1
                counter16_A +=1
            elif doc.find_all("textarea"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag title","</br>")
                print('</div>')
```

```
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error17.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error17.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter18_B +=1
                htmlH_91 =+1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag textarea","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error17.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error17.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter18_B +=1
                htmlH_91 =+1

            break

        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter20_C +=1

        print("<hr>")

        #WGAG 2.1 // 2 operable/ 2.1 Keyborard Access / 2.1.1 keyboard
        imgs_tag20 =  doc.find_all("fieldset")
        print("Validation for Success Crition 2.1.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H91.html">Click here for
critirion by WCAG 2.1</a>','</br>')
```

```
        print("Critirion : 2.1.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use standard HTML form
controls and link elements to provide keyboard operation and assistive
technology interoperability of interactive user interface elements.')
        print('User agents provide the keyboard operation of HTML form controls
and links. In addition, the user agent maps the form controls and links to an
accessibility API. Assistive technologies use the accessibility API to extract
appropriate accessibility ')
        print('information, as role, name, state, and value, and present them to
users. The role is provided by the HTML element, and the name is provided by the
text associated with that element. Elements for which values and states are
appropriate also expose the values and states via multiple mechanisms.')
        print('In some cases, the text is already associated with the control
through a required attribute. For example, submit buttons use the button element
text or image "alt" attribute as the name. In the case of form controls, label
elements or "title" attributes are used. The following table describes how the
role, name, value, and state are determined for HTML links and form controls.')
        print('</p>')
        for tag1 in imgs_tag20:
            if  doc.find_all("fieldset",{'legend':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error18.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error18.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter17_A +=1
                htmlJ_91 =+1
            elif doc.find_all("fieldset"):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR Missing tag legend","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error18.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error18.html','w') as secondfile:
```

```
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter19_B +=1
                htmlJ_91 =+1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR :Missing tag fieldset","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error18.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error18.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter19_B +=1
                htmlJ_91 =+1

            break
        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter21_C +=1

        print("<hr>")

        #WGAG 2.1 / 2 Operable / 2.2 Enough time / Critirion 2.2.1,2.2.2.
        imgs_tag21 = doc.find_all('button')
        print("Validation for Success Crition 2.2.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G4.html">Click here for
critirion by WCAG 2.2.1 , 2.2.2</a>','</br>')
        print("Critirion : 2.2.1 - 2.2.2")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide a way to pause
movement or scrolling of content. If the user needs to pause the movement, to
reduce distraction or to have time to read it, they can do so, and then restart
it as needed. This mechanism can be provided either through interactive controls
```

```
that conform to WCAG or through keyboard shortcuts. If keyboard shortcuts are
used, they are documented.')
        print('</p>')
        for tag1 in imgs_tag21:
            if  doc.find_all('button',{'id':'refresh','scrolling':True}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error19.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error19.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                General_4 +=1
                counter18_A +=1
            elif doc.find_all('button'):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR Missing tags id|scrolling","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error19.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error19.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter20_B +=1
                General_4 +=1


            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error19.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error19.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter20_B +=1
                General_4 +=1

            break

        else:
            print('<h4>'"Result of Validation:"'</h4>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter22_C +=1

        print("<hr>")


        #WGAG 2.1 / 2 Operable / 2.2 Enough time / Critirion 2.2.5
        imgs_tag22 = doc.find_all('form')
        print("Validation for Success Crition 2.2.1 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G105.html">Click here for
critirion by WCAG 2.2.5</a>','</br>')
        print("Critirion : 2.2.5")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('Web servers that require user authentication often terminate the
session after a set period of time if there is no activity from the user. If the
user is unable to input the data quickly enough and the session times out before
they submit, the server will require re-authentication before proceeding. When
this happens, the server stores the data in a temporary cache while the user
logs in, and when the user has re-authenticated, the data is made available from
the cache and the form is processed as if there had never been a session time-
out. The server does not keep the cache indefinitely, merely long enough to
ensure success after re-authentication in a single user session, as one day.')
        print('</p>')
        for tag1 in imgs_tag22:
            if  doc.find_all('form',{'autocomplete':'on'}):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error7.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter19_A +-1
                General_105 +=1
            elif doc.find_all('form'):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag autocomplete","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter21_B +=1
                General_105 +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                General_105 +=1
                counter21_B +=1

            break

        else:
            print('<h4>'"Result of Validation:"'</h4>')
```

```
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter23_C +=1

        print("<hr>")


        #WGAG 2.1 / 2.4 Navigable / Critirion 2.4.2.
        imgs_tag23 = doc.find_all('title')
        print("Validation for Success Crition 2.4.2 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G88.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.2")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to give each Web page a
descriptive title. Descriptive titles help users find content, orient themselves
within it, and navigate through it. A descriptive title allows a user to easily
identify what Web page they are using and to tell when the Web page has changed.
The title can be used to identify the Web page without requiring users to read
or interpret page content. Users can more quickly identify the content they need
when accurate, descriptive titles appear in site maps or lists of search
results. When descriptive titles are used within link text, they help users
navigate more precisely to the content they are interested in.')
        print('</p>')
        print('<p>')
        print('The title of each Web page should:')
        print('</p>')
        print('<ol>')
        print("<li>")
        print('Identify the subject of the Web page')
        print('</li>')
        print('<li>')
        print('Make sense when read out of context, for example by a screen
reader or in a site map or list of search results')
        print('</li>')
        print('<li>')
        print('Be short')
        print('</li>')
        print('</ol>')
        print("<p>")
        print('It may also be helpful for the title to')
        print('</p>')
        print('<ol>')
```

```
        print('<li>')
        print('Identify the site or other resource to which the Web page
belongs')
        print('</li>')
        print('<li>')
        print('Be unique within the site or other resource to which the Web page
belongs')
        print('</li>')
        print('</ol>')
        for tag1 in imgs_tag23:
            if  doc.find_all('title'):
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id = "ok">')
                print('<a href = error21.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error21.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                General_88 +=1
                counter20_A +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>')
                print('<div id="error">')
                print("ERROR : Missing tag title","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error21.html#error1>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error21.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter22_B +=1
                General_88 +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>','</br>')
            print('<div id="warning">')
```

```
            print("Warning","</br>")
            print('</div>')
            counter24_C +=1


        print("<hr>")


