Software Requirements Specification

for

Dotted Chart Visualizer

Version 1.0 approved

Prepared by Elisabeth Buttkus, Ariane Chu, Samir Majeri, Pieter Patonedi, Victor Férnandez

Chair of Process Mining and Data Science, RWTH

09.5.2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

Our product is a Python based Dotted Chart Visualizer for Process Mining. The user is able to load an even log file in either the XES or CSV format into the app, and display selected process data in the dotted chart visual style. Furthermore, the user will be able to interactively modify certain aspects of the visual, like changing the range of the displayed data, select colors and switching between different attributes of the event data that he chooses to display. Also a legend is provided to indicate relevant information about the process data that is shown in the chart. Both the legend and the dotted chart visual can be downloaded as a jpg or png file by the user. This way, we provide an intuitive, but powerful tool for the analysis of event logs that is essential and needed in the field of Process Mining.

1.2 Intended Audience and Reading Suggestions

The final product will be open-source and available for public researchers and therefore accessible for researchers in the field of Process Mining worldwide.

1.3 Product Scope

Process mining is an important practice in the fields of data science and business process management. The results can be used to solve problems or optimize performance of process-driven systems and is therefore highly valued in companies and institutions. Companies aiming to identify and model process activities require a comprehensive tool that caters to this task and that allows to create an expressive and accurate model of the activities in such a system. The first step in this process is visualizing the information available which is necessary during the process discovery phase.
The process information needed for this task is extracted from event logs. Our finished tool will be available as a web application. It supports the import of event logs in both the CSV and XES format which are the most important formats in which event data is available. The data can then be visualized as a dotted chart and customized by the user for the individual purpose. Each of the created visualizations can be saved by the user and downloaded to their device for further use.

1.4 References

In the following we provide a list of the referenced framework, libraries and tools:

Django UI framework using pm4py: *https://github.com/madhubs08/UIFramework_pm4py*
ProM: *http://www.promtools.org/doku.php?id=prom69*
Pm4py: *https://pm4py.fit.fraunhofer.de/documentation#discovery*

2. Overall Description

2.1 Product Perspective

One of the tools for process mining that is currently being used by researchers around the globe is ProM. ProM is an open-source java based framework that among other functionalities provides a dotted chart visualization of event log data. ProM is provided as a desktop application and its functionality has proven to be highly dependent on the operating system of the device. The ProM application is not a small app, many packages have to be downloaded to ensure functionality. Many of these tools might not be needed by the user who only wishes to plot the data for Process Disovery. Our dotted chart visualizer therefore aims to offer an alternative visualization tool that is more flexible and can be easily integrated into pre-existing python based process mining applications, such as the Tool for Process Mining created by PADS, RWTH Aachen.

2.2 Product Functions

- Import CSV/XES files
  - Converting one type of file into the other
- Provide dotted chart visualization
  - x-y-coordinate system needs to be created (value range adjusted to possible values)
  - Points need to be drawn into the coordinate system (x axis equals date/time)
  - Points are assigned a color/shape based on their type (8 shape maximum in ProM)
  - User can change the attribute displayed on the x- and y-axis
  - User can change the color and shape attribute
  - User can zoom and adjust the scaling of the x-axis if time:timestamp is selected
- Provide legend for the chart
  - Name the value type for x- and y-axis and add values along the axis
  - Legend for the different colors/shapes of points
- Allow to save images of the different visualizations
  - Save to website
  - Download

The use case diagram to the right illustrates the functions of the our product from the user's perspective.

