Software Design Specification

Label Refinement by Behavioral Similarity

Document owners:
Bianka Bakullari
Christopher Beine
Nicole Ventsch
Juan Garza

Last edited: May 10, 2019

1 Introduction

1.1 System Overview

1.2 Design Map

1.3 Supporting Materials

1.4 Definitions and Acronyms

2 Design Considerations

2.1 Assumptions and Dependencies

2.2 General Constraints

2.3 Goals and Guidelines

2.4 Development Methods

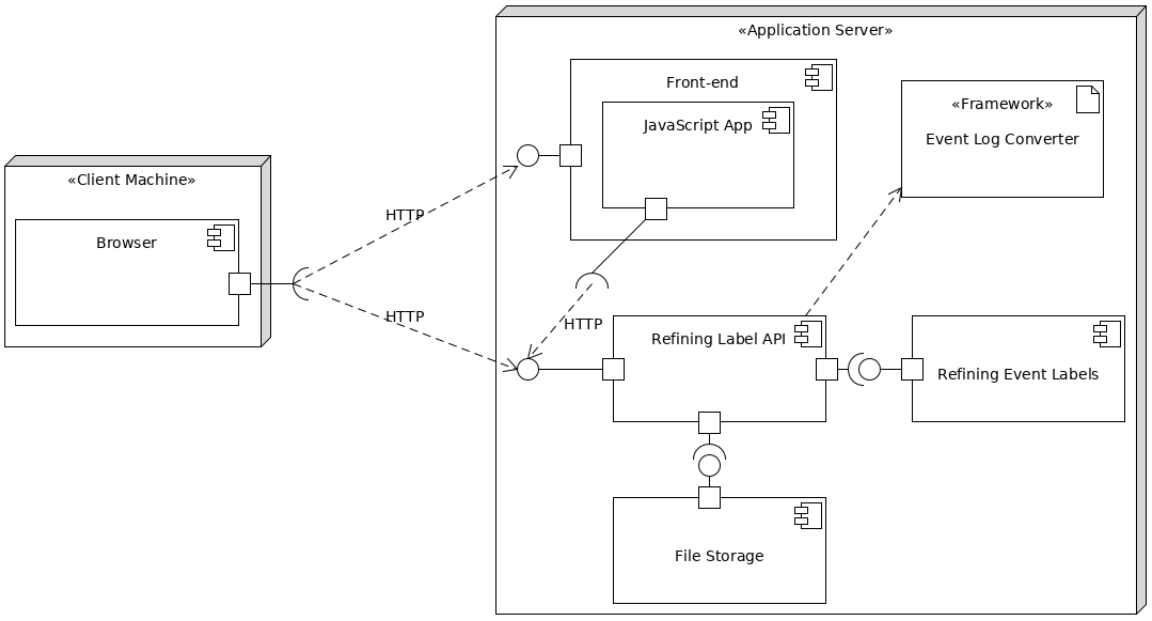
3 Architectural Strategies

**** under construction ****

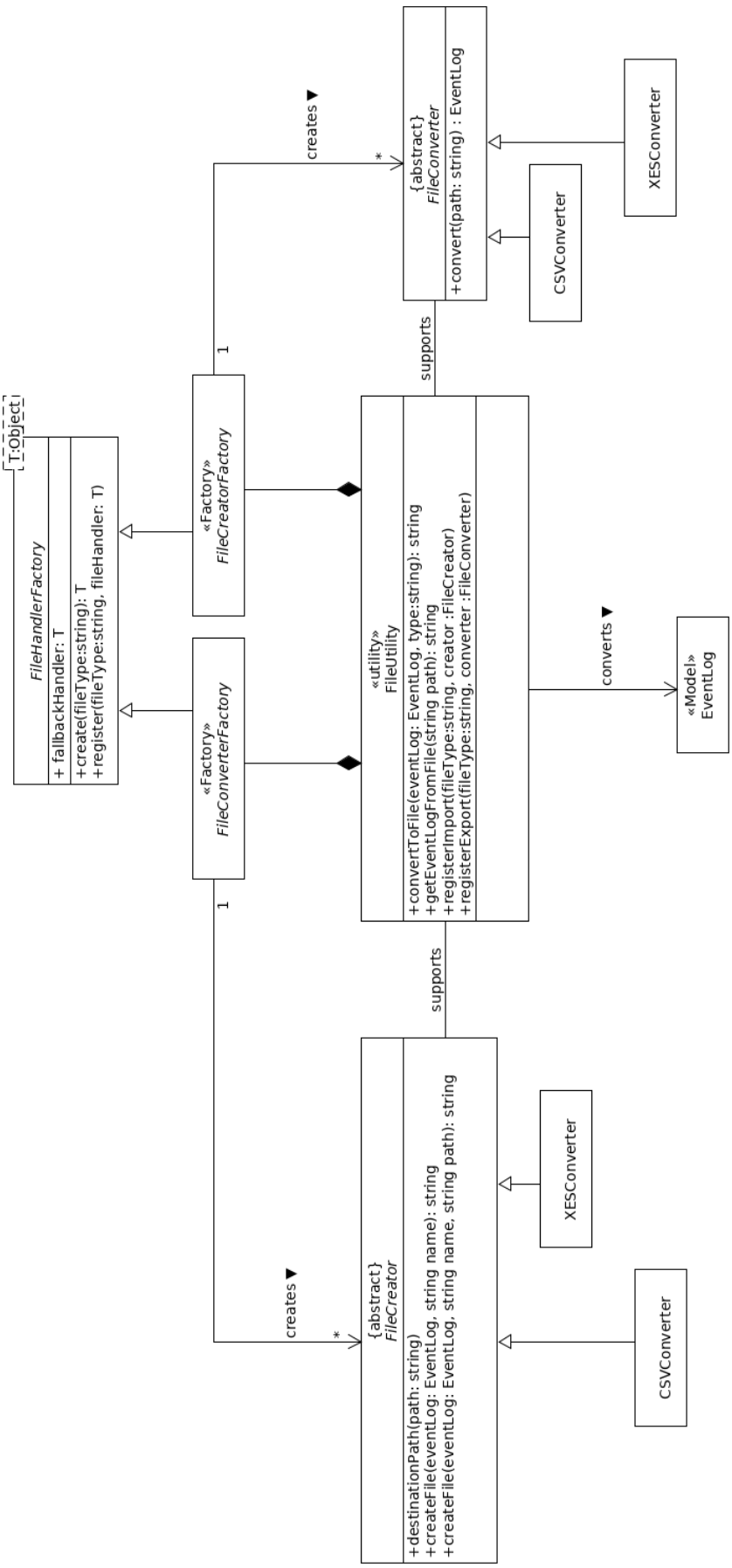
- Factory Pattern
- Decorator (Variaton)

****

4 System Architecture



4.6 Event Log Converter



5 Policies and Tactics

6 Detailed System Design

The main algorithm "Refining Event Labels" will be split up into multiple modules that contain the main parts of the algorithm. These modules will be explained in detail in the following subsections.

6.1 Module 1: Preprocessing

Name: Preprocessing
Type: module
Description: This module is responsible for preprocessing the data. The event log provided by the user is read and stored internally. Moreover, it is checked if it has the right format and an error is produced if is does not have.
Attributes: None
Resources: None
Operations:

Name: read_csv()
Arguments: path to a log file in csv format
Returns: event log
Description: The csv file provided by the user is read and stored as a table containing the original columns.
Precondition: the path leads to a csv file
Postcondition: the table is stored internally
Exceptions: None

Name: read_XES()
Arguments: path to a log file in XES format
Returns: event log
Description: The XES file provided by the user is read and stored as a table containing the original columns.
Precondition: the path leads to an XES file, the provided column names exist in the file
Postcondition: an event log is stored internally
Exceptions: None

Name: check_event_log()
Arguments: table imported using read_XES() or read_csv(), name of the activity column, ID column and time stamp column provided by the user
Returns: True or False
Description: The table we created by reading in the file is checked for actually being an event log, i.e., containing an activity column, an ID column and a time stamp column. The names for these columns are entered by the user. If these exist in the table, True will be returned, otherwise False will be returned.
Precondition: a file was read and the names for the columns were entered by the user
Postcondition: the process can continue if True was returned, else an error occurs
Exceptions: None

6.2 Module 2: Storing the Traces

Name: Storing the Traces
Type: module
Description: This module is responsible for getting the unique traces from the event log and storing the traces for each activity in a look-up table that can be used for refining the original log later.
Attributes: None
Resources: None
Operations:

