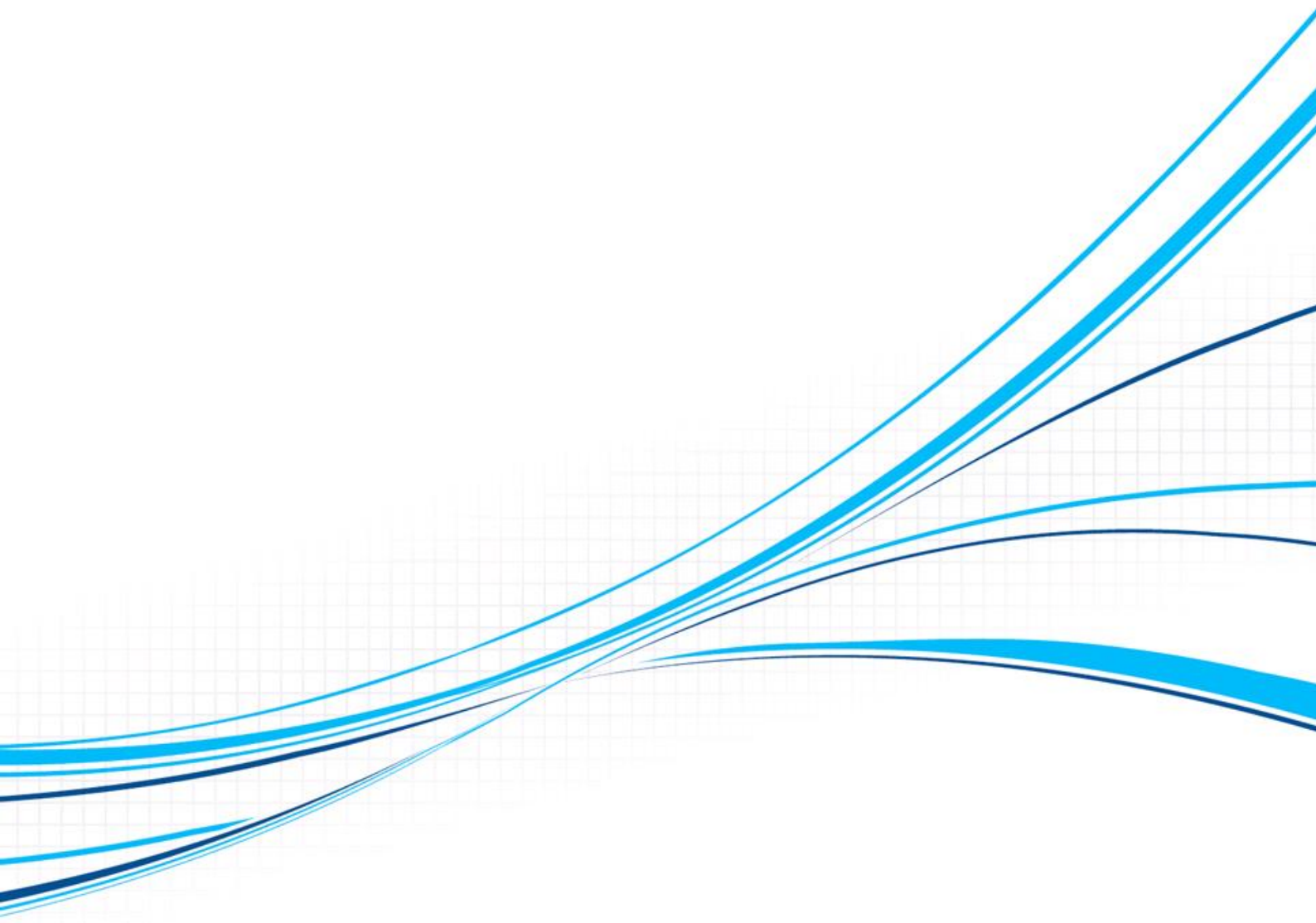
Software Requirements Specification

Advisor Project

Created by Antoine Ratat

Version 1.0 - Issued December 03, 2020

This requirements specification is used to record the user requirements.



Contents

1 VERSIONS

1.1 VERSIONS

Ver.	Author(s)	Date	Description
1.0	Antoine RATAT	03/12/2020	Creating Document

2 INTRODUCTION

2.1 PURPOSE

Financial is a tool to verify the current exchange rate and finance related information. This application to check the current finance rate depending on multiple currencies, convert in multiple ways, check finance related news and observe graph representation of exchange rate values.

2.2 PROJECT SCOPE

Financial is a programming project, that allows its developers to practice client-side programming using API calls, authentication and user management.

2.3 REFERENCES

React - https://reactjs.org/
React Router - https://reactrouter.com/web/guides/quick-start
React Bootstrap - https://react-bootstrap.github.io/
React Spinners - http://www.davidhu.io/react-spinners/
React Toastify - https://www.npmjs.com/package/react-toastify/v/1.4.3
Formik - https://www.npmjs.com/package/formik
Yup - https://www.npmjs.com/package/yup
React Dark Mode Toggle - https://www.npmjs.com/package/react-dark-mode-toggle
React Token Auth - https://www.npmjs.com/package/react-token-auth
Flask - https://flask.palletsprojects.com/en/1.1.x/
Flask Mail - https://flask-mail.readthedocs.io/en/latest/
ItsDangerous - https://pypi.org/project/itsdangerous/
Flask SQLalchemy - https://flask-sqlalchemy.palletsprojects.com/en/2.x/
Flask JWT Extended - https://flask-jwt-extended.readthedocs.io/en/stable/
Flask Bcrypt - https://flask-bcrypt.readthedocs.io/en/latest/
Cloudinary - https://cloudinary.com/documentation/image_upload_api_reference

3 DESCRIPTION

3.1 PRODUCT PERSPECTIVE

The product is a full stack web application implemented on both front and back end, it is using API calls to operate the database.