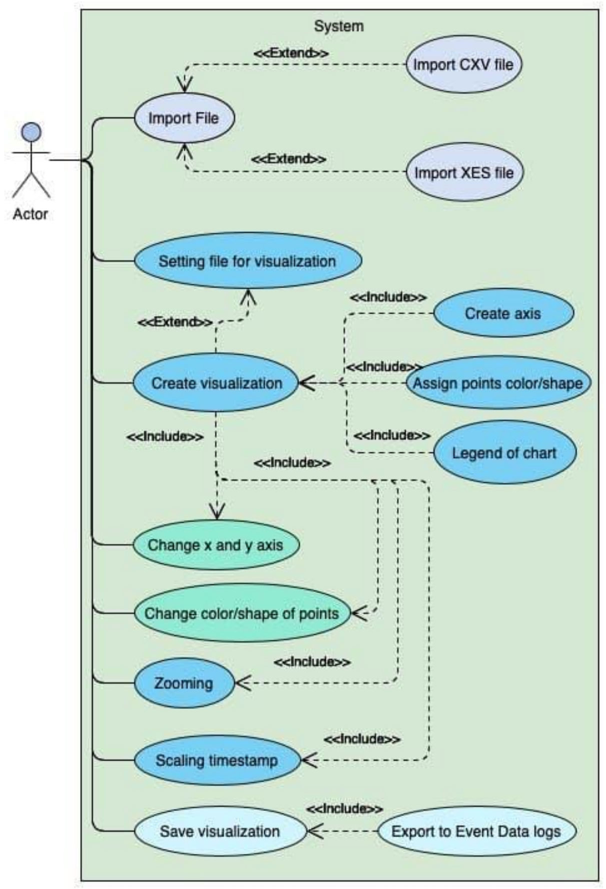

Figure 1: Use Case Diagram for the Product Functions

2.3 User Classes and Characteristics

Our intended tool is relevant for all researchers in the field of Process Mining, especially in the academic field and the area of teaching. The advantage of our product is that it is a stand-alone app with a clear purpose, the visualization of event data. This restriction to one specific goal makes it lightweight, intuitive and easy to use. Therefore, the tool is perfect for teachers who want to demonstrate process analysis techniques in class, as well as scholars who are instructed to learn about Process Mining without having to download tools, that are bigger in size and filled with more elaborate or specific functions that are not needed and make the app less intuitive and accesible. Overall, the intended app is useful for all researchers in the field of Process Mining who look for a web-based app that is specialized on the process visualization aspect.

2.4 Operating Environment

The app is projected to run on all modern operating systems, namely Windows, Linux and macOS. The final project will be available as a Docker software encapsulating all needed libraries and packages. The app is a web application used inside of a web browser and employs Javascript to enable to interactivity with the user. Therefore, Javascript needs to be enabled in the user's web browser to make sure the app runs properly.

2.5 Design and Implementation Constraints

Django was chosen as a framework for the intended app which makes it, by design, a Client/Server application that is run inside a web browser by the user. This implies the use of web based languages like HTML, CSS and Javascript for the structuring, visual design and interactivity of the app as design and implementation constraints.

2.6 User Documentation

A documentation of the software will be provided, as well as a manual how to launch and use the app. Since the use of the app is controlled by the user-interface that is provided within the app, like buttons and knobs to achieve functionality, the user is not dependent on understanding the back-end of the software and will be enabled to start and use the app without online help or tutorials.

2.7 Assumptions and Dependencies

Our app is intended to work on all modern operating systems. This is achieved by making the final product available as a dockerized package that contains all dependencies. Nevertheless, the functionality of the app depends on the availability of libraries and packages that were used in the program code to build the project, namely, Django 3.2, PM4Py 2.2.4 and Python Pandas 1.2.4.. For the visualization we will use CanvasJS Chart 3.2.16. The functionality of the app will depend on the availability of these components. Since, we only use well-known, established libraries in our project, this is not anticipated to be a crucial factor.
It is assumed that only files of type CSV or XES will be uploaded to the Event Data page.

3. External Interface Requirements

3.1 User Interfaces

The dotted chart visualizer will be available within the pm4py-UI framework through localhost:8000/dcv/

3.2 Software Interfaces

The dotted chart visualizer will be provided as an app within the pm4py-UI framework, but otherwise it can be used independently.

4. System Features

In the following, the system features will be explained. We start with the user's perspective to give a detailed high-level overview over the requirements regarding the system features. In the second part, we will give an insight from the system's perspective going into more technical details regarding feature requirements.

4.1 User Perspective

In this section, the system features from a user's perspective are described along with detailed system requirements.

4.1.1 Importing log files

This feature is of high priority as the user needs to be able to work with their own event log files. Importing the log files entails choosing the desired file from the devices file system and uploading it into the framework's database.