Name: create_lookup_table()

Arguments: table, name of the activity column, ID column and time stamp column
Returns: lookup table containing one trace for every ID , where the trace is stored as an array
Description: The table is converted, so that we get the activity traces for each ID ordered by their time stamp.
Precondition: check_event_log() returned True
Postcondition: a lookup table is stored
Exceptions: None

Name: get_unique_traces()
Arguments: look-up table created using create_lookup_table()
Returns: an array containing the unique traces, i.e., an array of arrays
Description: The traces from the look-up table are checked for duplicates and the unique traces are stored in an array
Precondition: a lookup table was created using create_lookup_table()
Postcondition: an array of arrays containing the unique traces is stored
Exceptions: None

6.3 Module 3

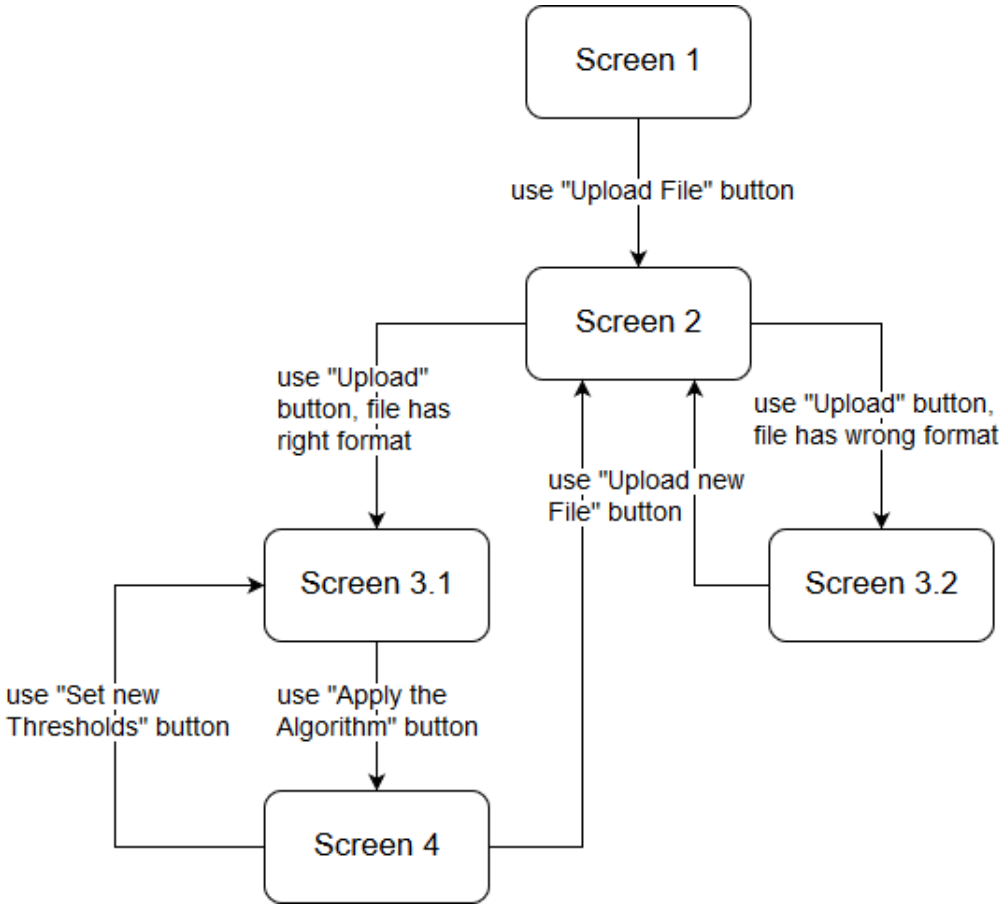
7 User Interface Design

7.1 Application Control

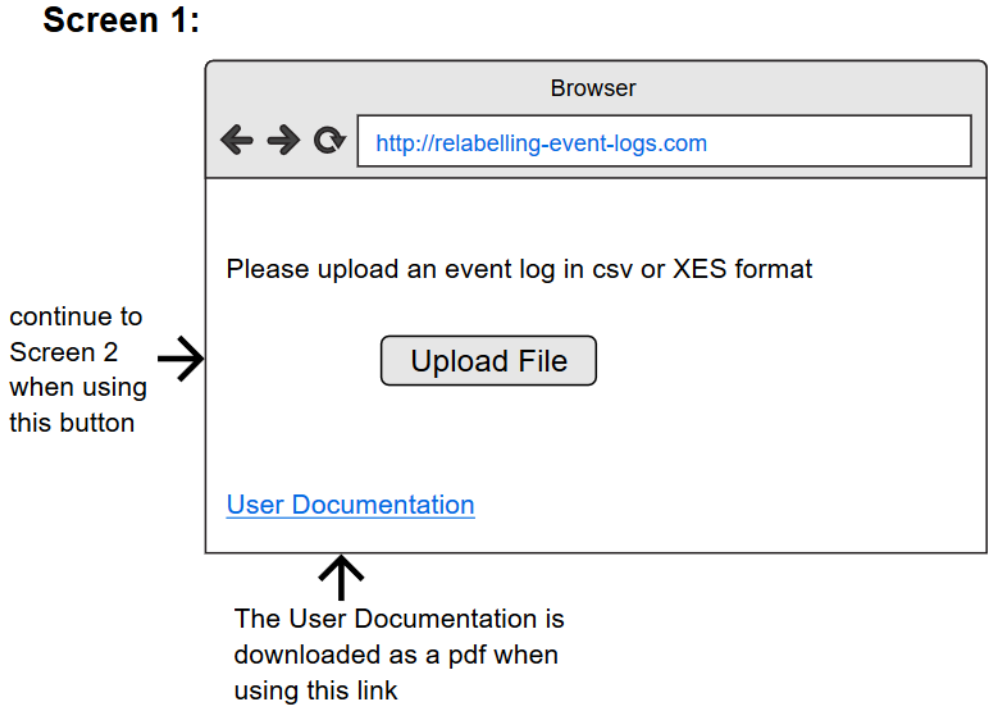
In the project, we will implement a web service. The web client will have a rather plain design that should focus on the main activities the service should provide, which are uploading event logs in csv or XES format, setting the