Design document Software Design COMP.SE.110-2022-2023-1

Emil Kaskikallio 050106789, Ilmari Marttila 265040, Joonas Paajanen 050115767, Tomi Lotila 274802

Contents

1. Introduction

This is the design document for the task to design and implement a desktop application for traffic data visualization. In this document we will introduce the graphical user interface (GUI) and a high-level description for classes and interfaces to represent how individual components handle our data flow. We will be using Digitraffic API to fetch data to our programs use. The Programs outline is that it is done with C++ as a QT application which already gives us a good starting point for the communication between the user interface and the program itself. The prototype of the graphical user interface was implemented with Figma.

2. Software outline

Our goal is to make a piece of software that helps users to monitor weather and road conditions in Finland. Users can study how different weather conditions affect road maintenance requirements. The software fetches data such as weather forecasts, temperature, visibility, road slippage, road traffic, and traffic cameras. The road data is fetched from Digitraffic and weather data from Ilmatieteenlaitos. Software allows users to select locations and study data in graphical format. Users can also save data for later study.

3. Graphical user interface prototype

A graphical user interface prototype is implemented with Figma prototyping software. For the home screen, in figure 1, we wanted to show a collection of everything such as road conditions currently and forecast, weather, and current ongoing maintenance work. If user wants to study more about a specific area he/she can press the 'more'-button. Selecting or reselecting location is on the top bar.

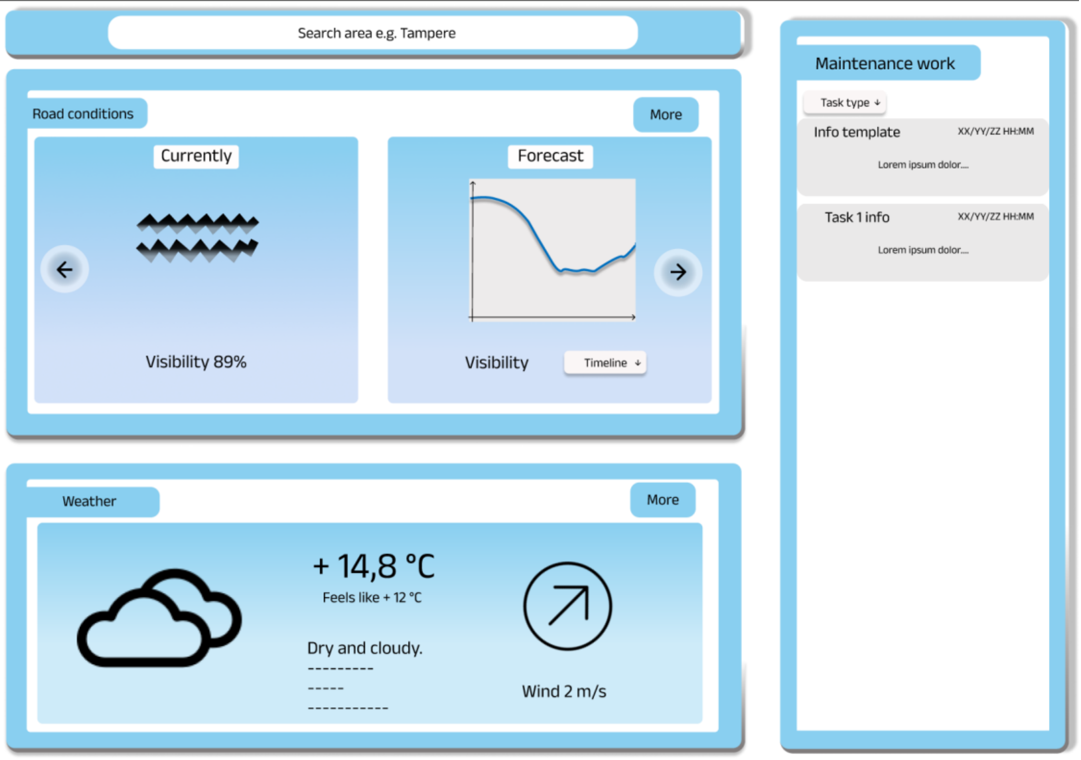

Figure 1. GUI prototype home screen.

