**MyBank**

Software Design Document

Prepared by Antici Maria-Daniela,

Calotă Vlad-Marin,

Vasile Radu-Mihail

Software Engineering Laboratory

ACE CEN 3

**1. INTRODUCTION**

1.1 Purpose

This software design document describes the architecture and system design of the MyBank web application. Here you can find everything a user of the application might want to know and even a potential client of the Bank.

1.2 Scope

The aim of MyBank is to facilitate easy transactions and storage of currencies, to have loyal customers by digitizing the banking process and to bring to clients the offering of a digital variance of the classic banking process. One more thing that MyBank does is that it reduces the number of employees a bank needs.

1.3 Overview

The overview of this document is that it’s structured on various levels with multiple sub points that are defined strictly by a normal convention.

1.4 Reference Material

*https://drive.google.com/file/d/1-MFI1kc6vR3nHrBzc27yoOFp0U4tJmPX/view*

1.5 Definitions and Acronyms

The document conventions of the SDD are: Times New Roman, size 14.

The bold words represent critical technical data or signs of important paragraphs.

The italic words represent under discussion terms that may be changed at a later edit. ( are temporary )

2. SYSTEM OVERVIEW

2.1 Product Perspective

The benefits of using our system is instant updates because we will use a dedicated database that is linked to the website. It will have a clear interface with no additional corporate functions that confuse the client. Another benefit is the lack of ads.

2.2 Product Functions

● Login

● Balance infos

● Changing the address

● Changing the password

● Financial operations between clients

● Utilities payment

● Transaction history

2.3 Operating Environment

A web application (or web app) is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system (OS) of the device. Web applications are accessed by the user through a web browser with an active network connection.

Google Chrome Version 89.0.4389.82 is required.

2.4 Design and Implementation Constraints

Timing requirement: 11 weeks.

IDE: Visual Studio 2019

Front-end: HTML5, CSS3, JavaScript, Bootstrap 4.6

Back-end: C Sharp, ASP.NET MVC Core 3.1

Database: SQL Server on a Linux Container in Docker 3.2.1

Database Management: Azure Data Studio 1.29.1

Other tools: Postman 7.30.1 for creating and saving simple and complex HTTP/s requests, as well

as read their responses.

Testing: NUnit, Selenium

3. SYSTEM ARCHITECTURE

**----- RADUUUUUUU RADUUUUU RADUUUU**

3.1 Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Provide a diagram showing the major subsystems and data repositories and their interconnections. Describe the diagram if required.

**----- RADUUUUUUU RADUUUUU RADUUUU**

3.2 Decomposition Description

Provide a decomposition of the subsystems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object oriented description. For a functional description, put top level data flow diagram (DFD) and structural decomposition diagrams. For an OO description, put subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.

**----- RADUUUUUUU RADUUUUU RADUUUU**

3.3 Design Rationale

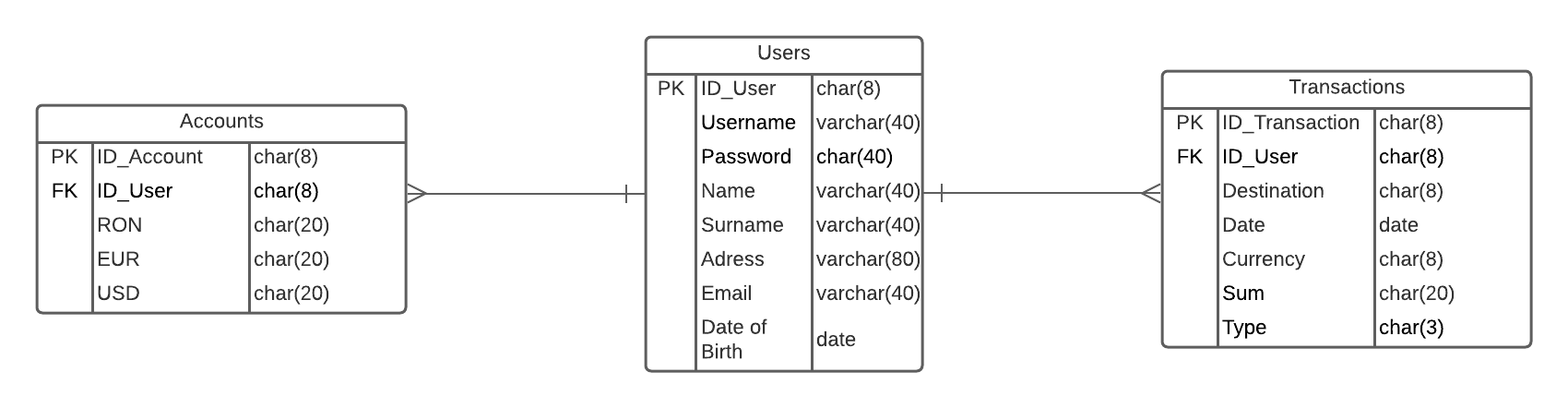
Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.