On the GUI the user can click on the "Datei auswählen" button and is presented with a file navigator. Using the navigator, the user can choose their desired file. By clicking "Upload EventLog" uploading the XES or CSV file to the database. The file name then appears in the overview and management section below, where the user can delete files and choose a file as input for tool of choice. When setting a file as input basic information about the log file is displayed.

This is already provided by the pm4py-UI, for details please see References.

4.1.2 Basic visualization as dotted chart with default attributes

This feature is of high priority as the visualization of an event log is the key feature of our product. By applying this feature, the user can initiate the visualization.

Once the user has chosen a file to visualize (see 4.1.1- REQ-5) by clicking the "Dotted Chart Visualizer" button on the sidebar, the user is presented a basic dotted chart based on default settings for the x- and y-axes. The basic chart maps the two minimal attributes concept:name (event-level) and time:timestamp against each other. The data is displayed as round dots of uniform color. For a first impression, see the mock UI in (Fig. 2). Furthermore, the sequence chart below (Fig. 3) illustrates the activity sequence.

REQ-1: Provide “Dotted Chart Visualizer” on sidebar
REQ-2: Upon navigating to “Dotted Chart Visualizer” present default dotted chart of chosen event log
REQ-3: Provide drop down menus for configuration of x-axis, y-axis, dot colors and dot shapes
REQ-4: x-axis and y-axis are set to default attributes case:ID and event:concept:name respectively