        #WGAG 2.1 / 2.4 Navigable / Critirion 2.4.3.
        imgs_tag24 = doc.find_all('input')
        print("Validation for Success Crition 2.4.3 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H4.html">Click here for critirion
by WCAG 2.4.3</a>','</br>')
        print("Critirion : 2.4.3")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide a logical tab order
when the default tab order does not suffice. Often, G59: Placing the interactive
elements in an order that follows sequences and relationships within the content
is sufficient and this technique is not necessary. It can be very easy to
introduce usability bugs when setting the tab order explicitly.')
        print('</p>')
        print('<p>')
        print('In some cases, the author may want to specify a tab order that
follows relationships in the content without following the order of the
interactive elements in the code. In these cases, an alternative order can be
specified using the tabindex attribute of the interactive element. The tabindex
is given a value between 0 and 32767.')
        print('</p>')
        print('<p>')
        print('When the interactive elements are navigated using the tab key,
the elements are given focus in increasing order of the value of their tabindex
attribute. Elements that have a tabindex value higher than zero will receive
focus before elements without a tabindex or a tabindex of 0. After all of the
elements with a tabindex higher than 0 have received focus, the rest of the
interactive elements are given focus in the order in which they appear in the
Web page.')
        print('</p>')
        for tag1 in imgs_tag24:
            if  doc.find_all('input'  ,{'tabindex' :True }):
                print('<div id = "ok">')
                print('<a href = error10.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                print("OK",'</br>')
                counter21_A +-1
                html_4 +=1
            elif doc.find_all('input'):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missing tag tabindex","</br>")
                print('</div>')
                html_4 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html">''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter23_B +=-1
            else:
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR","</br>")
                print('</div>')
                html_4 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter23_B +=-1
            break

        else:

            print('<h4>'"Result of Validation:"'</h4>','</br>')
```

```
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')

            counter25_C +=1


        print("<hr>")


        #WGAG 2.1 / 2.4 Navigable / Critirion 2.4.3.
        imgs_tag25 = doc.find_all('a')
        print("Validation for Success Crition 2.4.3 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H4.html">Click here for critirion
by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.3")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide a logical tab order
when the default tab order does not suffice. Often, G59: Placing the interactive
elements in an order that follows sequences and relationships within the content
is sufficient and this technique is not necessary. It can be very easy to
introduce usability bugs when setting the tab order explicitly.')
        print('</p>')
        print('<p>')
        print('In some cases, the author may want to specify a tab order that
follows relationships in the content without following the order of the
interactive elements in the code. In these cases, an alternative order can be
specified using the tabindex attribute of the interactive element. The tabindex
is given a value between 0 and 32767.')
        print('</p>')
        print('<p>')
        print('When the interactive elements are navigated using the tab key,
the elements are given focus in increasing order of the value of their tabindex
attribute. Elements that have a tabindex value higher than zero will receive
focus before elements without a tabindex or a tabindex of 0. After all of the
elements with a tabindex higher than 0 have received focus, the rest of the
interactive elements are given focus in the order in which they appear in the
Web page.')
        print('</p>')
        for tag1 in imgs_tag25:
            if  doc.find_all('a',{'href':True,'tabindex' :True }):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id = "ok">')
```

```
                print('<a href = error23.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlA_4 +=1
                counter22_A +=1
            elif doc.find_all('a',{'href':True}):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missing tag tabindex","</br>")
                print('</div>')
                htmlA_4 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error23.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter24_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missing tag href","</br>")
                print('</div>')
                htmlA_4 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error23.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter24_B +=1
            break

        else:
```

```
            print('<h4>'"Result of Validation:"'</h4>','</br>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter26_C +=1

        print("<hr>")


        #WGAG 2.1 / 2.4 Navigable / Critirion 2.4.6
        print("Validation for Success Crition 2.4.6"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G131.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.6")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that the label for
any interactive component within Web content makes the component purpose clear.
Using the appropriate technology-specific techniques for technologies for
associating labels with interactive controls allows assistive technology to
recognize the label and present it to the user, therefore allowing the user to
identify the purpose of the control.The label may also be used to include text
or a text symbol indicating that input is required.')
        print('</p>')
        imgs_tag26 = doc.find_all('input')
        for tag1 in imgs_tag26:
            if  doc.find_all('input',{'type':'text','required':True }):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id = "ok">')
                print('<a href = error10.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                General_131 +=1
                counter23_A +=1
            elif doc.find_all('input',{'type':'text'}):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missing tags: type|required","</br>")
                print('</div>')
                x=(tag1.attrs)
```

```
                y1=json.dumps(x)
                print('<a href = error10.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter25_B +=1
                General_131 +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Mising tag type","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter25_B +=1
                General_131 +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>','</br>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter27_C +=1

        print("<hr>")


        #WGAG 2.1 / 2.4 Navigable / Critirion 2.4.4
        print("Validation for Success Crition 2.4.4"'<a
href="http://w3.org/WAI/WCAG21/Techniques/aria/ARIA8.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
```

```
        print('The objective of this technique is to describe the purpose of a
link using the aria-label attribute. The aria-label attribute provides a way to
place a descriptive text label on an object, as a link, when there are no
elements visible on the page that describe the object. If descriptive elements
are visible on the page, the aria-labelledby attribute should be used instead of
aria-label. Providing a descriptive text label lets a user distinguish the link
from links in the Web page that lead to other destinations and helps the user
determine whether to follow the link. In some assistive technologies the aria-
label value will show in the list of links instead of the actual link text.')
        print('</p>')
        print('<p>')
        print('Per the Accessible Name and Description Computation and the HTML
to Platform Accessibility APIs Implementation Guide, the aria-label text will
override the text supplied within the link. As the text supplied will be used
instead of the link text by AT. Due to this it is recommended to start the text
used in aria-label with the text used within the link. This will allow
consistent communication between users.')
        print('</p>')
        imgs_tag27 = doc.find_all('a')
        for tag1 in imgs_tag27:
            if  doc.find_all('a',{'href':True,'tabindex' :True }):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id = "ok">')
                print('<a href = error23.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                Aria_8 +=1
                counter24_A +=1
            elif doc.find_all('a'):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missing tag href|tabindex","</br>")
                print('</div>')
                Aria_8 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error23.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
```

```
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter26_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missing tab href","</br>")
                print('</div>')
                Aria_8 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error23.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter26_B +=1
            break

        else:

            print('<h4>'"Result of Validation:"'</h4>','</br>')
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter28_C +=1

        print("<hr>")

        #WGAG 2.1 / 2.4 Navigable / Critirion 2.4.10
        print("Validation for Success Crition 2.4.10"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G141.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that sections have
headings that identify them. Success Criterion 1.3.1 requires that the headings
be marked that they can be programmatically identified.')
        print('</p>')
        print('<p>')
```

