Roger That Project

Software Engineering

Architecture Document

Monday 14th June, 2021

Authors:
Gasan Rzaev
(s3553213)
Valeria Mavceanscaia
(s3673952)
Denis Garabajiu
(s4142551)
Constantin Cainarean
(s4142152)

Lecturer:
A. Capiluppi
Teaching Assistant
D. Plamadeala

rijksuniversiteit
groningen

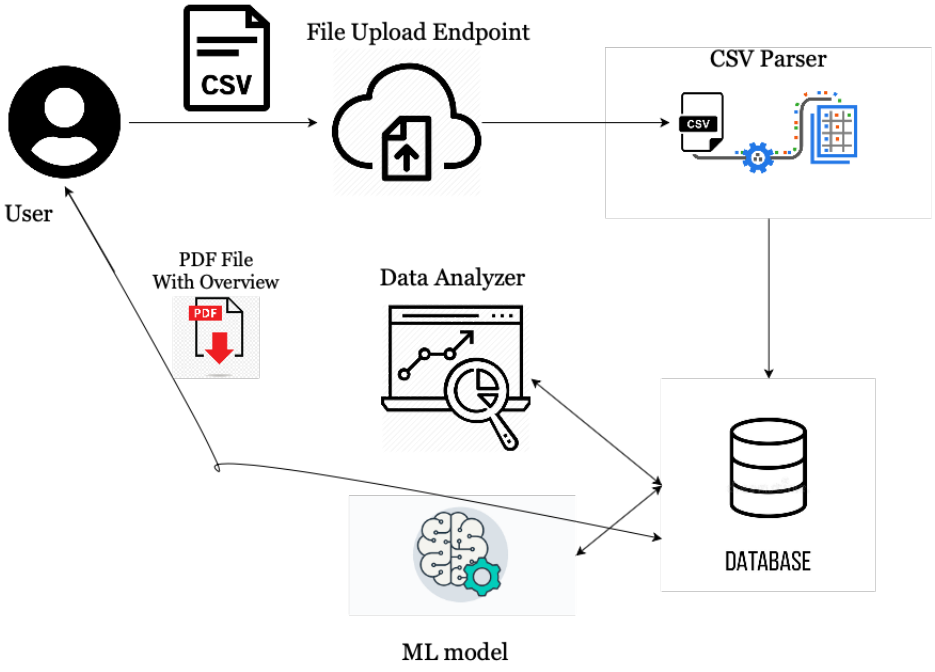
Contents

1 Introduction

An actual economical and social problem in the Netherlands is a relatively big number of people, especially young adults (18 - 35 y.o) with debts or bad financial situations. A lot of people do not even know how they are doing at the moment from a financial point of view. **RogerThat Project** is a tool for organizing financial information of an individual and providing feedback on the user's financial overview based on the user's transactions. The aim of the platform is to offer the user a better insight on his financial life.

2 General Overview of the Architecture

Figure 1: Graphical representation of the Project's architecture



The project is structured as a pattern of an MVP Architecture. The Database, which in our case is the **Model**, stores all the information that is necessary for the application to run as intended. Here we store all of the user's data, starting from the login credentials, which are of course encrypted, and ending with the transactions previously uploaded by the user.

The **View** layer is responsible for receiving the input from the user. The layer itself contains zero logic, it helps and eases the interaction of the user with the core functionality of the platform. Here the user is able to login or register, access his data, upload his files with the financial transactions, receive the overview of his financial activity in a PDF format. Since the RogerThat app is designed as a web app, the web page itself serves as the View layer.

The Back End (with tools such as Data Analyzer, CSV Parser etc) - serves as the **Presenter**. This is the part where all the checks and operations related to the functionality of the Web App will be processed and executed. The financial transactions are analyzed from various points of view, and in different scenarios. The required information will be later on retrieved from the Model, and after the calculations, displayed in the View layer, and updated in Model. Since Model and View do not communicate directly, the interaction between these two parts is happening here.

3 Front End

In this project, the front end is the web application which is an user-friendly interface that helps the user to interact with the functionality of the project. This is possible due to communication with the back end via HTTP requests and responses. It is divided into several parts which will be discussed in details in the sections below. For this part we have chosen to use the Angular Framework combined with the NG-Zorro library. This decision was taken by us after thinking from all of the perspectives. First of all, from the client side, our team did not have any requirements nor any recommendations regarding the front-end framework. Secondly, for most of our members, this framework was familiar. Also Angular Framework has a bunch of advantaged, we found some to be :

- Considerably reducing the development time, as the framework does not require writing additional code to provide continual View and Model synchronization. Seeing the output "live" as You change the code is a strong benefit!
- The dependencies that we are able to implement in Angular. They define how different pieces of code interact with each other and how the changes in one component impact the other ones. Usually, depen-

dencies are directly defined in the components themselves. So that every change in dependency requires changing components as well. With Angular, we can just use the injectors that defined dependencies as external elements decoupling components from their dependencies. Dependency injection made components more reusable, easier to manage and test.