4. DATA DESIGN

4.1 Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

The data is data provided by a SQLServer database managed by Azure Data Studios.

The data is stored in our system by having various HTTP Post methods that are used to send data in our backend. We will use Postman for testing the various routes, be either GET or POST routes. With Entity Framework we use the database so that we add the sent data (after it has been worked on for the necessary tasks) to various tables that the application uses.

4.2 Data Dictionary

Accounts - The accounts entity stores the various aspects that a user has in his usage of the application. Here you have the RON, EUR and USD variables that store his money.

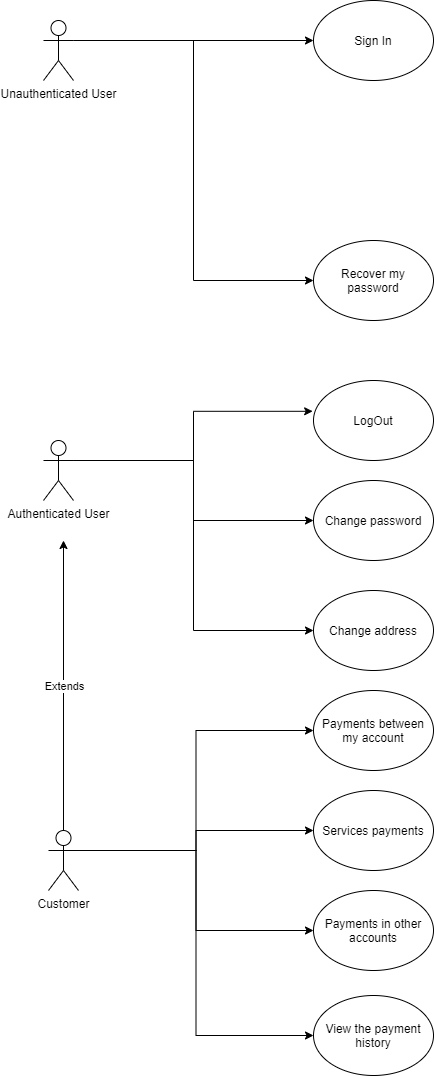
Users - The users entity is having the login functionality of the application closely linked to it as it has the entries that a user has to use.

Transactions - The transactions entity is the entity that stores every details about a transaction that has been done.

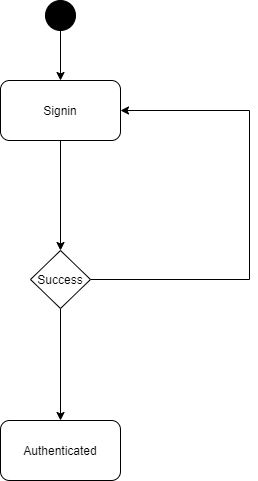
5. COMPONENT DESIGN

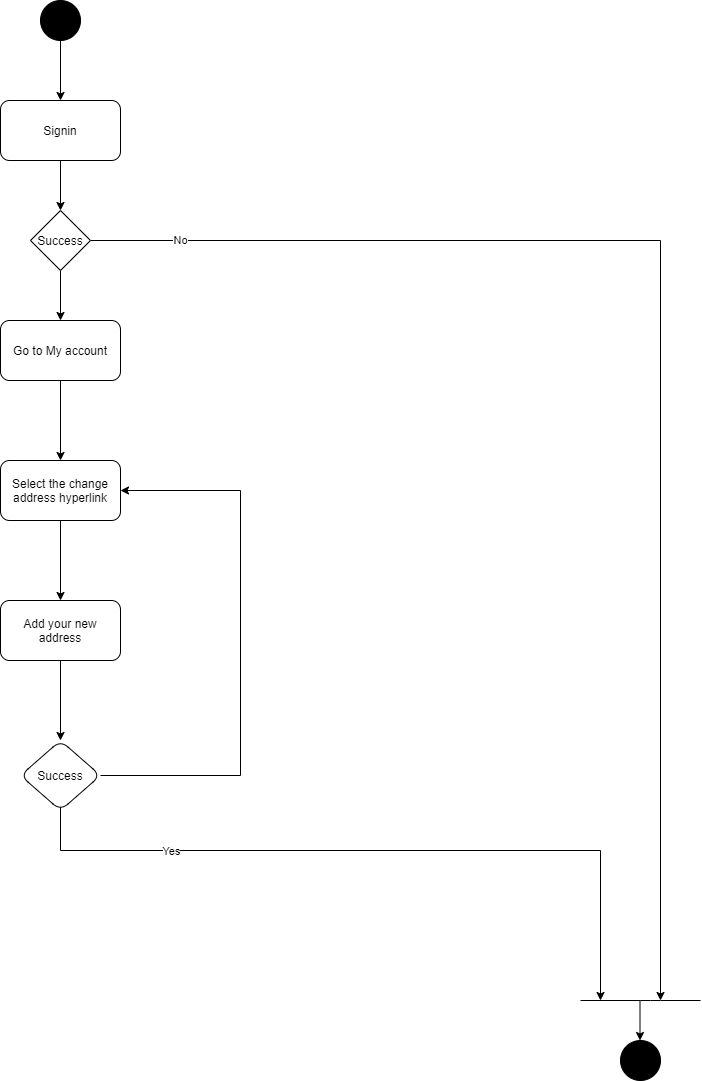
In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

