IBU | International Burch University

IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

Currency Converter

Prepared by:
Adnan Mutilovic

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

20/06/2023

Table of Contents

1. Introduction 3
    1.1. About the Project 3
    1.2. Project Functionalities and Screenshots 3
2. Project Structure 8
    2.1. Technologies 8
    2.2. Database Entities 9
    2.3. Design Patterns 10
    2.4. Tests 11
3. Conclusion 11

1. Introduction

This is a brief introduction guide of my "Currency Converter" application. It's a full stack JavaScript project which main goal is to provide users with real time currency exchange rates. Here in this file, I will cover the functionalities, design patterns, software architecture and details about how to use this app.

So, as already mentioned, this application is designed to show currency conversion. It retrieves the latest conversion rates from an API, it provides an interface for users to input their currency conversion need and then outputs the result of the conversion.

For the development of this project, I used: React.js for the frontend, Node.js and Express for the backend and MongoDB for the database. Through this project documentation I will dive deeper into the design, implementation and provide more explanation on how this project works and how it was built.

1.1. About the Project

I built my currency conversion project as a full stack web application using MERN stack (MongoDB, Express, React, Node). The main goal of the application is to be able to help users convert currencies and check exchange rates.

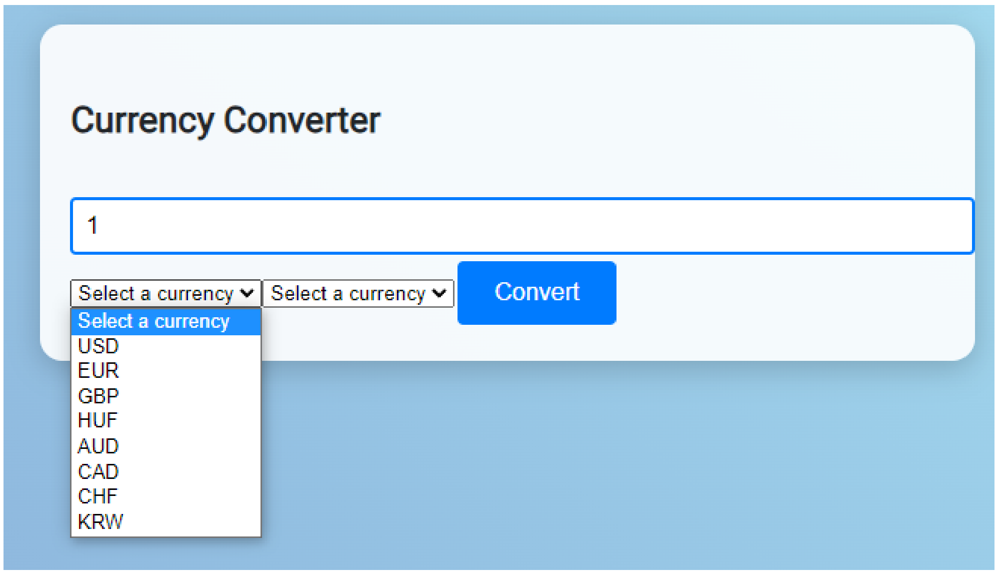
On my frontend, React.js builds a user-friendly interface and react router is used for efficient page navigation. To get the latest currency rates, I use Axios and for the development of backend, I used Express.js and Node.js which interact with MongoDB to retrieve data.

The main goal is to make converting currency simple and easy. Plus, it's built in a way that we can add new features in the future without a lot of trouble.