- The framework is simply fast and performant.
- Readability of the code and the way the files are structured, helped our team a lot when it comes to division of the work. A person could easily complete the task that was originally started by someone else.

The only disadvantage of this framework is that there is always more than one way to complete any task or to implement a feature, which sometimes differ due to the versioning of the framework. This sometimes leads to confusions and struggles, however, thanks to a pleasant community and the abundance of tutorials and issue discussions most of the problems can be resolved.

3.1 Login and Register

The first thing that the user sees when he enters on the platform is the form that will be used to login or register to the platform. All credentials are sent to the back end via HTTP requests, and in the back end they're encrypted. If the login/register action was performed with success, the user will be redirected to the main panel of platform, else an error message will appear describing what went wrong.

Figure 2: Authentication process

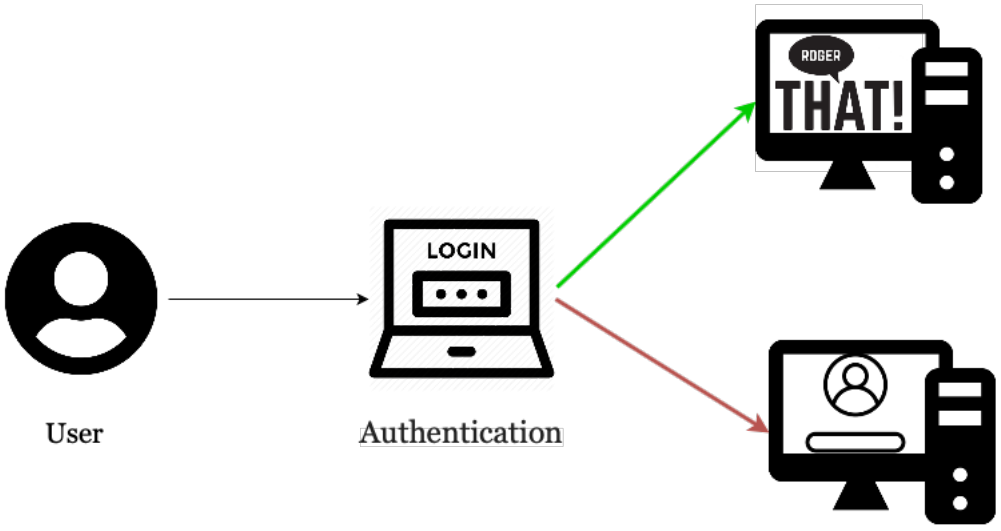


Figure 3: Graphical SSD for Logging in

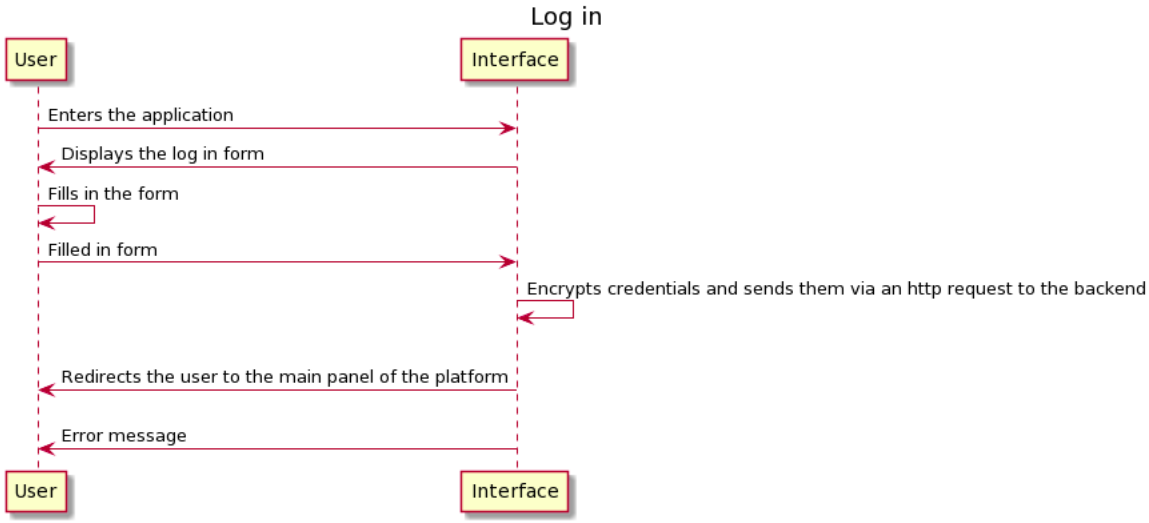