```
        print('In HTML, this could be done using the HTML heading elements (h1,
h2, h3, h4, h5, and h6). These allow user agents to automatically identify
section headings. Other technologies use other techniques for identifying
headers. To facilitate navigation and understanding of overall document
structure, authors should use headings that are properly nested (e.g., h1
followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).')
        print('</p>')
        imgs_tag28=doc.find_all('h1')
        for tag1 in imgs_tag28:
            if  doc.find_all('h1'):
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                General_141 +=1
                print('<div id = "ok">')
                print('<a href = error26.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error26.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter25_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>','</br>')
                print('<div id="error">')
                print("ERROR : Missisn tag h1 ","</br>")
                print('</div>')
                General_141 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error26.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error26.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter27_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter29_C +=1
```

```
        print("<hr>")

        print("Validation for Success Crition 2.4.10"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G141.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that sections have
headings that identify them. Success Criterion 1.3.1 requires that the headings
be marked that they can be programmatically identified.')
        print('</p>')
        print('<p>')
        print('In HTML, this could be done using the HTML heading elements (h1,
h2, h3, h4, h5, and h6). These allow user agents to automatically identify
section headings. Other technologies use other techniques for identifying
headers. To facilitate navigation and understanding of overall document
structure, authors should use headings that are properly nested (e.g., h1
followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).')
        print('</p>')

        imgs_tag29=doc.find_all('h2')
        for tag1 in imgs_tag29:
            if  doc.find_all('h2'):
                GeneralA_141 +=1
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id = "ok">')
                print('<a href = error27.html>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error27.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter26_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag h2","</br>")
                print('</div>')
                GeneralA_141 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
```

```
                print('<a href = error27.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error27.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter28_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter30_C +=1

        print("<hr>")

        print("Validation for Success Crition 2.4.10"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G141.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that sections have
headings that identify them. Success Criterion 1.3.1 requires that the headings
be marked that they can be programmatically identified.')
        print('</p>')
        print('<p>')
        print('In HTML, this could be done using the HTML heading elements (h1,
h2, h3, h4, h5, and h6). These allow user agents to automatically identify
section headings. Other technologies use other techniques for identifying
headers. To facilitate navigation and understanding of overall document
structure, authors should use headings that are properly nested (e.g., h1
followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).')
        print('</p>')
        imgs_tag30=doc.find_all('h3')
        for tag1 in imgs_tag30:
            if  doc.find_all('h3'):
                GeneralB_141 +=1
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id = "ok">')
                print('<a href = error28.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error28.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter27_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR Missing tag h3","</br>")
                print('</div>')
                GeneralB_141 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error28.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error28.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter29_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter31_C +=1

        print("<hr>")


        print("Validation for Success Crition 2.4.10"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G141.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that sections have
headings that identify them. Success Criterion 1.3.1 requires that the headings
be marked that they can be programmatically identified.')
        print('</p>')
        print('<p>')
```

```
        print('In HTML, this could be done using the HTML heading elements (h1,
h2, h3, h4, h5, and h6). These allow user agents to automatically identify
section headings. Other technologies use other techniques for identifying
headers. To facilitate navigation and understanding of overall document
structure, authors should use headings that are properly nested (e.g., h1
followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).')
        print('</p>')
        imgs_tag31=doc.find_all('h4')
        for tag1 in imgs_tag31:
            if  doc.find_all('h4'):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id = "ok">')
                print('<a href = error29.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error29.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                GeneralC_141 +=1
                counter28_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag h4","</br>")
                print('</div>')
                GeneralC_141 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error29.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error29.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter30_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter32_C +=1
```

```
        print("<hr>")


        print("Validation for Success Crition 2.4.10"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G141.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that sections have
headings that identify them. Success Criterion 1.3.1 requires that the headings
be marked that they can be programmatically identified.')
        print('</p>')
        print('<p>')
        print('In HTML, this could be done using the HTML heading elements (h1,
h2, h3, h4, h5, and h6). These allow user agents to automatically identify
section headings. Other technologies use other techniques for identifying
headers. To facilitate navigation and understanding of overall document
structure, authors should use headings that are properly nested (e.g., h1
followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).')
        print('</p>')
        imgs_tag32=doc.find_all('h5')
        for tag1 in imgs_tag32:
            if  doc.find_all('h5'):
                print('<h4>'"Result of Validation:"'</h4>' )
                GeneralD_141 +=1
                print('<div id = "ok">')
                print('<a href = error30.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error30.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter29_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR Missing tag h5","</br>")
                print('</div>')
                GeneralD_141 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
```

```
                print('<a href = error30.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error30.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter31_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter33_C +=1
        print('<hr>')

        print("Validation for Success Crition 2.4.10"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G141.html">Click here for
critirion by WCAG 2.1 </a>','</br>')
        print("Critirion : 2.4.10")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that sections have
headings that identify them. Success Criterion 1.3.1 requires that the headings
be marked that they can be programmatically identified.')
        print('</p>')
        print('<p>')
        print('In HTML, this could be done using the HTML heading elements (h1,
h2, h3, h4, h5, and h6). These allow user agents to automatically identify
section headings. Other technologies use other techniques for identifying
headers. To facilitate navigation and understanding of overall document
structure, authors should use headings that are properly nested (e.g., h1
followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).')
        print('</p>')
        imgs_tag33=doc.find_all('h6')
        for tag1 in imgs_tag33:
            if  doc.find_all('h6'):
                print('<h4>'"Result of Validation:"'</h4>' )
                GeneralE_141 +=1
                print('<div id = "ok">')
                print('<a href = error31.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
```

```
                with open('submit_v1.py','r') as firstfile ,
open('error31.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter30_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR Missing tag h6","</br>")
                print('</div>')
                GeneralE_141 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error31.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error31.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter32_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter34_C +=1
        print('<hr>')


        #WGAG 2.1 / 2.5 Motion Actuation / Critirion 2.5.4
        imgs_tag34 = doc.find_all('input')
        print("Validation for Success Crition 2.5.4 "'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G213.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 2.5.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to ensure that:')
        print('</p>')
        print('<ol>')
        print("<li>")
```