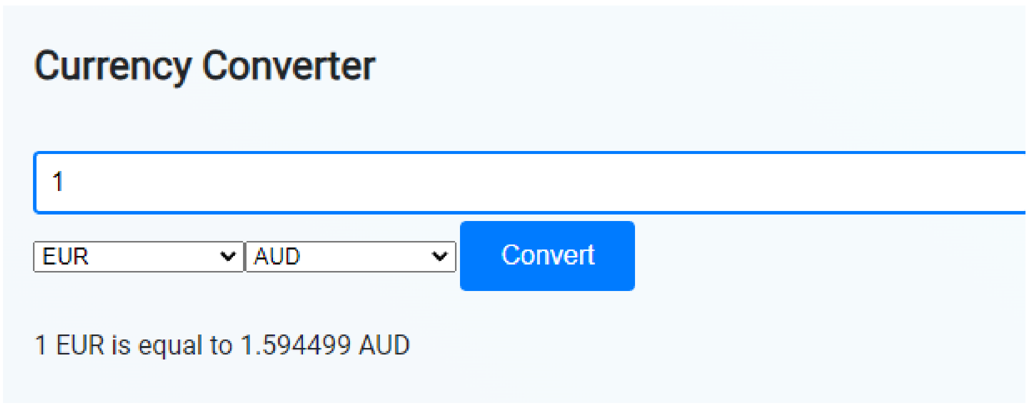
1.2. Project Functionalities and Screenshots

Main features of my currency converter web application include the following:

1. **Currency Conversion:** Main part and most important function of the project. There are 2 drop down menus (currency to convert from and currency to convert to). Above the drop down menus user can input any value of the currency to see the exchange rate. Once the users click the "Convert" button the conversion is performed.



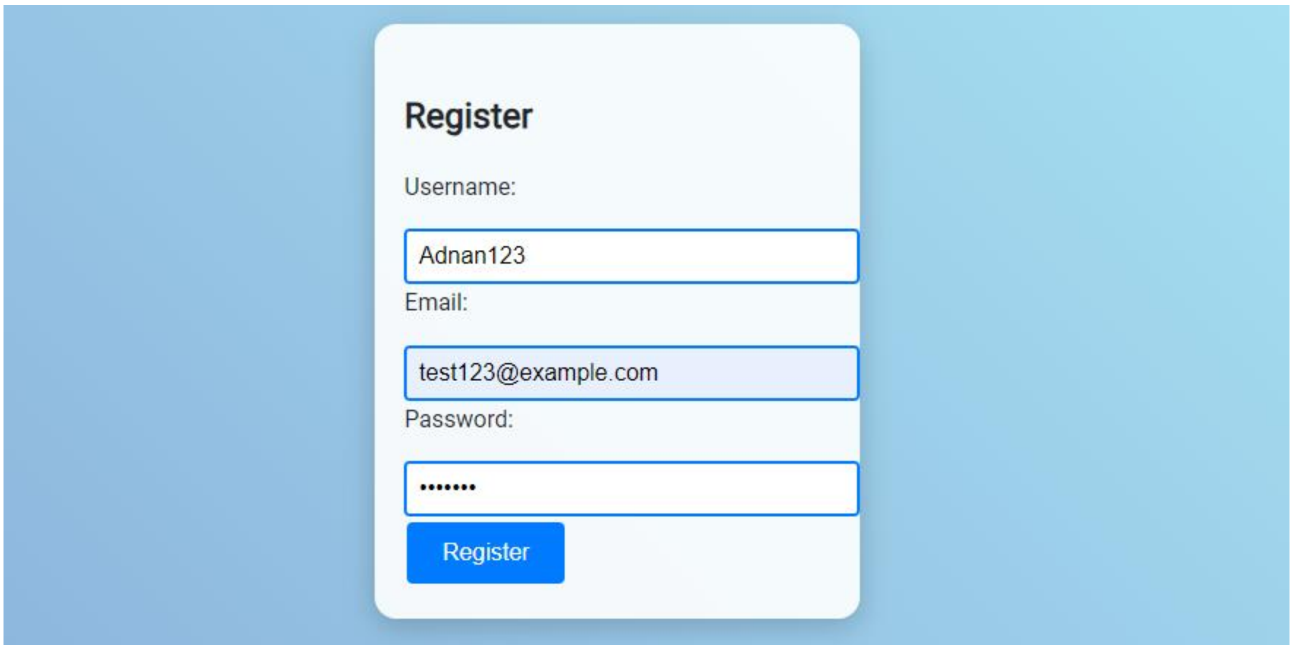
In the next screenshot I show the functionality of the convert button and the currency conversion function by converting 1 EUR to AUD (Australian Dollar)



1 EUR is equal to 1.594499 AUD

As we can see the currency conversion has been performed without any problems or issues, which was basically the main component and core of this application. Now we will move on to the rest of the functionalities which I built in this project.