threshold for the label refinement algorithm and download the refined log after the algorithm is finished. There will be buttons used to upload the file, apply the algorithm and download the refined event log. Moreover, the screens will have short explanations telling the user what to do (if not self-explanatory). For setting the thresholds, two boxes will be provided that include the default thresholds, but new values can be entered by the user. A draft of each of the main screen can be found in the next section, section 6.2.

7.2 The Screens

The main screens will be visualized in the following subsections. In these screens include the main functionalities, which are described in the former section. The following diagram will show the flow of control through the screens.



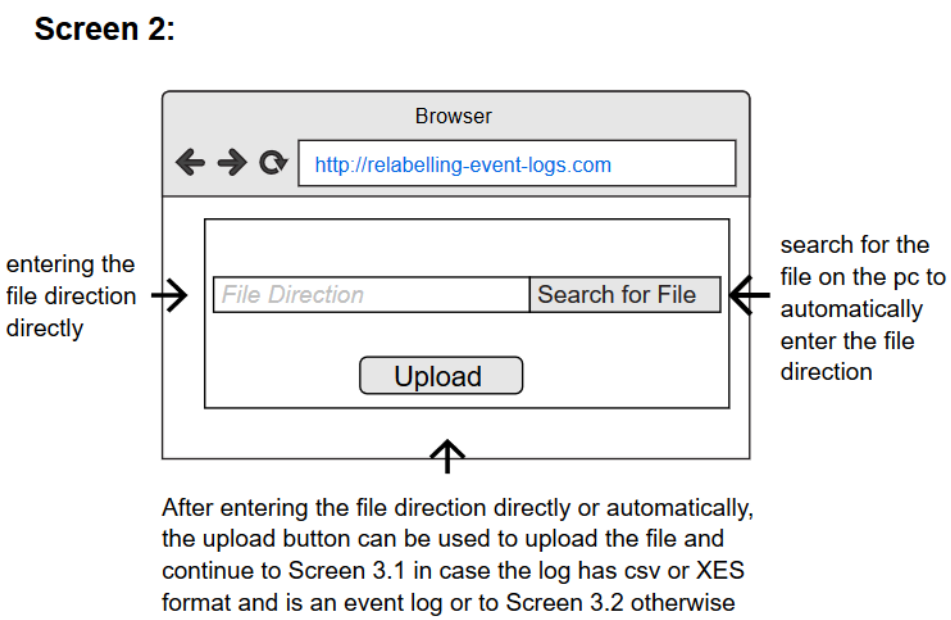
7.2.1 Screen 1



The first screen visible to the user will show a description saying that an event log in csv of XES format should be uploaded. Moreover, a button "Upload File" is visible. By using this button, the user will continue

to Screen 2. At the end of the page, there will be a link called "User Documentation". By clicking on this link, the User Documentation will be downloaded in pdf format.

