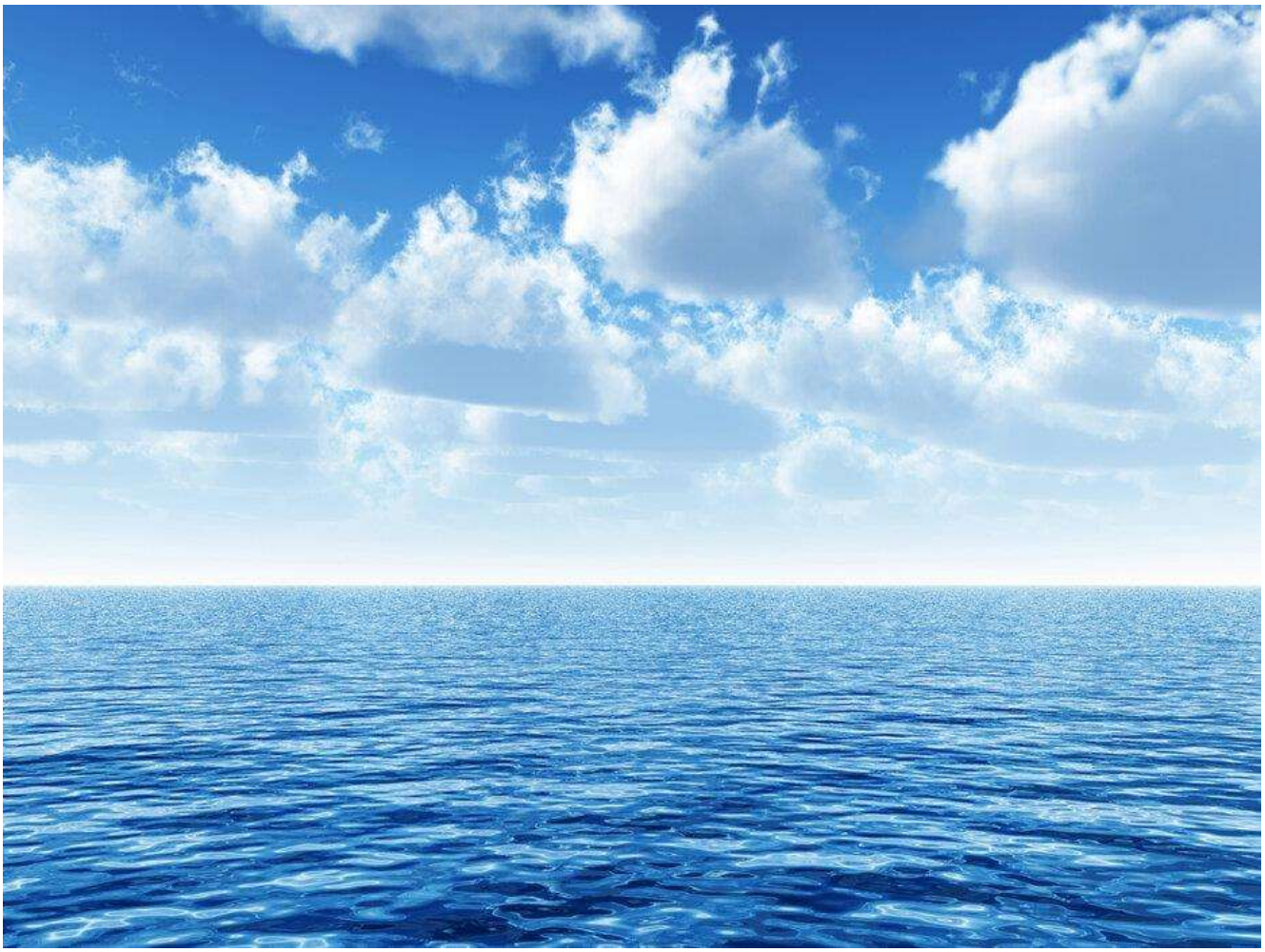
Project Horizon



Team Members: Niilo Hautamäki, Joni Hämäläinen, Max Jauhiainen, Tenho Laakkio

1. Introduction

For this project we will be using Openstreetmap API as well as Vipunen API. The main idea of this project is to allow the user to easily see the points needed to enter a college or a high school. The user can filter their wanted fields of study and see from the map where the field is taught in. The user can also give their maximum number of points or use the software calculator that calculates the points and relevant subject scores from their matriculation examination to the map filtering system. The marker popup for the selected college or high school will include overall a variety of useful information that also includes a simplifying rating system that will be based on our algorithm. The algorithm will at least look into the number of students to the staff, the overall success in the research department but also the financial situation of the college.