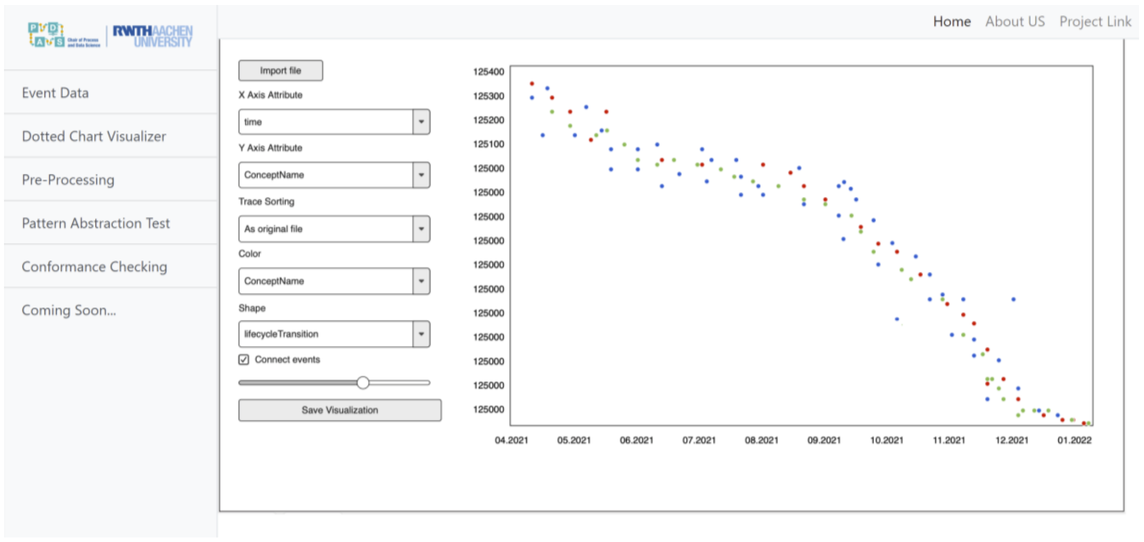

Figure 2: Mock UI of the Intgrated Dotted Chart Visualizer Application

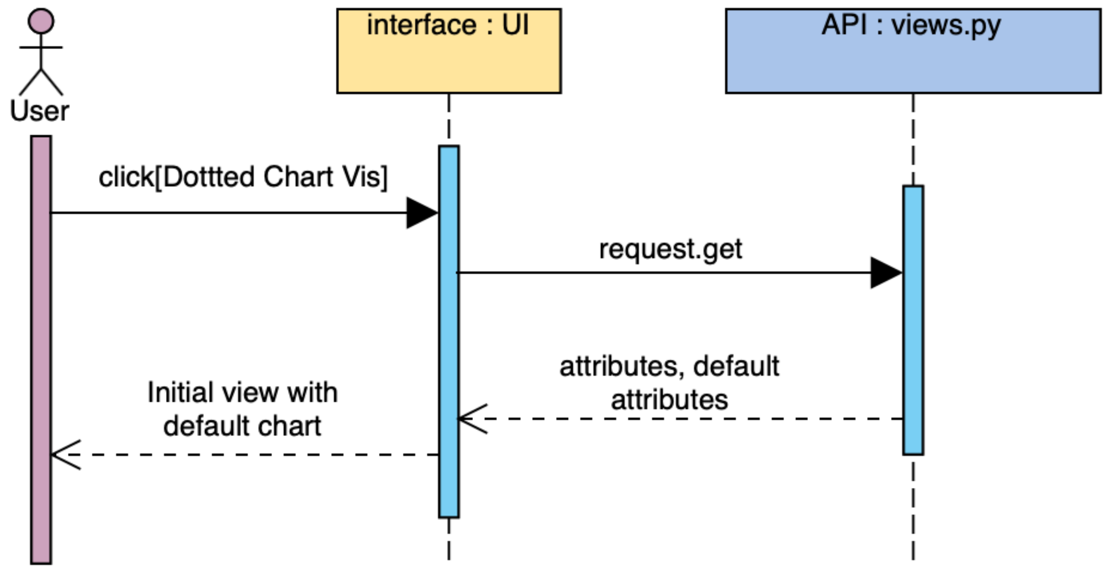

Figure 3: Sequence Diagram for the Default Visualization

4.1.3 Manipulation of the dotted chart

This feature is of medium priority by comparison. It allows the user to create a “4D”-visualization of the event log data by introducing dot colors and geometric shapes. Each color and shape corresponds to an attribute value. For a better insights, a zooming function will be provided as well as an option to further adjust the x-axis if time:timestamp is selected.

The user is offered four drop down menus, one of each axis, one for the colors and one for the shapes. Using the menu, the user can choose which attribute they want to visualize at the same time. After choosing the attribute, the dotted chart is updated and a legend labelling the colors and shapes is created or updated. This action sequence is illustrated by the sequence diagram on the next page (Fig. 4).

REQ-5: Upon selecting from drop down menu, immediately update/display chart
REQ-6: If color and shapes are set, display corresponding legend
REQ-7: The user can zoom in and out of the graph, the axes are adjusted accordingly
REQ-8: time:timestamp: The user can choose to adjust the scale to days, months or years

