Development of a web-interface and web service API to support process mining techniques.

Ilias Merkoureas
Mcse19006

Supervisor:
Dr Georgios Miaoulis, Professor
Development of a web-interface and web service API to support process mining techniques
Development of a web-interface and web service API to support process mining techniques

MASTER THESIS

Development of a web-interface and web service API to support process mining techniques.

Ilias Merkoureas
Mcse19006

Supervisor:

Dr Georgios Miaoulis, Professor

Examination Committee:

Georgios Miaoulis, Professor
Nikolaos Vasilas, Professor
Athanasios Voulodimos, Assistant Professor

Examination Date 09/04/2021
Development of a web-interface and web service API to support process mining techniques
POSTGRADUATE WORK AUTHOR’S STATEMENT

The following undersigned Merkoureas Ilias of Vasileiou, with registration number mcse19006 student of the Postgraduate Program Studies Computer Science and Engineering of the School Department of Informatics and Computer Engineering of West Attica, I declare that:
«I am the author of this master’s thesis and that all the help I had for its preparation is fully recognized and refers to the work. Also, any sources from which I used data, ideas, or words, whether exact or paraphrased, are listed in the entirety, with full reference to the authors, publisher, or magazine, including any sources that may have been used by the Internet. I also certify that this work has been written exclusively by me and is an intellectual property product of both myself and the Foundation.
Violation of my above academic responsibility is an essential reason for the revocation of my degree».
I wish to be denied access to the full text of my work until ............... and upon my request to the Library and approval of the supervising professor.

The Declarant
Development of a web-interface and web service API to support process mining techniques
SPECIAL THANKS

This master thesis was completed after persistent efforts, in an interesting subject, such as process mining. This effort was supported by my supervising teacher, whom I would like to thank.

I would also like to thank Mrs. Georgia Theodoropoulou for her valuable advice.
Development of a web-interface and web service API to support process mining techniques
ABSTRACT

This master thesis deals with the development of a web service application (web API) which will use the functions of process mining that have already been implemented in python with library pm4py. A web interface was also developed which calls with http requests the web service in which the user can apply process mining techniques online. In this environment there is the possibility of selecting event logs and applying multiple process automatic retrieval algorithms by selecting various parameters based on the algorithm. The interface allows the user to select parameters for the corresponding procedures.
CONTENTS

1. **Introduction** ........................................................................................................ 14
   1.1 Introduction ................................................................................................... 14
   1.2 Theoretical Background ............................................................................. 15
      1.2.1 Xes Standard .......................................................................................... 15
      1.2.2 Event Logs ............................................................................................ 15
      1.2.3 Process Mining for Python (PM4PY) .................................................. 16
      1.2.4 Architecture and features ...................................................................... 16
      1.2.5 Object Management .............................................................................. 17
      1.2.6 Algorithms ............................................................................................ 17
   1.3 Process Discovery ............................................................................................. 18
      1.3.1 Alpha Miner ........................................................................................... 18
      1.3.2 Inductive Miner ...................................................................................... 18
      1.3.3 Heuristics Miner .................................................................................... 19
   1.4 Conformance Checking .................................................................................... 19
      1.4.1 Token-based replay ............................................................................... 19
      1.4.2 Alignments ............................................................................................. 20
   1.5 Evaluation ......................................................................................................... 21
      1.5.1 Replay Fitness ......................................................................................... 21
      1.5.2 Precision ................................................................................................. 21
      1.5.3 Generalization ........................................................................................ 22
      1.5.4 Simplicity ............................................................................................... 22

2. **APPLICATION PROGRAMMING INTERFACE** .................................................. 23
   2.1 API Definition .................................................................................................. 23
   2.2 Rest ................................................................................................................... 23
   2.3 Project’s API .................................................................................................... 23
      2.3.1.1 Convert xes files to csv ..................................................................... 26
      2.3.1.2 Convert csv files to xes ..................................................................... 27
      2.3.2.1 View xes file to content .................................................................... 29
      2.3.2.2 View csv file to content .................................................................... 30
      2.3.3.1 View information of xes file .............................................................. 31
      2.3.3.2 View information of csv file .............................................................. 34
      2.3.4.1 Discovery algorithms for xes files ..................................................... 37
      2.3.4.2 Discovery algorithms for csv files ..................................................... 43
      2.3.5.1 Replay results for xes files ............................................................... 51
      2.3.5.2 Replay results for csv files ............................................................... 53
      2.3.6.1 Get alignments for xes files .............................................................. 55
Development of a web-interface and web service API to support process mining techniques

2.3.6.2 Get alignments for csv files.................................................................56

3. WEB INTERFACE........................................................................................................59
   3.1 React Js..........................................................................................................................59
   3.2 React Js and flask..........................................................................................................60
   3.3 Web Pages ......................................................................................................................62
   3.3.1 First Page ....................................................................................................................62
   3.3.2 File menu ....................................................................................................................63
   3.3.3 Discover menu ..........................................................................................................73
   3.3.4 Conformance menu ....................................................................................................77
   3.4 System Walkthrough .....................................................................................................83

4. CONCLUSION......................................................................................................................90

5. APPENDIX ..........................................................................................................................91

6. REFERENCES ....................................................................................................................108
Development of a web-interface and web service API to support process mining techniques

Image Catalogue

Image 2.1: Browser Server Communication ................................................................. 25
Image 2.2: System Architecture .................................................................................. 26
Image 2.3: Xes File Contents ...................................................................................... 29
Image 2.4: Csv File Contents ...................................................................................... 30
Image 2.5: Xes File Statistics ....................................................................................... 32
Image 2.6: Csv File Statistics ...................................................................................... 34
Image 2.7: API Alpha Miner ......................................................................................... 37
Image 2.8: API Inductive Miner ................................................................................... 38
Image 2.9: API heuristic miner .................................................................................... 39
Image 2.10: API csv Alpha ........................................................................................ 44
Image 2.11: API csv Inductive .................................................................................... 45
Image 2.12: API csv Heuristic ..................................................................................... 46
Image 2.13: API replay results .................................................................................... 51
Image 2.14: API csv replay results .............................................................................. 53
Image 2.15: API alignments ........................................................................................ 55
Image 2.16: API csv alignments ................................................................................ 57
Image 3.1: First Page .................................................................................................. 62
Image 3.2: File Upload ................................................................................................ 62
Image 3.3: File Upload Complete ............................................................................... 63
Image 3.4: File Upload State ....................................................................................... 63
Image 3.5: File Menu .................................................................................................. 64
Image 3.6: File upload ................................................................................................ 64
Image 3.7: Convert to xes .......................................................................................... 65
Image 3.8: Wrong csv input ....................................................................................... 65
Image 3.9: Choose Headers ......................................................................................... 66
Image 3.10: No seperator ............................................................................................ 67
Image 3.11: Wrong seperator ..................................................................................... 68
Image 3.12: Convert completed .................................................................................. 69
Image 3.13: Convert to csv ......................................................................................... 69
Development of a web-interface and web service API to support process mining techniques

Image 3.14: Convert to csv completed................................................................. 70
Image 3.15: File Content................................................................................. 70
Image 3.16: View State..................................................................................... 71
Image 3.17: File Info......................................................................................... 72
Image 3.18: File Info State................................................................................ 72
Image 3.19: Choose header for file info......................................................... 73
Image 3.20: Discover Menu............................................................................ 74
Image 3.21: Alpha miner site......................................................................... 75
Image 3.22: Heuristics miner site................................................................. 75
Image 3.23: Inductive miner site................................................................. 76
Image 3.24: Discovery State.......................................................................... 77
Image 3.25: Conformance menu................................................................. 78
Image 3.26: Replay results site.................................................................... 79
Image 3.27: Replay Fitness State................................................................. 80
Image 3.28: Aligments site.......................................................................... 81
Image 3.29: Aligments state.......................................................................... 82
Image 3.30: Running example file upload.................................................... 83
Image 3.31: Running example file upload selected....................................... 83
Image 3.32: Running example file uploaded................................................ 83
Image 3.33: Running example convert to csv............................................. 84
Image 3.34: Running example convert to csv question.............................. 84
Image 3.35: Running example csv converted............................................. 84
Image 3.36: Running example view............................................................. 85
Image 3.37: Running example rows............................................................. 85
Image 3.38: Running example information menu....................................... 85
Image 3.39: Running example information................................................ 86
Image 3.40: Running example discovery menu.......................................... 86
Image 3.41: Running example Alpha Miner............................................... 86
Image 3.42: Running example Heuristics Miner Menu............................... 87
Image 3.43: Running example Heuristics Miner......................................... 87
Development of a web-interface and web service API to support process mining techniques

**Image 3.43**: Running example Heuristics Miner............................................. 87
**Image 3.45**: Running example Inductive Miner................................................ 88
**Image 3.46**: Replay Results Menu.................................................................. 88
**Image 3.47**: Running example algorithm selection for replay.......................... 88
**Image 3.48**: Running example Replay Results..................................................... 88
**Image 3.49**: Running example Alignments menu................................................ 89
**Image 3.50**: Running example algorithm select for alignments............................ 89
**Image 3.51**: Running example Alignments........................................................... 89
Introduction

1.1 Introduction

The field of process mining provides tools and techniques to increase the overall knowledge of a (business) process, by means of analyzing the event data stored during the execution of the process. Process mining received a lot of attention from both academia and industry, which led to the development of several commercial and open-source process mining tools. The majority of these tools supports process discovery, i.e., discovering a process model that accurately describes the process under study, as captured within the analyzed event data. However, process mining also comprises conformance checking, i.e., checking to what degree a given process model is accurately describing event data, and process enhancement, i.e., techniques that enhance process models by projecting interesting information, e.g., case flow and/or performance measures, on top of a model. The support of such types of process mining analysis is typically limited to open-source, academic process mining tools such as the ProM Framework, Apromore, Disco, Celonis and others [1].

The purpose of this master thesis was to develop a system in order to help a user to apply process mining techniques online. There are already systems that serve this need offline. This system is implemented in two levels:

- the level of the API, which helps the user to receive data after data mining techniques so he can view or analyze them in his own way
- the level of the web, at which the user can use it from any device he wants and doesn’t need to have a system on his computer

Both the API and the interface provide the user data such as the events, the start events and the end events and how many times they appear in the file (in absolute value and as a percentage), the transitions and places in order to construct a petri net, conformance checking results and evaluation data (log fitness, precision, generalization, simplicity). Also, the project provides functions to help the users view and convert the file they want to use.

Finally, this master thesis deals with the user’s need to change the petri net that the process mining techniques produce. This thesis website does not provide a static image of a petri net, but an html element of the petri net to help the user zoom in/out and also to give him the ability of drag every transition and place so he can get a clear result of the process discovery.
1.2 Theoretical Background

1.2.1 XES Standard

The XES standard defines a grammar for a tag-based language whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems behaviors by means of event logs and event streams is defined in the XES standard. An XML Schema describing the structure of an XES event log/stream and a XML Schema describing the structure of an extension of such a log/stream are included in this standard. Moreover, a basic collection of so-called XES extension prototypes that provide semantics to certain attributes as recorded in the event log/stream is included in this standard [2].

Several formats have been proposed during the years for the standard storage of event logs in process mining. The IEEE standard is XES, for which different implementations exist in the ProM6 process mining framework. Among noticeable implementations, we can cite XES Lite, that provides a memory-efficient handling of event logs, while supports the XES standard on relational databases, albeit with a performance deficit, and DBXES that use relational databases to support some intermediate calculations. OpyenXES took XES in an open-source Python implementation and the PM4Py Python process mining library followed obtaining a full certification [3].

1.2.2 Event Logs

We assume the existence of an event log where each event refers to a case, an activity, and a point in time. An event log can be seen as a collection of cases. A case can be seen as a trace/sequence of events.