**Use Case**

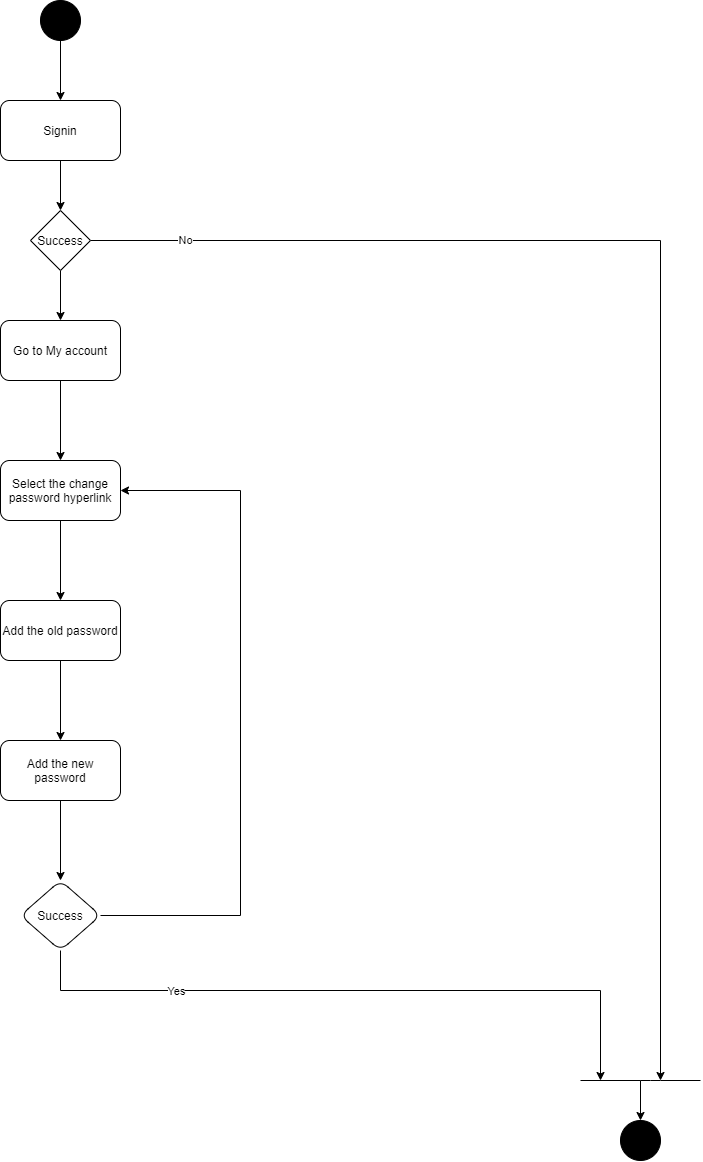


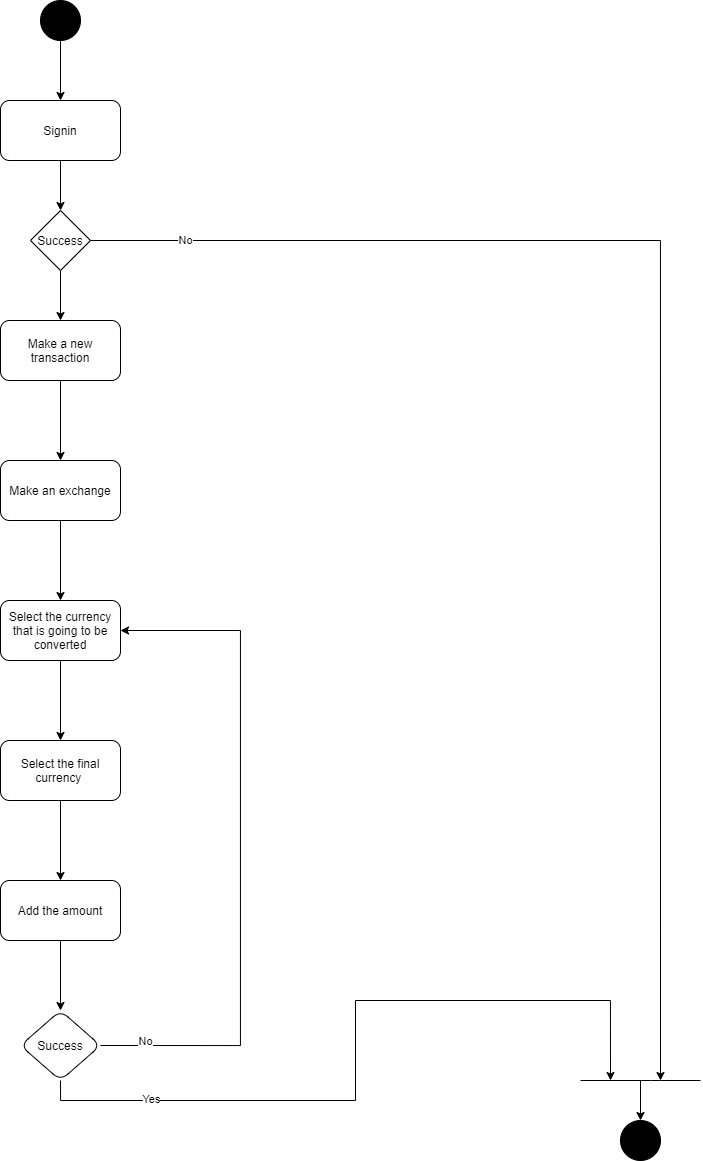
**Activity Diagram**

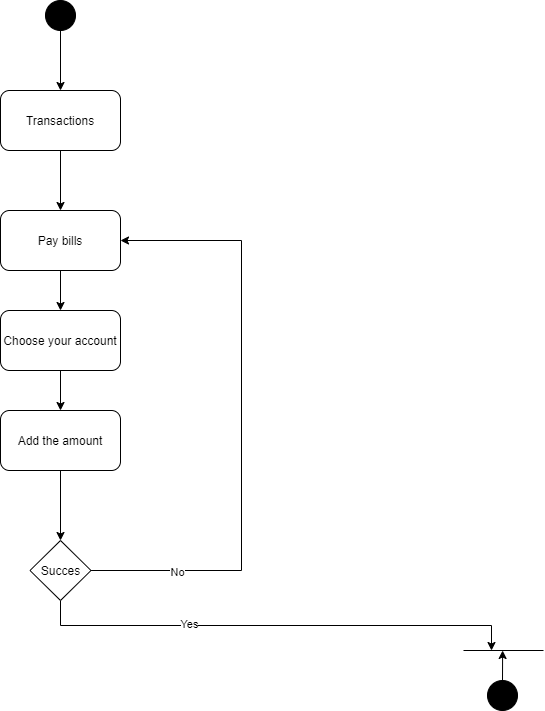
**Login  
**

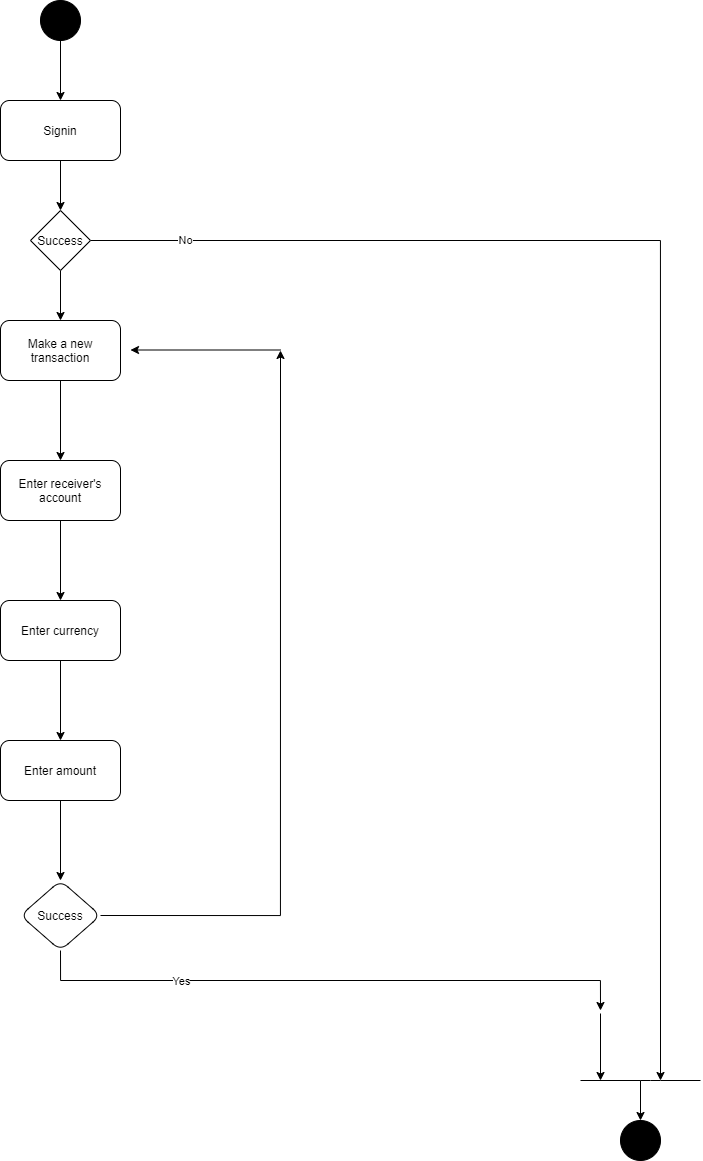
**Change Address  
  
**

**Change password**

****

**Exchange  
**

**Pay bills  
**

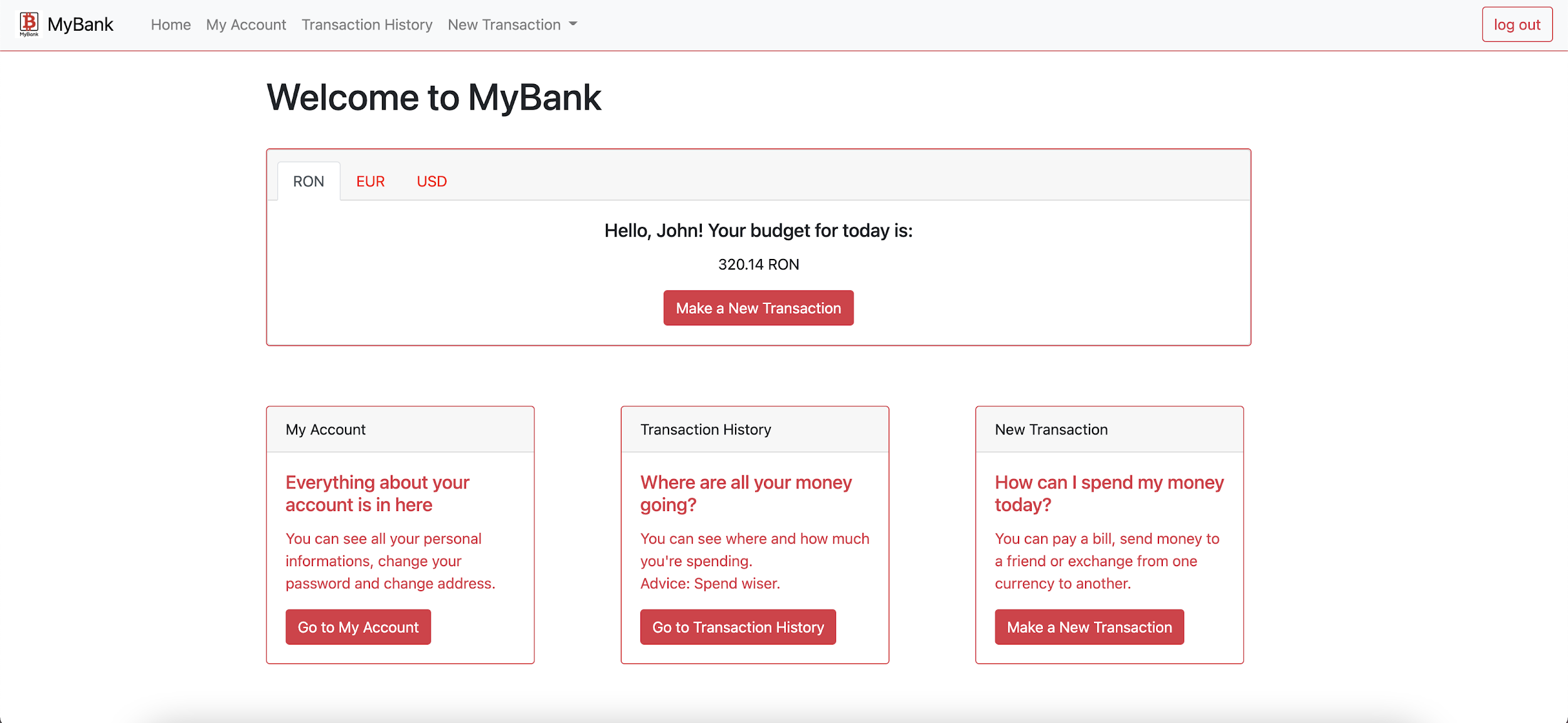