```
        print('People who use mounted devices or who cannot precisely manipulate
a device are able to provide inputs through more conventional user interface
components, and')
        print('</li>')
        print('<li>')
        print('People who may accidentally activate sensors due to tremors or
other motor impairments have the ability to turn off motion actuation to prevent
accidental triggering of functions.')
        print('</li>')
        print('</ol>')
        print("<p>")
        print('It may also be helpful for the title to')
        print('</p>')
        print('<ol>')
        print('<li>')
        print('When a device sensor as an accelerometer or gyroscope is used to
gather input:')
        print('</li>')
        print('<li>')
        print('A button, link, or other conventional control is provided that
does not rely on sensor input, and')
        print('</li>')
        print('<li>')
        print('A setting is provided to allow the user to turn off the sensor
data.')
        print('</li>')
        print('</ol>')
        for tag1 in imgs_tag34:
            if  doc.find_all('input',{'type':'text','id':'clear' }):
                print('<h4>'"Result of Validation:"'</h4>' )
                General_213 +=1
                print('<div id = "ok">')
                print('<a href = error10.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter31_A +=1
            elif  doc.find_all('input',{'type':'text'}):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag id","</br>")
```

```
                print('</div>')
                General_213 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter33_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missina tags id|type","</br>")
                print('</div>')
                General_213 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter33_B +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter35_C +=1

        print('<hr>')

        #WGAG 2.1 / Readable / Critirion 3.1.2
        imgs_tag35 = doc.find_all("blockquote" )
        print("Validation for Success Crition 3.1.2"'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H58.html">Click here for
critirion by WCAG 2.1 </a>','</br>')
        print("Critirion : 3.1.2")
```

```
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to clearly identify any
changes in language on a page by using the lang attribute.')
        print('Allowed values for the lang attribute are indicated in the
resources referenced below. Language tags use a primary code to indicate the
language, and optional sub-codes (separated by hyphen characters) to indicate
variants of the language. For instance, English is indicated with the primary
code "en"; British English and American English can be distinguished by using
"en-GB" and "en-US", respectively. Use of the primary code is important for this
technique. Use of subcodes is optional but may be helpful in certain
circumstances.')
        print('</p>')
        for tag1 in imgs_tag35:
            if  doc.find_all("blockquote" ,{'lang':True}):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id = "ok">')
                print('<a href = error33.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error33.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                html_58 +=1
                counter32_A +=1
            elif doc.find_all("blockquote" ):
                print('<h4>'"Result of Validation:"'</h4>' )
                html_58 +=1
                print('<div id="error">')
                print("ERROR : Missing tag lang","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error33.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error33.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter34_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
```

```
                html_58 +=1
                print('<div id="error">')
                print("ERROR : Missing tag blockquote","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error33.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error33.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter34_B +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter36_C +=1

        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.1. Readable / Critirions 3.1.3 -3.1.4
        imgs_tag36 = doc.find_all('a')
        print("Validation for Success Crition 3.1.3-3.1.4"'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H58.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.1.3 - 3.1.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide the definition of
words, phrases, jargon, or abbreviation expansions by adding a mechanism to
access an on-line dictionary to the Web page. This technique uses existing
resources on the Web to provide the definition rather than requiring the author
to create a glossary or other mechanism within the site. By providing access
from within the Web page, a user can easily locate the desired definition. This
technique can only be used if the online dictionary returns the correct
definition.')
        print('</p>')
        for tag1 in imgs_tag36:
            if  doc.find_all('a',{'href':('[a-zA-Z]+dictionary.gr|com') }):
                print('<h4>'"Result of Validation:"'</h4>' )
```

```
                htmlA_58 +=1
                print('<div id = "ok">')
                print('<a href = error23.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter33_A +=1
            elif doc.find_all('a',{'href'}):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing Dictionary for tag","</br>")
                print('</div>')
                print(tag1.attrs)
                htmlA_58 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html#error7>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter35_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing Dictionary for tag","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html#error7>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                htmlA_58 +=1
                counter35_B +=1
            break
```

```
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter37_C +=1

        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.1. Readable / Critirion 3.1.6
        imgs_tag37 = doc.find_all('ruby')
        print("Validation for Success Crition 3.1.6"'<a
href="http://w3.org/WAI/WCAG21/Techniques/html/H62.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.1.6")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to use ruby annotation to
provide information about the pronunciation and meaning of a run of text where
meaning is determined by pronunciation.')
        print('</p>')
        print('<p>')
        print('There are many languages in which a run of text may mean
different things depending on how the text is pronounced. This is common in East
Asian languages as well as Hebrew, Arabic, and other languages; it also occurs
in English and other Western European languages.')
        print('</p>')
        print('<p>')
        print('Ruby Annotation allows the author to annotate a "base text,"
providing a guide to pronunciation and, in some cases, a definition as well.
Ruby is commonly used for text in Japanese and other East Asian languages. Ruby
Annotation is defined as a module for XHTML 1.1 or HTML5.')
        print('</p>')
        print('<p>')
        print('There are two types of Ruby markup: simple and complex. Simple
Ruby markup applies to a run of text as a complete word or phrase. This is known
as the "base" text (rb element). The Ruby annotation that indicates how to
pronounce the term (the rt element, or Ruby text) is shown in a smaller font.
(The term "Ruby" is derived from a small font used for this purpose in printed
texts.) The Ruby text is usually rendered above or immediately before the base
text, that is, immediately above horizontal text or immediately to the right of
vertical text. Sometimes Japanese uses Ruby to provide the meaning of text on
the other side of the base text (visually) from the phonetic annotation. Simple
Ruby markup also provides a "fallback" option for user agents that do not
```

```
support Ruby markup (that is, user agents that do not support XHTML 1.1 or
HTML5).')
        print('</p>')
        print('<p>')
        print('Complex Ruby markup makes it possible to divide the base text
into smaller units, each of which may be associated with a separate Ruby
annotation. Complex Ruby markup does not support the fallback option.')
        print('</p>')
        print('<p>')
        print('Ruby annotation is uncommon in languages as Hebrew, where Unicode
fonts can include diacritical marks that convey pronunciation. It is also
uncommon in English and European languages.')
        print('</p>')
        print('<p>')
        print('Note: The primary reason for indicating pronunciation through
Ruby or any other means is to make the content accessible to people with
disabilities who could read and understand the language of the content if
information about pronunciation were provided. It is not necessary to provide
information about pronunciation for use by people who are not familiar with the
language of the content.')
        print('</p>')
        for tag1 in imgs_tag37:
            if  doc.find_all('ruby'):
                print('<h4>'"Result of Validation:"'</h4>' )
                html_62 +=1
                print('<div id = "ok">')
                print('<a href = error34.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error34.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter34_A +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag rubi","</br>")
                print('</div>')
                html_62 =+1
                x=(tag1.attrs)
                y1=json.dumps(x)
```