3.2 FEATURES

The Financial system provides a simple mechanism for users to acquire information.

The following are the main features that are included in the system:

- **Convert Currencies**: allow the user to convert from any source to any destination currency
- **Get Latest Rates**: search for the latest currency rate from the European Central Bank
- **Exchange Rate Graph**: draw an exchange graph for the selected currency available for different period of time.
- **Historical Exchange Rate Graph**: draw historical exchange rate graph for the selected currency in the last year
- **Historical Exchange Rate**: fetch and selected currency's rate on all the available destination currencies, along with the historical rate of the last month.
- **Finance NewsFeed**: fetch and provide the latest finance news feed related to finance trending topics.
- **User management**: allow the user to connect and save preferences such as theme color and settings like default currency. That information and preferences about users are saved for future sessions.

3.3 USER OVERVIEW

It is considered that the user does have the basic knowledge of operating the internet and to have access to it.

The administrator is expected to be familiar with HTML, CSS, JavaScript, React library, AJAX (Asynchronous JavaScript and XML), React Router, React Pagination, React Token Authentication, React ChartJS2, React Bootstrap, JWT Decode, Formik, Yup, Styled components to handle Front-End's side. Regarding Back-End perspective the administrator is expected to be familiar with Python, Flask, Flask-SQLalchemy, Flask-Mail, SMTP Satellite Configuration, CORS configuration, JWT Tokens, Authentication, and Password hashing mechanisms.

3.4 OPERATING ENVIRONMENT

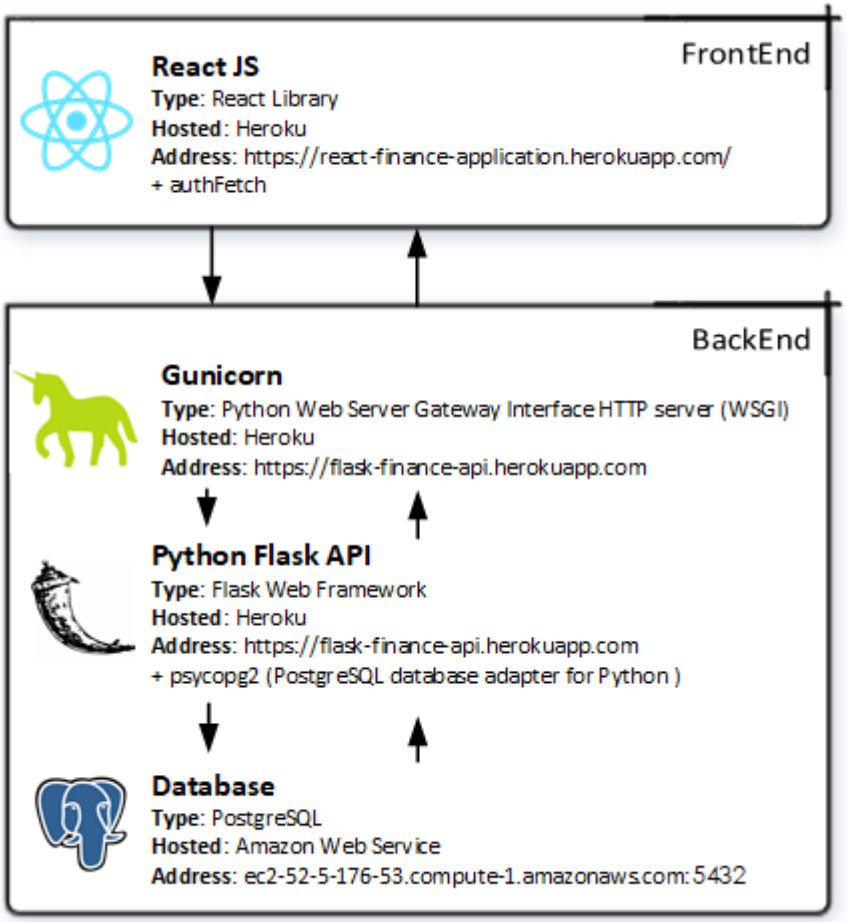
This is a web-based system and hence will require the operating environment for a client and server GUI.

Dependencies

- This software highly depends on the type and version of the browser being installed in the system i.e. browser version should be used which has HTML5 support.

3.5 CONTRAINTS : IMPLEMENTATION / DESIGN

3.5.1 Application Communication Schema



3.5.2 Client-side rendering (frontend)

This system is provisioned to be built in JavaScript using React library which is highly flexible.

The browser will be in charge of rendering this application in its final form, HTML. Some of the logic involved in creating the web page, especially the one in charge of dealing with presentation logic is handled on the client-side.

List of frontend dependencies and version used:

react-fontawesome: V0.1.11,
bootstrap: V4.5.3,
chart.js: V2.9.3,
formik: V2.2.5,
jwt-decode: V3.1.2,
react: 16.12.0,
react-bootstrap: V1.4.0,

react-chartjs-2: 2.9.0,
react-dark-mode-toggle: 0.0.10,
react-datetime: 2.16.3,
react-dom: 16.12.0,
react-dropdown-select: V4.6.0,
react-helmet: V6.1.0,
react-js-pagination: V3.0.3,
react-router-dom: 5.1.2,
react-scripts: V3.4.4,
react-spinners: V0.9.0,
react-toastify: V6.1.0,
react-token-auth: V1.1.7,
reactstrap: V8.4.1,
styled-components: V5.2.1,
yup: V0.29.3

3.5.1 Server-side rendering (backend)

The system is also provisioned to connect and contact a custom made API. This API handles CRUD request and manage operations on the database.

