HOMEBAKNING
Requirements and Report

for

Sysnovare Challenge

Prepared by : Pedro Marçal

June 7, 2023

Contents

1 Introduction

1.1 Purpose

For instance of the proposed challenge by the company to develop a front-end application of a Home-banking System using Angular for employment admission.
This systems' purpose is to provide the user the possibility to monitor his bank account and perform basic operations, like deposit and withdraw to add or remove funds from the user balance.

1.2 Intended Audience

This home-banking system can be used by any client who wants to adopt it an customise it for their services as well as it can be used for personal financial control by any interested individual who understands the necessity of keep personal finances under control, monitoring every movement - incoming and outgoing.

2 Overall Description

2.1 Product Overview

This application creates a space where registered users can access and monitors their financial movements.
After register and therefore logged in the system the user must be able to check the realised bank movements, such as deposits and withdraws as well as the account balance. Entering the main page, the user will find two containers. The wider container centred on the screen displays the entire list of movements made by the logged user. Its first operation is a deposit of 100€ at the sign up moment - *typical bank condition is to deposit a defined amount to start the account* - and after that will come all the others realised by the user.
The user will find on the left sid of the page a smaller container that will have two user informations - name and userBalance - and two buttons corresponding to the movement operations available for the user - deposit and withdraw. Clicking on each button, the page will open a prompt for the user to introduce the amount to deposit or the amount to withdraw, when clicking "Ok" at the end, the user balance gets updated and a new entry will show on the movement list.

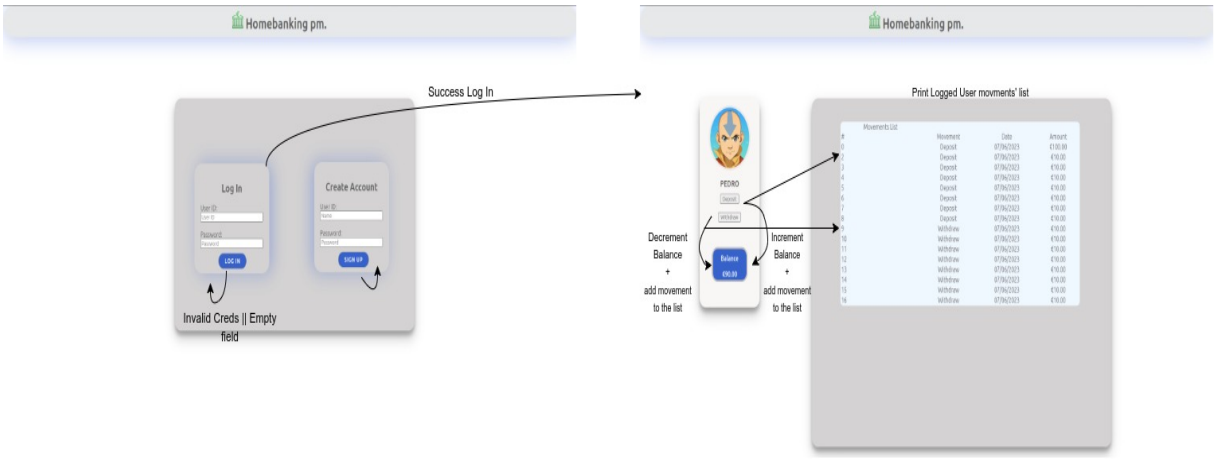

Figure 2.1: Homebanking diagram

2.2 User Class

"Homebanking pm." has 1 type of user. The user has 5 components

- User
  - name: string,
  - password: string,
  - userID: number,
  - userBalance: number,
  - userMovs: Movements[],

Movements is defined as an object. This type of object is defined this way,

- Movements
  - movNumber: number,
  - movType: string,
  - movDate: Date,
  - movAmount: number,

Each user has its own list of movements. That list will be shown in the main page on the Movements container.

2.3 Product Functions

3 System Features

3.1 Description

This home-banking system consists of a web application coded in Angular TypeScript. The system allows the user to create an account, introducing the name and the password for the account.
After the register process the user can log into the account page and monitor the movements made, check the balances and perform new movements. The movements available are deposit - it increments the user balance by adding the amount indicated by the user on the opening prompt - and withdraw - it decrements the user balance by adding the amount indicated by the user on the opening prompt.
Every time the user realises an operation, the balance will increase or decrease accordingly with the operation and a new entry will appear on the movement list.

Resuming, this systems allows:

1. **User Registration** : This allows users to sign up in order to log in. User data is stored on a test array working as a "fake backend".

2. **User LogIn** : This allows registered users to have access to the accounts. User registers array is read and if the introduced data match the existing one, user is allowed to proceed to the main-page.

3. **Movement Display** : The core feature of this system is the movement list display. Big container displays every movement stored in a specific type - Movements - array. Every account first movement is a *100€* deposit - account starter. Every time an operation takes place, the list will increase with the new entry.

4. **Deposit Funds** : This function will generate a new entry on the movement list and will increment the user balance. It colects the amount to add via a prompt for the user to indicate it and the current date, the movement type is defined directly inside the function.

5. **Withdraw Funds** : This function will generate a new entry on the movement list and will decrement the user balance. It colects the amount to deduct via a prompt for the user to indicate it, collects the current date and the movement type is defined directly inside the function.

3.2 Functional Requirements

"Homebanking pm." website is being build on Angular framework.

1. Font-End - Angular, HTML, CSS, TypeScript.
2. User must log in;
3. User must see the movements list;
4. User must add funds;
5. User must take funds;

4 Other Nonfunctional Requirements

4.1 Security Requirements

There is no implementations of data security for the accounts created. But the page can only be accessed only by registred users.

5 App Interaction and Conclusions

5.1 App Interaction

Opening the app, the user must first sign up. The log in does not recognise the test array initially user with the pair [admin: admin]. After sign in the user must introduce the registered account and press log in. If the app finds the match, a new page is loaded presenting the user area and the movement list.
On the user area, the user will find an avatar icon - later might be possible to give user the opportunity to add its own photo - the registered name in upper case below the icon. After that, the user have two buttons for the two available operations and at the bottom the account balance. By clicking any of the buttons the browser opens a prompt for the user to introduce the amount to operate. If the button is deposit, the user will add the introduced amount to the balance, otherwise the introduced amount will be taken from the account balance.

5.2 Conclusions

It is a simple web app to monitor balance, view the list of movements and also allow to manipulate the balance and add items to the list of movements by depositing and withdrawing funds.
It allows the user to register and log in, even though the test user implemented [admin, admin] is not working and the users array gets lost after the page refresh or moving back to the log in page. It needs an improve. Other than that, security on the passwords was not possible to implement and the authentication process is really basic, consisting only on find a match from the users stored on the array.
As my first experience with Angular, this helped me to try a new language and learning a little more about web development.