```
                print('<a href = error34.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error34.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter36_B +=1

            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter38_C +=1


        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.1. Readable / Critirion 3.2.1
        imgs_tag38 = doc.find_all('button')
        print("Validation for Success Crition 3.2.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G107.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.2.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide a method for
activating things that is predictable by the user. Users with cognitive
disabilities and people using screen readers or screen magnifiers may be
confused by an unexpected event as automatic form submission or activation of a
function that causes a change of context.')
        print('</p>')
        print('<p>')
        print('With this technique, all changes of context would be triggered
only by a specific action on the part of the user. Further, that action would be
one that usually causes changes in context, as clicking on a link or pressing a
submit button. Actions that simply move the focus to an element would not cause
a change of context.')
        print('</p>')
        for tag1 in imgs_tag38:
            if  doc.find_all('button',{'label':'next step' }):
                print('<h4>'"Result of Validation:"'</h4>' )
```

```
                General_107 +=1
                print('<div id = "ok">')
                print('<a href = error35.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error35.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter35_A +=1
            elif doc.find_all('button'):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag label","</br>")
                print('</div>')
                General_107 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error35.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error35.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter37_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag button","</br>")
                print('</div>')
                General_107 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error35.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error35.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter37_B +=1
            break
```

```
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter39_C +=1


        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.1. Readable / Critirion 3.2.2
        imgs_tag39 = doc.find_all('form')
        print("Validation for Success Crition 3.2.2"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G80.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.2.2")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide a mechanism that
allows users to explicitly request changes of context. Since the intended use of
a submit button is to generate an HTTP request that submits data entered in a
form, this is an appropriate control to use for causing a change of context and
is a practice that does not create confusion for users.')
        print('</p>')
        for tag1 in imgs_tag39:
            if  doc.find_all('form',{'select':True}):
                print('<h4>'"Result of Validation:"'</h4>' )
                General_80 +=1
                print('<div id = "ok">')
                print('<a href = error7.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter36_A +=1
            elif doc.find_all('form'):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag select ","</br>")
                print('</div>')
                General_80 +=1
                x=(tag1.attrs)
```

```
                y1=json.dumps(x)
                print('<a href = error7.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter38_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag form","</br>")
                print('</div>')
                General_80 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter38_B +=1
            break
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter40_C +=1


        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.2 Rredictable / Critirion 3.2.4
        imgs_tag40 = doc.find_all('form')
        print("Validation for Success Crition 3.2.4"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G197.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.2.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
```

```
        print('The purpose of this technique is to help users with cognitive
disabilities, blindness and vision loss to understand what will happen when they
interact with a function on a Web page. If there are different labels on user
interface components (i.e., elements, links, JavaScript objects, etc.) that have
the same function, the user will not know that they have encountered a component
with the same function and will not know what to expect. This could lead to many
unnecessary errors. It is also recommended that this approach to consistent
labelling be applied across the Web site.')
        print('</p>')
        for tag1 in imgs_tag40:
            if  doc.find_all('form',{'label':True}):
                    print('<h4>'"Result of Validation:"'</h4>' )
                    General_197 +=1
                    print('<div id = "ok">')
                    print('<a href = error7.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                    print('</div>')
                    with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                         for line in firstfile:
                            secondfile.writelines(y1)
                            break
                    counter37_A +=1
            elif doc.find_all('form'):
                    print('<h4>'"Result of Validation:"'</h4>' )
                    General_197 +=1
                    print('<div id="error">')
                    print("ERROR : Missing tag label","</br>")
                    print('</div>')
                    x=(tag1.attrs)
                    y1=json.dumps(x)
                    print('<a href = error7.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                    with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                            for line in firstfile:
                              secondfile.writelines(y1)
                              break
                    counter39_B +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR","</br>")
                print('</div>')
                General_197 +=1
```

```
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error7.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error7.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter39_B +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter41_C +=1


        print('<hr>')


        #WGAG 2.1 / 3 Understable /3.3 Input Assistance / Critirion 3.3.1
        imgs_tag41 = doc.find_all('input')
        print("Validation for Success Crition 3.3.1"'<a
href="http://w3.org/WAI/WCAG21/Techniques/Aria/ARIA21.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.3.1")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('This technique demonstrates how aria-invalid may be employed to
specifically identify fields that have failed validation. Its use is most
suitable when:')
        print('</p>')
        print('<ul>')
        print('<li>')
        print('The error description does not programmatically identify the
failed fields')
        print('</li>')
        print('<li>')
        print('The failed fields are identified in a manner that is not
available to some users - for example by using an error-icon rendered visually
by some technique that does not rely on color as a visual cue like a border.')
        print('</li>')
```

```
        print('</ul>')
        print('<p>')
        print('While it is always preferable to programmatically associate
specific error description with the failed field, the page design or the
framework employed may sometimes constrain the author ability to do so. In these
cases, authors may programmatically set aria-invalid to "true" on the fields
that have failed validation. This is interpretable mainly by assistive
technologies (like screen readers / screen magnifiers) employed by users who are
vision impaired. When a field has aria-invalid set to “true”, VoiceOver in
Safari announces “invalid data” when the field gets focus; JAWS and NVDA notify
the error as an “invalid entry”.')
        print('</p>')
        print('<p>')
        print('This ARIA attribute has to be set / turned on programmatically.
It should not be set to “true” before input validation is performed or the form
is submitted. Setting aria-invalid to “false” is the same as not placing the
attribute for the form control at all. Quite understandably, nothing is conveyed
by assistive technology to users in this case.')
        print('</p>')
        print('<p>')
        print('When visible text is used to programmatically identify a failed
field and / or convey how the error can be corrected, setting aria-invalid to
"true" is not required from a strict compliance standpoint but may still provide
helpful information for users.')
        print('</p>')
        for tag1 in imgs_tag41:
            if  doc.find_all('input',{'aria-invalid':True,'aria-
describebdy':('^error$'),'id':('^error$')}):
                print('<h4>'"Result of Validation:"'</h4>' )
                Aria_21 +=1
                print('<div id = "ok">')
                print('<a href = error10.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter38_A +=1
            elif doc.find_all('input',{'aria-invalid':True,'aria-
describebdy':('^error$')}):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag id","</br>")
```

```
                print('</div>')
                Aria_21 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter40_B +=1
            elif doc.find_all('input',{'aria-invalid':True,'id':('^error$')}):
                print('<h4>'"Result of Validation:"'</h4>' )
                Aria_21 +=1
                print('<div id="error">')
                print("ERROR : Missing tag aria-describedby ","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter40_B +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tags aria-invalid|aria-
describebdy","</br>")
                print('</div>')
                Aria_21 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
```