2. **Register/Login:** Second most important component of the application (which basically allows access to the rest of the functionalities) is the register and login system. For registration the user is necessary to enter his name, email and password. Once this information is entered correctly, the user gets immediately redirected to the login page, where he has to enter his email address and password. If the information is entered correctly the user gets logged in, this register information is of course saved in MongoDB.



Register

Username:

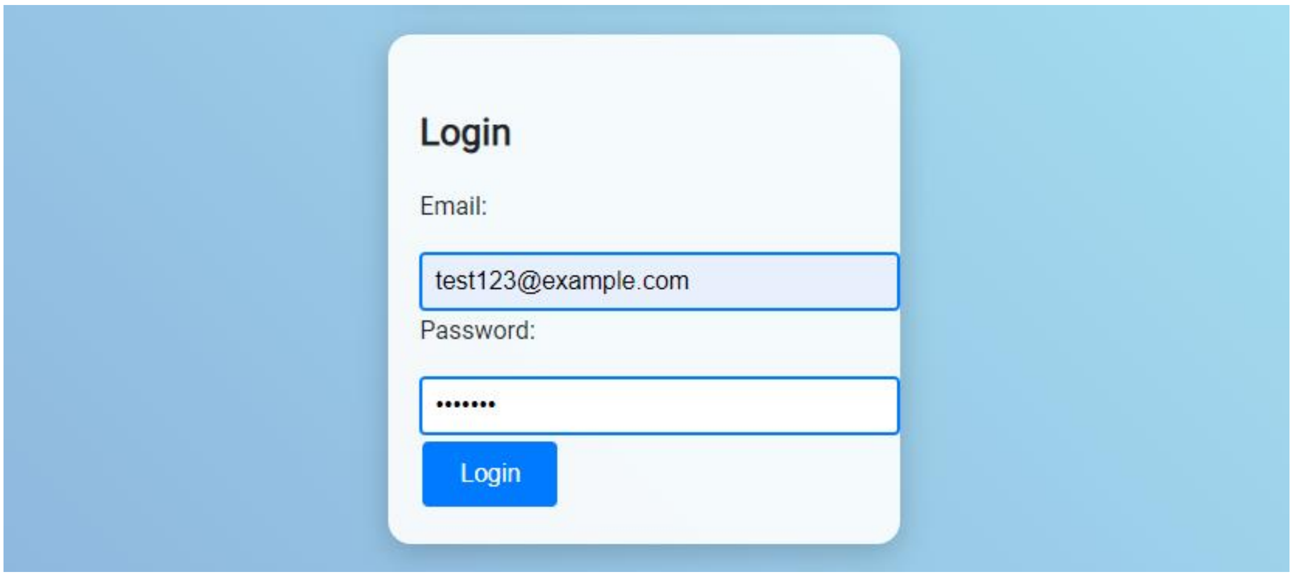
Adnan123

Email:

test123@example.com

Password:

•••••••



Login

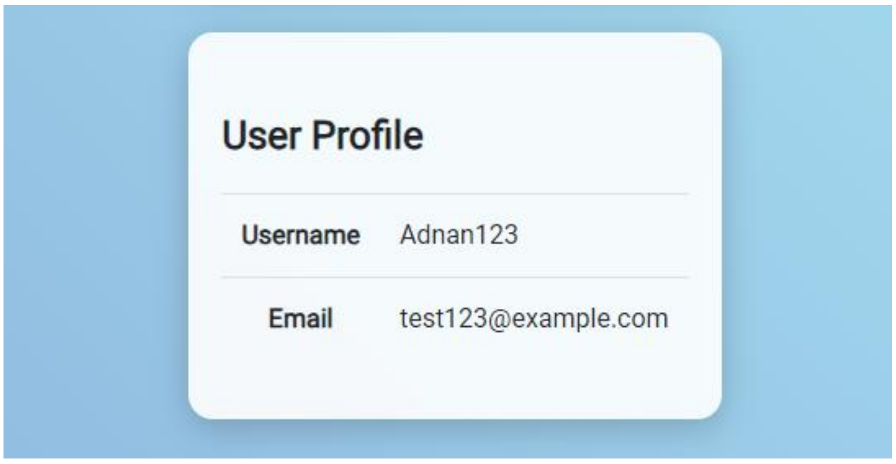
Email:

test123@example.com

Password:

•••••••

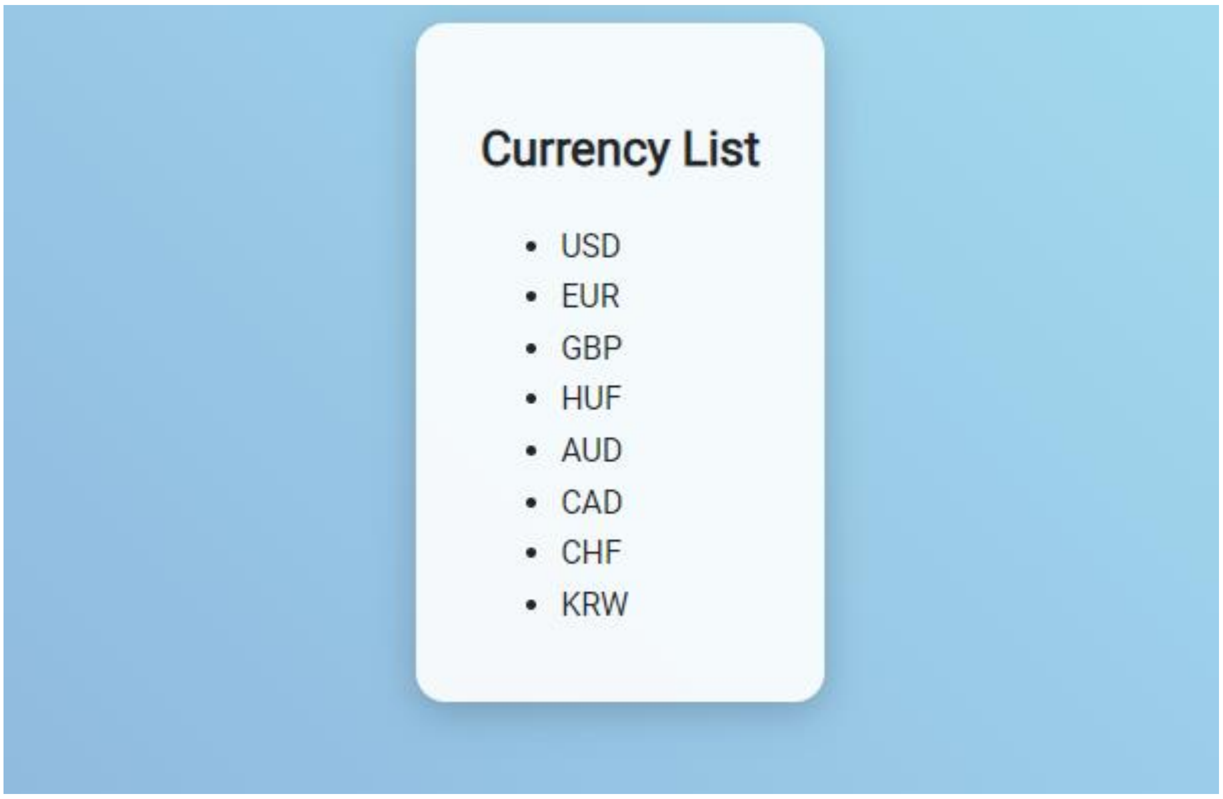
3. **User Profile:** Once the user completes the registration and login process, he gets full access to the rest of the functionalities and options of the application. One of these options is User Profile, where the user can see to which account he's logged currently and what his username is.