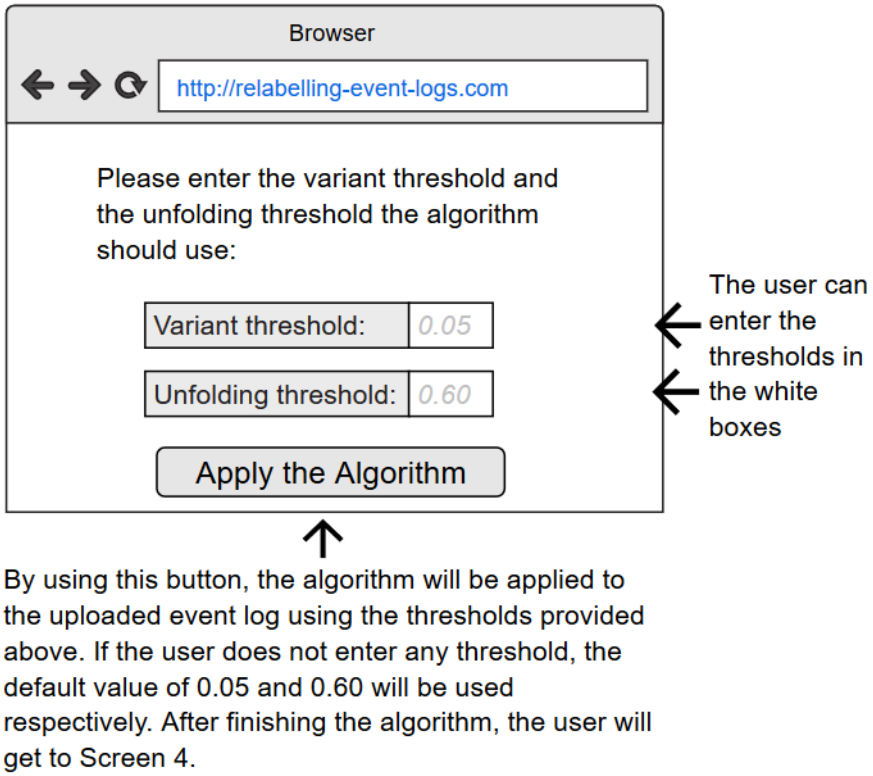
7.2.2 Screen 2



In the second screen visible to the user, the user can enter the file direction of the event log he wants to upload. He can either directly type in the direction into the "File Direction" field or use the "search for File" button to search for a file on his pc, so that the direction will automatically be filled in after selecting a file. After using one of this alternatives, he can use the "Upload" button to upload the file with the given directory. In case this file is an event log, i.e., the data contains at least the attributes "id", "time stamp" and "activity name", and has either csv or XES format, the user will continue to Screen 3.1. If one of these conditions is not satisfied, he will continue to Screen 3.2.

7.2.3 Screen 3.1

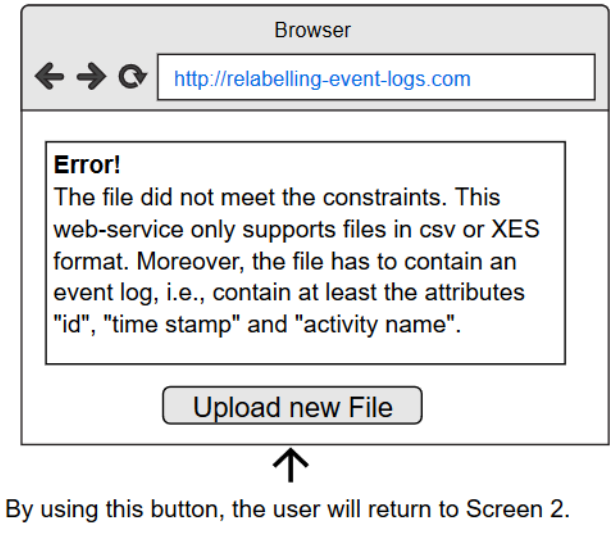
Screen 3.1:



This screen appears if the file uploaded by the user meets the requirements. In this screen, the user can set the thresholds for the algorithm, i.e., the variant and the unfolding threshold. He can enter these in the corresponding white boxes. If he does not enter the thresholds, the default values of 0.05 and 0.60 will be used respectively. Using the button "Apply the Algorithm", the web service will start applying the algorithm using the provided thresholds. After the algorithm is finished, the user will get to Screen 4.