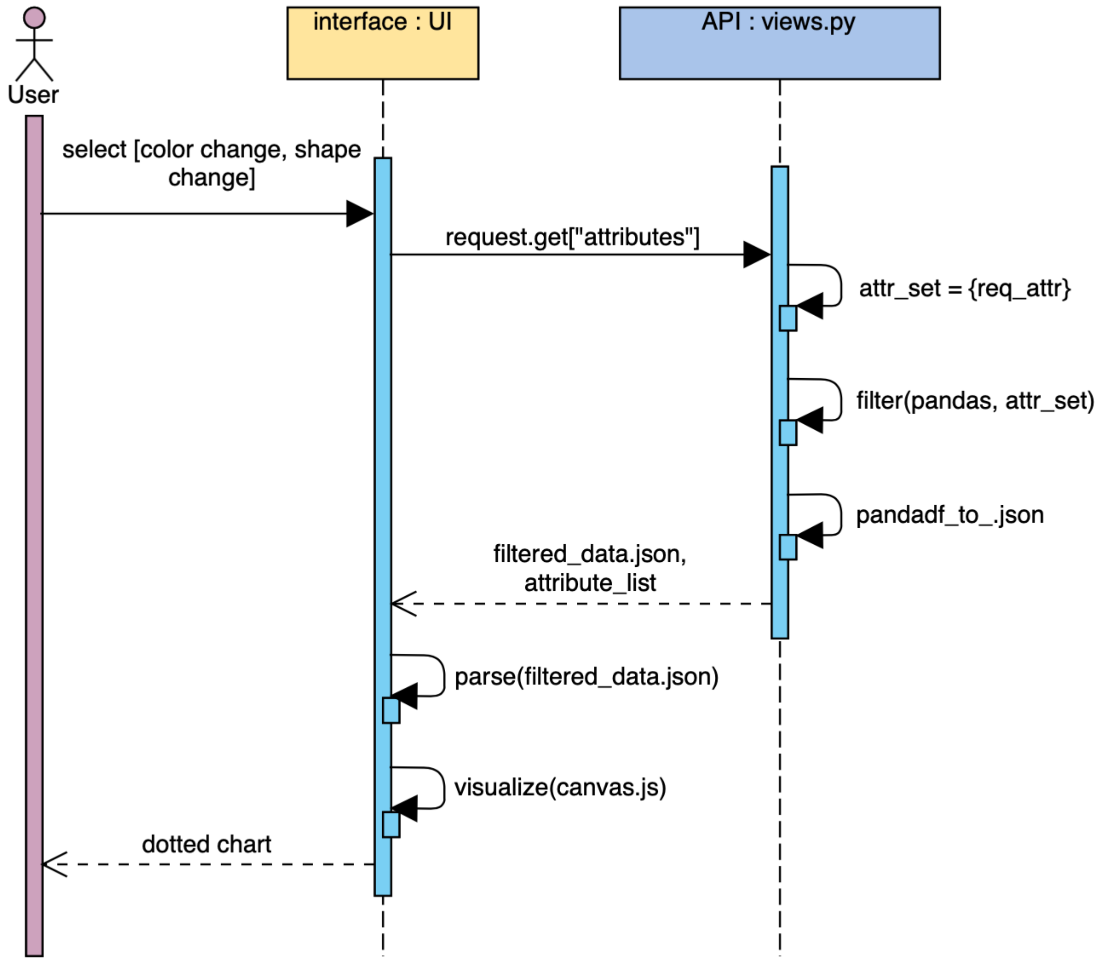

Figure 4: Sequence Diagram for the Manipulation of the Displayed Features

4.1.4 Exporting the dotted chart visualization

This is of high priority as requested by the costumer. The user can save the created chart to the “None-Event Logs” section of the Event Data page and be downloaded from there.

If the user plans to download the dotted chart, they can first save it using the “Save” button and then download by using the “Download” Button. The user is presented with a file manager which allows them to choose the designated location. The action sequence is further illustrated in Fig. 5.

REQ-7: Allow storage of visualization as png to Event Data page under the “None-Event Logs” section

The download from there is already provided in the pm4py-UI framework. For details, please see References.