After the user has pressed the more-button the screen will update to show bigger graphs and allows the user to change different variables such as the timeline. The ongoing maintenance work window and the location search bar will always show on different screens because users may want to go back to those. A home button will appear on the top bar to allow the user to return to the home screen.

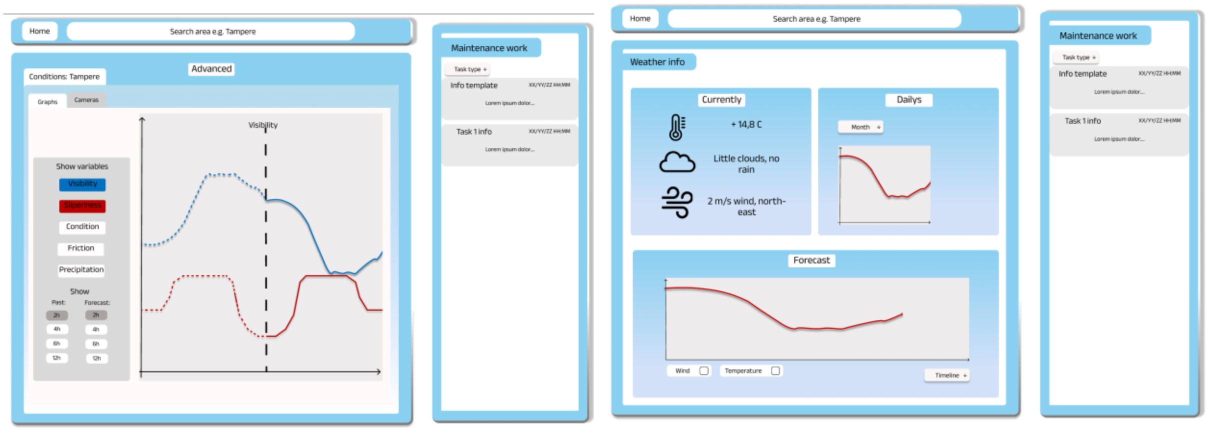

Figure 2. The road condition screen is on the left and the weather info screen is on the right.

4. Component diagram and boundaries

We need to separate UI, logic, and data from each other. To accomplish this, we choose model-view-controller (MVC) as our software architecture. MVC architecture is very structural and doesn't allow circumventing the structure. The architecture allows us to work on different parts at the same time.

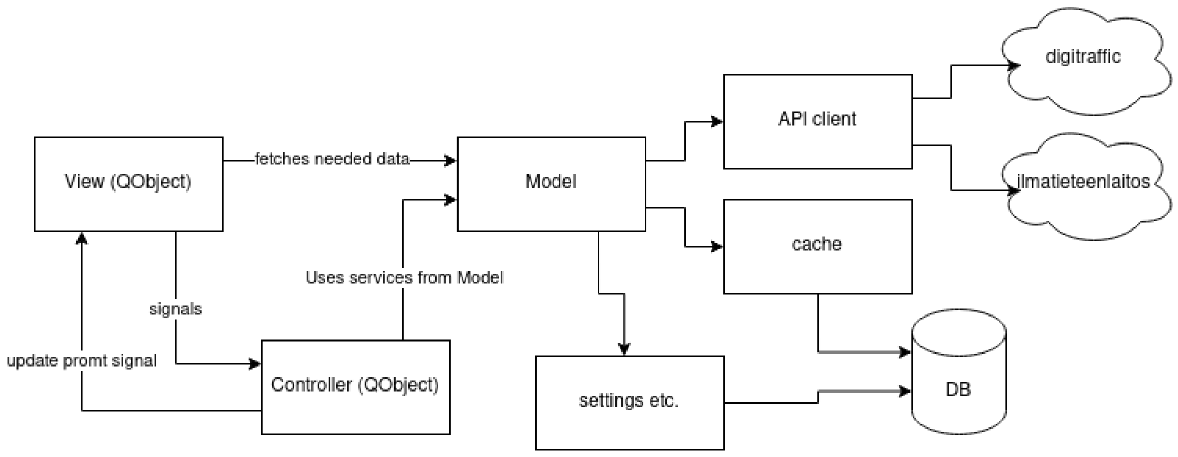

Figure 3. The software structure.