7.2.4 Screen 3.2

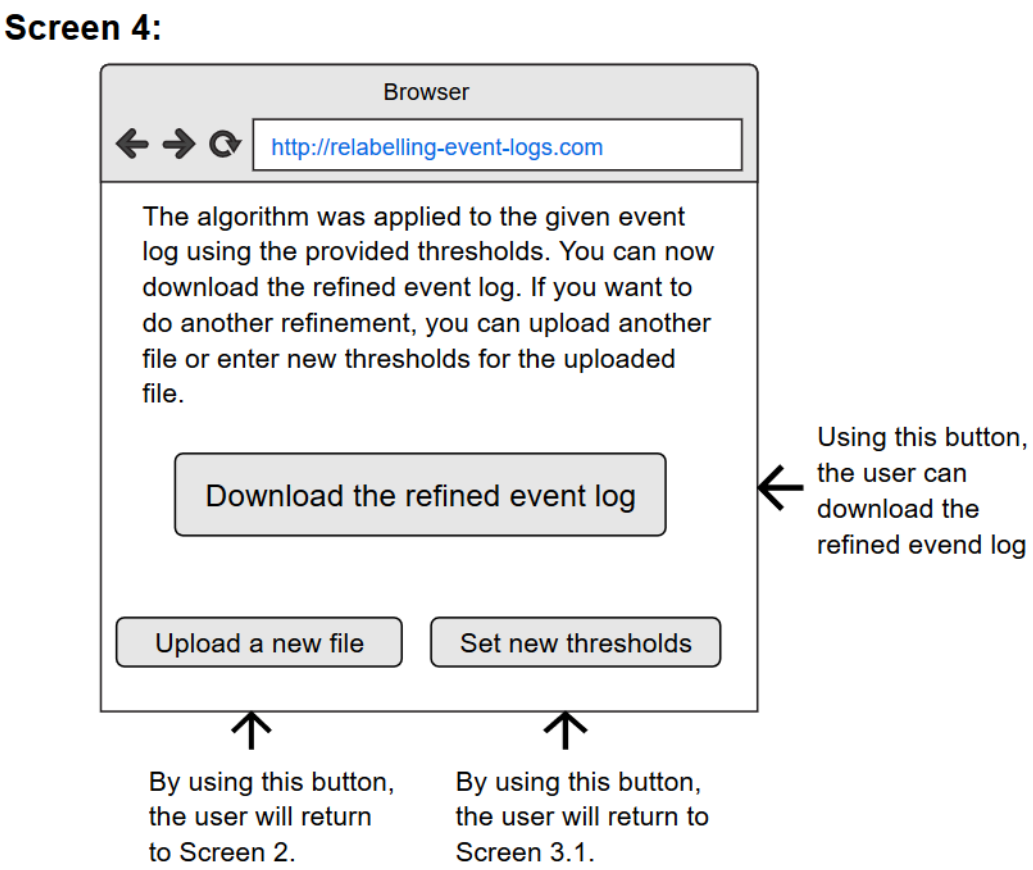
Screen 3.2:



This screen appears if the upload was not successful because the file did not meet the assumptions. If the file

does not have the right format, the user can click on the button "Upload new File" to return to Screen 2 and upload a file that meets the constraints.

7.2.5 Screen 4



This Screen will be shown after finishing the algorithm. The user can now download the refined log using the corresponding button. After this step, the user is done and can exit the page, but if he also wants to apply the algorithm to another event log or to the same event log using different thresholds, he can use the corresponding buttons and will be redirected to Screen 2 or Screen 3.1 respectively.

References

[1] Lu, Xixi, et al. "Handling duplicated tasks in process discovery by refining event labels." International Conference on Business Process Management. Springer, Cham, 2016.