The marker popup will also include the possibility to see the history and the progression of the points needed to enter the school institute.

The tech stack for this project is the following:
Backend: Java with Spring Boot

Frontend: React with Vite

2. API usage

Like we previously mentioned, our project will utilize two distinct API's: OpenStreetMap and Vipunen API. One is very famous for its free-to-use API implementation and the other one is distinctly Finnish it allows us to utilize Finnish school data to incorporate varies of Finnish Department of Education –data for our usages.

OpenStreetMap will allow us to use the Vipunen's provided school data to get more general information of the school; that includes the contact information and the school website but also the coordinates for the school to plot to the map.

<u>Update for mid-term submission</u>

It turned out that the best API for finding coordinates for schools was Nominatim API. Nominatim is linked to OpenStreetMap, but it still is a separate API. With Nominatim API, it is easy to get coordinates for schools with just using the name of the school as query word. We get the names of the high schools from Vipunen API and names of the universities/colleges we get from parsed Vipunen's excel files.

We still use OpenStreetMap for the map and by combining Nominatim API, Vipunen API and parsed excel files we were able to get the schools showing in the map. Also, it's

worth noting that at least one college (Humanistinen ammattikorkeakoulu) and several high schools don't show up in the map, because Nominatim API didn't return any information for them. We will try to make these schools searchable for the final submission.

3. Interfaces and general design

The map page works as an interface, the given data will affect the available filter options which correspondingly affect the markers on the map. That means that we will use at least the component design patterns like composite since it allows us to utilize the component-based GUI design. This can also be seen to utilize the factory design pattern, our MapComponent will act sort of as an interface and the given data will complete it.

The information which dataset will be used, is driven by user choice. The user can choose from two options at the landing page: high schools or universities. This option ultimately decides the dataset, if the user somehow goes to the map screen without a chosen dataset, university dataset will be selected as default. That is a good example, why MVC is such a useful design principle in this project, we need to separate data and view to ensure that filtering and dataset changing is possible with little effort.

<u>Update for mid-term submission</u>

The user can open a side panel by selecting a marker from the map and then clicking the button that has the schools name on it. When the user has selected the "universities" view, the side panel will show all the fields of study in the selected university and by clicking a field it will show the different types of admission methods and the required points to be accepted to the field from the last year's results.

In addition to showing the last year's results we plan on implementing a separate view in the sidebar that shows the graph for the last five years of required points to be accepted to the selected field.

In this submission we have utilized generic programming to provide an easy platform to expand upon. The Cache class provides a base class where the load and save functions are provided. They allow the programmer to create sub cache classes and provide their own datatypes to the head class. This makes the general programming flow easier to maintain and expand. This can be seen with CoordinateCache for our coordinate data but other caches will be implemented later in the development of this software.

We have also utilized web scraping which has allowed us to learn how to combine datasets to provide one cohesive dataset. It was also critical to follow the single responsibility principle for this since it made the class hierarchy clearer.