```
                counter40_B +=1

            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter42_C +=1


        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.3 Input Assistance / Critirion 3.3.3
        imgs_tag42 = doc.find_all("div")
        print("Validation for Success Crition 3.3.3"'<a
href="http://w3.org/WAI/WCAG21/Techniques/Aria/ARIA18.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.3.3")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The purpose of this technique is to alert people that an input
error has occured. Using role="alertdialog" creates a notification. This
notification should be modal with the following characteristics:')
        print('</p>')
        print('<ul>')
        print('<li>')
        print('aria-label or aria-labelledby attribute gives the alertdialog an
accessible name.')
        print('</li')
        print('li')
        print('aria-describedby provides a reference to the text of the alert.')
        print('</li>')
        print('li')
        print('The alertdialog contains at least one focusable control, and the
focus should move to that control when the alertdialog opens.')
        print('</li>')
        print('<li>')
        print('The tab order is constrained within the alertdialog whilst it is
open.')
        print('</li>')
        print('<li>')
        print('When the dialog is dismissed, the focus moves back to the
position it had before the dialog opened, if possible.')
```

```
        print('</li>')
        print('</ul>')
        print('<p>')
        print('Note that the alertdialog should not be present in a way that it
will be accessed by AT until it is needed. One way to do this is not to include
it in the static HTML and instead to insert it into the DOM via script when the
error condition is triggered. The insertion would correspond to the following
HTML sample.')
        print('</p>')
        for tag1 in imgs_tag42:
            if  doc.find_all('div',{'role':('^alert$'),'aria-
labelebdy':('^alert$'),'aria-describedby':('^alert$')}):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id = "ok">')
                print('<a href = error38.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error38.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                Aria_18 +=1
                counter39_A +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tags: role|aria-labelbdy|aria-
describebdy",'</br>')
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error38.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error38.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                Aria_18 +=1
                counter41_B +=1
            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
```

```
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter43_C +=1

        print('<hr>')


        #WGAG 2.1 / 3 Understable /3.3 Input Assistance / Critirion 3.3.4
        imgs_tag43 = doc.find_all('input')
        print("Validation for Success Crition 3.3.4"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G98.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.3.4")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide users with a way to
ensure their input is correct before completing an irreversible transaction.
Testing, financial, and legal applications permit transactions to occur which
cannot be "undone". It is therefore important that there be no errors in the
data submission, as the user will not have the opportunity to correct the error
once the transaction has been committed.')
        print('</p>')
        print('<p>')
        print('On a simple, 1-page form this is easy because the user can review
the form before submitting. On a form that spans multiple Web pages, however,
data is collected from the user in multiple steps before the transaction is
committed. The user may not recall all of the data that was entered in previous
steps before the step which commits the transaction.')
        print('</p>')
        print('<p>')
        print('One approach is to cache the results of each individual step and
allow the user to navigate back and forth at will to review all data entered.
Another approach is to provide a summary of all data collected in all steps for
the user to review prior to the final commitment of the transaction.')
        print('</p>')
        print('<p>')
        print('Before the final step that commits the transaction to occur,
instructions are provided to prompt the user to review the data entered and
confirm. Once the user confirms, the transaction is completed.')
        print('</p>')
        for tag1 in imgs_tag43:
            if
doc.find_all('input',{'type':'text','label':'cancel','label':'back','label':'nex
t','label':'submit'}):
```

```
                print('<h4>'"Result of Validation:"'</h4>' )
                General_98 +=1
                print('<div id = "ok">')
                print('<a href = error10.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter40_A +=1
            elif doc.find_all('input',{'type':'text'}):
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag label","</br>")
                print('</div>')
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter42_B +=1
                General_98 +=1
            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
                print("ERROR : Missing tag type","</br>")
                print('</div>')
                General_98 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error10.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error10.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter42_B +=1
            break
```

```
        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter44_C +=1

        print('<hr>')

        #WGAG 2.1 / 3 Understable /3.3 Input Assistance / Critirion 3.3.5
        imgs_tag44 = doc.find_all('a')
        print("Validation for Success Crition 3.3.5"'<a
href="http://w3.org/WAI/WCAG21/Techniques/general/G71.html">Click here for
critirion by WCAG 2.1</a>','</br>')
        print("Critirion : 3.3.5")
        print('<h3>'"Description:"'</h3>')
        print('<p>')
        print('The objective of this technique is to provide context sensitive
help for users as they enter data in forms by providing at least one link to the
help information on each Web page. The link targets a help page with information
specific to that Web page. Another approach is to provide a help link for every
interactive control. Positioning this link immediately before or after the
control allows users to easily tab to it if they have problems in the control.
Displaying the help information in a new browser window ensures that any data
that has already been entered into the form will not be lost. NOTE: A link is
not the only means to provide help.')
        print('</p>')
        for tag1 in imgs_tag44:
            if  doc.find_all('a',{'href':('^help$.html|^HELP&.html')}):
                print('<h4>'"Result of Validation:"'</h4>' )
                General_71 +=1
                print('<div id = "ok">')
                print('<a href = error23.html#>''OK
line:',tag1.sourceline,'</a>','</br>')
                print('</div>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter41_A +=1

            else:
                print('<h4>'"Result of Validation:"'</h4>' )
                print('<div id="error">')
```

```
                print("ERROR : Missing tag help.html","</br>")
                print('</div>')
                General_71 +=1
                x=(tag1.attrs)
                y1=json.dumps(x)
                print('<a href = error23.html#>''Error in
line:',tag1.sourceline,'</a>','</br>')
                with open('submit_v1.py','r') as firstfile ,
open('error23.html','w') as secondfile:
                    for line in firstfile:
                        secondfile.writelines(y1)
                        break
                counter43_B +=1

            break

        else:
            print('<h4>'"Result of Validation:"'</h4>' )
            print('<div id="warning">')
            print("Warning","</br>")
            print('</div>')
            counter45_C +=1

        print('<hr>')


        print('</div>')



        print('<div class="split right">')
        counter_yes =
(counter_A+counter1_A+counter2_A+counter3_A+counter4_A1+counter5_A+counter6_A+co
unter7_A+counter8_A+counter9_A+counter10_A+counter11_A+counter12_A+counter13_A+c
ounter15_A+counter16_A