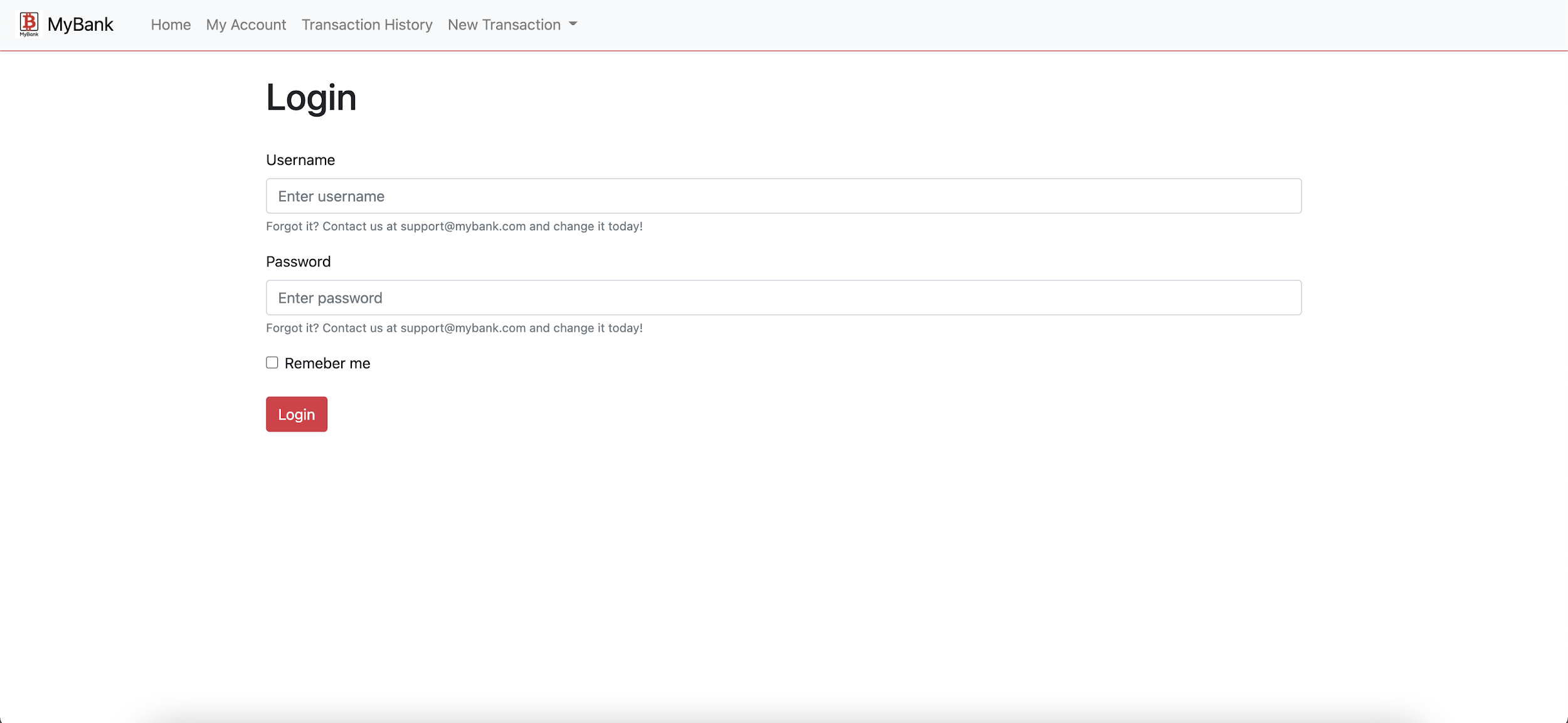
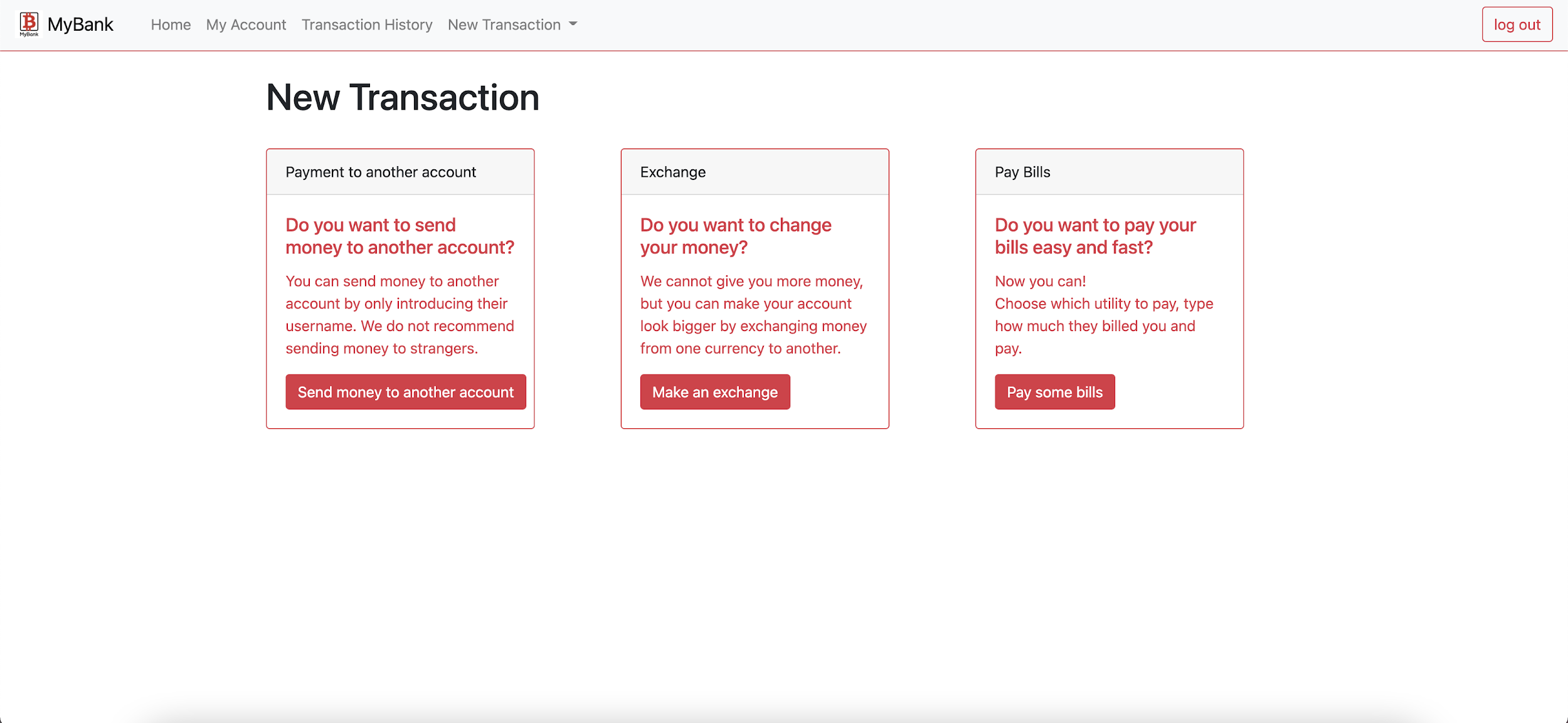
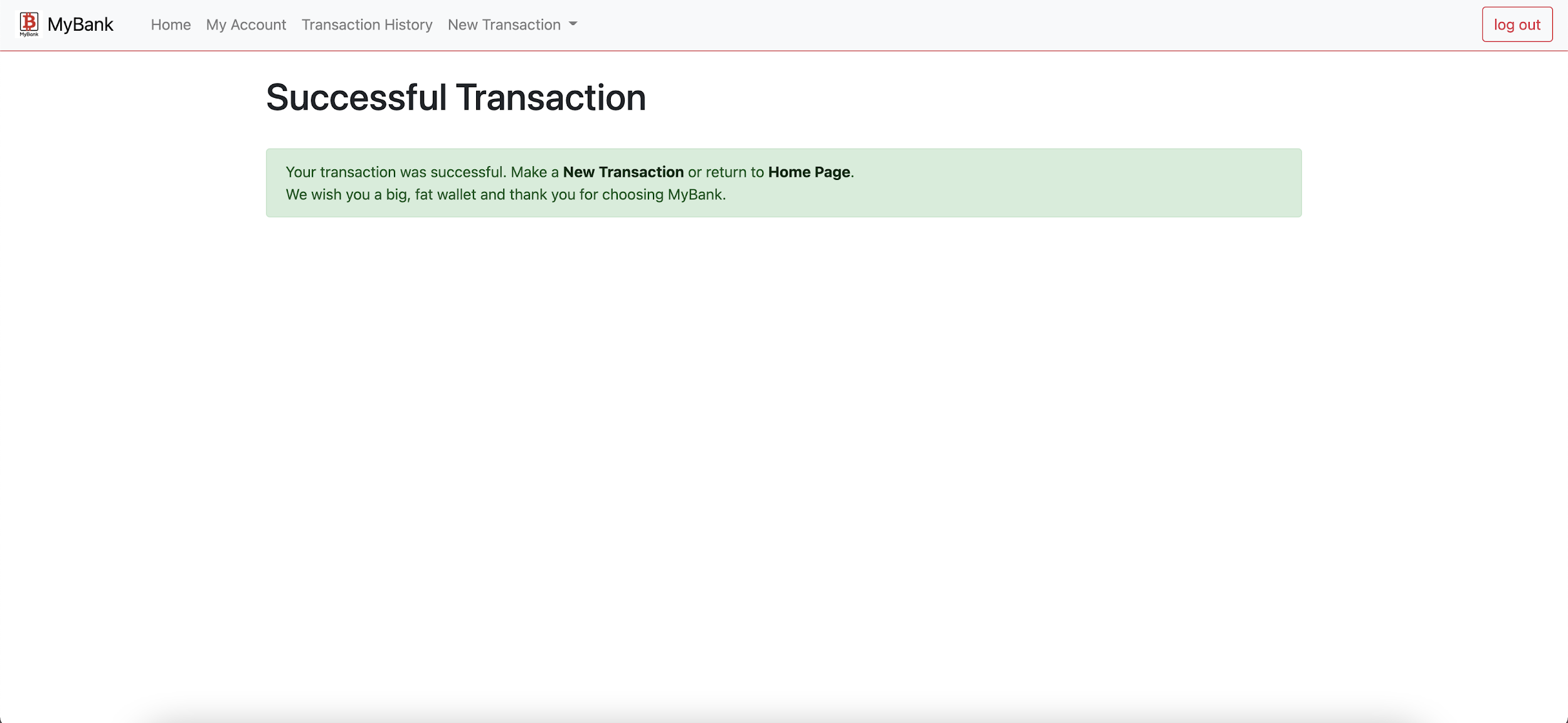
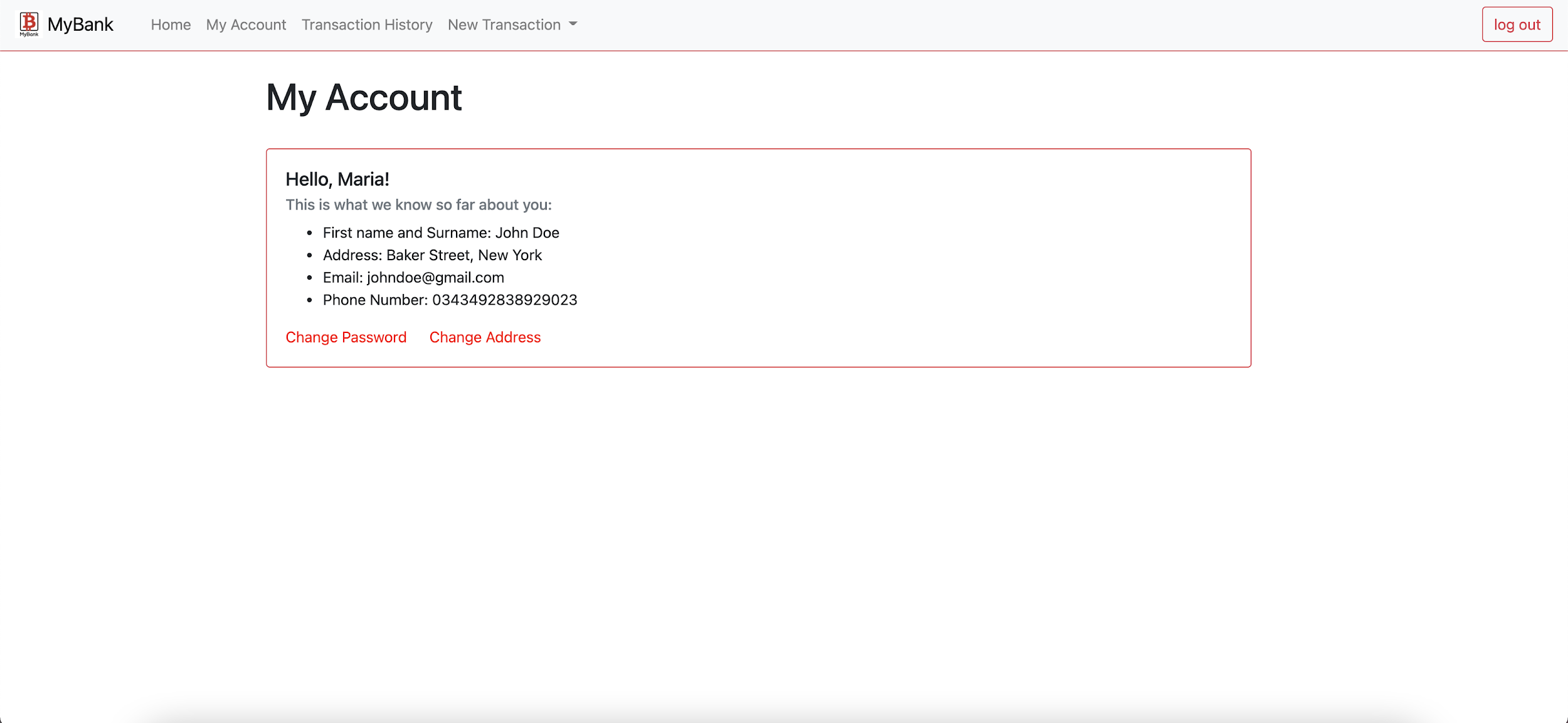
**Send money  
**

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

The functionality of the system from the user’s perspective is very complex. Even though you have a clean look you can fulfill all actions of the application from every page. Each functionality is implemented to have a specific page and a response page with no dead end. If you use the application you won’t get stuck in a page not knowing what to do next. The UI is user-friendly.

6.2 Screen Images

6.3 Screen Objects and Actions

You have a navigation bar that has most of the functionalities of the application set on it. You can also log out of the application from it.

On the main page you have the cards that lead you to various functions.

My Account leads you to your account page.

The top card shows you your balances.

Transaction History leads you to the transactions that you have done.

New transaction leads you to the page that has the new transaction function.

On the login page you have the usual login page with it’s typical elements.

Username, password and login. With a little remember me tick for the obvious function.

The new transaction page has the 3 different transactions you can do. Transfer to another account, exchange currencies and pay your bills.

MyAccount page has the credentials of the user presented in a typical html container.

7. REQUIREMENTS MATRIX

SRS: The web application will use a maximum of 2 gigabytes ( GB ). It’s loading time will have a variation between 0.05 ms and 60s. These performance requirements are deemed right only when the required software has it’s versions up-to-our-date.

SDD:

8. APPENDICES