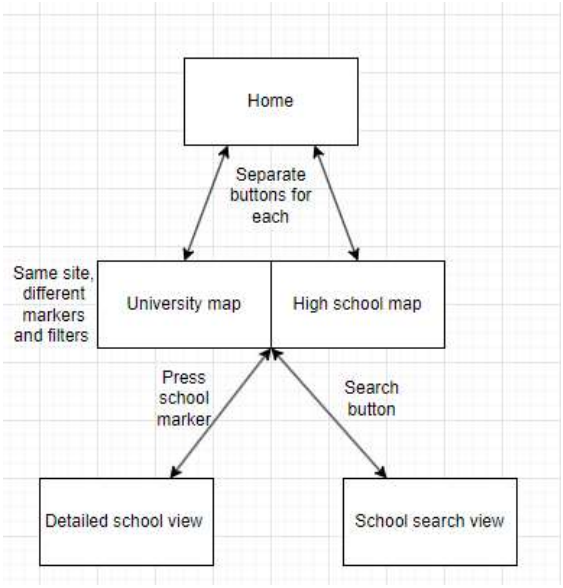


Figure 1: Navigation diagram

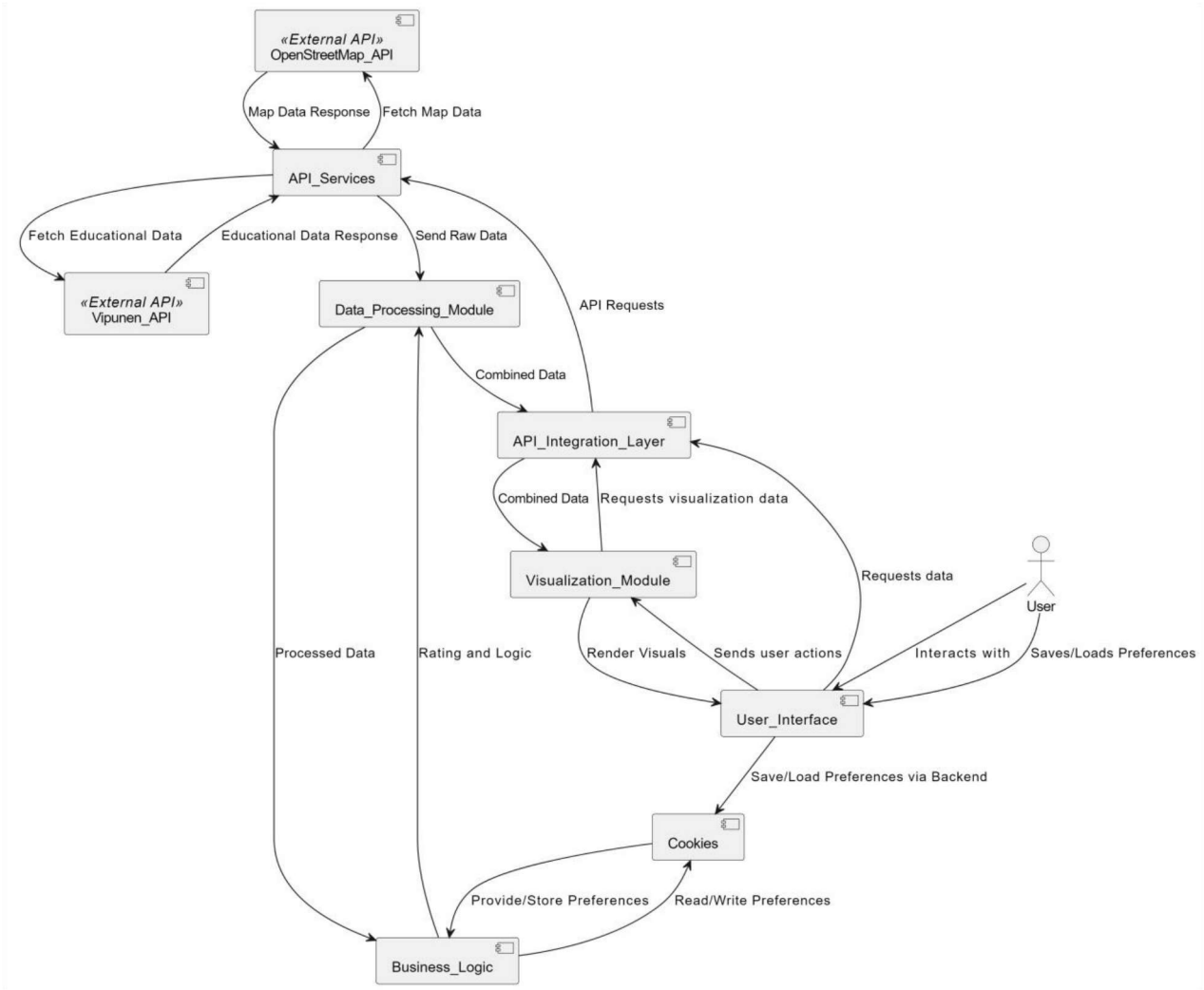


Figure 2: High-level design diagram

Figure 2 describes how our program's structure could look like, when all the features have been added.

