Software design document

COMP.SE.110 Software Design

Weather and traffic data manipulation

Written by: Veikka Peltomäki, Niklas Kanetele, Eemil Lehto, Samuel Ahopelto

Introduction

This document works as a design document on our groups project in the course COMP.SE.110 Software Design. The purpose of this document is to provide a description of our system which is comprehensive enough to work as a source of details for how the software is to be built.

The software project will be implemented using programming language C++20 and the Qt framework. The project is to be implemented using agile methods where the group will continuously evaluate the gathered requirements and iterate on those.

System Overview

The goal of this software project is to create a software which allows its user to study weather conditions and its history as well as road conditions and its history and their correlation by studying graphs, tables, and other visualizations in the software UI. These visualizations will be based on data gathered from Digitraffic (road condition data) and FMI (Finnish Meteorological Institute, weather data). The data is gathered in real time using public APIs provided by Digitraffic and FMI.

The software will be based on a UI where the user is able to select if they wish to view weather data, road conditions or the correlation between these (weathers effect on road conditions). The user will be able to input the coordinates of the area which he wishes to study as well as select a timeline of the data involved in the visualizations. The user should also be able to compare the data between different timelines and coordinate areas.

Requirements

These requirements will be iterated on continuously during the development process as the software development group identifies more of them.

The software system shall be able to gather real-time as well as historical data on weather and road conditions using APIs provided by Digitraffic and FMI

The software system shall have a GUI which allows the user to select which of the provided data to study

The data shall be visualized in various forms such as tables, graphs, and plots.

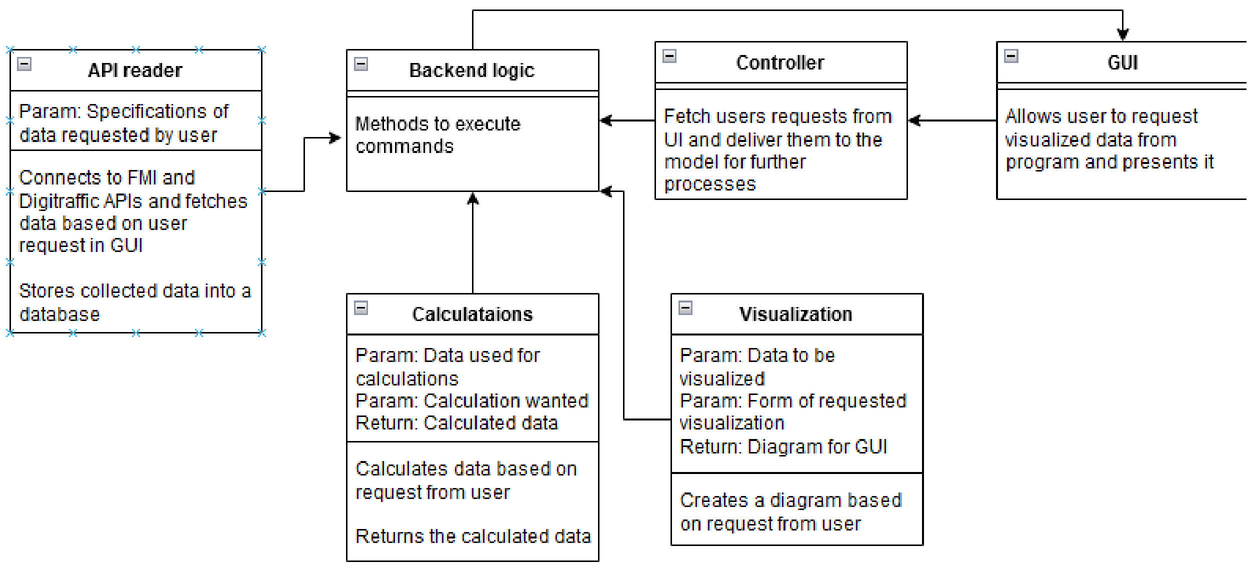
The user should be able to select the timeline of the data they wish to see in these visualizations

The user should be able to specify coordinates of the area of where the visualized data will be based on

The system shall be able to process the data gathered from the API and do calculations based on it to provide the user with relevant information based on the calculations

Architecture

The software system is to be implemented using C++20


API reader
Param: Specifications of data requested by user
Connects to FMI and Digitraffic APIs and fetches data based on user request in GUI
Stores collected data into a database
Backend logic
Methods to execute commands
Controller
Fetch users requests from UI and deliver them to the model for further processes
GUI
Allows user to request visualized data from program and presents it
Calculataions
Param: Data used for calculations
Param: Calculation wanted
Return: Calculated data
Calculates data based on request from user
Returns the calculated data
Visualization
Param: Data to be visualized
Param: Form of requested visualization
Return: Diagram for GUI
Creates a diagram based on request from user

1 Diagram of software classes

- Backend logic model
  - Executes other backend classes (Data gathering, Calculation and visualization, comparison)
  - Sends results and visualizations to the GUI
- Controller
  - Handles users' requests from UI and sends them to the backend
- GUI
  - Frontend of software
  - Based on prototype created in Figma with final implementation being iterated on that
  - Possibly based on Qt GUI
  - Visualizations of data collected and processed
- Data gathering component
  - Gathers and processes data provided by Digitraffic and FMI APIs
    - Data will be processed into a form which is easy to use in calculations
  - Saves gathered data into a database for later use (Not a must-implement)
- Calculation ~~and visualization component~~
  - Gathered data is calculated to properly visualize for example the correlation between weather and road conditions
  - ~~Data is turned into different forms of visualization~~
- Visualization
  - Forms presented diagrams to present to the user
  - Diagrams created based on raw data from API (temperature, …) or calculations (correlation)