Event data may come from

- a database system (e.g., patient data in a hospital),
- a comma-separated values (CSV) file or spreadsheet,
- a transaction log (e.g., a trading system),
- a business suite/ERP system (SAP, Oracle, etc.),
- a message log (e.g., from IBM middleware),
- an open API providing data from websites or social media and others [4]

Events are listed together with their attributes in an event log. Attributes that are typically listed are the case ID, the time stamps of the start and end times, and other attributes of the event recorded by the IT system. An event log can also be the documentation of several related business processes [5].
1.2.3 Process Mining for Python (PM4PY)

Pm4py provides a process mining software which is easily extendable, allows for algorithmic customization and allows user to easily conduct large scale experiments.

The data science world, both for classic data science and for cutting-edge machine learning research is heavily using Python. Other libraries, albeit with a lower number of features, exist already for the Python language. The bupaR library supports process mining in the statistical language R, that is widely used in data science. The main focal points of the novel PM4Py library are:

- Lowering the barrier for algorithmic development and customization when performing a process mining analysis compared to existing academic tools such as ProM, RAPIdProM and Apromore.
- Allow for the easy integration of process mining algorithms with algorithms from other data science fields, implemented in various state-of-the-art Python packages.
- Create a collaborative eco-system that easily allows researchers and practitioners to share valuable code and results with the process mining world.
- Provide accurate user-support by means of a rich body of documentation on the process mining techniques made available in the library.
- Algorithmic stability by means of rigorous testing. [1]

1.2.4 Architecture and features

In order to maximize the possibility to understand and re-use the code, and to be able to execute large-scale experiments, the following architectural guidelines have been adopted on the development of PM4Py:

- A strict separation between objects, algorithms (Alpha Miner, Inductive Miner, alignments) and visualizations in different packages. In the pm4py.object package, classes to import/export and to store the information related to the objects are provided, along with some utilities to convert objects (e.g. process trees into Petri nets); while in the pm4py.algo package, algorithms to discover, perform conformance checking, enhancement and evaluation are provided. All visualizations of objects are provided in the pm4py.visualization package.
- Most functionality in PM4Py has been realized through factory methods. These factory methods provide a single access point for each algorithm, with a standardized set of input objects, e.g., event data and a parameters object. Consider the factory method of the Alpha Miner. Factory methods allow for the extension of existing algorithms whilst ensuring backward-compatibility. The factory methods typically accept the name of the variant
Development of a web-interface and web service API to support process mining techniques

of the algorithm to use, and some parameters (shared among variants, or variant-specific). [1]

1.2.5 Object management

Within process mining, the main source of data are event data, often referred to as an event log. Such an event log, represents a collection of events, describing what activities have been performed for different instances of the process under study. PM4Py provides support for different types of event data structures:

- Event logs, i.e., representing a list of traces. Each trace, in turn, is a list of events. The events are structured as key-value maps.
- Event Streams representing one list of events (again represented as key-value maps) that are not (yet) organized in cases.

Conversion utilities are provided to convert event data objects from one format to the other. Furthermore, PM4Py supports the use of pandas data frames, which are efficient in case of using larger event data. Other objects currently supported by PM4Py include: heuristic nets, accepting Petri nets, process trees and transition systems. [1]

1.2.6 Algorithms

The PM4Py library provides several mainstream process mining techniques, including:

- Process discovery: Alpha (+) Miner and Inductive Miner.
- Conformance Checking: Token-based replay and alignments.
- Measurement of fitness, precision, generalization and simplicity of process models.
- Filtering based on time-frame, case performance, trace endpoints, trace variants, attributes, and paths.
- Case management: statistics on variants and cases.
- Graphs: case duration, events per time, distribution of a numeric attribute’s values.
- Social Network Analysis: handover of work, working together, subcontracting and similar activities networks.[1]
1.3 Process Discovery

Process Discovery algorithms want to find a suitable process model that describes the order of events/activities that are executed during a process execution. [6]

1.3.1 Alpha Miner

The alpha miner is one of the most known Process Discovery algorithm and is able to find:

- A Petri net model where all the transitions are visible and unique and correspond to classified events (for example, to activities).
- An initial marking that describes the status of the Petri net model when a execution starts.
- A final marking that describes the status of the Petri net model when a execution ends.

Although this algorithm has the following disadvantage:

- Cannot handle loops of length one and length two
- Invisible and duplicated tasks cannot be discovered
- Discovered model might not be sound
- Weak against noise [6]

1.3.2 Inductive Miner

The basic idea of Inductive Miner is about detecting a 'cut' in the log (e.g. sequential cut, parallel cut, concurrent cut and loop cut) and then recur on sublogs, which were found applying the cut, until a base case is found. The Directly-Follows variant avoids the recursion on the sublogs but uses the Directly Follows graph.

Inductive miner models usually make extensive use of hidden transitions, especially for skipping/looping on a portion on the model. Furthermore, each visible transition has a unique label (there are no transitions in the model that share the same label).

Two process models can be derived: Petri Net and Process Tree.

Some advantages of this algorithm are:

- Can handle invisible tasks
- Model is sound
- Most used process mining algorithm [6]
1.3.3 Heuristic Miner

Heuristics Miner is an algorithm that acts on the Directly-Follows Graph, providing way to handle with noise and to find common constructs (dependency between two activities, AND). The output of the Heuristics Miner is a Heuristics Net, so an object that contains the activities and the relationships between them. The Heuristics Net can be then converted into a Petri net.

It is possible to obtain a Heuristic Net and a Petri Net.

To apply the Heuristics Miner to discover a Heuristics Net, it is necessary to import a log. Then, a Heuristic Net can be found. There are also numerous possible parameters that can be inspected by clicking on the following button.

In addition, this algorithm takes frequency into account, detects short loops but does not guarantee a sound model.[6]

1.4 Conformance checking

Conformance checking is a technique to compare a process model with an event log of the same process. The goal is to check if the event log conforms to the model, and, vice versa.

In PM4Py, two fundamental techniques are implemented: token-based replay and alignments. [6]

1.4.1 Token-based replay

Token-based replay matches a trace and a Petri net model, starting from the initial place, in order to discover which transitions are executed and in which places we have remaining or missing tokens for the given process instance. Token-based replay is useful for Conformance Checking: indeed, a trace is fitting according to the model if, during its execution, the transitions can be fired without the need to insert any missing token. If the reaching of the final marking is imposed, then a trace is fitting if it reaches the final marking without any missing or remaining tokens.

In PM4Py there is an implementation of a token that is able to go across hidden transitions (calculating shortest paths between places) and can be used with any Petri net model with unique visible transitions and hidden transitions. When a visible transition needs to be fired and not all places in the preset are provided with the correct number of tokens, starting from the current marking it is checked if for some place there is a sequence of hidden transitions that could be
fired in order to enable the visible transition. The hidden transitions are then fired and a marking that permits to enable the visible transition is reached.

First, the log is loaded. Then, the Alpha Miner is applied in order to discover a Petri net. Eventually, the token-based replay is applied. The output of the token-based replay, stored in the variable `replayed_traces`, contains for each trace of the log:

- **trace_is_fit**: boolean value (True/False) that is true when the trace is according to the model.
- **activated_transitions**: list of transitions activated in the model by the token-based replay.
- **reached_marking**: marking reached at the end of the replay.
- **missing_tokens**: number of missing tokens.
- **consumed_tokens**: number of consumed tokens.
- **remaining_tokens**: number of remaining tokens.
- **produced_tokens**: number of produced tokens.

The token-based replay supports different parameters.

### 1.4.2 Alignments

Alignment-based replay aims to find one of the best alignment between the trace and the model. For each trace, the output of an alignment is a list of couples where the first element is an event (of the trace) or » and the second element is a transition (of the model) or ». For each couple, the following classification could be provided:

- Sync move: the classification of the event corresponds to the transition label; in this case, both the trace and the model advance in the same way during the replay.
- Move on log: for couples where the second element is », it corresponds to a replay move in the trace that is not mimicked in the model. This kind of move is unfit and signal a deviation between the trace and the model.
- Move on model: for couples where the first element is », it corresponds to a replay move in the model that is not mimicked in the trace. For moves on model, we can have the following distinction:
  - Moves on model involving hidden transitions: in this case, even if it is not a sync move, the move is fit.
  - Moves on model not involving hidden transitions: in this case, the move is unfit and signals a deviation between the trace and the model.

First, we have to import the log. Subsequently, we apply the Inductive Miner on the imported log. In addition, we compute the alignments.
With each trace, a dictionary containing among the others the following information is associated:

- **alignment**: contains the alignment (sync moves, moves on log, moves on model)
- **cost**: contains the cost of the alignment according to the provided cost function
- **fitness**: is equal to 1 if the trace is perfectly fitting

Then, a process model is computed, and alignments are also calculated. Besides, the fitness value is calculated and the resulting values are printed.

### 1.5 Evaluation

In PM4Py, it is possible to compare the behavior contained in the log and the behavior contained in the model, in order to see if and how they match. Four different dimensions exist in process mining, including the measurement of replay fitness, the measurement of precision, the measurement of generalization, the measurement of simplicity.[6]

#### 1.5.1 Replay Fitness

The quality dimension of replay fitness describes the fraction of the behavior in the event log that can be replayed by the process model. Several different measures exist for this quality dimension. Some measures consider traces of behavior as whole, checking if the whole trace can be replayed by the process model. Other measures consider the more detailed level of events within a trace and try to get a more fine-grained idea of where the deviations are. Another important difference between existing measures is that some enforce a process model to be in an accepted end state when the whole trace is replayed. Others ignore this and allow the process model to remain in an active state when the trace ends. The most recent and robust technique uses a cost-based alignment between the traces in the event log and the most optimal execution of the process model. This allows for more flexibility and distinction between more and less important activities by changing the costs [7].

#### 1.5.2 Precision

Precision is estimated by confronting model and log behavior: imprecisions between the model and the log (i.e., situations where the model allows more behavior than the one reflected in the log) are detected by juxtaposing behavior allowed by the log and the one allowed by the model. This juxtaposition is done in
Development of a web-interface and web service API to support process mining techniques

terms of an automaton: first, an automaton is built from the alignments. Then, the automaton is enhanced with behavioral information of the model. Finally, the enhanced automaton is used to compute the precision [8].

1.5.3 Generalization

Replay fitness and precision only consider the relationship between the event log and the process model. However, the event log only contains a part of all the possible behavior that is allowed by the system. Generalization therefore should indicate if the process model is not “overfitting” to the behavior seen in the event log and describes the actual system. Another explanation for generalization is the likelihood that the process model is able to describe yet unseen behavior of the observed system. To date only few measures for generalization exist [7].

1.5.4 Simplicity

The simplicity dimension evaluates how simple the process model is to understand for a human. This dimension is therefore not directly related to the observed behavior but can consider the process model solitarily. Since there are different ways to describe the same behavior using different process models, choosing the simplest one is obviously best. This is also expressed by Occam’s razor: “one should not increase, beyond what is necessary, the number of entities required to explain anything”. However, sometimes a complex process model can only be simplified by changing the behavior, hence influencing the other quality dimensions. Several measures exist to measure how simple a process model is, for an overview we refer to. However, research has also shown that size is the main complexity indicator [7].
CHAPTER 2

APPLICATION PROGRAMMING INTERFACE

2.1 API Definition

API stands for Application Programming Interface. A Web API is an application programming interface for the Web. A Browser API can extend the functionality of a web browser. A Server API can extend the functionality of a web server [9].

2.2 REST

The REST architecture was introduced in the year 2000, by Thomas Fielding, and is based on the principles that support the World Wide Web. In summary, according to the REST principles, REST interfaces rely exclusively on Uniform Resource Identifiers (URI) for resource detection and interaction, and usually on the Hypertext Trans-fer Protocol (HTTP) for message transfer. A REST service URI only provides location and name of the resource, which serves as a unique resource identifier. The predefined HTTP verbs are used to define the type of operation that should be performed on the selected resource (e.g., GET to retrieve, DELETE to remove a resource).