The API is build using Python with Flask web framework. The application server would be served with GUnicorn (Python WSGI HTTP Server for UNIX)

The API will interface the database using SQLAlchemy. SQLAlchemy is an object-relational mapper (ORM) and provides the data mapper pattern, where classes can be mapped to the database in open-ended, multiple ways - allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

Authentication will be handled using Flask-JWT-Extended which adds support for using JSON Web Tokens (JWT) to Flask for protecting views.

List of used dependencies and version used:

astroid V2.4.2
bcrypt V3.2.0
blinker V1.4
certifi V2020.11.8
cffi V1.14.3
click V7.1.2
Flask V1.1.2
Flask-Bcrypt V0.7.1
flask-buzz V0.1.15
Flask-Cors V3.0.9
Flask-JWT-Extended V3.24.1
Flask-Mail V0.9.1
Flask-SQLAlchemy V2.4.4
gunicorn V20.0.4
inflection V0.3.1
isort V5.6.4
itsdangerous V1.1.0
Jinja2 V2.11.2
lazy-object-proxy V1.4.3
MarkupSafe V1.1.1
mccabe V0.6.1
passlib V1.7.4

pendulum V2.1.2
psycopg2 V2.8.6
py-buzz V1.0.3
pycparser V2.20
PyJWT V1.7.1
pylint V2.6.0
python-dateutil V2.8.1
pytzdata V2020.1
six V1.15.0
SQLAlchemy V1.3.20
toml V0.10.2
urllib3 V1.26.2
Werkzeug V1.0.1
wrapt V1.12.1

The database itself is using PostgreSQL and is accessible for UI management tools like PgAdmin. It would be hosted on AWS (Amazon Web Service) and available on port 5432.

The database would contain two main tables: User and Settings. The table user would contain User information and the Setting table would contain associated user settings.

You can find an ER Entity Relationship Diagram to understand its structure:

Hardware Interface

The device should be enabled with the Internet.

Software interface

The user's browser should be HTML5 compatible for a satisfactory user experience.

SYSTEM FEATURES

3.6 SYSTEM FEATURE FRONT END

3.6.1 Not Connected Features

- Create an account
- Login
- Forgot Password
- Reset Password

3.6.2 Connected Features

Dashboard

- Consult Financial Dashboard
- Switch to Dark / Light mode in the sidebar
- Navigate to other website pages in the sidebar
- Access to Profile
- Disconnect

Profile

- Update Profile Information
- Upload Profile Picture
- Choose Default Currency (settings)
- Choose Theme (settings)
- Delete Profile Information
- Delete Account
- Consult User current information

Convert

- Convert source currency to destination currency
- Calculate input value to output value depending on the currency
- Change input and output currency
- Inverse source currency with destination currency
- Check the daily exchange rate for selected currencies

Exchange Rate Graph

- Check the exchange rate graph for selected currencies from 1 week, 1 month, 3 months, 6 months to a year time period.

Historical Exchange Rate Graph

- Check the history rate graph for selected currencies for a year time period

Historical Rate

- Check the history rate currency for selected currencies from 1 week, 1 month, 3 months, 6 months to a year time period.
- Check the historical exchange rate from input currency to all existing currencies
- Sort the historical exchange rate per currency, rate or, history value
- Filter the historical exchange rate per currency label or the currency name
- Paginate the results of the historical exchange rate currency to display only 10 results at once

- Graphic representation in the historical exchange rate of the negative and positive using variable indicative color arrows.

Finance Feed

- Check latest news feed related to Finance
- Sort the news per Brand, Date or, Score
- Filter the news per Article Name, Article Description or Brand
- Click the Article Name to read the new article (external link)
- Paginate the results of the finance feed to display only 12 results at once

3.7 SYSTEM FEATURE BACK END

3.7.1 User features

User account is only available after the first login with User account authenticated with User JWT

- Allow user to create an account
- Allow user to login
- Allow user to disconnect
- Allow user to save information in the database
- Allow the user to update or delete their information (name, password, address, etc…)
- Allow user to delete their account
- Allow user to reset their password through Email
- Allow user to upload a profile picture
- Allow user to save setting in the database
- Allow user to update or delete their settings (theme, default currency)

3.7.2 Admin features

Admin features are only available to Admin account authenticated with Admin JWT

- Allow admin to get a list of existing users
- Allow admin to update specific user information
- Allow admin to update specific user setting
- Allow admin to delete a specific user
- Allow admin to create specific user

3.8 SYSTEM FEATURE DATABASE

3.8.1 User Table

For each subscribed user store information such as user, email, hashed password, created time, first name, last name, birthdate, position, education, about user, address, city, postcode, country, profile picture URL.