+counter17_A+counter18_A+counter19_A+counter20_A+counter21_A+counter22_A+counter
23_A+counter24_A+counter25_A+counter26_A+counter27_A+

counter28_A+counter29_A+counter30_A+counter31_A+counter32_A+counter33_A+counter3
4_A+counter35_A+counter36_A+counter37_A+counter38_A+counter39_A+counter40_A+coun
ter41_A+counter42_A)
        counter_error =
(counter1_B+counter2_B+counter3_B+counter4_B+counter5_B+counter6_B+counter7_B+co
unter8_B+counter9_B+counter10_B
```

```
+counter11_B+counter12_B+counter13_B+counter14_B+counter15_B+counter16_B+counter
17_B+counter18_B+counter19_B+counter20_B+counter21_B+counter22_B+counter23_B

+counter24_B+counter25_B+counter26_B+counter27_B+counter28_B+counter29_B+counter
30_B+counter31_B+counter32_B+counter33_B+counter34_B+counter35_B+counter36_B

+counter37_B+counter38_B+counter39_B+counter40_B++counter41_B++counter42_B++coun
ter43_B+counter44_B+counter45_B)
        counter_No =
(counter_C+counter1_C+counter2_C+counter3_C+counter4_C+counter5_C+counter6_C+cou
nter7_C+counter8_C+counter9_C+counter10_C+counter11_C

+counter12_C+counter13_C+counter14_C+counter15_C+counter16_C+counter17_C+counter
18_C+counter19_C+counter20_C+counter21_C+counter22_C+counter23_C+counter24_C

+counter25_C+counter26_C+counter27_C+counter29_C+counter30_C+counter31_C+counter
32_C+counter33_C+counter34_C+counter35_C+counter37_C+counter38_C+counter39_C

+counter40_C+counter41_C+counter42_C+counter43_C+counter44_C+counter45_C+counter
46_C)
        print("<h1>Summary</h1>")
        print('<table style = "width:70%" align="center">')
        print('<tr>')
        print('<th>')
        print('<div id="ok">')
        print('Correct Tags')
        print('</div>')
        print('</th>')
        print('<th>')
        print('<div id="error">')
        print('Error Tags')
        print('</div>')
        print('</th>')
        print('<th>')
        print('<div id="warning">')
        print("Warnings","</br>")
        print('</div>')
        print('</th>')
        print('</tr>')
        print('<tr>')
        print('<th>')
        print(counter_yes)
        print('</th>')
        print('<th>')
```

```
        print(counter_error)
        print('</th>')
        print('<th>')
        print(counter_No)
        print('</th>')
        print('</tr>')
        print('</table>')


        print('<div>')

        print('</br>')
        print('</div>')
        print('<div>')

        Aria_Tags=(Aria_8+Aria_21+Aria_18+Aria_10+Aria_11)
        Aria_Tags_Summary = (Aria_Tags / (counter_yes+counter_error)*100)

        General_Tags=(General_134+General_159+General_96+General_171
+General_174 +General_4+General_105+General_88
+General_131+General_141+GeneralA_141+GeneralB_141+

GeneralC_141+GeneralD_141+GeneralE_141+General_213+General_107+General_80+Genera
l_197+General_98+General_57 +General_199+General_91+
        General_14 +General_17 +General_195+General_215+General_71 )
        Ceneral_Tags_Summary = (General_Tags / (counter_yes+counter_error)*100)

Html_Tags=(html_91+htmlA_91+htmlB_91+htmlC_91+htmlD_91+htmlE_91+htmlF_91+htmlH_9
1+
        htmlJ_91+htmlG_91+html_4+htmlA_4+html_62+html_58+html_57+htmlA_58)
        Html_Tags_Summary = (Html_Tags / (counter_yes+counter_error)*100)
        Css_Tags=(Css_22 +Css_32+Css_8+Css_39+Css_15)
        formatting_string1 = "{:.2f}".format(Aria_Tags_Summary)
        float_value1 = float(formatting_string1)
        Css_Tags_Summary =(Css_Tags / (counter_yes+counter_error)*100)
        formatting_string2 = "{:.2f}".format(Ceneral_Tags_Summary)
        float_value2 = float(formatting_string2)
        formatting_string3= "{:.2f}".format(Html_Tags_Summary)
        float_value1 = float(formatting_string3)
        formatting_string4= "{:.2f}".format(Css_Tags_Summary)
        float_value1 = float(formatting_string4)
        formatting_string5= "{:.2f}".format((counter_yes /
(counter_yes+counter_error+counter_No))*100)
        float_value1 = float(formatting_string5)
        print('</br>')
```

```
        print('</br>')
        print('<h2>'"Techniques by WCAG21"'</h2>')
        print('<table style = "width:70%" align="center">')
        print('<tr>')
        print('<th>')
        print('Aria')
        print('</th>')
        print('<th>')
        print('Gereral')
        print('</th>')
        print('<th>')
        print('Html')
        print('</th>')
        print('<th>')
        print('Css')
        print('</th>')
        print('</tr>')
        print('<tr>')
        print('<th>')
        print(float_value1)
        print('%')
        print('</th>')
        print('<th>')
        print(formatting_string2 )
        print('%')
        print('</th>')
        print('<th>')
        print(formatting_string3)
        print('%')
        print('</th>')
        print('<th>')
        print(formatting_string4)
        print('%')
        print('</th>')
        print('</tr>')
        print('</table>')
        print('</div>')
        print('</br>')
        print('</br>')
        print('<h5>''Rate of success by WCAG21 QUICK REFERENCES
CRITIRIONS''</h5>')
        print('<table style = "width:70%" align="center">')
        print('<tr>')
        print('<th>')
        print('Percent %')
```

```
        print('</th>')
        print('<td>')
        print(formatting_string5)
        print('</td>')
        print('</tr>')
        print('</table>')
        print('</br>')
        print ('<form action = "http://localhost/codeMcs/index.html"method =
"POST" class ="form">')
        print('<button class="button">New Validation</button>')
        print("</form>")
        print('</div>')
except requests. ConnectionError as exception:
        print ('<h1>'"The url does not exist"'</h1>')
        print ('<form action = "http://localhost/codeMcs/index.html"method =
"POST" class ="form">')
        print('<button class="button">Back</button>')
        print("</form>")
print ('</body>')
print ('</html>')
```

Αρχεία CSS

1ο:

```
#header {
    background-color: rebeccapurple;
    height: 100px;
}
#main {
    border: solid 2px blue;
    margin-left: 100px;
    min-height: 300px;
    margin-right: 10px;