Possibly due to HTTP’s features (which fit the REST architecture rather well), long-term presence, and general understandability, REST has become a de facto standard way for offering a service on the Web. Despite this, REST is merely an architectural style, provided without standard specifications. This implies that several decisions have to be made by developers when exposing service APIs, which may result in diverse APIs and, in some cases, in poor design decisions (e.g., using a single HTTP verb for retrieving or deleting a resource) [10].

2.3 Project’s API

For this project an API was developed so the user can get various data such as petri nets, event dictionaries and others on his browser with a GET method and also return this data to a website with a POST method. This master thesis API was developed in python because it is a powerful language for data analysis and provides many ready-made and useful libraries. For the development of this API flask was used (https://flask.palletsprojects.com/en/1.1.x/).
Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools [11].

When we run flask web application with the command python API.py. A local IP address is return. In this case it is http://127.0.0.1:5000/. This IP is our endpoint. We can call this endpoint with two ways POST and GET.

Below there is an example of output when a user starts the API:

* Serving Flask app "api" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with windowsapi reloader
* Debugger is active!
* Debugger PIN: 282-941-890
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Below there is a UML example of Web Interface communication with the API and a diagram for the web Interface.
Development of a web-interface and web service API to support process mining techniques

Image 2.1: Browser Server Communication
Development of a web-interface and web service API to support process mining techniques

And an image with the architecture of the system.

![System Architecture Image](image2.png)

**Image 2.2: System Architecture**

Next, I'll display some examples of this thesis endpoint.

### 2.3.1.1 Convert xes files to csv

The user can convert his uploaded .xes files to .csv, on the server side of this application. The user has to call the endpoint: http://127.0.0.1:5000/convertocsv?filename=(name_of_file).xes.

With the help of Swagger Inspector [13] below there is a definition of the specific endpoint of the API:

```json
get:
  description: Auto generated using Swagger Inspector
  parameters:
    - name: filename
      in: query
      schema:
        type: string
        example: repairexample.xes
  responses:
```

26
Development of a web-interface and web service API to support process mining techniques

'200':
    description: Auto generated using Swagger Inspector
    content:
        application/json:
            schema:
                type: object
                properties: {}
            examples:
                '0':
                    value: |
                        |
                        |
                        |
                        |

2.3.1.2 Convert csv files to xes

The user can convert his uploaded .csv files to xes, on the server side of this application. The user has to call the endpoint:
http://127.0.0.1:5000/convertocsv?filename=(name_of_file).csv&seperator=(csv_column_separator)&caseconcept=(the_column to rename)&conceptname=(the_column to rename)&timestamp=(the_column to rename)&startevent=(the_column to rename)

get:
    description: Auto generated using Swagger Inspector

parameters:
    - name: caseconcept
      in: query
      schema:
        type: string
        example: Case%20ID
    - name: filename
      in: query
      schema:
        type: string
        example: running-example.csv
    - name: startevent
      in: query
Development of a web-interface and web service API to support process mining techniques

```json
  schema:
    type: string
    example: Costs
- name: conceptname
  in: query
  schema:
    type: string
    example: Activity
- name: seperator
  in: query
  schema:
    type: string
    example: ;
- name: timestamp
  in: query
  schema:
    type: string
    example: 'dd-MM-yyyy:HH.mm'

responses:
  '200':
    description: Auto generated using Swagger Inspector
    content:
      application/json:
        schema:
          type: object
          properties: {}
        examples:
          '0':
            value: |
            [  
              {  
                "xesname": "running-example.xes"
              }
            ]
```
2.3.2.1 View xes file content

The user can view the content of his uploaded .xes file. The user has to call the endpoint: http://127.0.0.1:5000/view?filename=(name_of_file).csv.

The following screenshot is an example with the contents of running-example.xes file.

```
<table>
<thead>
<tr>
<th>Event ID:</th>
<th>36654423</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource:</td>
<td>&quot;Rex&quot;</td>
</tr>
<tr>
<td>case:concept:name:</td>
<td>1</td>
</tr>
<tr>
<td>concept:name:</td>
<td>&quot;register request&quot;</td>
</tr>
<tr>
<td>start_event:</td>
<td>50</td>
</tr>
<tr>
<td>time:timestamp:</td>
<td>&quot;20-12-2010:11.02&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event ID:</th>
<th>36654424</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource:</td>
<td>&quot;Sue&quot;</td>
</tr>
<tr>
<td>case:concept:name:</td>
<td>1</td>
</tr>
<tr>
<td>concept:name:</td>
<td>&quot;examine thoroughly&quot;</td>
</tr>
<tr>
<td>start_event:</td>
<td>466</td>
</tr>
<tr>
<td>time:timestamp:</td>
<td>&quot;31-12-2010:10.05&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event ID:</th>
<th>36654425</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource:</td>
<td>&quot;Mike&quot;</td>
</tr>
<tr>
<td>case:concept:name:</td>
<td>1</td>
</tr>
<tr>
<td>concept:name:</td>
<td>&quot;check ticket&quot;</td>
</tr>
<tr>
<td>start_event:</td>
<td>100</td>
</tr>
<tr>
<td>time:timestamp:</td>
<td>&quot;05-01-2011:15.12&quot;</td>
</tr>
</tbody>
</table>
```

Image 2.3: Xes File Contents

get:

description: Auto generated using Swagger Inspector

parameters:
- name: filename
  in: query
  schema:
    type: string
  example: repairexample.xes
Development of a web-interface and web service API to support process mining techniques

2.3.2.2 View csv file content

The user can view the content of his uploaded .csv file. The user has to call the endpoint:


The following screenshot is an example with the contents of running-example.csv file.

Image 2.4: Csv File Contents
get:

description: Auto generated using Swagger Inspector

parameters:
- name: caseconcept
  in: query
  schema:
    type: string
    example: Case%20ID
- name: filename
  in: query
  schema:
    type: string
    example: running-example.csv
- name: startevent
  in: query
  schema:
    type: string
    example: Costs
- name: conceptname
  in: query
  schema:
    type: string
    example: Activity
- name: seperator
  in: query
  schema:
    type: string
    example: ;
- name: timestamp
  in: query
  schema:
    type: string
    example: 'dd-MM-yyyy:HH.mm'

2.3.3.1 View information of xes file

The user can view the content of his uploaded .xes file. The user has to call the endpoint:

The following screenshot is an example with the contents of repair-example.csv file.

Image 2.5: Xes File Statistics

Dictionary key has value all the activities. Activities key has value all the end activities. Start activities key has value all the start activities. Noevents key has value the number of all the events. Notraces key has value the number of traces.

get:
description: Auto generated using Swagger Inspector

parameters:
- name: filename
  in: query
  schema:
    type: string
    example: repairexample.xes

responses:
  '200':
Development of a web-interface and web service API to support process mining techniques

description: Auto generated using Swagger Inspector
content:
application/json:
schema:
type: object
properties: {}
examples:
'0':
  value: |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
  |
Development of a web-interface and web service API to support process mining techniques

2.3.3.2 View information of csv file

The user can view the content of his uploaded .csv file. The user has to call the endpoint:

http://127.0.0.1:5000/csvstatistics?filename=(name_of_file).csv&separator=(csv_column_separator)&caseconcept=(the_column_to_rename)&conceptname=(the_column_to_rename)&timestamp=(the_column_to_rename)&startevent=(the_column_to_rename).

The following screenshot is an example with the contents of repair-example.csv file.

```

dictionary:
  check ticket: 9
  decide: 9
  examine casually: 6
  examine thoroughly: 3
  pay compensation: 3
  register request: 6
  reinitiate request: 3
  reject request: 3

endactivities:
  pay compensation: 3
  reject request: 3
  noevents: 42
  notraces: 6

startactivities:
  register request: 6

tracelist: "(\"[\"concept\":name\"]\")"
```

Image 2.6: Csv File Statistics

Test scenarios

get:
  description: Auto generated using Swagger Inspector
  parameters:
Development of a web-interface and web service API to support process mining techniques

- name: caseconcept
  in: query
  schema:
    type: string
    example: Case%20ID

- name: filename
  in: query
  schema:
    type: string
    example: running-example.csv

- name: startevent
  in: query
  schema:
    type: string
    example: Costs

- name: conceptname
  in: query
  schema:
    type: string
    example: Activity

- name: seperator
  in: query
  schema:
    type: string
    example: ;

- name: timestamp
  in: query
  schema:
    type: string
    example: 'dd-MM-yyyy:HH.mm'

responses:

'200':
  description: Auto generated using Swagger Inspector
  content:
    application/json:
      schema:
        type: object
        properties: {}
        examples:
          '0':

Development of a web-interface and web service API to support process mining techniques

```json
{
  "dictionary": {
    "check ticket": 9,
    "decide": 9,
    "examine casually": 6,
    "examine thoroughly": 3,
    "pay compensation": 3,
    "register request": 6,
    "reinitiate request": 3,
    "reject request": 3
  },
  "endactivities": {
    "pay compensation": 3,
    "reject request": 3
  },
  "noevents": 42,
  "notraces": 6,
  "startactivities": {
    "register request": 6
  },
  "tracelist": "\"[{\"concept:name\"}]\""
}
```
2.3.4.1 Discovery Algorithms for xes files

The web API is using the same endpoint for all three of algorithms (Alpha, Heuristics, Inductive Miner). Although, you should provide different parameters for each algorithm. For the Alpha Miner algorithm, the user has to call the endpoint:

http://127.0.0.1:5000/getimage?filename=(name_of_xesfile)&algorithm=1

```
0:
  evaluation_result: 0.2394858778531464
  generalization: 0.970941777942248
  image: "http://127.0.0.1:8000/static/temp.svg"
  log_fitness: 0.68283694282967
  nets:
    0: "(t)Archive Repair-->p[end"
    1: "(p):start-->t:Register"
    2: "(t):Inform User-->p[end"
    3: "(t):Repair (Complex)->p[end"
    4: "(t):Inform User-->p(['Inform User'], ['Archive Repair'])"
    5: "(t):Test Repair-->p[end"
    6: "(p):(['Inform User'], ['Archive Repair'])-->t:Archive Repair"
  netplaces:
    0: "end"
    1: "(['Inform User'], ['Archive Repair'])"
    2: "start"
  nettransitions:
    0: "Analyze Defect"
    1: "Restart Repair"
    2: "Test Repair"
    3: "Repair (Complex)"
    4: "Repair (Simple)"
    5: "Inform User"
    6: "Register"
    7: "Archive Repair"

similarity: 1
```

Image 2.7: API Alpha Miner
For the Inductive Miner algorithm, the user has to call the endpoint:

http://127.0.0.1:5000/getimage?filename=(name_of_xesfile)&algorithm=2
Development of a web-interface and web service API to support process mining techniques

http://127.0.0.1:5000/getimage?filename=(name_of_xesfile)&algorithm=3

```python
evaluation_result: 0.607361938657466
generalization: 0.523490361033926
image: "http://127.0.0.1:5000/static/temp.svg"
log_fitness: 0.846652368993691
```

Image 2.9: API heuristic miner

The endpoint returns log_fitness, precision, generalization, simplicity for the specific algorithm and log file. Next, it returns a petri net as svg file (the endpoint saves the petri net on server side), netplaces a list of places of petri net,
Development of a web-interface and web service API to support process mining techniques

nettransitions a list of transitions of petrinet and netarcs a list of all the arcs between places and transitions.