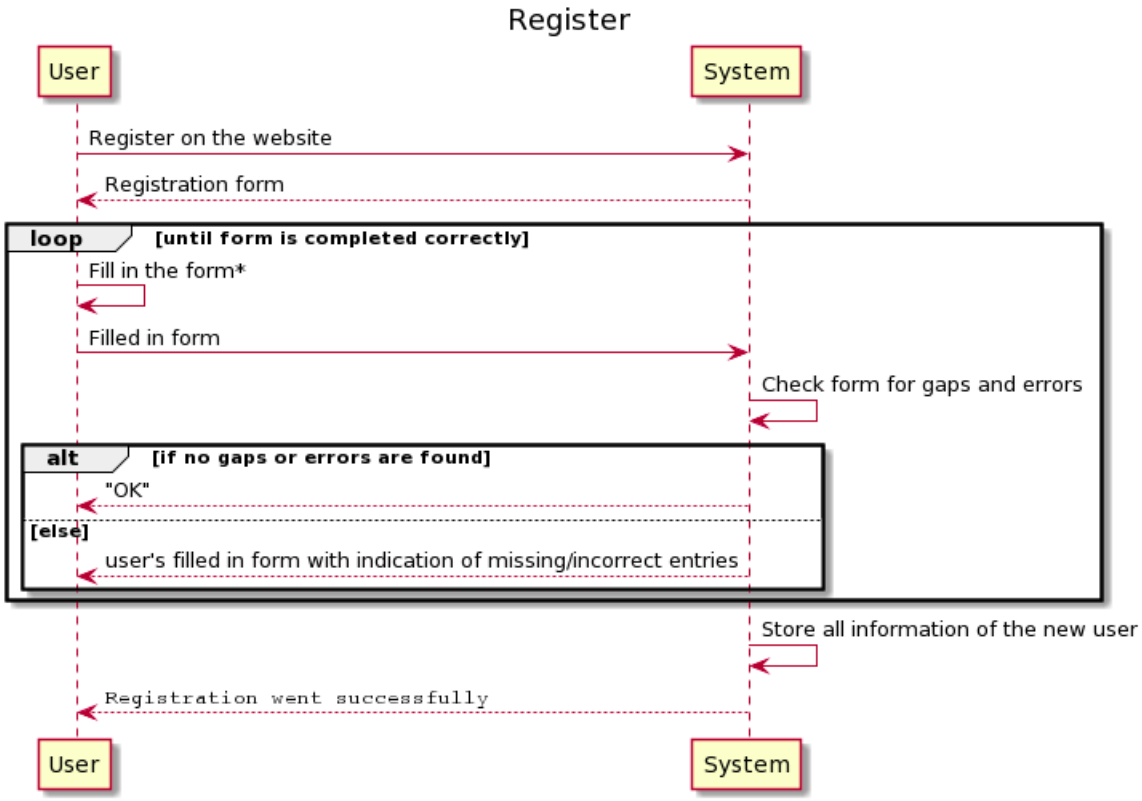


Figure 4: Graphical SSD for Register



3.2 Main Panel

Main Panel is the working space of the user and it consists of various sections, in each of them the user will be able to perform different actions. The panel is a folding slide menu, which contains all the functionality of the interface.

3.2.1 My Profile Section

Here the user will be able to view own personal information. The information displayed is the one that is kept in the database, which was added after the user has registered to the app and has input the information in the register form.

3.2.2 Settings

The settings section is meant to improve the usability of our web application. On this page the user is able to change the current e-mail that is registered to the account to a new one. Also the user has the possibility to enter a new password that will be replaced with the old one for the authentication process. To confirm the choice of the new password, it should be entered twice, and in both fields the passwords should be the same. In this section the user is also able to bind a new phone number to the account, the number should be entered with the plus sign(+),the country-code and the actual number itself.