View

The view contains graphical interface elements. The view takes update command from controller and updates the graphical interface with the latest data. The view has access to data from model. The view will be implemented with QWidgets from Qt library. The model interacts with the controller with Qt signals and with the model trough pointer.

Controller

User interaction towards user interface triggers controller to call services from the model. After or during that controller – model interaction the controller tells the view to update its content from the model. The controller inherits from the QObject so that it can interact with the view with signals. The controller also interacts with model trough pointer.

Model

Model executes service calls from controller. The model might inform the controller if new data differs from the old data. The model should not have anything to do with the Qt. This makes it possible to change GUI modules later if one want to. Ultimately GUI could be replaced with CLI.

5. Class structure

Model is mainly used trough one main Model class that is composed of separate model submodules.

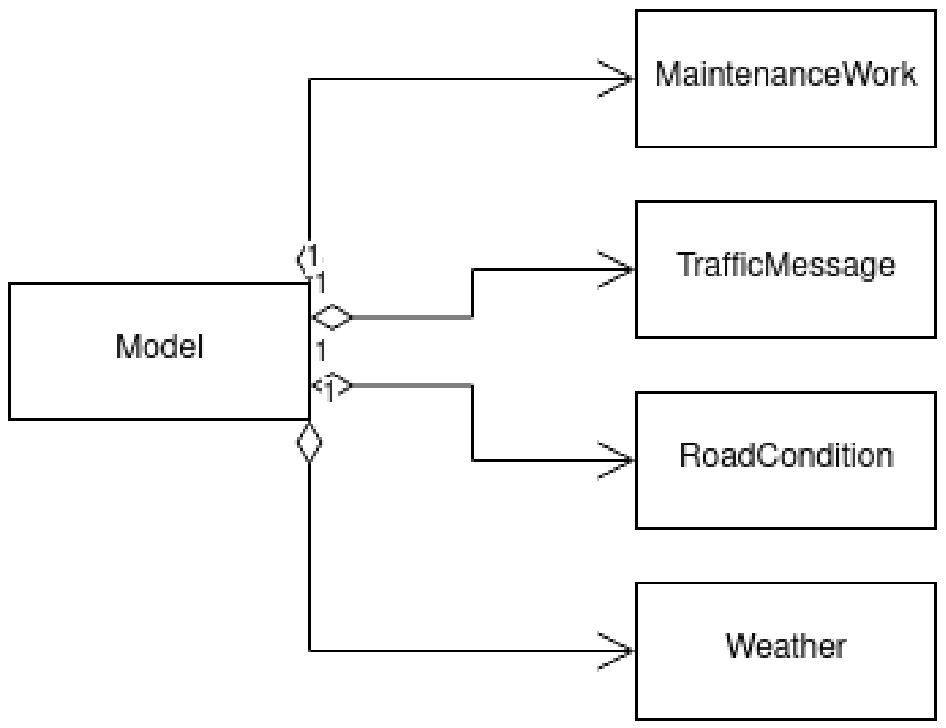

Figure 4. Class structure.

Submodules are separated from each other's because they will have different internal logic or interfaces. Most of these submodules will have methods that take a location and a time slot as parameters.

6. Libraries

GUI parts of this software will be made with Qt6 library. It is not yet decided what library will be used to parse incoming web responses or what library is used to send web requests. Cache will be probably done with SQLite (or not at all if we do not have time to implement it).

7. Sources for weather and road data

https://www.digitraffic.fi/en/road-traffic/

https://tie.digitraffic.fi/swagger/

https://tie.digitraffic.fi/swagger/#/Maintenance

https://tie.digitraffic.fi/swagger/#/Data%20v3/roadConditions

https://tie.digitraffic.fi/swagger/#/Traffic%20message

https://en.ilmatieteenlaitos.fi/open-data-manual