**parameters:**
- name: filename
  
in: query
  
schema:
  
type: string
  
example: repairexample.xes
- name: algorithm
  
in: query
  
schema:
  
type: string
  
example: '3'

**responses:**

'200':
  
description: Auto generated using Swagger Inspector
  
content:
  
application/json:
  
schema:
  
type: object
  
properties: {}
  
examples:

'0':
  
value: |
  |
  |
  |
  |
  |
  |
  |

  "evaluation_result": 0.5073613988657465,
  "generalization": 0.9224983618335266,
  "image": "http://127.0.0.1:5000/static/temp.svg",
  "log_fitness": 0.9445528689593501,
  "netarcs": [
    "(t)Register->(p)pre_Analyze Defect",
    "(t)hid_23->(p)pre_Repair (Simple)",
    "(t)hid_42->(p)pre_Test Repair",
    "(t)hid_30->(p)pre_Inform User",
    "(t)hid_35->(p)pre_Repair (Simple)",
    "(p)intplace_Inform User->(t)Restart Repair",
    "(p)intplace_Test Repair->(t)Archive Repair",
    "(p)intplace_Inform User->(t)Archive Repair",
  ]
Development of a web interface and web service API to support process mining techniques

"(p)intplace_Test Repair->(t)Restart Repair",
"(p)pre_Analyze Defect->(t)Analyze Defect",
"(p)source0->(t)Register",
"(p)pre_Repair (Complex)->(t)Repair (Complex)",
"(p)intplace_Analyze Defect->(t)hid_22",
"(p)intplace_Analyze Defect->(t)hid_23",
"(t)hid_22->(p)splace_in_Inform User_0",
"(p)intplace_Inform User->(t)hid_37",
"(p)pre_Test Repair->(t)Test Repair",
"(t)hid_23->(p)splace_in_Inform User_0",
"(p)intplace_Inform User->(t)hid_38",
"(t)hid_38->(p)sink0",
"(t)Test Repair->(p)intplace_Test Repair",
"(t)Archive Repair->(p)sink0",
"(t)Repair (Complex)->(p)intplace_Repair (Complex)",
"(p)intplace_Test Repair->(t)hid_40",
"(p)intplace_Repair (Complex)->(t)hid_26",
"(p)pre_Repair (Simple)->(t)Repair (Simple)",
"(p)intplace_Repair (Complex)->(t)hid_27",
"(p)intplace_Test Repair->(t)hid_42",
"(p)intplace_Repair (Complex)->(t)hid_28",
"(p)intplace_Test Repair->(t)hid_43",
"(p)intplace_Repair (Complex)->(t)hid_29",
"(t)hid_43->(p)sink0",
"(t)hid_22->(p)pre_Repair (Complex)",
"(t)hid_29->(p)sink0",
"(p)intplace_Analyze Defect->(t)hid_21",
"(t)Restart Repair->(p)intplace_Restart Repair",
"(t)Repair (Simple)->(p)intplace_Repair (Simple)",
"(p)intplace_Repair (Simple)->(t)hid_30",
"(p)intplace_Restart Repair->(t)hid_44",
"(p)intplace_Repair (Simple)->(t)hid_31",
"(p)intplace_Restart Repair->(t)hid_45",
"(p)pre_Inform User->(t)Inform User",
"(t)hid_40->(p)pre_Inform User",
"(p)intplace_Repair (Simple)->(t)hid_32",
"(t)Analyze Defect->(p)intplace_Analyze Defect",
"(t)hid_31->(p)pre_Repair (Simple)",
"(t)hid_11->(p)pre_Inform User",
"(t)hid_34->(p)pre_Repair (Complex)"
Development of a web-interface and web service API to support process mining techniques

"(t)Inform User->(p)intplace_Inform User",
"(t)hid_45->(p)pre_Repair (Simple)",
"(p)splace_in_Inform User_0->(t)hid_11",
"(t)hid_37->(p)pre_Test Repair",
"(t)hid_27->(p)pre_Repair (Complex)",
"(p)intplace_Inform User->(t)hid_35",
"(t)hid_28->(p)pre_Test Repair",
"(t)hid_44->(p)pre_Repair (Complex)",
"(p)intplace_Inform User->(t)hid_34",
"(t)hid_21->(p)pre_Analyze Defect",
"(t)hid_32->(p)pre_Test Repair",
"(t)hid_26->(p)pre_Inform User"
],

"netplaces": [ 
  "pre_Repair (Simple)",
  "source0",
  "intplace_Restart Repair",
  "sink0",
  "intplace_Test Repair",
  "pre_Repair (Complex)",
  "pre_Test Repair",
  "pre_Analyze Defect",
  "splace_in_Inform User_0",
  "intplace_Repair (Complex)",
  "intplace_Inform User",
  "intplace_Analyze Defect",
  "pre_Inform User",
  "intplace_Repair (Simple)"
],

"nettransitions": [
  "hid_43",
  "hid_22",
  "hid_29",
  "Register",
  "hid_37",
  "hid_30",
  "hid_44",
  "Analyze Defect",
  "Repair (Complex)",
  "hid_38",
  "hid_36",
  "hid_45",
  "hid_44",
  "hid_37",
  "hid_30",
  "hid_22",
  "Register",
  "hid_29",
  "hid_27",
  "hid_21",
  "hid_11",
  "hid_34",
  "hid_35"
]
Development of a web-interface and web service API to support process mining techniques

"Repair (Simple)",
"Inform User",
"Test Repair",
"hid_31",
"Archive Repair",
"hid_45",
"Restart Repair",
"hid_32",
"hid_11",
"hid_40",
"hid_26",
"hid_23",
"hid_27",
"hid_34",
"hid_21",
"hid_42",
"hid_28",
"hid_35"
}
"simplicity": 0.5384615384615384

2.3.4.2 Discovery Algorithms for csv files

The web API is using the same endpoint for all three of algorithms (Alpha, Heuristics, Inductive Miner). Although, you should provide different parameters for each algorithm. For the Alpha Miner algorithm, the user has to call the endpoint:

http://127.0.0.1:5000/getimagecsv?filename=(name_of_csvfile)&algorithm=1&seperator=(csv_column_separtor)&caseconcept=(the_column to rename)&conceptname=(the_column to rename)&timestamp=(the_column to rename)&startevent=(the_column to rename)
For the Inductive Miner algorithm, the user has to call the endpoint:

http://127.0.0.1:5000/getimagecsv?filename=(name_of_csvfile)&algorithm=2&seperator=(csv_column_separtor)&caseconcept=(the_column_to_rename)&conceptname=(the_column_to_rename)&timestamp=(the_column_to_rename)&startevent=(the_column_to_rename)
Development of a web-interface and web service API to support process mining techniques

Image 2.11: API csv Inductive
Development of a web-interface and web service API to support process mining techniques

For the Heuristic Miner algorithm, the user has to call the endpoint:

http://127.0.0.1:5000/getimagecsv?filename=(name_of_csvfile)&algorithm=3&seperator=(csv_column_separator)&caseconcept=(the_column_to_rename)&conceptname=(the_column_to_rename)&timestamp=(the_column_to_rename)&startevent=(the_column_to_rename)

Image 2.1: API csv Heuristic

Image 2.12: API csv Heuristic
Development of a web-interface and web service API to support process mining techniques

get:
description: Auto generated using Swagger Inspector

parameters:
- name: caseconcept
  in: query
  schema:
    type: string
    example: Case%20ID
- name: filename
  in: query
  schema:
    type: string
    example: running-example.csv
- name: startevent
  in: query
  schema:
    type: string
    example: Costs
- name: conceptname
  in: query
  schema:
    type: string
    example: Activity
- name: seperator
  in: query
  schema:
    type: string
    example: ;
- name: timestamp
  in: query
  schema:
    type: string
    example: 'dd-MM-yyyy:HH:mm'
- name: algorithm
  in: query
  schema:
    type: string
    example: '3'

responses:
Development of a web-interface and web service API to support process mining techniques

'200':
description: Auto generated using Swagger Inspector
content:
application/json:
schema:
type: object
properties: {}
examples:
'0':
value: |
[
  
  "evaluation_result": 0.75,
  "generalization": 0.46345210039759677,
  "image": "http://127.0.0.1:5000/static/temp.svg",
  "log_fitness": 0.9688983855650523,
  "netarcs": [
    "(t)reinitiate request->(p)intplace_reinitiate request",
    "(p)source0->(t)register request",
    "(t)register request->(p)intplace_register request",
    "(t)examine casually->(p)splace_in_decide_examine casually_0",
    "(p)intplace_decide->(t)reject request",
    "(t)reject request->(p)sink0",
    "(p)pre_examine thoroughly->(t)examine thoroughly",
    "(p)intplace_decide->(t)pay compensation",
    "(p)splace_in_decide_examine casually_0->(t)hid_7",
    "(t)pay compensation->(p)sink0",
    "(t)hid_10->(p)pre_examine casually",
    "(p)intplace_register request->(t)hid_9",
    "(p)splace_in_check ticket_0->(t)hid_3",
    "(p)pre_examine casually->(t)examine casually",
    "(t)hid_10->(p)splace_in_check ticket_0",
    "(t)decide->(p)intplace_decide",
    "(t)hid_9->(p)pre_examine thoroughly",
    "(t)hid_4->(p)pre_check ticket",
    "(t)examine thoroughly->(p)pre_decide",
    "(t)check ticket->(p)splace_in_decide_check ticket_0",
    "(t)hid_16->(p)pre_examine thoroughly",
    "(t)hid_17->(p)splace_in_check ticket_1",
    "(p)intplace_decide->(t)reinitiate request",
  ]
Development of a web-interface and web service API to support process mining techniques

"(p)intplace_reinitiate request->(t)hid_17",
"(t)hid_17->(p)pre_examine casually",
"(p)splace_in_check ticket_1->(t)hid_4",
"(t)hid_3->(p)pre_check ticket",
"(p)intplace_reinitiate request->(t)hid_16",
"(p)splace_in_decide_check ticket_0->(t)hid_7",
"(t)hid_9->(p)splace_in_check ticket_0",
"(p)intplace_register request->(t)hid_10",
"(t)hid_16->(p)splace_in_check ticket_1",
"(p)pre_check ticket->(t)check ticket",
"(p)pre_decide->(t)decide",
"(t)hid_7->(p)pre_decide"
],

"netplaces": [
"splace_in_decide_check ticket_0",
"pre_check ticket",
"pre_examine casually",
"intplace_register request",
"splace_in_decide_examine casually_0",
"source0",
"pre_examine thoroughly",
"intplace_reinitiate request",
"pre_decide",
"intplace_decide",
"splace_in_check ticket_1",
"splace_in_check ticket_0",
"sink0"
],

"nettransitions": [
"hid_7",
"hid_10",
"hid_9",
"hid_3",
"hid_16",
"register request",
"hid_4",
"reject request",
"pay compensation",
"hid_17",
"decide",
"(p)intplace_reinitiate request->(t)hid_17",
"(t)hid_17->(p)pre_examine casually",
"(p)splace_in_check ticket_1->(t)hid_4",
"(t)hid_3->(p)pre_check ticket",
"(p)intplace_reinitiate request->(t)hid_16",
"(p)splace_in_decide_check ticket_0->(t)hid_7",
"(t)hid_9->(p)splace_in_check ticket_0",
"(p)intplace_register request->(t)hid_10",
"(t)hid_16->(p)splace_in_check ticket_1",
"(p)pre_check ticket->(t)check ticket",
"(p)pre_decide->(t)decide",
"(t)hid_7->(p)pre_decide"}
Development of a web-interface and web service API to support process mining techniques

"reinitiate request",
"examine casually",
"examine thoroughly",
"check ticket"
],
"simplicity": 0.6666666666666666
}
Development of a web-interface and web service API to support process mining techniques

2.3.5.1 Replay results for xes files

For the replay results, the user has to call the endpoint:

http://127.0.0.1:5000/getreplayresults?filename=(name_of_csvfile)&algorithm=(exactly_the_same_numbers_as_discover_process)

Image 2.13: API replay results

get:

description: Auto generated using Swagger Inspector

parameters:

- name: filename
Development of a web-interface and web service API to support process mining techniques

in: query
schema:
  type: string
  example: repairexample.xes
- name: algorithm
  in: query
  schema:
    type: string
    example: '3'
Development of a web-interface and web service API to support process mining techniques

2.3.5.1 Replay results for csv files

For the replay results for csv files, the user has to call the endpoint:

http://127.0.0.1:5000/getreplayresultscsv?filename=(name_of_csvfile)&sitealgo=(exactly_the_same_numbers_as_discover_process)&separator=(csv_column_separator)&caseconcept=(the_column_to_rename)&conceptname=(the_column_to_rename)&timestamp=(the_column_to_rename)&startevent=(the_column_to_rename)