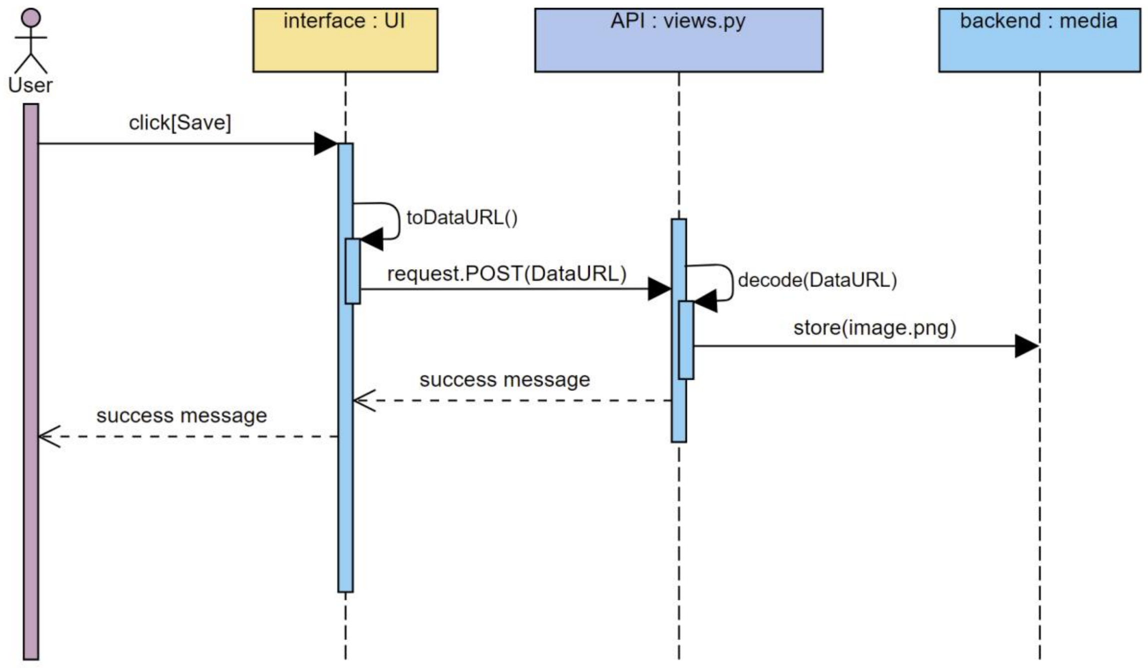

Figure 5: Sequence Diagram for Saving the Dotted Chart to Event Logs

4.2 System Perspective

4.2.1 Import/Export

Already provided, see 4.1 User Perspective.

4.2.2 Visualization

In order to visualize the event log data, the event log needs to be filtered according to the required attributes and converted to a pandas.dataframe first.

Depending on whether the uploaded file is in the XES or CSV format, the file is first converted to a pandas dataframe. In order to extract attribute information, the traces contained in the log file are filtered using the pm4py filter function creating a new dataframe. The resulting dataframe is passed to the front end for visualization. See the sequence diagram on the next page for an illustration of the action sequence (Fig. 5).

REQ-8: Convert CSV or XES event log file to dataframe
REQ-9: Validity check of event log
REQ-10: Extract attributes
REQ-11: Execute filtering on dataframe
REQ-12: Create scatter graph and display it