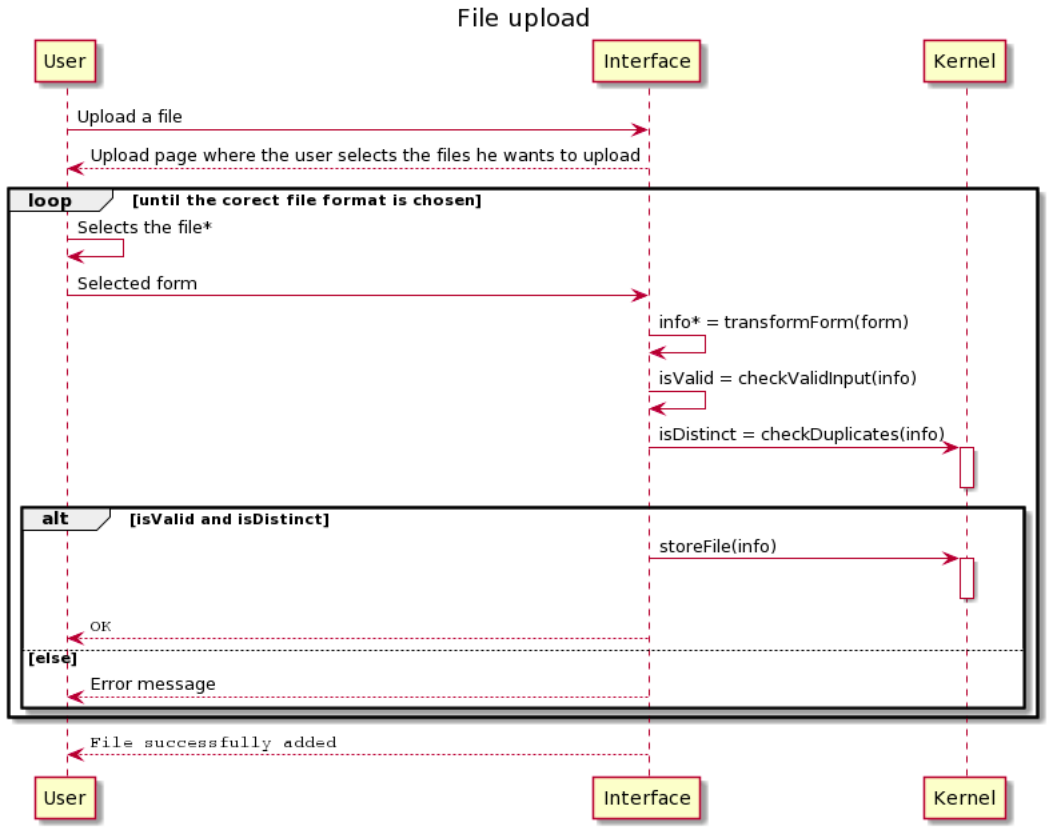
3.2.3 Contact us

The contact us sections, as the title suggests, is meant to offer the user the possibility to get in touch with the RogerThat team.

3.2.4 Status Section

This section is the most important part of our front end in terms of app functionality. The user will upload a CSV file containing his transactions. This file will be sent as a byte stream to the back end and will be stored in the root of the program. The following operations done on the uploaded file will be discussed in the Back end section. In this section the user will be able to see his short current financial status (a bar from red to green) and will be able to download a PDF file containing a detailed overview of his financial situation.

Figure 5: Graphical SSD for File upload



4 Back End

As previously mentioned all the computation and data processing is done in the back end of the application. The computations and data processing are divided into smaller parts of the back end. Additionally, there are several endpoints which are there for the purpose of communication with the front end. The following parts will be discussed in the subsections below.

4.1 CSV Parser

CSV Parser is the part of the application that accepts CSV files from the endpoint, parses the file and persists it to the database. CSV Parser accepts the line of CSV file, converts it into the token list. Afterwards, the token list's elements are passed on to the Transaction object created specifically for the Database, with its attributes being equivalent and in right order to the columns of the database table for each transaction. Finally, every transaction is being persisted, using Panache ORM method $persist()$ to the database right after passing the values of the line are read.

4.2 Database

MySQL is used as a service for the database development and manipulation. And the main purpose of it is to store all the bank transactions of all users. However, apart from transactions the database will hold a purpose of holding the rest of the information about users. For instance, user's personal information, user credentials for logging in and registering is stored in different tables. Additionally, database holds information about the transactions that were generated by analyser, such as spending and income classification.