---

get:

description: Auto generated using Swagger Inspector

parameters:

- name: caseconcept
  in: query
  schema:
    type: string
Development of a web-interface and web service API to support process mining techniques

- name: caseID
  in: query
  schema:
  type: string
  example: Case%20ID

- name: filename
  in: query
  schema:
  type: string
  example: running-example.csv

- name: startevent
  in: query
  schema:
  type: string
  example: Costs

- name: conceptname
  in: query
  schema:
  type: string
  example: Activity

- name: separator
  in: query
  schema:
  type: string
  example: ;

- name: sitealgo
  in: query
  schema:
  type: string
  example: '3'

- name: timestamp
  in: query
  schema:
  type: string
  example: 'dd-MM-yyyy:HH.mm'
2.3.6.1 Get alignments for xes files

For the alignments, the user has to call the endpoint:

http://127.0.0.1:5000/getalignments?filename=(name_of_csvfile)&algorithm=(exactly_the_same_numbers_as_discover_process)

get:

description: Auto generated using Swagger Inspector

parameters:
- name: filename
  in: query

schema:
Development of a web-interface and web service API to support process mining techniques

```
type: string
example: repairexample.xes

- name: algorithm
  in: query
  schema:
    type: string
    example: '3'
```

2.3.6.2 Get alignments for csv files

For the alignments of csv files, the user has to call the endpoint:

```
http://127.0.0.1:5000/getalignmentscsv?filename=(name_of_csvfile)&sitealgo=(exactly_the_same_numbers_as_discover
process)&seperator=(csv_column_separtor)&caseconcept=(the_column_to
rename)&conceptname=(the_column_to_rename)&timestamp=(the_column_to
rename)&startevent=(the_column_to_rename)
```
Development of a web-interface and web service API to support process mining techniques

Image 2.16: API csv alignments
Development of a web-interface and web service API to support process mining techniques

get:

description: Auto generated using Swagger Inspector

parameters:
- name: caseconcept
  in: query
  schema:
    type: string
    example: Case%20ID
- name: filename
  in: query
  schema:
    type: string
    example: running-example.csv
- name: startevent
  in: query
  schema:
    type: string
    example: Costs
- name: conceptname
  in: query
  schema:
    type: string
    example: Activity
- name: seperator
  in: query
  schema:
    type: string
    example: ;
- name: sitealgo
  in: query
  schema:
    type: string
    example: '3'
- name: timestamp
  in: query
  schema:
    type: string
    example: 'dd-MM-yyyy:HH.mm'
CHAPTER 3

WEB INTERFACE

3.1 React Js

React JS is a JavaScript library for building user interfaces.

The reasons why React Js was chosen to build the interface are:

- React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug.

- Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, the user can easily pass rich data through the app and keep state out of the DOM.

- The user can develop new features in React without rewriting existing code [12].

- It is easy to make POST calls to the API and retrieve returning JSON objects.

- Thanks to react flow library (https://reactflow.dev/) the developer can create a petri net, in which the user can zoom in/out and also move around its elements, places and transitions.

- Also, thanks to react loader spinner (https://www.npmjs.com/package/react-loader-spinner) the developer can create a customized loader to let the user know that the API is processing data.
3.2 React Js and Flask

The API has two responsibilities. The first is to respond to post or get request made to this endpoint and the second is to response web interface requests.

Next, we need to install the flask-cors [14] with the pip install flask-cors command. Flask-cors is looking for requests for a different application.

Functions that are available on server side:

- **savefile**: `@app.route('/savefile', methods=['POST'])`.
  It is used only from web interface so the user can upload his xes or csv files and then save them to server.

- **headers**: `@app.route('/headers', methods=['POST'])`.
  It is used only from web interface to return the headers of files.

- **convertocsv**: `@app.route('/convertocsv', methods=['POST', 'GET'])`.
  The user can convert a selected xes file to csv.

- **convertoxes**: `@app.route('/convertoxes', methods=['POST', 'GET'])`.
  The user can convert a selected csv file to xes.

- **view**: `@app.route('/view', methods=['POST', 'GET'])`.
  API returns the header and each row of xes file.

- **viewcsv**: `@app.route('/viewcsv', methods=['POST', 'GET'])`.
  API returns the header and each row of csv file.

- **statistics**: `@app.route('/statistics', methods=['POST', 'GET'])`.
  API returns statistics of xes file, such as number of traces, events, start events, end events.

- **csvstatistics**: `@app.route('/csvstatistics', methods=['POST', 'GET'])`.
  API returns statistics of csv file, such as number of traces, events, start events, end events.
Development of a web-interface and web service API to support process mining techniques

- **getimage**: @app.route('/getimage', methods=['POST', 'GET']).
  API returns list of places, transitions and edges of petri net importing a xes file and also evaluation results (log_fitness, precision, simplicity, generalization).

- **getimagecsv**: @app.route('/getimagecsv', methods=['POST', 'GET']).
  API returns list of places, transitions and edges of petri net importing a csv file and also evaluation results (log_fitness, precision, simplicity, generalization).

- **getreplayresults**: @app.route('/getreplayresults', methods=['POST', 'GET]).
  API returns an array with data such as: Trace is fit, Trace fitness, Activated Transitions, Reached Marking, Enabled transitions in marking, Transitions with problems, Missing Tokens, Consumed tokens, Remaining Tokens και Produced tokens for the selected xes file.

- **getreplayresultscsv**: @app.route('/getreplayresultscsv', methods=['POST', 'GET']).
  API returns an array with data such as: Trace is fit, Trace fitness, Activated Transitions, Reached Marking, Enabled transitions in marking, Transitions with problems, Missing Tokens, Consumed tokens, Remaining Tokens και Produced tokens for the selected csv file.

- **getalignments**: @app.route('/getalignments', methods=['POST', 'GET']).
  API returns a lot of arrays with two rows in purpose of conformance checking of imported xes file.

- **getalignmentscsv**: @app.route('/getalignmentscsv', methods=['POST', 'GET']).
  API returns a lot of arrays with two rows in purpose of conformance checking of imported csv file.
Development of a web-interface and web service API to support process mining techniques

3.3 Web pages

3.3.1 First Page

Xes and csv files are stored at server. There is not a session management and when you press refresh the application redirects you to this page.

When user hovers File from menu there is only one option upload. The reason is that he has not upload any file so he can do any other action. When the user presses the upload option the next page is showed up:

Image 3.2: File Upload
Development of a web-interface and web service API to support process mining techniques

The user can upload any xes or csv file by pressing one click on the box and then choose the file from windows browser or with drag and drop. In case the file is csv or xes the image will change to this:

![Image 3.3: File Upload Complete](image)

When the user presses the complete button, web interface redirects to the main screen.

![Image 3.4: File Upload State](image)
3.3.2 File Menu

Under the file menu there is the option upload so the user can upload more than one file.

Under file menu there are more option if the user has already uploaded at least one file. The user should select a file from the left menu and then select one of the options. If the user has not selected any file and presses an option then a blank screen will appear on the right until the user selects one file from the left menu. The first option is convert to xes.
Development of a web-interface and web service API to support process mining techniques

Image 3.7: Convert to xes

In case user does not select a csv file:

Image 3.8: Wrong csv input

In the opposite view a webpage with four dropdowns will appear so the user can declare which csv column will be: concept:name, concept:name, start_event and time:timestamp.
Development of a web-interface and web service API to support process mining techniques

Image 3.9: Choose Headers
Development of a web-interface and web service API to support process mining techniques

In this case the user has to provide a separator. If he leaves the input blank then the dropdowns will have null value:

<table>
<thead>
<tr>
<th>File</th>
<th>Discover Model</th>
<th>Statistics</th>
<th>Conformance</th>
</tr>
</thead>
</table>

Files
Choose sepeartor
running-example.csv

Choose an element for case:concept:name
null

Choose an element for concept:name
null

Choose an element for start_event
null

Choose an element for time:timestamp
null

Are you sure you want to convert your file

[YES] [NO]

Image 3.10: No seperator

In case user gives wrong separator, the next screen will appear:
Development of a web-interface and web service API to support process mining techniques

Image 3.1: Wrong separator

Otherwise, when the user presses Yes an extra file with the type .xes will appear on the left menu (Files).
Development of a web-interface and web service API to support process mining techniques

Similar to the last option, convert to csv does not allow the user to convert a csv file and when the user selects an xes file the following web page appears:

Image 3.12: Convert Completed

Image 3.13: Convert to csv
Development of a web-interface and web service API to support process mining techniques

In case the user presses yes, on the left files list the converted file appears with the repairExample.csv.

![Image 3.1: Convert to csv completed](image)

When the user selects the view option, he can either choose a xes or csv file and an html table will appear with the content of file.

![Image 3.15: File Content](image)

The web interface reads the whole table with post from python flask and when the user changes the number to results dropdown to 10, then the html table shows the first 10 rows, when the user changes the number to results dropdown
Development of a web-interface and web service API to support process mining techniques

to 50, then the html table shows the first 10 rows and when the user changes to the dropdown to all then html table shows all the rows. Until the response is returned from the API react-loader-spinner is used (https://www.npmjs.com/package/react-loader-spinner).

About the Info option, the user can retrieve the next data: number of traces, number of events, file structure, an event array and how many times each event appears in the file, a table with start events and one with end events and how many traces start or end with those events.

Image 3.16: View State
Development of a web-interface and web service API to support process mining techniques

![Image 3.17: File Info](image)

![Image 3.18: File Info State](image)
Development of a web-interface and web service API to support process mining techniques

In case the user selects a csv file then the web interface will first appear a select header webpage.

3.3.3 Discover Menu

When the user hovers on Discover Model, then he can choose one of the discover algorithms as an option: alpha miner, inductive miner, heuristics miner.
In case the user chooses alpha miner then a webpage with a table will appear which show the Log Fitness, Precision, Generalization and Simplicity evaluation for the specific file. Also, petri net of the algorithm will appear under the table. The petri net is developed with react flow, an open-source library which allow the web user to drag and drop any transition and place of the graph.
Development of a web-interface and web service API to support process mining techniques

For Heuristic Miner:

Image 3.21: Alpha miner site

Image 3.22: Heuristics miner site
Development of a web-interface and web service API to support process mining techniques

And Inductive Miner:

<table>
<thead>
<tr>
<th>Files</th>
<th>Discover Model</th>
<th>Statistics</th>
<th>Conformance</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogFitness</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>LogPrecision</td>
<td>0.730064197530864</td>
<td>0.5445854851849998</td>
<td>0.7398444444444444</td>
</tr>
<tr>
<td>Generalization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simplicity</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Image 3.23: Inductive miner site

In case the user chooses a csv file and then selects one of the above three options, then the choose heeder screen will appear and then the web interface will display the webpage with the table and the petri net.
3.3.4 Conformance Menu

When the user hovers the menu Conformance then he can select the option replay results or alignments
In case the user selects replay results, then a dropdown appears with the three discovery models (alpha miner, Inductive Miner, Heuristics Miner) as option. If the file is csv, then the other four dropdowns from header select will appear in this screen as well.

![Image 3.2: Conformance menu]

When the user selects one of the three algorithms then a table appears with the following data:

- Trace is fit
- Trace fitness
- Activated Transitions
- Reached Marking
- Enabled transitions in marking
- Transitions with problems
- Missing Tokens
- Consumed Tokens
- Remaining Tokens
- Produced Tokens
Development of a web-interface and web service API to support process mining techniques

Image 3.26: Replay results site
Development of a web-interface and web service API to support process mining techniques.

Image 3.27: Replay Fitness State
Development of a web-interface and web service API to support process mining techniques

In case the user selects the other option (alignments) then the same dropdown appears with the discovery models as option. When the user selects one algorithm then a lot of tables appear, each one with two rows so the user can tell if the trace is fit on the process model.