}
#sidebar {
    background: color #0ff;
    width: 180px;
    position:relative;
    top: 10px;
    min-height: 300px;
}
#footer {
    border: 2px solid grey;
    padding-bottom: 5%;
    margin: auto;
}

#hide {
    display: none;
```

```
    background: rebeccapurple;
}

#MyDIV:hover + #hide {
    display: block;
    color: red;
}

.split {
    height: 100%;
    width: 50%;
    position: fixed;
    z-index: 1;
    top: 0;
    overflow-x: hidden;
    padding-top: 20px;
  }

  /* Control the left side */
  .left {
    left: 0;
    right: 50%;
    background-color:rgb(236, 238, 227);
    border: 1px solid rgb(168, 60, 60);
    padding-left: 3px;
    padding-top: 3px;
    margin-right: 10px;
    }

  .down{
      nav-down: auto;
  }


  /* Control the right side */
  .right {
    right: 0;
    left:50%;
    background-color:rgb(236, 238, 227);
    padding-top: 3px;
    border: 1px solid rgb(168, 60, 60);
    margin-left: 0px;
    padding-left:0px;




  }

  /* If you want the content centered horizontally and vertically */
  .centered {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    text-align: center;
  }


  #false{
    color: powderblue;
  }


  #top{
     position: absolute;
     top:0;
    right: 0;
```

```
    left: 0;
    bottom: 50%;
    background-color: rgb(207, 241, 241);
    text-align: center;

   }

   #bottom{
    position: absolute;
    top:30%;
   right: 0;
   left: 0;
   bottom: 0;
   background-color: bisque;
   text-align:center;
   color: #ffffff;

  }

  #p {
    margin :0;
    padding: 0;
  }
h1{
  text-align: center;

}
h2 {
  text-align: center;
}
h5 {
  text-align: center;
}
.tags {
    background-color: red;
}
table , th ,td {
  border: 1px solid black;

}

#error{
  color: red;
}
#ok a{
  color: green;
}
#ok {
  color: green;
}
.button {
  background-color:rosybrown;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 260px;
  cursor: pointer;
}
#warning{
  color: orange;
}
```

2ο :

```
body{background:#59ABE3;margin:0}
.form{width:340px;height:440px;background:#e6e6e6;border-radius:8px;box-shadow:0
0 40px -10px #000;margin:calc(50vh - 220px) auto;padding:20px 30px;max-
width:calc(100vw - 40px);box-sizing:border-box;font-family:'Montserrat',sans-
serif;position:relative}
h2{margin:10px 0;padding-bottom:10px;width:180px;color:#78788c;border-bottom:3px
solid #78788c}
input{width:100%;padding:10px;box-sizing:border-
box;background:none;outline:none;resize:none;border:0;font-
family:'Montserrat',sans-serif;transition:all .3s;border-bottom:2px solid
#bebed2}
input:focus{border-bottom:2px solid #78788c}
p:before{content:attr(type);display:block;margin:28px 0 0;font-
size:14px;color:#5a5a5a}
button{float:right;padding:8px 12px;margin:8px 0 0;font-
family:'Montserrat',sans-serif;border:2px solid
#78788c;background:0;color:#5a5a6e;cursor:pointer;transition:all .3s}
button:hover{background:#78788c;color:#fff}
div{content:'Hi';position:absolute;bottom:-15px;right:-
20px;background:#50505a;color:#fff;width:320px;padding:16px 4px 16px 0;border-
radius:6px;font-size:13px;box-shadow:10px 10px 40px -14px #000}
span{margin:0 5px 0 15px}
```

Αρχείο Script:

```
function replace(){
  var myLoc = window.prompt('Please enter a web site address', 'http://');
getHttp =   myLoc.substring(0, 7);

if ( getHttp == "http://")
{
finalUrl = myLoc;
} else {
finalUrl = 'http://' + myLoc;
}

window.location = 'view-source:' + finalUrl;
};

function urls(){
 var los = ["http://w3.org/WAI/WCAG21/understanding/resize-test",
 "http://w3.org/WAI/WCAG21/Understanding/images-of-text",
 "http://w3.org/WAI/WCAG21/Understanding/visual-presentation",
```

```
 "http://w3.org/WAI/WCAG21/Understanding/images-of-text-no-exception"]

for (let i=0; i<los.length; i++){
 window.open(los[i])}
};
function urls1(){
  var los = ["http://w3.org/WAI/WCAG21/Tequiques/general/G4.html",
  "http://w3.org/WAI/WCAG21/Tequiques/general/G4.html"]

 for (let i=0; i<los.length; i++){
  window.open(los[i])}
};


function urls2(){
  var los = ["http://w3.org/WAI/WCAG21/Techniques/general/G57.html",
  "http://w3.org/WAI/WCAG21/Techniques/general/G14.html",
  "http://w3.org/WAI/WCAG21/Techniques/general/G17.html",
  "http://w3.org/WAI/WCAG21/Techniques/css/C22.html",
  "http://w3.org/WAI/WCAG21/Techniques/css/C32.html",
  "http://w3.org/WAI/WCAG21/Techniques/general/G195.html",
  "http://w3.org/WAI/WCAG21/Techniques/css/C8.html",
  "http://w3.org/WAI/WCAG21/Techniques/css/C39.html",
  "http://w3.org/WAI/WCAG21/Techniques/general/G213.html",
  "http://w3.org/WAI/WCAG21/Techniques/css/C15.html"
]

 for (let i=0; i<los.length; i++){
  window.open(los[i])}
};


function urls3(){
  var los = ["http://w3.org/WAI/WCAG21/Techniques/html/H91.html",
  "http://w3.org/WAI/WCAG21/Techniques/General/G91.html"
]

 for (let i=0; i<los.length; i++){
  window.open(los[i])}
};


function refresh(){
  setTimeout(function(){
    location.replace("http://localhost/codeMcs/index.html")
```

```
  },100);
};
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.w3.org/WAI/fundamentals/accessiblity-intro/

(accessed 20 December 2021)

[2] https://www.w3.org/WAI/standards-guidelines/wcag ,

https://www.w3.org/WAI/standards-guidelines/atag,

https://www.w3.org/WAI/standards-guidelines/uaag,

https://www.w3.org/WAI/standards-guidelines/aria,

https://www.w3.org/WAI/standards-guidelines/wcag

(accesed 2 February 2022)

[3] G. Broccia, M. Manca , F. Paterno , F. Pulina , “Flexible Automatic Support For Web Accessibility Validation” , June 2020

[4] Brajnic, G(2004). Comparing accessibility evaluation tools : a method for tools effectivnsess. *Universal access in information society ,* 3(3-4)