~~Comparison component~~

- ~~User is able to save a coordinate+timeframe combination of their choice and compare this to another combination of coordinates+timeframe~~

- ~~The comparison will be visualized in a manner where the user is easily able to understand and differentiate the data~~

Example run case

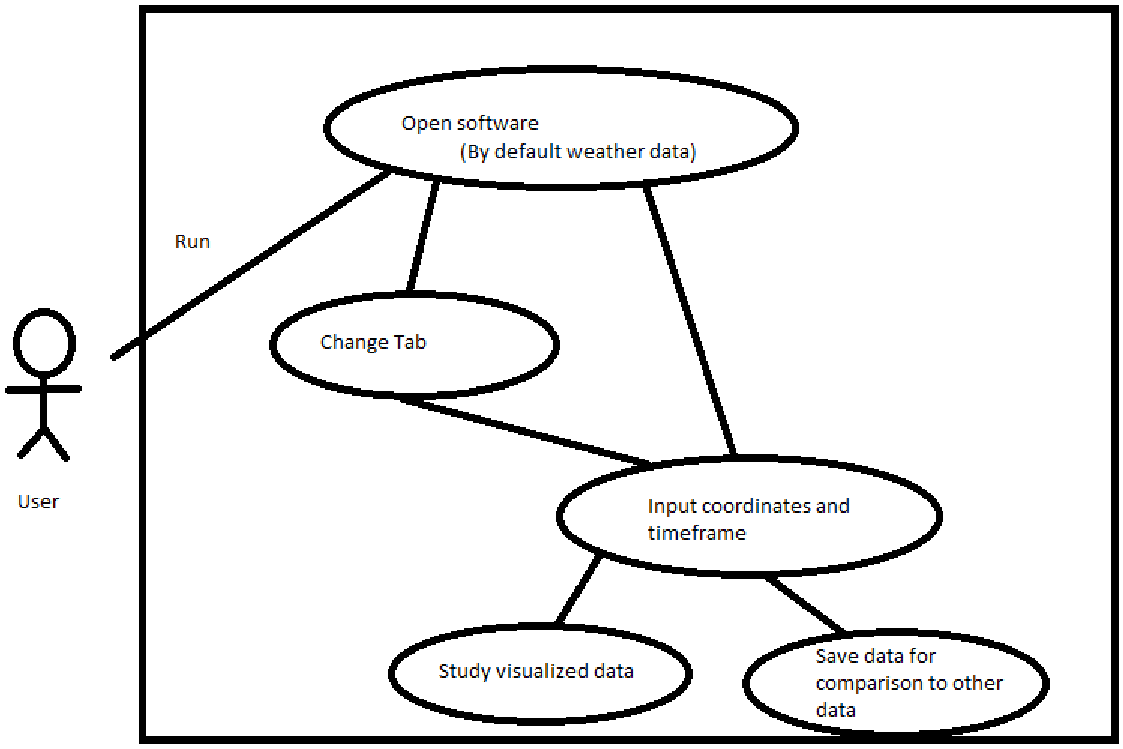

Open software
(By default weather data)
Run
Change Tab
User
Input coordinates and timeframe
Study visualized data
Save data for comparison to other data

Figure 2 Very simple use-case diagram

- Software is opened
- Software is automatically opened to provide weather data of Tampere with a short timeline
    - This data is by default gathered from the API and visualized on the GUI
- The user can either:
    - Input coordinates and select a timeline to study similar weather data with different specifications
    - Change to a different tab
        - Selection of two
            - Road conditions
            - Weathers effect on road conditions
        - Both have a default location (Tampere) and a default timeframe
    - After inputting coordinates and selecting timeframe of choice user is able to save his selections to later compare them to another choice

The run case and architecture will be iterated on heavily as the project continues as for now only the basic functionality has been planned out.

Software project timeline

By 2.10.2022: Prototype of system UI and base design document done

By 30.10.2022: Somewhat working implementation of UI to be deployed with some functionalities. Working API calls to gather data from sources. Updated design document with self-evaluation based on the created prototype.

By 2.12.2022: Working final product

Self-evaluation as of 30.10.2022

As of the mid-term point of the project it has had a lot of ups and downs. Even though everyone in the group is mainly using the same IDE (QT Creator) with the same operating system QT Creator has proven a little problematic at times especially when trying to fetch data from the API due to OpenSSL having some issues requiring specific .dll files to be copied into the build folder for it to work. Due to figuring this problem out as well as the project group having pretty busy ends of 1st period, we haven't had the best of progresses with the backend side of the project.

As far as the design goes, we haven't had too many changes from our prototype. Our front end is very similar to what we envisioned in our Figma, and the biggest backend changes have been to our classes. Instead of combining visualization and calculations we concluded that these should be divided to their own classes. I believe our current design will work well in this project. It's simple in a way that we shouldn't end up with bloated classes with too much functionality or with too many classes each doing a small thing which a function in another class could do. I anticipate the larger changes in the future to be with our frontend as we dive deeper into the data the APIs are able to provide (datatypes, forms of diagrams, timelines and locations).