Project: Software Engineering

Course DLMCSPSE

Portfolio Project

SENOUCI BRIKSI Djelloul

Registration number: 9225022

dd.mm.yyyy

Fribourg, Switzerland

Table of contents

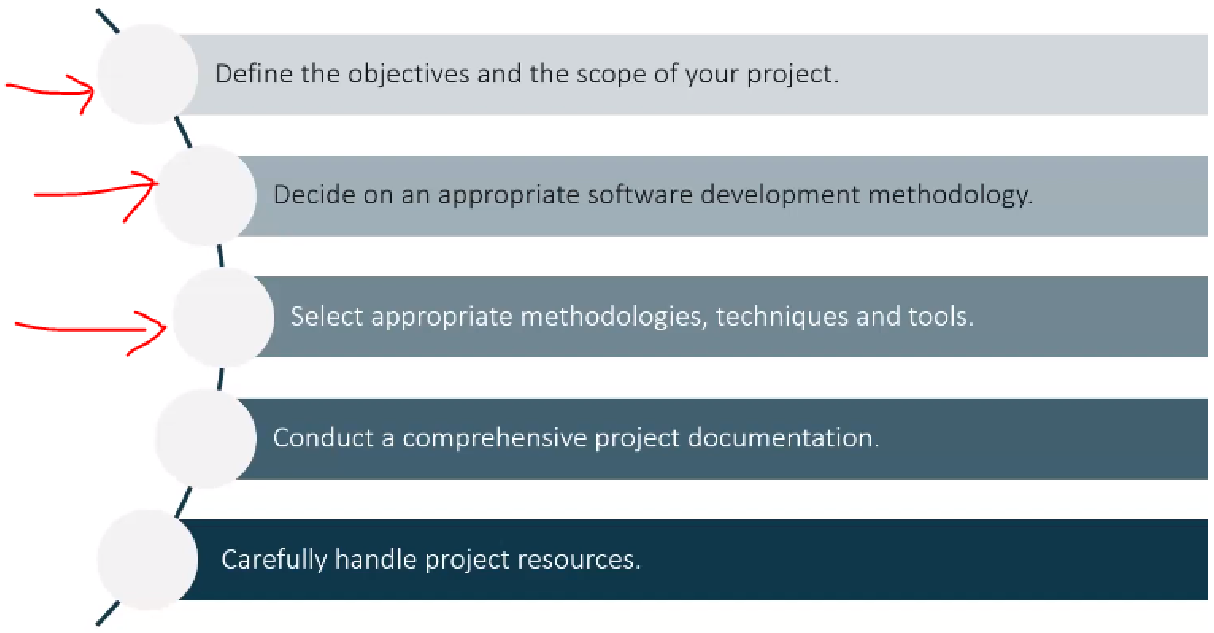
Image directory

Glossary

CAB	
Complex Action	
mobile SQLite database	
PIS	Passenger Information System
Route	
Train Number	

Introduction

Tips for a Successful Software Engineering Project



- Define the objectives and the scope of your project.
- Decide on an appropriate software development methodology.
- Select appropriate methodologies, techniques and tools.
- Conduct a comprehensive project documentation.
- Carefully handle project resources.

Requirements

This section contains the functional and non-functional requirements of our project. The requirements below are ordered following a top-down view. This order does *not* define the priority of the requirement. It is intended to implement all requirements listed in this section within the realization of this project.

Functional Requirements

The CAB is web-based application that allows browsing complex action trees.

The CAB loads all complex actions contained in a PIS mobile SQLite database.

Complex actions are linked to train numbers (routes). The CAB shall show the corresponding list of train numbers contained in the mobile SQLite database.

Train numbers (routes) are usually valid on certain dates or range of dates. The CAB shall give a possibility to choose a date to filter with.

Furthermore, the CAB shall give the possibility to filter the train number (route) to search into.

The CAB shall enable browsing a complex action tree by collapsing and expanding the tree nodes (complex actions).