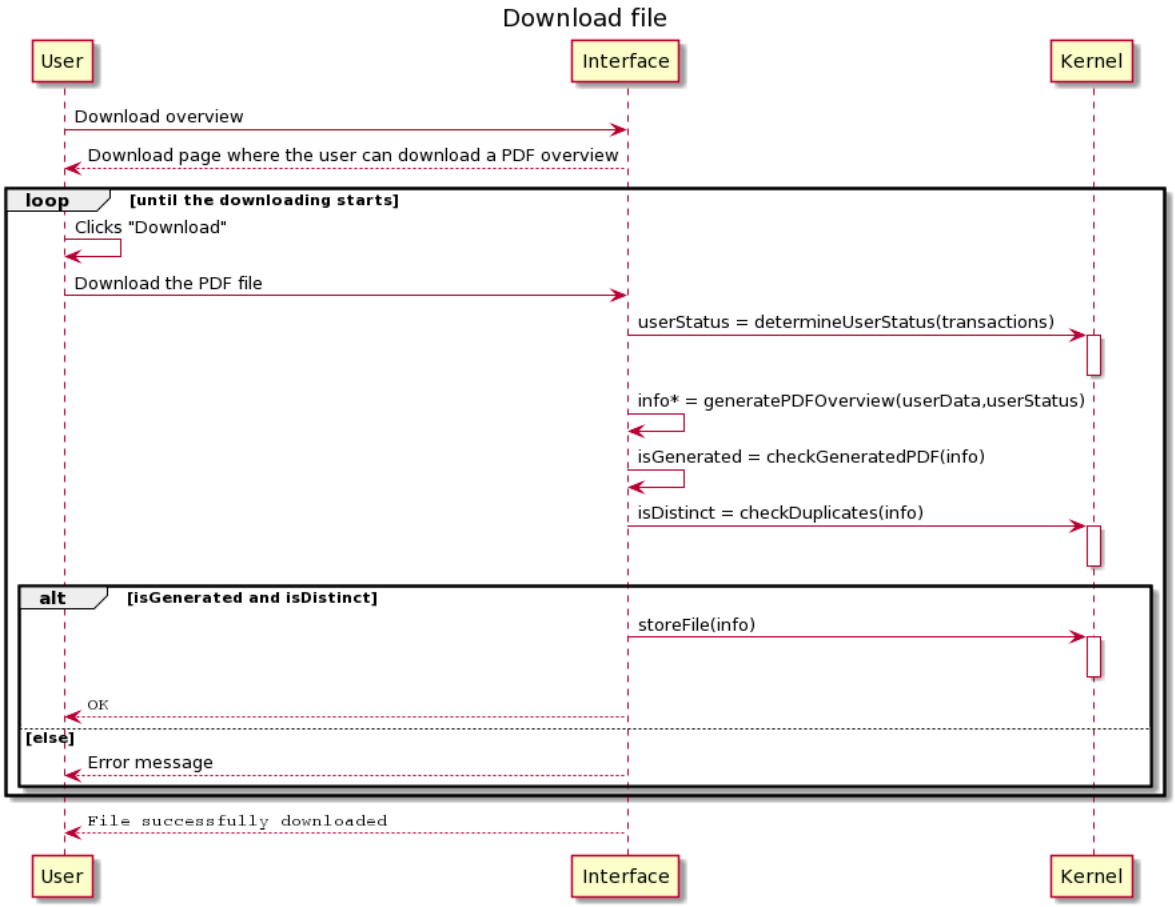
4.3 Data analyser

Data analyser is one of the vital parts of the back end, since its purpose would fulfill the main feature of this project. One of the first and simplest tasks that it solves is the check of income and spending recurrence. The check will mainly test the ratio between two in the past $1 - 6$ months. Moreover, the classification of each transaction is going to be introduced to make it easier for the PDF generator to sum the financial overview up.

4.4 PDF generator

PDF Generation is using a java library called *IText* (version 5.5.10) which is a tool that let's us import tables and images for a report of analysed bank transactions and an overall overview of a financial situation in a PDF document. In comparison with other tools like *Apache PDFBox*, this tool is easier to use and allows us to import tables from database queries, that are generated by Data Analyser, while *Apache PDFBox* does not support insertion of tables, hence would need a manual drawing for it. This part of the back end is a final step in order to give an overview and status of financial situation of a user.

Figure 6: Graphical SSD for Download PDF File



4.5 Authentication endpoint

The endpoint has a functionality of connecting front end with the back end and database manipulation. It is a vital part that apart from accepting HTTP request from the front end, also encrypts user credentials before persisting them to the database. For credential encryption *Java Base64* module was used.

4.6 File endpoint

File endpoint starts its functionality as soon as the user has access to the main page after successful authentication process. The functionality of the endpoint allows front end to upload a file, parse the file into an object that is later on sent into the database.

5 Technology Stack

Due to the fact that this project involves a web application, a full functional back end and a DB for storing data we have a vast stack of technologies. As mentioned above, since the client gave us the full freedom of choice we have the following :

For the front end we use Angular + NG-Zorro library. We chose this combination of technology because it gave us the possibility to develop a qualitative web application without any design skills. Also because it uses typescript language it was fast in development.

For the back end we have used Java with Quarkus. This stack was chosen because we were all familiar with Java and it is a language that fits perfectly for this kind of requirements and offers a lot of possibilities to work on with. Quarkus was chosen because it has bunch of functionality that fit our project, for example Panache ORM (a tool that makes the work with the DB much more capable). Also it ensures a simple compilation of the Java code.