4. AI USAGE

AI has been used for creating the first version of Figure 2 diagram when given prompt with the specifications of our project, including this document and the given project specification. Figure 2 was then improved "by hand".

Update for mid-term submission

AI has been used for some help in coding, especially for API calls and general improvements and debugging.

5. SELF-EVALUATION

The design we made supported implementing the code and it will also support implementing the remaining code as well. We had to branch from the original plan slightly, but the big picture is still relatively the same. We have been able to implement features from the original design in some extent, apart from some small changes, but many features are still yet to be implemented. Many of these yet-to-be implemented features (for example graphs for point limits) need to be implemented in the front-end. That's why most of the work that we have ahead of us, is in front-end. Backend has to also improved to be able to offer the needed data for frontend.

We anticipate that we will have to improve the communication between the back- and front-end of the application, as well as the communication between the different parts of back-end.

We have had our fair share of challenges in the data management side. There have been multiple apis and sources for our data. This has made combining them a real challenge. This however has not disheartened us; we have developed wide variety of solutions to combat these issues. First of all, our OpenStreetMap data usage was too naive, we could not be able to find a do-it-all solution to our needs, for instance contact information was not able to be found, as it was on the official OpenStreetMap, since we had to resort to Nominatim. Second of all, our data collection has grown also, in the manner we do it. Our first ventures into web scraping were also done, that will be a challenge later on the when the data will be integrated into frontend and when the backend will be finalized.

We have also learned a lot on the software design side of the things. We have utilized the Singleton pattern and seen how it is effectively used. Some people had doubts about it and its use in this project, but later on everyone has agreed that is has been valuable, especially for our utility-classes which make the use of them everywhere in the program easier. Another good use that we learned, is the singleton class as a context class for instance for university data since there will only be one data object.

We have also focused on the single-responsibility principle as it is a major topic of this course. We have divided our classes especially careful to not to break this principle. This has also helped us to divide our functionality to different packages and folders, since we try to make it clear which parts of the program are done and where.

To improve clarity and reusability even more, we have also tried to use generic programming and the use of template method principle. We utilized it on our cache system. There is the head cache class that where the load and save functions are generically, we can provide for them any of our data class types and the loading and saving works. For further functionality we have used inheritation to provide more specific features to our caches depending on the use case.

Finally, we have learned web scraping and the combining and creating a new dataset. The yliopistovalinnat.fi website managed by University of Helsinki did not provide an api, but for some of our functionalities we need data for the points that provided based on the matriculation examination diploma. Furthermore, it also gives us the restrictions for the programme, for instance the needed grade for admission from certain subjects. Also, this allows us more easily divide the programmes to different sections since they are automatically divided on the website.

Some of the features that we plan on implementing

Preference saving

User can select their favourite schools and fields, and these preferences show in the UI, so the user can click them, and the detail side bar opens up. We have a plan to store these preferences in cookies, and if we fail to make it work with cookies, we probably use a JSON file to store them.

Search bar

User can search schools and if they are found from the map, map centres on it and the sidebar for details opens. If the school is not found from the map, only the sidebar opens. From the sidebar, fields can be searched for the chosen school through another search bar or maybe a dropdown menu.

Filters

User will have an option to use various filters. For universities/colleges, user will have an option to filter universities and colleges, respectively. In the sidebar, where the university's/college's fields are listed, there is an option to show only Finnish or English programmes or both (same for Bachelor's/Master's programmes). For universities/colleges, we have a plan to implement a feature, where the user can feed their matriculation examination grades to the system and all the fields that you wouldn't have gotten in, filter out.

High school statistics

We would like to show the results of matriculation examination per school and the number of students who have taken the exam as well as the average grade from that school.

Graphs for point limits (for universities/colleges)

After user has selected school and a field from that school, detailed information about that field can be shown. We will visualize the point limit data to show the point limits have changed in the last 5 years.

Rating system (probably only for universities, maybe for colleges)

We will create an algorithm that gives each field a star rating, based on factors like how many points are needed to get in compared to the maximum possible points. Each university will get their rating based on the average rating of their fields.

Expand the web scraping features

Currently only universities data for matriculation examination admission are scraped, excluding DIA – engineering fields, which need to be scraped from another site. There is also need for scraping the points given if applied for universities of applied sciences with a matriculation examination diploma. Also apply the caching also for web scraped data.