How this tree of complex actions is visually presented will be defined during the implementation phase.

The CAB shall give the possibility to search complex actions by their attributes.

The following complex actions attributes (search criteria) are defined:

- 1111
- 2222
- 3333
- 4444
- 5555

The complete list of these attributes will be finalized during the implementation phase.

The CAB shall also give a way to check the plausibility of a given complex action tree.

The following plausibility checks are defined:

- 1111
- 2222
- 3333

- 4444
- 5555

The complete list of plausibility checks will be finalized during the implementation phase.

The following diagram shows the main features of the Complex Action Browser:

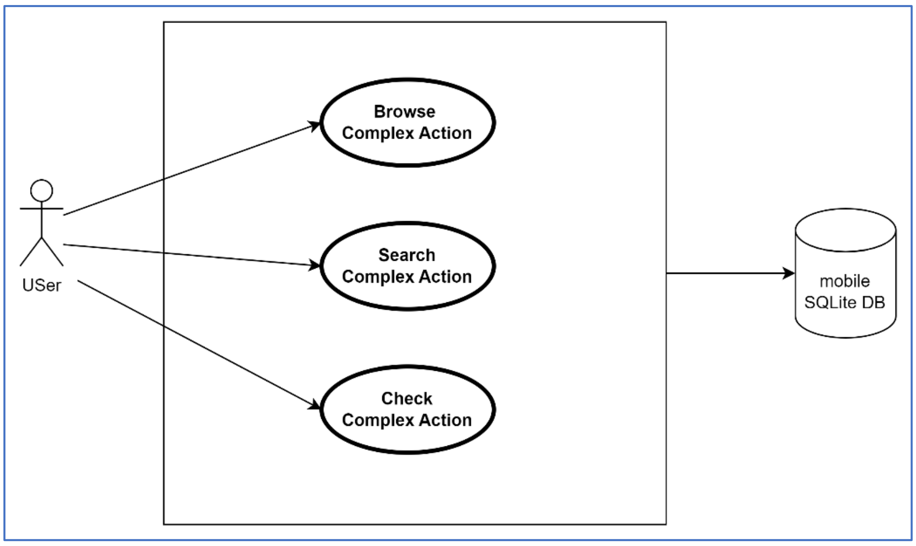


Figure 1 – CAB: Main Features

Non-Functional Requirements

The CAB shall be operable on Chrome, Firefox and Edge browsers.

The CAB shall be able to load mobile SQLite databases up to 1 GB.

When collapsing or expanding a complex action tree, the CAB shall render with a maximum of 300ms response time.

If the mobile SQLite database cannot be read (eg. corrupt or wrong/unknown structure), the CAB shall show a corresponding error message.

The CAB shall present the features (browsing a complex action tree, searching possibilities, etc) in an intuitive and meaningful way. → how to test ?

The CAB shall be easily updated, when e.g. the structure of the mobile SQLite database changes.

Design

Scope and Context

The Complex Action Browser (orange box in Figure 2) is the scope of our project.

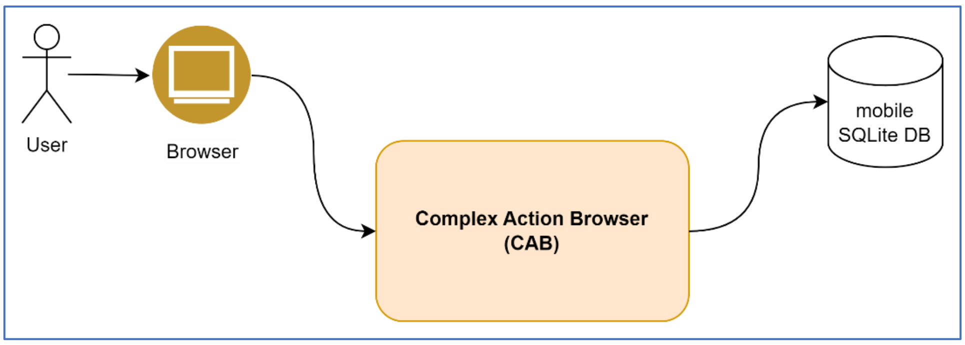


Figure 2 – CAB: Context Diagram

It is mainly accessed via a standard browser. It loads data from a mobile SQLite database.