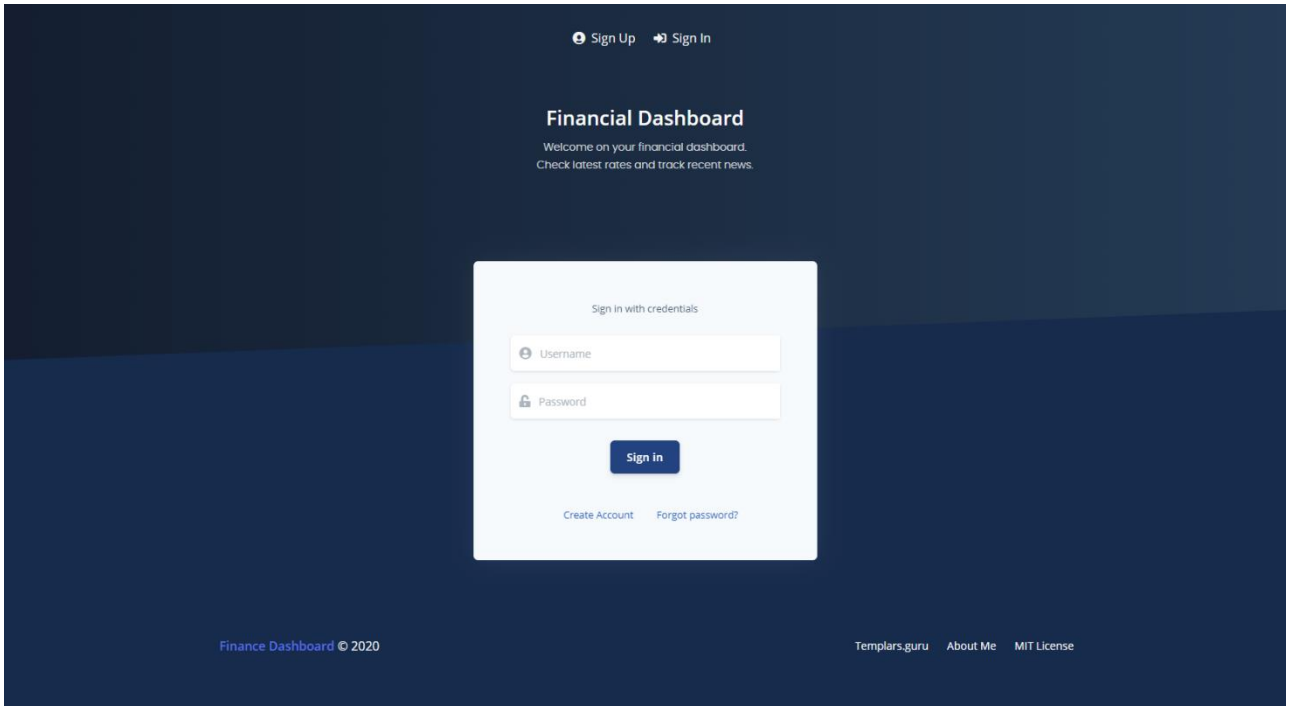
3.8.2 Setting Table

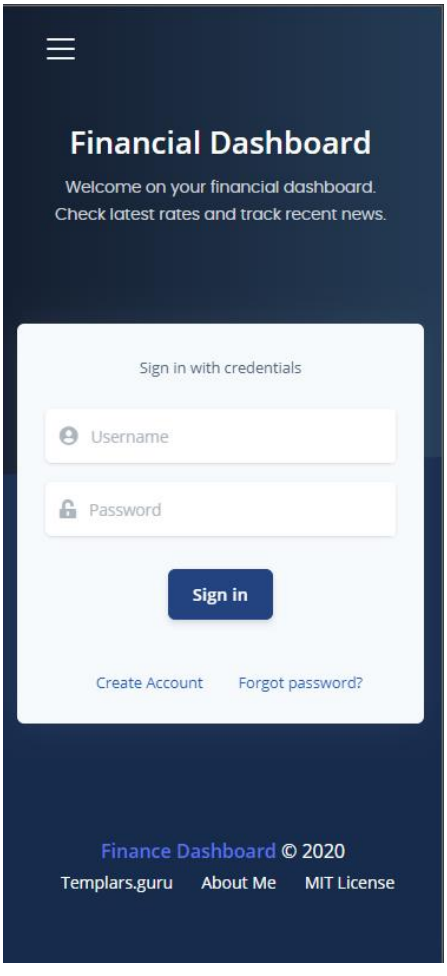
For each subscribed user store settings such as user theme and default currency