For the Data Base we chose MySQL because it ensures the functionality we need for the project. The main point in this choice was that all of our team members have previously worked with it. Overall, it is a secure and performant database.

6 Design of the project

6.1 Front End Design decisions

From the architectural point of view, the decision of choosing the Angular framework was clearly explained in the sections above. In this part of the document, we would like to discuss some of the minor visual design decisions we have taken. To begin with the library NG-Zorro, which is an UI design component.

We have chosen this particular library due to the fact that it helped us to achieve a pleasant, animated and at the same time a personalized design of the application that suits the thematic and the colors of the client's company. Since the functionality of the platform is mainly happening on the back end, we have decided to keep the front end as simple and intuitive as possible, in order to ease the interaction with the app.

6.2 Back End Design decisions

Behind the "scenes", on the back end side of our application, we are using Java Framework Quarkus - which acts as a compiler, that helps us to run our project. This framework is working in combination with ORM Panache which is an utility that helps us to connect and maintain the connection between the Data Base and the Java code, which in the context of our project makes the operations on user's transactions much easier.

Quarkus has a lot of libraries that is used in order to connect the front end with the back end using end-points. Additionally, the end-points that connect front end with the back end, let us manipulate the database directly, which is useful for all of the features that involve the database usage.

Compared to other Java Application Frameworks, such as Spring for example, Quarkus has a much faster start up time, this is due to the fact that it uses "Ahead of Time compilation" strategy, which basically means that the compilation occurs during the building of the application. This gives the benefit to loading only the necessary classes which helps to start the application. Also, after consulting some of the internet sources, we found Quarkus to be more performant, moreover using less memory than it's concurents.

7 Team Organization

To better organize all of our common and individual tasks, we have used Trello, which is a list-making applications that allows to organize the projects into boards, so everyone can see what's being worked on, and who is working on.

To be able to better organize the Git repository of our team we have set and adhered to a set of rules. In short, before starting to work on a task, one must create an issue with the name of the task, create a branch related to the task, and when the task is finished and is fully operational, a merge request with master branch is created.

Communication with group members was essential for a better understanding of some problems that have occurred during the development of the application, as well as the faster solution of them. We had regular meetings and group calls with the team, where we discussed all of the tasks we have to do, the ideas that we have, some optimisations on the current stage of the application.

After every sprint, we had a group meeting where we have discussed and analyzed the mistakes we have commited, and of course we were deciding on how to refactor the code or the documentation implementing the received feedback.