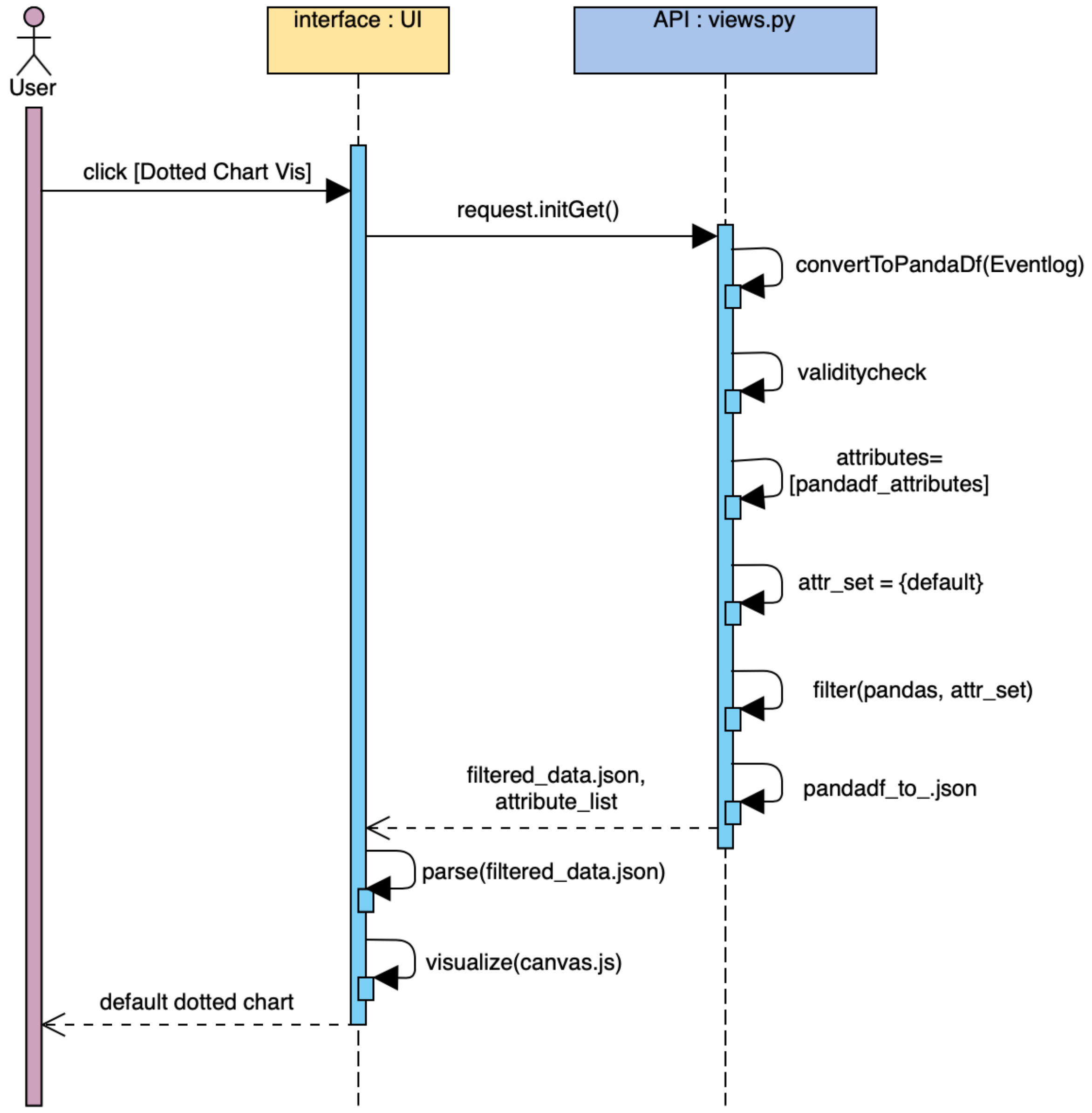

Figure 6: Sequence Diagram for the Dotted Chart Visualization from the System's Perspective

5. Other Nonfunctional Requirements

5.1 Performance Requirements

This application could only be used by 1 user per host at a time, due to the nature of the web application, which only runs as a client server application and only opens in a web browser.

Furthermore, our dotted chart visualizer also offers more flexibility in terms of platform independency, as it could be easily integrated into other pre-existing python-based process mining applications without going through a different setup or installing additional packages.

Therefore, it is much more stable over time and will not need a lot of changes to be maintained and reused in various kinds of applications.

The optimal performance on this application would be being able to successfully import the XES or CSV file to the app in less than 1 second. The file should be accurately imported and perfectly visualized as a dotted chart with uniform color in under 1 second without failing or crashing the application.
The user can then change the color of the dots or change the dots with other geometric shapes, which should be successfully done in under 1,5 seconds, with respect to the amount of data and types that are imported to the chart.
The user can also choose which attributes they want to visualize at the same time using the menu. The chart should be successfully updated in under 1,5 seconds with respect to the number of attributes that needs to be visualized at the same time.

In exporting the chart, it needs to be exported and saved in the desired file format in less than 1 second. It should be also correctly saved in the expected file directory.

5.2 Safety Requirements

There are no specific safety requirements to consider.

5.3 Security Requirements

There are no specific security requirements to consider.

5.4 Software Quality Attributes

There are no specific quality attributes to consider.

5.5 Business Rules

There are no business rules to consider.