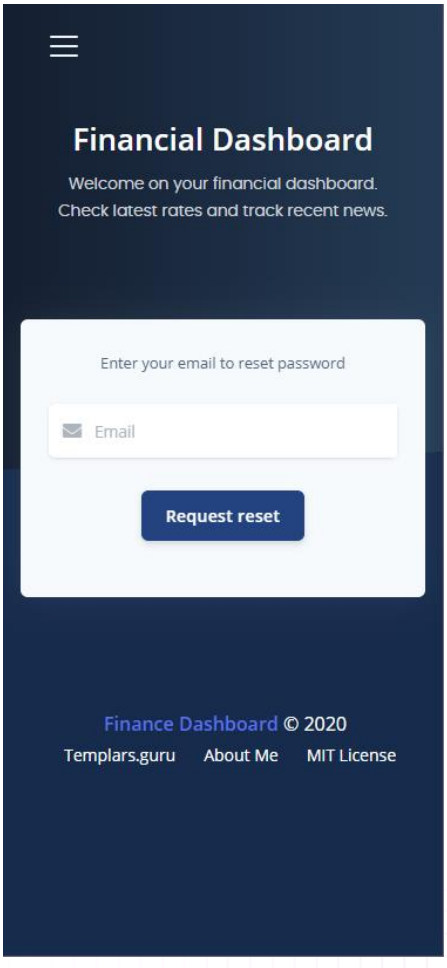
4 REQUIREMENTS OF EXTERNAL INTERFACE

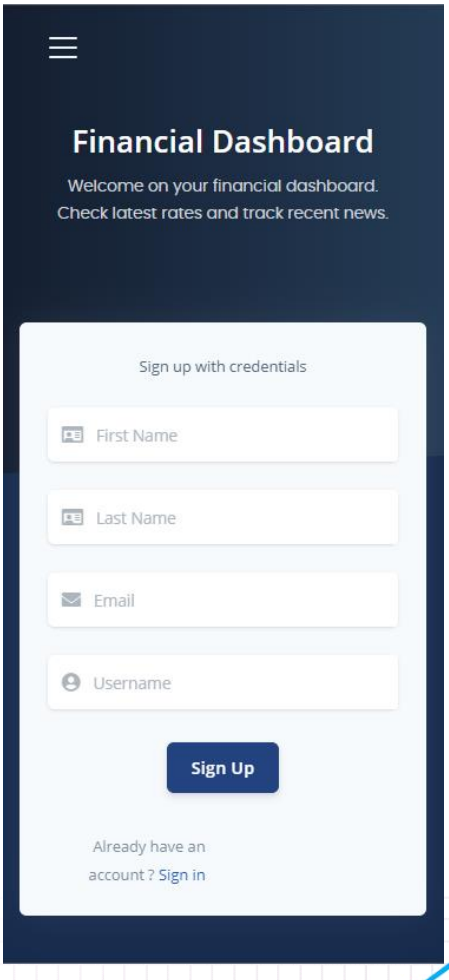
4.1 USER INTERFACES

Login Page – Desktop and Mobile Version

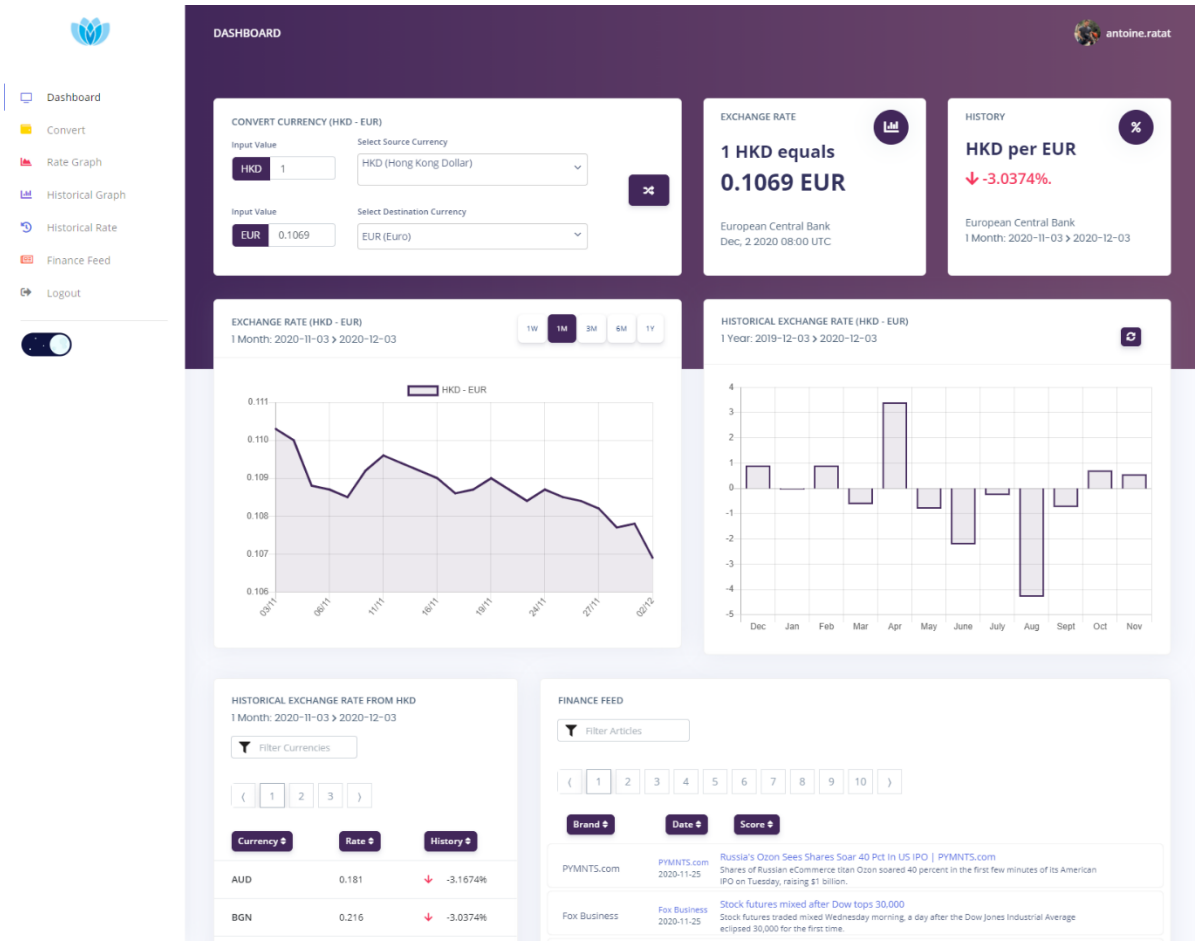


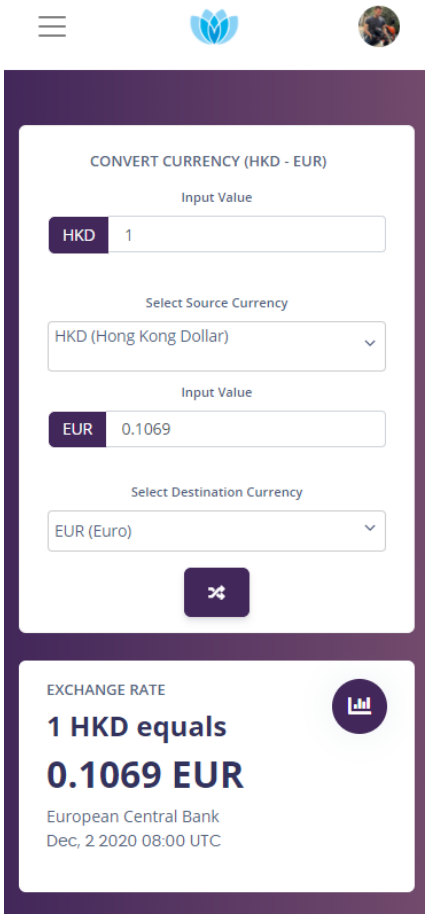


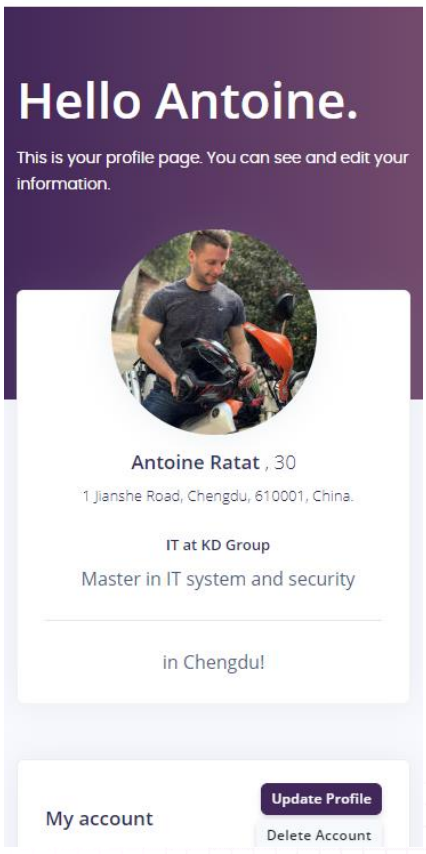


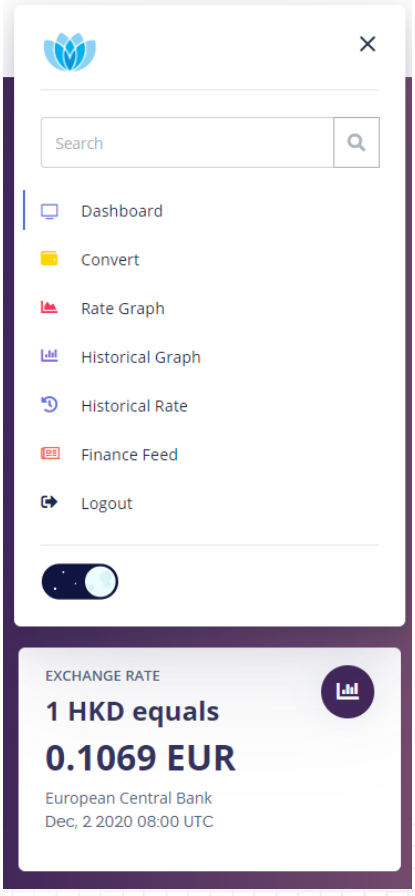


Dashboard - Purple Theme - Light Mode – Desktop and Mobile Version

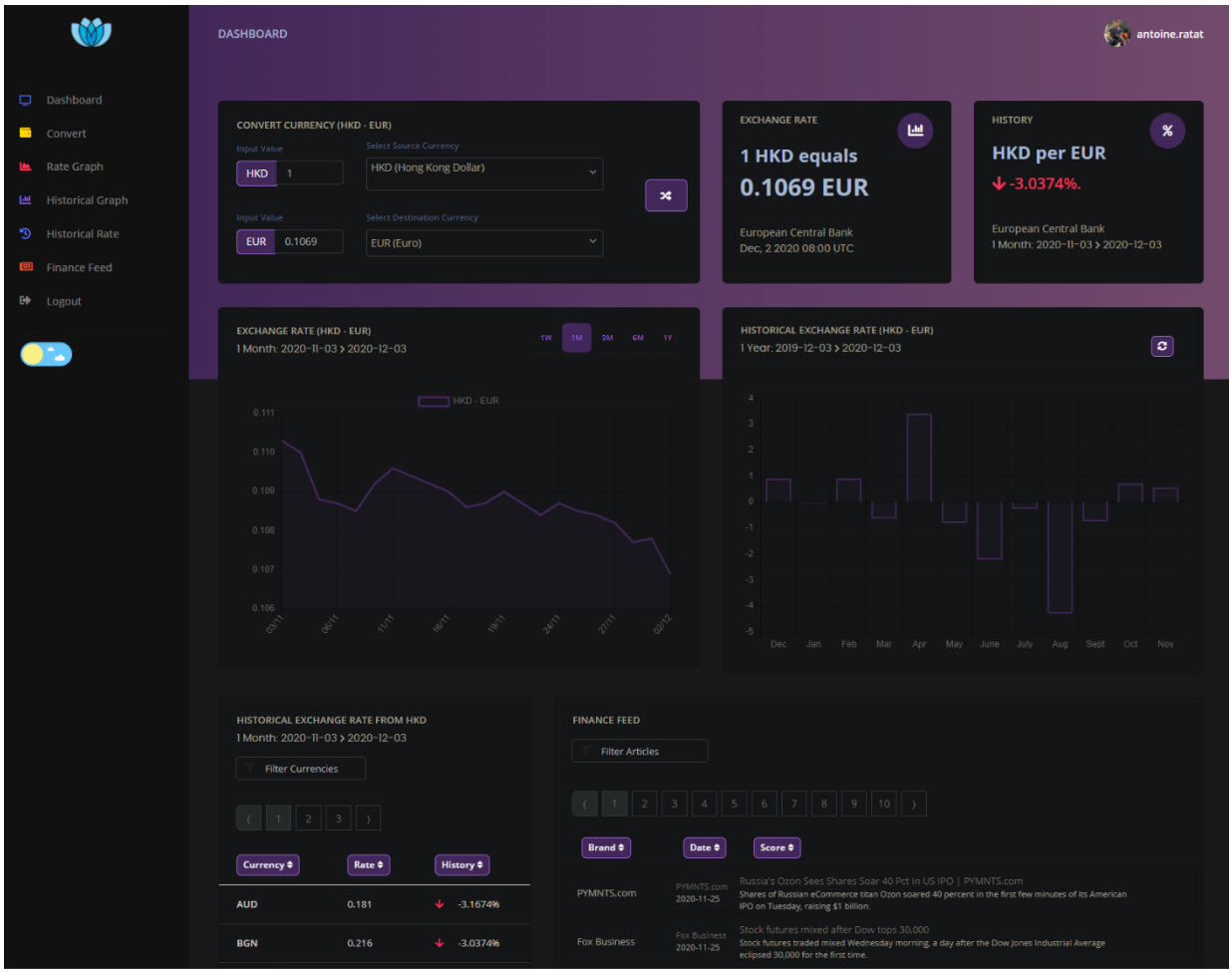


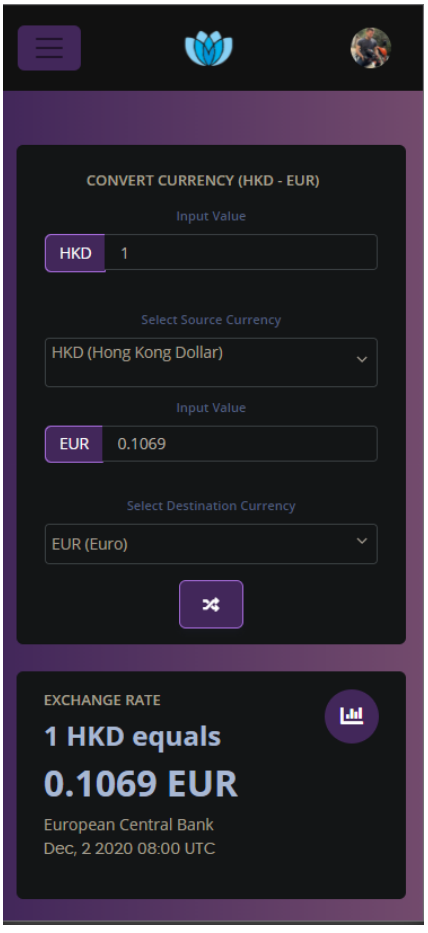


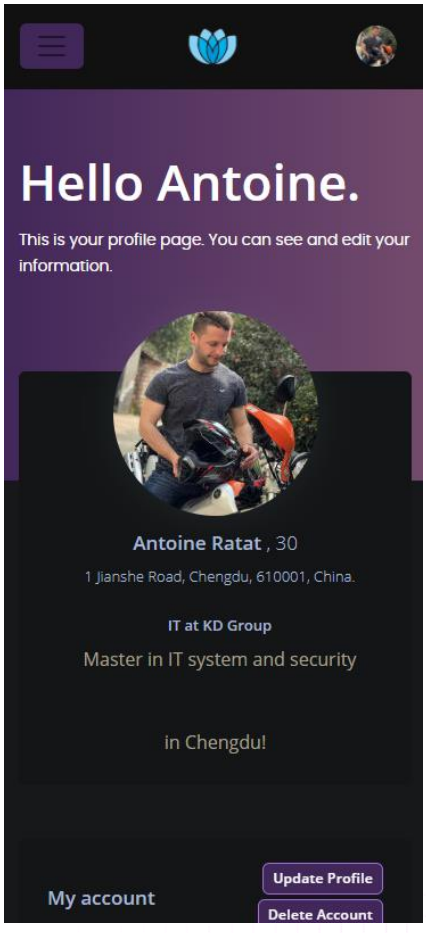


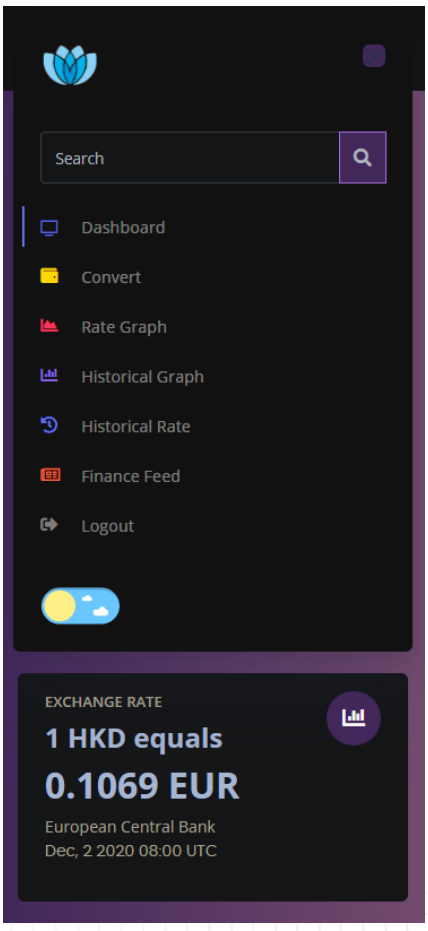


Dashboard - Purple Theme - Dark Mode – Desktop and Mobile Version





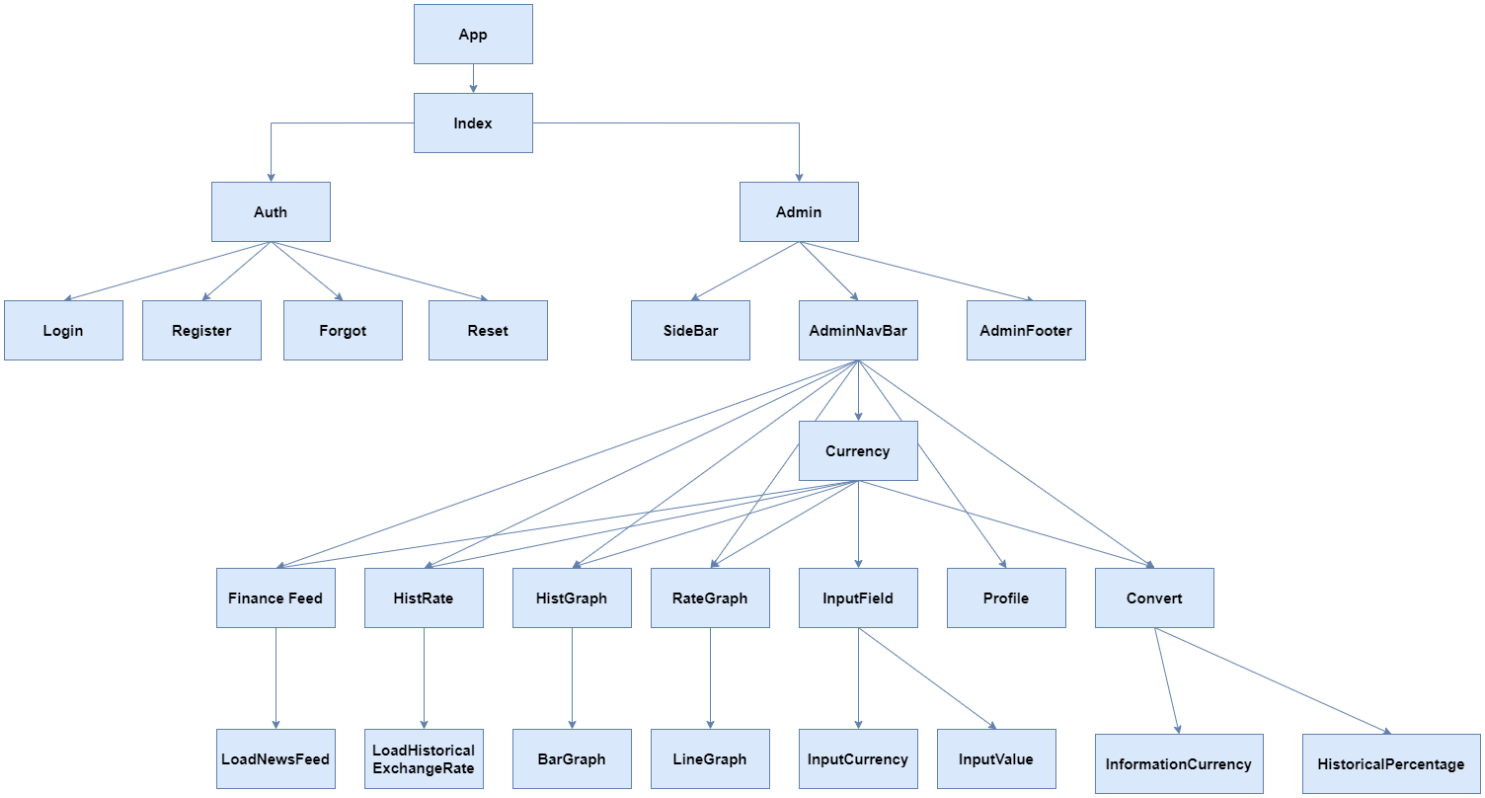




4.2 SOFTWARE INTERFACES

Software is designed in small fragmented atomic components. Each component has specific functionality and assembled together creates our application.

This is easier to maintain, replace, and re-use. The component organization of the Financial app is available in the scheme below:



4.3 COMMUNICATION INTERFACES