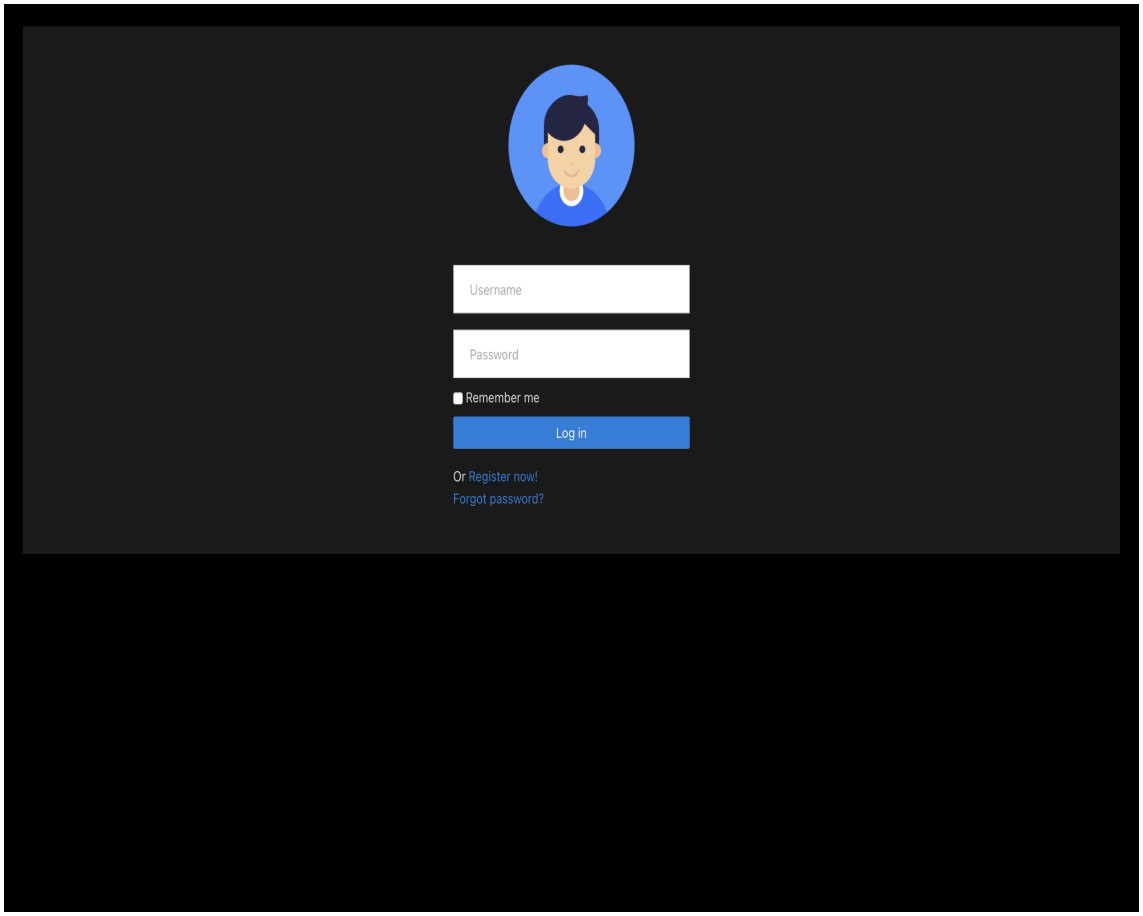
8 Future developments

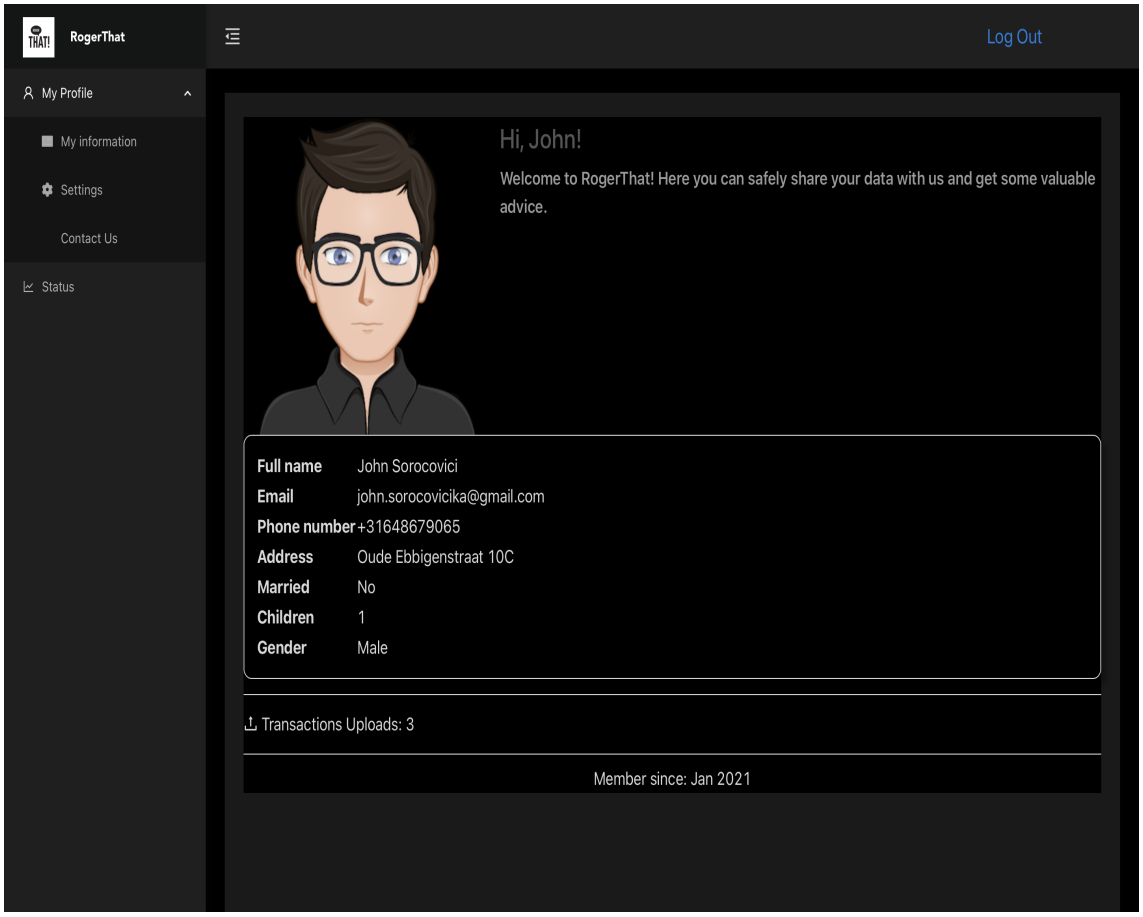
Due to the fact that the goal of the project is analyzing the financial information of the user, there are plenty of possible ideas that could be further implemented on the analyzer. Some of ideas that we thought of would be :

- To check whether the user is withdrawing cash more than paying by card.
- The total amount of money paid for house services, rent every month.
- The amount of money owed from or to someone.
- The current amount of money taken as a credit from a bank, or the remaining amount of money from mortgage (if applicable)

In the future, when the application will be launched to work with real users, the RogerThat team could implement some additional pages with policies, FAQ, "About us page".
Of course, a possible extension of this platform could be a mobile app.