![Image 3.28: Alignments site]
Development of a web-interface and web service API to support process mining techniques

Image 3.29: Alignment state
Development of a web-interface and web service API to support process mining techniques

3.4 System Walkthrough

For this Walkthrough I used the file running-example.xes which can be found in the following link https://github.com/pm4py/pm4py-ws/blob/master/files/event_logs/running-example.xes

- File upload

![Image 3.30: Running example file upload](image-url)

![Image 3.31: Running example file upload selected](image-url)

![Image 3.32: Running example file uploaded](image-url)
Development of a web-interface and web service API to support process mining techniques

- Convert to csv

Image 3.33: Running example convert to csv

Image 3.34: Running example convert to csv question

Image 3.35: Running example csv converted
Development of a web-interface and web service API to support process mining techniques

- View

Image 3.36: Running example view

<table>
<thead>
<tr>
<th>Activity</th>
<th>Costs</th>
<th>Resource</th>
<th>caseconceptname</th>
<th>conceptname</th>
<th>eventname</th>
<th>eventinstanceid</th>
<th>timetamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>register request</td>
<td>50</td>
<td>Note</td>
<td>register request</td>
<td>register request</td>
<td>Note</td>
<td>2019-12-30T13:25:00.000Z</td>
<td></td>
</tr>
<tr>
<td>examine casualty</td>
<td>400</td>
<td>Mile</td>
<td>examine casualty</td>
<td>examine casualty</td>
<td>Mile</td>
<td>2019-12-30T04:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>check input</td>
<td>100</td>
<td>Ellen</td>
<td>check input</td>
<td>check input</td>
<td>Ellen</td>
<td>2019-12-30T13:25:00.000Z</td>
<td></td>
</tr>
<tr>
<td>decide</td>
<td>200</td>
<td>Sara</td>
<td>decide</td>
<td>decide</td>
<td>Sara</td>
<td>2011-05-30T03:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>reimburse request</td>
<td>200</td>
<td>Sara</td>
<td>reimburse request</td>
<td>reimburse request</td>
<td>Sara</td>
<td>2011-05-30T03:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>examine thoroughly</td>
<td>400</td>
<td>Ivan</td>
<td>examine thoroughly</td>
<td>examine thoroughly</td>
<td>Ivan</td>
<td>2011-05-30T03:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>check input</td>
<td>100</td>
<td>Milan</td>
<td>check input</td>
<td>check input</td>
<td>Milan</td>
<td>2011-05-30T03:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>decide</td>
<td>200</td>
<td>Sara</td>
<td>decide</td>
<td>decide</td>
<td>Sara</td>
<td>2011-05-30T03:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>pay compensation</td>
<td>200</td>
<td>Ellen</td>
<td>pay compensation</td>
<td>pay compensation</td>
<td>Ellen</td>
<td>2011-05-30T03:00:00.000Z</td>
<td></td>
</tr>
<tr>
<td>register request</td>
<td>50</td>
<td>Note</td>
<td>register request</td>
<td>register request</td>
<td>Note</td>
<td>2019-12-30T13:25:00.000Z</td>
<td></td>
</tr>
</tbody>
</table>

Image 3.37: Running example rows

- Info

Image 3.38: Running example information menu
Development of a web-interface and web service API to support process mining techniques

- **Alpha Miner**

  ![Image 3.40: Running example discovery menu](Image 3.40)

  ![Image 3.41: Running example Alpha Miner](Image 3.41)

---

Image 3.39: Running example information

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Events</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>check ticket</td>
<td>9</td>
<td>21.429%</td>
</tr>
<tr>
<td>decide</td>
<td>9</td>
<td>21.429%</td>
</tr>
<tr>
<td>examine casually</td>
<td>6</td>
<td>14.286%</td>
</tr>
<tr>
<td>examine thoroughly</td>
<td>3</td>
<td>7.143%</td>
</tr>
<tr>
<td>pay compensation</td>
<td>3</td>
<td>7.143%</td>
</tr>
<tr>
<td>register request</td>
<td>6</td>
<td>14.286%</td>
</tr>
<tr>
<td>reinitialize request</td>
<td>3</td>
<td>7.143%</td>
</tr>
<tr>
<td>reject request</td>
<td>3</td>
<td>7.143%</td>
</tr>
</tbody>
</table>

Start Events:

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Events</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>register request</td>
<td>6</td>
<td>14.286%</td>
</tr>
</tbody>
</table>

End Events:

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Events</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>pay compensation</td>
<td>3</td>
<td>7.143%</td>
</tr>
<tr>
<td>reject request</td>
<td>3</td>
<td>7.143%</td>
</tr>
</tbody>
</table>
Development of a web-interface and web service API to support process mining techniques

- **Heuristics Miner**

![Running example Heuristics Miner Menu](image3_42.png)

![Running example Heuristics Miner](image3_43.png)

- **Inductive Miner**

![Running example Inductive Miner Menu](image3_44.png)
Development of a web-interface and web service API to support process mining techniques

Image 3.45: Running example Inductive Miner

- Replay Results

Image 3.46: Replay Results Menu

Image 3.47: Running example algorithm selection for replay

Image 3.48: Running example Replay Results
Development of a web-interface and web service API to support process mining techniques

- Alignments

Image 3.49: Running example Alignments menu

Image 3.50: Running example algorithm select for alignments

Image 3.51: Running example Alignments
CHAPTER 4

CONCLUSION

This master thesis deals with the development of a web service application (web API) for process mining. As I mentioned in chapter two the API is developed with python and pm4py library. The user can easily call the end point and get response for many functions. He can get json information of the file with one endpoint (number of traces, number of events, events of file, start and end events) instead of calling each pm4py function and he can also get percentage of each event of file. He can also get xes files content in a json objects. Instead of getting the petri net’s image from p4mpy he can get as json objects all the information needed to draw his own petri net and instead of using different functions to get each algorithms evaluation he now gets four json objects.

This master thesis also deals with the development of a web interface to represent the json data from API’s response. The interface is created for the user that don’t want to hit an endpoint to get Json results but want visualized responses of their actions. The interface is user-friendly, easily accessible and easy to navigate. The only thing the user has to do is to upload a file for process mining and then he views the contents of his file as table rows, event information as table rows, evaluation for a specific algorithm as table rows, a petri net that is not static but he can even drag places and transitions on screen and conformance results in table rows.

In addition, because the project is agile, future recommendations can easily be implemented either on API or interface. If in the future it is decided to create a new web interface, the API can be reused. One suggestion is to do conformance checking using a standard petri net and choosing a log file to run on top of it. A way to do this, is to make every edge between the nodes animated for (example change the color of the edge) when a token passed by, so the user can understand the path of each token. Another future recommendation is to create a login form for the web interface and an authenticate function in the API. Finally, colleagues can easily implement their own process mining functions to the API and create a web page to show their results.
Appendix

Appendix A) Upload / Save file

Server Side:

```python
file = request.files['file']  # the file react sends
filename = file.filename  # name of react sends
splitedfile = filename.split('.')
if(splitedfile[len(splitedfile)-1]) == 'xes' or
(splitedfile[len(splitedfile)-1]) == 'csv':
    # we only save on server side xes csv files
    if not os.path.exists('filesfolder/'):  
        os.makedirs('filesfolder/')
    if path.exists('filesfolder/' + filename):
        # overwrite file if exists
        os.remove('filesfolder/' + filename)
    file.save('filesfolder/' + filename)  # save file
APIresults = [{'filename': str(filename)}]
response = jsonify(APIresults)
response.headers.set('Access-Control-Allow-Origin', '*')
return response;
```

Client Side:

```javascript
StatechangeFileDone = event => {
    if(this.file.filename!==null){
        this.sendData();
        if (this.file.filename.includes('xes')){
            // value of xes files in dictionary is true else false
            this.userfiles.allfiles[this.file.filename]=true
        }else{
            this.userfiles.allfiles[this.file.filename]=false
        }
        this.setState({ fileforupload: false});
        // change state when file is uploaded
        const data = new FormData();
        data.append('file', this.file.selectedfile);
        data.append('filename', this.file.filename);
        fetch('http://127.0.0.1:5000/savefile', {
            method: 'POST',
            body: data,
        })
        this.file.filename=null;
        this.file.selectedfile=null;
    }
}```
Appendix B) Convert Csv File

Server Side:

```python
if request.method == 'POST':
    csvinput = request.form.get('filename')
    case_concept_name = str(request.form.get('caseconcept'))
    time_timestamp = str(request.form.get('timestamp'))
    concept_name = str(request.form.get('conceptname'))
    start_event = str(request.form.get('startevent'))
    seperator = str(request.form.get('seperator'))
if request.method == 'GET':
    if (request.args.get('filename') is None) or
       (request.args.get('seperator') is None) or
       (request.args.get('caseconcept') is None) or
       (request.args.get('timestamp') is None) or
       (request.args.get('conceptname') is None) or
       (request.args.get('startevent') is None):
        return badrequest();
    csvinput = str(request.args.get('filename'))
    case_concept_name = str(request.args.get('caseconcept'))
    time_timestamp = str(request.args.get('timestamp'))
    concept_name = str(request.args.get('conceptname'))
    start_event = str(request.args.get('startevent'))
    seperator = str(request.args.get('seperator'))
if ".csv" not in csvinput:
        return badrequest();
log = ""
dataframe = None
start = csvinput.replace('.csv', '')
if path.exists('filesfolder/'+csvinput):
    log_file_path = 'filesfolder/'+csvinput
    log = pd.read_csv(log_file_path, sep=seperator)
dataframe=pm4py.convert_to_dataframe(log)
dataframe.rename(columns={case_concept_name: 'case:concept:name'}, inplace=True)
dataframe.rename(columns={time_timestamp: 'time:timestamp'}, inplace=True)
dataframe.rename(columns={concept_name: 'concept:name'}, inplace=True)
dataframe.rename(columns={start_event: 'start_event'}, inplace=True)
log=pm4py.convert_to_event_log(dataframe)
if path.exists('filesfolder/'+start + '.xes'):
        os.remove('filesfolder/'+start + '.xes')
        pm4py.write_xes(log,'filesfolder/'+ start+'.xes')
APIresults = [{'xesname':str(start + '.xes')}]  
response = jsonify(APIresults)
response.headers.set('Access-Control-Allow-Origin', '*')
return response;
else:
    return filenotfound();
```

Client Side:

```javascript
converttostate = () => {
    const data = new FormData();
```
data.append('filename',this.props.data);  //send filename to API
var caseconcept = document.getElementById('caseconcept').value;

//get values of dropdown
var start_event = document.getElementById('start_event').value;
var timestamp = document.getElementById('timestamp').value;
var conceptname = document.getElementById('conceptname').value;
data.append('caseconcept',caseconcept);
data.append('startevent',start_event);
data.append('timestamp',timestamp);
data.append('conceptname',conceptname);
data.append('seperator',this.props.value.seperator);
fetch('http://127.0.0.1:5000/convertoxes', {
    method: 'POST',
    body: data,
}).then(response => (response.json()))
.then(data => {
    if (data[0].error===undefined){
        this.props.parentCallback(data[0].xesname);
        var url =  window.location.href;
        url = url.replace("toxes", "mainmenu");
        window.location.href = url;
    //redirect to main menu after conversion
    }
});

Appendix C) Convert Xes File

Server Side:

if request.method == 'POST':
xes = request.form.get('filename')
if request.method == 'GET':
    if request.args.get('filename') is None:
        return badrequest();
xes = request.args.get('filename')
if ".xes" not in xes:  #file is not xes
    return badrequest();
if xes is not None:
    start = xes.replace('.xes', '')
    if path.exists('filesfolder/'+xes): #if it exists on server
        log = pm4py.read_xes('filesfolder/'+xes)    #get log
        dataframe = pm4py.convert_to_dataframe(log)
        #convert it to dataframe
        if path.exists('filesfolder/'+start + '.csv'):
            #overwrite if exists
            os.remove('filesfolder/'+start + '.csv')
dataframe.to_csv('filesfolder/'+start + '.csv',index=False,na_rep='false')  #convert to csv
        APIresults = [{'csvname':str(start + '.csv')}]  #convert to csv
        response = jsonify(APIresults)
        response.headers.set('Access-Control-Allow-Origin', '*')
        return response
    else:
        return filenotfound();

Client Side:

converttostate = () => {
    const data = new FormData();
data.append('filename',this.props.data);
Development of a web-interface and web service API to support process mining techniques

//send filename to API to convert it
fetch('http://127.0.0.1:5000/convertocsv', {
  method: 'POST',
  body: data,
}).then(response => (response.json()))
.then(data => {
  if (data[0].error===undefined){
    this.props.parentCallback(data[0].csvname);
    var url =  window.location.href;
    url = url.replace("tocsv", "mainmenu");
    window.location.href = url; //redirect to mainmenu after the file conversion
  }
});

Appendix D) View File

Server Side:

if request.method == 'POST':
xes = request.form.get('filename')
if request.method == 'GET':
xes = request.args.get('filename')
if (request.args.get('filename') is None) :
  return badrequest();
if ".xes" not in xes: #if file is not xes
  return badrequest();
if not path.exists('filesfolder/'+xes):
  #if file does not exist on server
  return filenotfound();
splited = xes.split(".")
log = pm4py.read_xes('filesfolder/'+xes)  #get log from file
dataframe = pm4py.convert_to_dataframe(log)
#convert to dataframe
dataframedict =
dataframe.to_json(orient="index",date_format='iso')
parsed = json.loads(dataframedict)
if splited[len(splited)-1] == 'xes':
  apiresults = [
    {'view':parsed
     }
  ]
  response = jsonify(apiresults)
  response.headers.set('Access-Control-Allow-Origin', '*')
  response.headers.set('cache-control', 'public, max-age=0')
  return response;

Client Side:

StateImage = () => {
  const data = new FormData();
  data.append('filename',this.props.data);
  fetch('http://127.0.0.1:5000/view', {
    method: 'POST',
    body: data,
Development of a web-interface and web service API to support process mining techniques