Communication is assured to external interfaces. The system is connected to several APIs using REST (representational state transfer), The payload is defined in the request itself and is formatted in JSON. Most of the operations are directed to a custom-made API. This API is a CRUD API created with Python and Flask technology in order to handle the Front-End requests and communicate safely with a PostgreSQL database.

4.3.1 Admin Operations

GET
Return JSON with users.
Authorization : Bearer Token auth as admin

PUT
Update a user from JSON.
Authorization : Bearer Token auth as admin

DELETE

Delete a user from ID
Authorization : Bearer Token auth as admin

4.3.2 User Operations

/api/login
POST username, password
Generate a JWT (Authentication Token) and store user's information as payload
Return as JSON 201

/api/users
POST email, username, password, first_name, last_name
Insert User in the database

/api/register
POST email, username, first_name, last_name
Insert User in the database

/api/reset
POST email
Send reset password message to email containing a link to reset form (expires after 30 minutes)

/api/reset_password
POST password token
Check if token is still valid
Store a bcrypt of the password in database

/api/user
Authorization : Bearer Token auth a user
Get user identity from the token's payload

GET
Return JSON information on User, 200

PUT
Update information in database
Return JSON User Updated, 200

DELETE
Delete username in database
Return JSON Removed User, 200

/api/refresh
Authorization : JWT Refresh Token
Refresh user's access token
Return JWT as JSON, 200

/api/user/setting
Authorization : Bearer Token auth a user

GET
Return JSON information on User Setting, 200

PUT
Update user Setting in database
Return JSON User Setting Updated, 200

5 ADDITIONAL NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE

The system must be interactive and the delays involved must be less. So, in every action-response of the system, there are no immediate delays. In the case of opening new windows, popping error messages, and saving the settings or sessions there is delay depending on the API response time and availability.

In the case of calling external APIs, the delay is based on editing on the distance of the secondary system and the configuration between them so there is a high probability that there will be or not a successful connection in less than 5 seconds for sake of good communication.

5.2 AVAILABILITY

If the internet service gets disrupted while fetching information from the external APIs, the information can be fetched again later.

The system is highly dependant on external APIs, if the communication between the client and the API is disrupted, the information could be requested again.

List of APIs contacted :

User Login, Register, Password Reset, User Settings : https://flask-finance-api.herokuapp.com/
Rate and Finance Information : https://api.exchangeratesapi.io/history
Picture Features : https://api.cloudinary.com/
Finance NewsFeed: https://api.cityfalcon.com

5.3 SECURITY

- Check input data with Form Validation using Formik and Yup
- Back end verification before any database operation
- Authentification mechanism with JWT
- HTTPS transfer protocol to secure communication over the network
- Hashing sensitive information with Bcrypt before storing in the database
- Storing application's sensitive information in environment variable on the server

5.4 USABILITY

As the system is easy to handle and navigates in the most expected way with no delays. In that case, the system program reacts accordingly and transverses quickly between its states.