9 Overview of the app





10 Traceability Matrices:

Business Requirements Document:

BR#	*Module Name*	*Applicable roles*	*Description*
BR1	Login and Logout	User	The user is able to log using the login page
BR2	Download PDF overview	User	The user is able to download an overview of his financial transactions with a conclusion on his current financial status
BR3	Upload transactions	User	The user is able to upload his bank transactions as a CSV file
BR4	Financial status	User	The user is able to observe his financial status (on a scale from red to green) based on the data collected from the bank transactions.

Technical Requirements Document:

Login T1:

Username is not blank

Password is not blank

If username and password are valid, log in

Download PDF Overview T2:

User has enough transactions uploaded

The financial status of the user has been analyzed and determined

The user generated the PDF Overview

Upload transactions T3:

The transaction being uploaded is in CSV format

The CSV file is a valid bank transaction

Financial status T4:

There exists provided data by the user to be analyzed

The transaction being analyzed is in the CSV format

Requirements Traceability Matrix:

Test Case#	*BR#*	*TR#*	*Test Steps*	*Test Data*	*Expected result*
1	BR1	T1	1) Go to login page 2) Enter username 3) Enter password 4) Click "log in"	**Username** = test **Password** = test	Login successful
2	BR2	T2	1) Go to "status" page 2) Find "Download detailed overview" section 3) Click "Download"	**Transactions > 0** **N = 1; N <7;N++** **Income** = user_income_month_N **Total_Income +=** user_income_month_N **Spending_recurrence** = user_sp_recc_month_N **Total_Spend_Recc +=** user_sp_recc_month_N **Financial_status** = ratio(Total_Spend_Recc, Total_Income) **User_data** = Financial_Status + User_Profile + Transactions **PDF_file** = generatePDF(User_data) **Download** = download(PDF_file)	File downloaded successfully
3	BR3	T3	1) Go to "status" page 2) Find "Upload your Transactions" section 3) Click "Upload" 4) Select a valid CSV file containing a bank transaction	**File** = selected_file **format(selected_file)** = CSV **selected_file** = isBankTransaction(selected_file)	File successfully uploaded
4	BR4	T4	1) Open downloaded PDF overview 2) Find the "Your financial status" section	**Financial_status** = ratio(Total_Spend_Recc, Total_Income)	The user is able to see his financial status categorized and analyzed

11 Change Log

Date	Name	Changes
25.04	Cainarean Constantin	Introduction Part
27.04	Cainarean Constantin	General Overview, Front end, Login and Register
28.04	Valeria Mavceanscaia	Login and Register diagram
28.04	Gasan Rzaev	Backend, CSV Parser, Database, Data analyser started
01.05	Cainarean Constantin	Main Panel, My Profile Section, Status Section
01.05	Denis Garabajiu	Settings section
03.05	Cainarean Constantin	Technology Stack
03.05	Valeria Mavceanscaia	File upload diagram
03.05	Valeria Mavceanscaia	General corrections/additions to the document
03.05	Denis Garabajiu	Improvements on previously added sections
03.05	Denis Garabajiu	Design section with the decisions on front and back end
28.05	Valeria Mavceanscaia	Layout improved, small corrections made within the text
28.05	Valeria Mavceanscaia	Front End section explained
31.05	Valeria Mavceanscaia	MVP Pattern explanation in the General Overview
04.06	Denis Garabajiu	Team Organization section added
05.06	Gasan Rzaev	Back End updated
06.06	Denis Garabajiu	Future developments section added
06.06	Denis Garabajiu	Added some schema's of architecture
08.06	Gasan Rzaev	Back End description added
12.06	Valeria Mavceanscaia	Traceability matrices created and added
13.06	Denis Garabajiu	Added new information / updated the old in all sections
13.06	Valeria Mavceanscaia	Created and added Graphical SSD for "Download File"
13.06	Valeria Mavceanscaia	Changed the layout
13.06	Denis Garabajiu	Updates on traceability matrix and overview of the app