Appendix A: Analysis Models

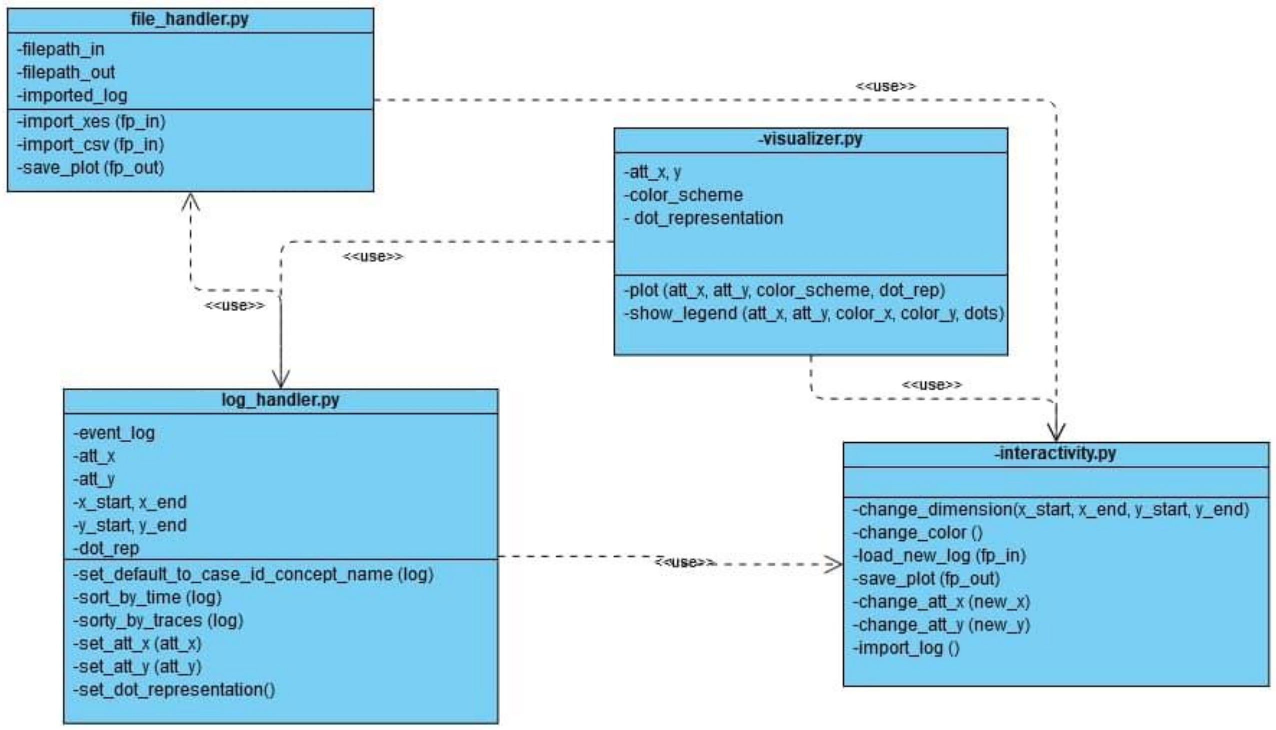

Figure 7: Class Diagram for the Low Level Structure of the Application

file_handler.py - This class is specialized on I/O tasks, specifically the import of log files in xes/csv format and the export of jpg/png files of the plot that the user wants to save to his system. For this matter, class attributes for the filepaths exist, as well as an attribute to that holds the log file which was imported by the import methods. This class provides information to *log_handler.py* which does further operations on the log file which was imported by the *file_handler.py*.

log_handler.py - The purpose of this class is to provide functionality to prepare the log data of the imported log file for a proper visualization and apply changes to the data when the user wants to modify the visualization. For this purpose, the class provides methods like *set_default_to_case_id_concept_name (log*) which prepares the log data for a visualization that shows the attributes case_id and concept_name. This class also handles when the user wants to choose another attribute for the visualition on either the x- or y-axis, provides methods to sort the data and is responsible for the dot representation of the data. How exactly, this is goint to be implemented is still left open in this draft. The *log_handler.py* receives its log file from the *file_handler.py*, it provides data to the visualizer.py which depends on the modifications of the logdata_handler.py for proper visualization. The *log_handler.py* is influenced by the *interactivity.py* which takes input from the user who wants to make changes to the visualization.

visualizer.py - This class realizes the plotting of the event data and the plot of the legend chart that shows information of the data that is plotted. It depends on the *log_handler.py* which modifies the data for the plot and it depends on the *interactivity.py* when the user wants to make changes to the plot.

interactivity.py - This class handles all the methods that realize interactivity with the user in the back-end. The user makes changes to the plot in the front-end, like changing the dimensions of the plot, switching out attributes that he wants to visualize, change colors or import a new log or save the plot as a picture file. The interactivity in the front-end is realized by HTML objects and event listeners in javascript that react when eg. a button is pressed to signal input by the user. When this happens, these changes by the user are handled by the *interactivity.py* by providing methods to realize the corresponding functionality, like changing dimensions of the plot. The *interactivity.py* therefore has influence on the *log_handler.py* and the *visualizer.py* whenever the user wants to make changes to the plot.