The CAB does not provide nor use any other interface.

Components

The CAB will be implemented as a backend and a frontend application.

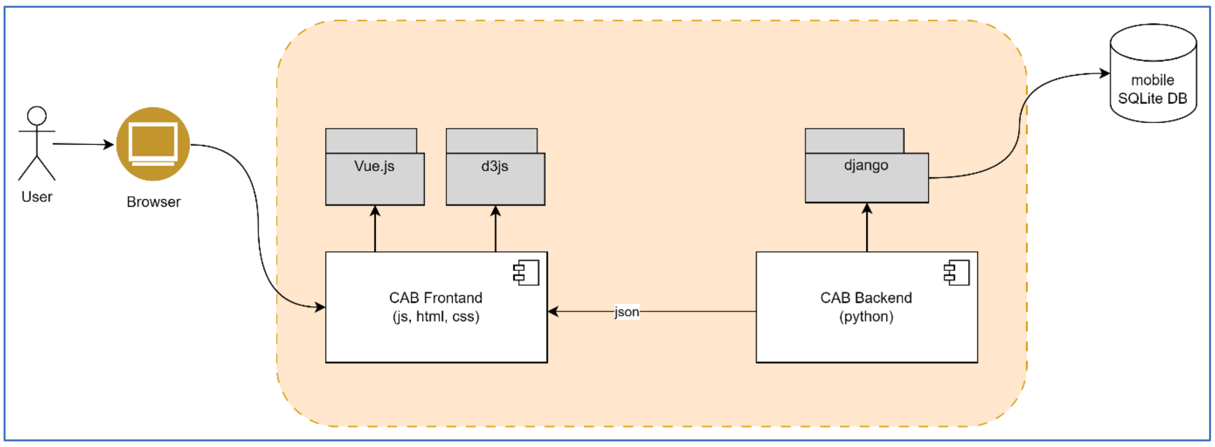


Figure 3 – CAB: Component Diagram

The backend application is written in python, and uses the django framework (django) to access the mobile SQLite database.

We have chosen The django frag

Realisation

Establish version and release control of the software and the documentation.

Testing

Conclusion - Lessons Learned

References

Tmp

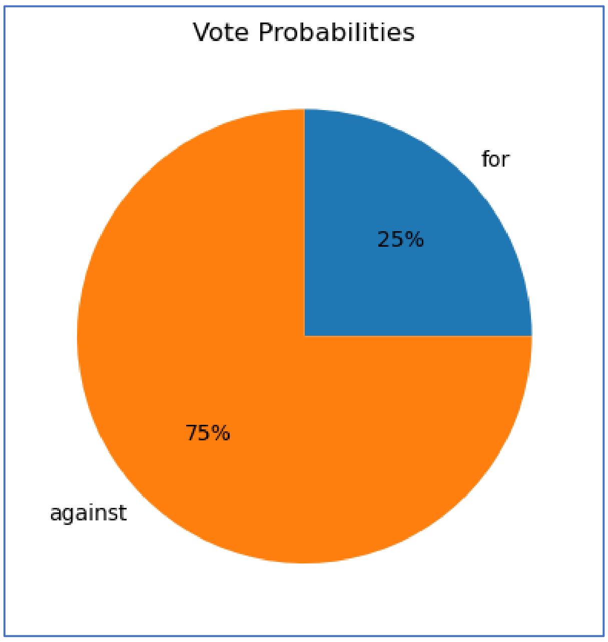


Figure 4 Vote Probabilities

Références

django. (n.d.). from https://www.djangoproject.com/.