```javascript
function builddf(dictionary, resultsno) {
    var header = [];
    var body = [];
    var tempinside = [];
    const htmlhead = [];
    const dataframe = [];
    const thead = 'stat_';
    var key = '';
    var headersaresok = false;
    var counter = 0;

    for (var i in dictionary) {
        if (resultsno !== -1) {
            if (counter === resultsno) {
                break;
            }
        }
        if (headersaresok === false) {
            // headers of file
            for (const [key] of Object.entries(dictionary[i])) {
                header.push(<td key={i}>{key}</td>);
            }
            headersaresok = true;
        }

        for (var j in dictionary[i]) {
            // content of file
            key = 'td' + i + '_' + j;
            if (dictionary[i][j] !== null) {
                tempinside.push(<td key={key}>{dictionary[i][j].toString()}</td>);
            } else {
                tempinside.push(<td key={key}></td>);
            }
            counter++;
            key = 'tr' + i;
            body.push(<tr key={key}>{tempinside}</tr>);
        }
        htmlhead.push(<thead key={thead}><tr key='0'>{header}</tr></thead>)
    }
    const tablekey = 'table_';
    dataframe.push(<table key={tablekey} className='resultstable'>{htmlhead}<tbody>{body}</tbody></table>);
    return dataframe;
}
```
Development of a web-interface and web service API to support process mining techniques

Appendix E) File Statistics

Server Side:

def firstassignment(log):
    dataframe = None
    stream = converter.apply(log, variant=converter.Variants.TO_EVENT_STREAM)
    dataframe = pm4py.convert_to_dataframe(log)

    lenlog = len(log)  #length of log
    lenevent = len(stream)  #number of events
    trace_list = []

    for trace in log:
        trace_list.append(str(list(trace.attributes.keys())))
    unique_trace_list = set(trace_list)  #trace list

    dictionary = dict()
    array = 1

    dataframe = dataframe.sort_values('concept:name')

    last_value = None
    for event in dataframe['concept:name']:
        if last_value is None:
            last_value = event
        else:
            if not last_value == event:
                dictionary[last_value] = array;
                array = 1
                last_value = event
            else:
                array = array + 1
                dictionary[last_value] = array;  #all file events

    end_activities = pm4py.get_end_activities(log)  #end events
    start_activities = pm4py.get_start_activities(log)  #start events
    return lenlog, lenevent, unique_trace_list, start_activities, end_activities, dictionary;

Client Side:

Infobuild(){
    const data = new FormData();
    data.append('filename', this.props.data);
    fetch('http://127.0.0.1:5000/statistics', {
        method: 'POST',
        body: data,
    }).then(response => (response.json()))
    .then(data => {

        if (data[0].error === undefined){
            this.statistics.notraces = data[0].notraces;
            this.statistics.noevents = data[0].noevents;
            this.statistics.tracelist = data[0].tracelist;
            this.statistics.events = data[0].dictionary;
    })}.	hen(response => (response.json()))
.then(data => {
Development of a web-interface and web service API to support process mining techniques

```javascript
this.statistics.startactivities=data[0].startactivities;
this.statistics.endactivities=data[0].endactivities;
this.setState({done:true}); //when API returns results
}
})

function buildevent(dictionary,noevents,number){
    const eventhtml=[];
    const htmlhead=[];
    const htmlbody=[];
    const thead = 'stat_')+number;
    htmlhead.push(<thead key={thead}><tr
key='${number}?${0}'>
<td>Name</td><td>Number of Events</td><td>Percentage</td></tr></thead>)
for (var i in dictionary) {
    const trkey='tr_')+number+'_'+i;
    const trkey2 = trkey+'_2';
    const trkey3 = trkey+'_3';
    htmlbody.push(<tr key={trkey}>
Number of Events</tr></tbody></table>)
}
}

function buildactivities(dictionary,notraces,number){
    const eventhtml=[];
    const htmlhead=[];
    const htmlbody=[];
    const thead = 'stat_')+number;
    htmlhead.push(<thead key={thead}><tr
key='${number}?${0}'>
<td>Name</td><td>Number of Traces</td><td>Percentage</td></tr></thead>)
for (var i in dictionary) {
    const trkey='tr_')+number+'_'+i;
    const trkey2 = trkey+'_2';
    const trkey3 = trkey+'_3';
    htmlbody.push(<tr key={trkey}>
Number of Traces</tr></tbody></table>)
}

```
Appendix F) Discovery Algorithms

Server Side:

```python
if request.method == 'POST':
    xes = request.form.get('filename')
    sitealgo = request.form.get('sitealgo')
if request.method == 'GET':
    xes = request.args.get('filename')
    sitealgo = request.args.get('algorithm')
if (request.args.get('filename')  is None) or (request.args.get('algorithm') is None):
    return badrequest();
if not path.exists('filesfolder/'+xes):
    return filenotfound();
splited = xes.split(".")
log = pm4py.read_xes('filesfolder/'+xes)  #get log of xes file
if splited[len(splited)-1] == 'xes':
    if int(sitealgo) == 1:
        #pm4py discover_algo functions for the three algorithms
        net,initial_marking,final_marking =
        discover_algo.discover_petri_net_alpha(log)
    elif int(sitealgo) == 2:
        net,initial_marking,final_marking =
        discover_algo.discover_petri_net_inductive(log)
    elif int(sitealgo) == 3:
        net,initial_marking,final_marking =
        discover_algo.discover_petri_net_heuristics(log)
    fitnesses= pm4py.evaluate_fitness_tbr(log, net, initial_marking, final_marking)
    evaluationresult = pm4py.evaluate_precision_tbr(log, net, initial_marking, final_marking)
    evaluation=evaluation_factory.apply(log, net, initial_marking, final_marking)  #evaluation object
    netplaces=list(net.places)  #petrinet places
    nettransitions=list(net.transitions) #petrinet transitions
netarcs=list(net.arcs)  #petrinet arcs
placeslist = []
transitionlist =[]
arcslist = []
for eachplaces in netplaces:
    placeslist.append(str(eachplaces))
for eachtransition in nettransitions:
    transitionlist.append(str(eachtransition))
for eacharc in netarcs:
    arcslist.append(str(eacharc))
file_path="static/temp.svg"
savesvgfromalgo.save_vis_petri_net(net, initial_marking, final_marking,file_path)
apiresults = [{
    'image':http://127.0.0.1:5000/static/temp.svg',
    'log_fitness':evaluation['fitness']]["log_fitness'],
    'evaluation_result':evaluation['precision'],
```
Development of a web-interface and web service API to support process mining techniques

'generalization':evaluation['generalization'],
'simplicity':evaluation['simplicity'],
'netplaces':placeslist,
'nettransitions':transitionlist,
'netarcs':arcslist
}

response = jsonify(APIresults)
response.headers.set('Access-Control-Allow-Origin', '*')
response.headers.set('cache-control', 'public, max-age=0')
return response;

Client Side:

StateImage = () => {
  const data = new FormData();
  data.append('filename', this.props.data);
  data.append('sitealgo', '1'); //the same endpoint different algorithm
  fetch('http://127.0.0.1:5000/getimage', {
    method: 'POST',
    body: data,
  }).then(response => (response.json()))
    .then(data => {
      if (data[0].error===undefined){
        this.image.nettransitions=data[0].nettransitions;
        this.image.netplaces=data[0].netplaces;
        this.image.netarcs=data[0].netarcs;
        this.image.image=data[0].image;
        this.image.imageHash= Date.now();
        this.setState({ image:
          true,fitness:data[0].log_fitness,precision:data[0].evaluation_result,generalization:data[0].generalization,simplicity:data[0].simplicity});
      }
    });
}

function removestrings(str){
  var newstr=str.replace('(p)','','');
  newstr=newstr.replace('(t)','','');
  newstr=newstr.trim();
  return newstr;
}

function buildminer(trans,places,arcs){
  var idstring=1;
  var ycount=100;
  var xcount=0;
  var elements=[];
  var trans_sort_list = trans.sort();
  var transdictionary = {};
  var moved = {};
  transdictionary['start'] = 0;
  transdictionary['end'] = -1;
  for (var x in trans_sort_list) //Drawing the transitions into graph
    {
Development of a web-interface and web service API to support process mining techniques

transdictionary[trans_sort_list[x]] = idstring;
moved[trans_sort_list[x]]=false;

    elements.push({ id: idstring.toString(),data: { label:
trans_sort_list[x] },style:{width: '100px',position: { x:xcount,
y:ycount }});

    ycount+=100;
    idstring++;
}

xcount+=200;
ycount=0;
var places_sort_list = places.sort();
for ( x in places_sort_list) //Drawing the places into graph
{
    if(places_sort_list[x].includes('start')){
        elements.push({id: transdictionary['start'].toString(),type: 'input',data: { label: 'start' },position: { x: 0, y: 0 },draggable: false,style:{width: '50px',height: '50px','border-radius': '50%', 'padding-top': '17px','border-color':'black','background-color': 'green'} });
    }
    else if(places_sort_list[x].includes('end')){
    }
    else{
        transdictionary[places_sort_list[x]] = idstring;
        elements.push({id: idstring.toString(),type: 'input',data: { label: ' ' },position: { x: xcount, y: ycount },draggable: true,style:{width: '50px',height: '50px','border-radius': '50%', 'padding-top': '17px','border-color':'black','background-color': 'white'} });
        idstring++;
    }
    moved[places_sort_list[x]]=false;
    ycount+=100;
}

xcount+=200;
elements.push({id: transdictionary['end'].toString(),type: 'input',data: { label: 'end' },position: { x: 950, y: 0 },draggable: false,style:{width: '50px',height: '50px','border-radius': '50%','padding-top': '17px','border-color': 'black','background-color': 'orange'} });

var arcs_sort_list = arcs.sort();
for ( x in arcs_sort_list){
    var left=arcs_sort_list[x].split('->')[0];
    var right=arcs_sort_list[x].split('->')[1];

    elements.push({ id: left.split(',')[0]+'~'+right, source: transdictionary[removestrings(left)], target: transdictionary[removestrings(right)],arrowHeadType:'arrowclosed' });
}

var rightelement=[];
Development of a web-interface and web service API to support process mining techniques

```javascript
for ( x in arcs_sort_list){ //Drawing the arcs into graph
    left=arcs_sort_list[x].split('->')[0];
    right=arcs_sort_list[x].split('->')[1];
    if(removestrings(left).includes('start')){
        rightelemt.push(removestrings(right));
    }
}
xcount=200;
ycount=0;
for ( x in rightelemt){
    for (var ele in elements) {
        if (parseInt(elements[ele].id)===parseInt(transdictionary[rightelemt[x]])){
            elements[ele].position={ x: xcount, y: ycount};
            ycount+=100;
        }
    }
    ycount=0;
xcount+=200;
if((rightelemt.length>0)){
    for (var y in rightelemt){
        if (!rightelemt[y].includes('end')){
            elements=recursive(arcs_sort_list,elements,rightelemt[y],ycount,xcount,transdictionary,moved);  //recursive function to change elements position from start to end
        }
    }
    return elements;
}
}
```

```
function recursive(arcs_sort_list,elements,rightelemt,ycount,xcount,transdictionary,moved){
    var newrightelemt=[];
    for (var x in arcs_sort_list){
        var left=removestrings(arcs_sort_list[x].split('->')[0]);
        var right=removestrings(arcs_sort_list[x].split('->')[1]);
        if (left===right){
            if (moved[right]===false){
                newrightelemt.push((right));
            }
        }
    }
    for ( x in newrightelemt){
        for (var ele in elements) {
            if (parseInt(elements[ele].id)===parseInt(transdictionary[newrightelemt[x]])){
                elements[ele].position={ x: xcount, y: ycount};
                moved[newrightelemt[x]]=true;
                ycount+=100;
            }
        }
    }
}
```
Development of a web-interface and web service API to support process mining techniques

```javascript
var tempyposition=0;
if (newrightelement.length>0){
  for (var y in newrightelement){
    if (!newrightelement[y].includes('end')){
      for (ele in elements){
        if (parseInt(elements[ele].id)==parseInt(transdictionary[newrightelement[y]])){
          tempyposition=parseInt(elements[ele].position.y);
        }
      }
    }
  }
  elements=recursive(arcs_sort_list,elements,newrightelement[y],tempyposition,xcount,transdictionary,moved);
  return elements;
}

<div>
  <table>
    <tr>
      <td>Log Fitness</td>
      <td>{this.state.fitness}</td>
    </tr>
    <tr>
      <td>Log Precision</td>
      <td>{this.state.precision}</td>
    </tr>
    <tr>
      <td>Generalization</td>
      <td>{this.state.generalization}</td>
    </tr>
    <tr>
      <td>Simplicity</td>
      <td>{this.state.simplicity}</td>
    </tr>
  </table>
</div>
```
Appendix G) Get Replay Results

Server Side:

```python
try:
    if request.method == 'POST':
        xes = request.form.get('filename')
        sitealgo = request.form.get('sitealgo')
    if request.method == 'GET':
        xes = request.args.get('filename')
        sitealgo = request.args.get('algorithm')
    if (request.args.get('filename') is None) or (request.args.get('algorithm') is None):
        return badrequest();
    if not path.exists('filesfolder/'+xes):
        return filenotfound();
    splited = xes.split(".")
    log = pm4py.read_xes('filesfolder/'+xes)
    #get log from xes file
    if splited[len(splited)-1] == 'xes':
        #get net from specific algorithm
        if int(sitealgo) == 1:
            net,initial_marking,final_marking =
            discover_algo.discover_petri_net_alpha(log)
        elif int(sitealgo) == 2:
            net,initial_marking,final_marking =
            discover_algo.discover_petri_net_inductive(log)
        elif int(sitealgo) == 3:
            net,initial_marking,final_marking =
            discover_algo.discover_petri_net_heuristics(log)
        confnbr =
        pm4py.conformance_tbr(log,net,initial_marking,final_marking)
        #Pm4py conformance
        returndict=[]
        for each in confnbr:
            tempactivate=str(each['activated_transitions']).replace('[', '').replace(']', '')
            tempactivate=tempactivate.replace('"', '')
            tempreached_marking=str(each['reached_marking']).replace('[', '').replace(']', '')
            tempreached_marking=tempreached_marking.replace('"', '')
            temptransitions_with_problems=str(each['transitions_with_problems']).replace('[', '').replace(']', '')
            temptransitions_with_problems=temptransitions_with_problems.replace('"', '')
            tempdict = {
                'trace_is_fit': each['trace_is_fit'],}
Development of a web-interface and web service API to support process mining techniques

```
'trace_fitness':
  'activated_transitions':
  'reached_marking':
  'enabled_transitions_in_marking':
  str(Each['enabled_transitions_in_marking']),
  'transitions_with_problems':temptransitions_with_problems,
  'missing_tokens':
  'consumed_tokens':
  'remaining_tokens':
  'produced_tokens':Each['produced_tokens']
}
returndict.append(tempdict)
apiresults = [{ 'dictionary': returndict }]
response = jsonify(apiresults)
response.headers.set('Access-Control-Allow-Origin', '*')
response.headers.set('cache-control', 'public,max-age=0')
return response;

Client Side:

StateImage = () => {
  const data = new FormData();
  data.append('filename', this.props.data);
  data.append('sitealgo', this.state.selectedalgo);
  fetch('http://127.0.0.1:5000/getreplayresults', {
    method: 'POST',
    body: data,
  }).then(response => (response.json()))
    .then(data => {
      if (data[0].error===undefined){
        this.setState({ image: true, dictionary: data[0].dictionary });
      }
    });
}

function buildEvent(dictionary){  ///converts API response into table
  var obj = dictionary;
  const eventhtml=[];
  const htmlhead=[];
  const htmlbody=[];
  const thead = 'stat';
  htmlhead.push(<thead key={thead}><tr key={0}><td>Trace is fit</td><td> Trace fitness </td><td> Activated Transitions </td><td> Reached Marking </td><td> Enabled transitions in marking </td><td> Transitions with problems </td><td> Missing Tokens </td><td> Consumed Tokens </td><td> Remaining Tokens </td><td> Produced Tokens </td></tr></thead>);
```
Development of a web-interface and web service API to support process mining techniques

for ( var i=0;i<obj.length;i++) {
    const trkey='tr_ '+i;
    const trkey2 = trkey+'_2';
    const trkey3 = trkey+'_3';
    const trkey4 = trkey+'_4';
    const trkey5 = trkey+'_5';
    const trkey7 = trkey+'_7';
    const trkey8 = trkey+'_8';
    const trkey9 = trkey+'_9';
    const trkey10 = trkey+'_10';
    const trkey11 = trkey+'_11';
    htmlbody.push(<tr key={i}><td key={trkey}>{obj[i].trace_is_fit.toString()}</td><td key={trkey2}>{obj[i].trace_fitness} </td><td key={trkey3}>{obj[i].activated_transitions} </td><td key={trkey4}>{obj[i].reached_marking}</td><td key={trkey5}>{obj[i].enabled_transitions_in_marking}</td><td key={trkey7}>{obj[i].transitions_with_problems}</td><td key={trkey8}>{obj[i].missing_tokens}</td><td key={trkey9}>{obj[i].consumed_tokens}</td><td key={trkey10}>{obj[i].remaining_tokens}</td><td key={trkey11}>{obj[i].produced_tokens}</td></tr>)
}
const tablekey='table';
eventhtml.push(<table key={tablekey}className='resultstable'>{htmlhead}<tbody>{htmlbody}</tbody></table>)
return eventhtml;

Appendix H) Get Alignments

Server Side:

if request.method == 'POST':
    xes = request.form.get('filename')
sitealgo = request.form.get('sitealgo')
if request.method == 'GET':
    xes = request.args.get('filename')
sitealgo = request.args.get('algorithm')
if (request.args.get('filename') is None) or (request.args.get('algorithm') is None):
    return badrequest();
if not path.exists('filesfolder/'+xes):
    return filenotfound();
splited = xes.split(".")
log = pm4py.read_xes('filesfolder/'+xes)  #get log from xes
if splited[len(splited)-1] == 'xes':
    if int(sitealgo) == 1:  #get net from specific algorithm
        net,initial_marking,final_marking = discover_algo.discover_petri_net_alpha(log)
elif int(sitealgo) == 2:
        net,initial_marking,final_marking = discover_algo.discover_petri_net_inductive(log)
elif int(sitealgo) == 3:
        net,initial_marking,final_marking = discover_algo.discover_petri_net_heuristics(log)
Development of a web-interface and web service API to support process mining techniques

```
alignments = pm4py.conformance_alignments(log, net, initial_marking, final_marking)
# pretty_print_alignments(alignments)
for each in alignments:
    tempdict = {'alignment': each['alignment']}
    returndict.append(tempdict)
apiresults = [{'dictionary': returndict}]
response = jsonify(apiresults)
response.headers.set('Access-Control-Allow-Origin', '*')
response.headers.set('cache-control', 'public,max-age=0')
return response;

Client Side:

StateImage = () => {
    const data = new FormData();
    data.append('filename', this.props.data);
    data.append('sitealgo', this.state.selectedalgo);
    fetch('http://127.0.0.1:5000/getalignments', {
        method: 'POST',
        body: data,
    }).then(response => (response.json()))
    .then(data => {
        if (data[0].error === undefined) {
            this.setState({ image: true, dictionary: data[0].dictionary });
        }
    });
}

function buildevent(dictionary) {
    var obj = dictionary; // gets response from API
    var returnhtml = [];
    for (var i = 0; i < obj.length; i++) {
        returnhtml.push(<div className="dropdown">{buildsinglealignemnt(obj[i].alignment, i)}</div>)
    } // creates a single table for every object of dictionary

    return returnhtml;
}

function buildsinglealignemnt(step_list, number) {
    var trace_steps = [];
    var model_steps = [];
    var max_label_length = 0;
    var rethtml = [];
    var rethtml2 = [];
    var dividerhtml = [];
    var step = [];
    var splitstep = step_list.toString().split(',');
```
var i,j;
for (i=0;i<splitstep.length;i+=2){
    step.push(splitstep[i]+','+splitstep[i+1]);
}
for (i=0;i<step.length;i++){
    var insidesplitstep=step[i].toString().split('');
    trace_steps.push(" "+ insidesplitstep[0].toString() + " ");
    model_steps.push(" "+ insidesplitstep[1].toString() + " ");
    if ((insidesplitstep[0].length) > max_label_length){
        max_label_length = insidesplitstep[0].toString().length;
    }
    if ( insidesplitstep[1].toString().length > max_label_length){
        max_label_length = insidesplitstep[1].toString().length;
    }
}
for (i=0;i<trace_steps.length;i++){
    if  (trace_steps[i].toString().length - 2 < max_label_length){
        var step_length = trace_steps[i].toString().length - 2;
        var spaces_to_add = max_label_length - step_length;
        for (j=0;j<spaces_to_add;j++){
            if (j % 2 === 0){
                trace_steps[i] = trace_steps[i] + " ";
            }
            else{
                trace_steps[i] = " " + trace_steps[i];
            }
        }
    }
    key++; 
    rethtml.push(<td key={key}>{trace_steps[i]}</td>);
}
dividerhtml.push(<tr>{rethtml}</tr>);
for (i=0;i<model_steps.length;i++){
    if ((model_steps[i].length - 2) < max_label_length){
        var step_lengthmodel = model_steps[i].length - 2;
        var spaces_to_addmodel = max_label_length - step_lengthmodel;
        for (j=0;j<spaces_to_addmodel;j++){
            if (j % 2 === 0){
                model_steps[i] = model_steps[i] + " ";
            }
            else{
                model_steps[i] = " " + model_steps[i];
            }
        }
    }
    key++; 
    rethtml2.push(<td key={key}>{model_steps[i]}</td>);
}
dividerhtml.push(<tr>{rethtml2}</tr>)
var finalhtml=[];
finalhtml.push(<table>{dividerhtml}</table>);
return finalhtml;}
REFERENCES


[4] Wil van der Aals, Event Logs What kind of data does process mining require?