User Profile

Username	Adnan123
Email	test123@example.com

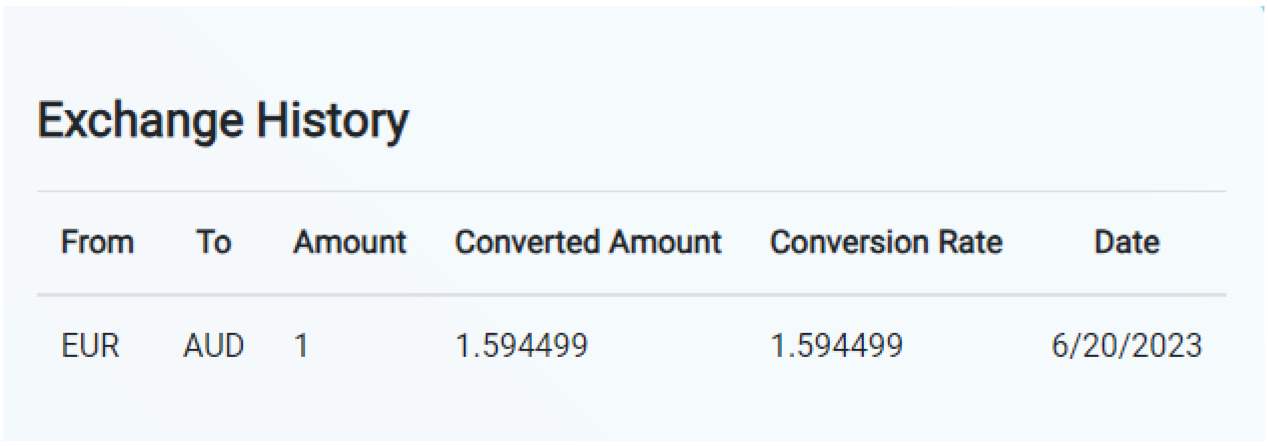
4. **Currency List:** By accessing this option, the user can get a full overview of the current currencies which are added to the database and supported by this website.



Currency List

- USD
- EUR
- GBP
- HUF
- AUD
- CAD
- CHF
- KRW

5. **Exchange History:** A great option to track the history of conversions for the user, as well as keep track of changes (rise or fall) of currencies. How this works is basically when the main function (Currency Conversion) is used, all of them get stored in the Exchange History. So, since I exchange EUR to AUD previously, I can see that now here.



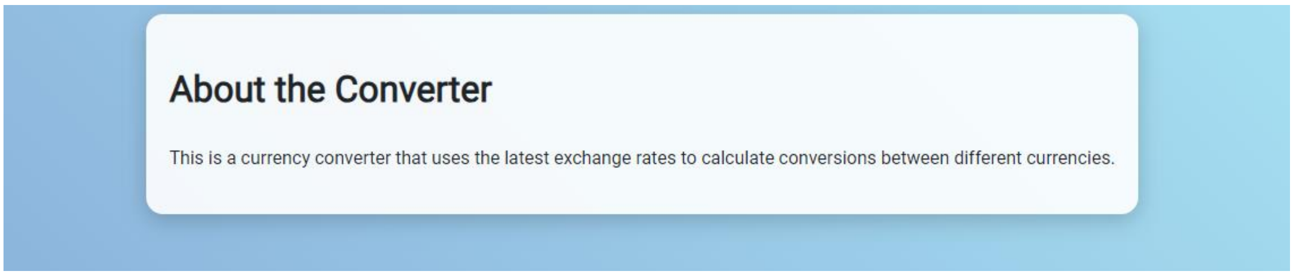
Exchange History

From	To	Amount	Converted Amount	Conversion Rate	Date
EUR	AUD	1	1.594499	1.594499	6/20/2023

6. **Favorites:** The main idea behind this option was to allow the user to mark currencies as his favorites so he can quick access and check their exchange rates, instead of going through the full list of currencies. It would also be a neat feature as some currencies have long names and it would be easier to remember them by checking the “Favorites” button, instead of

scrolling through hundreds of currencies. However, this option is not fully working currently.

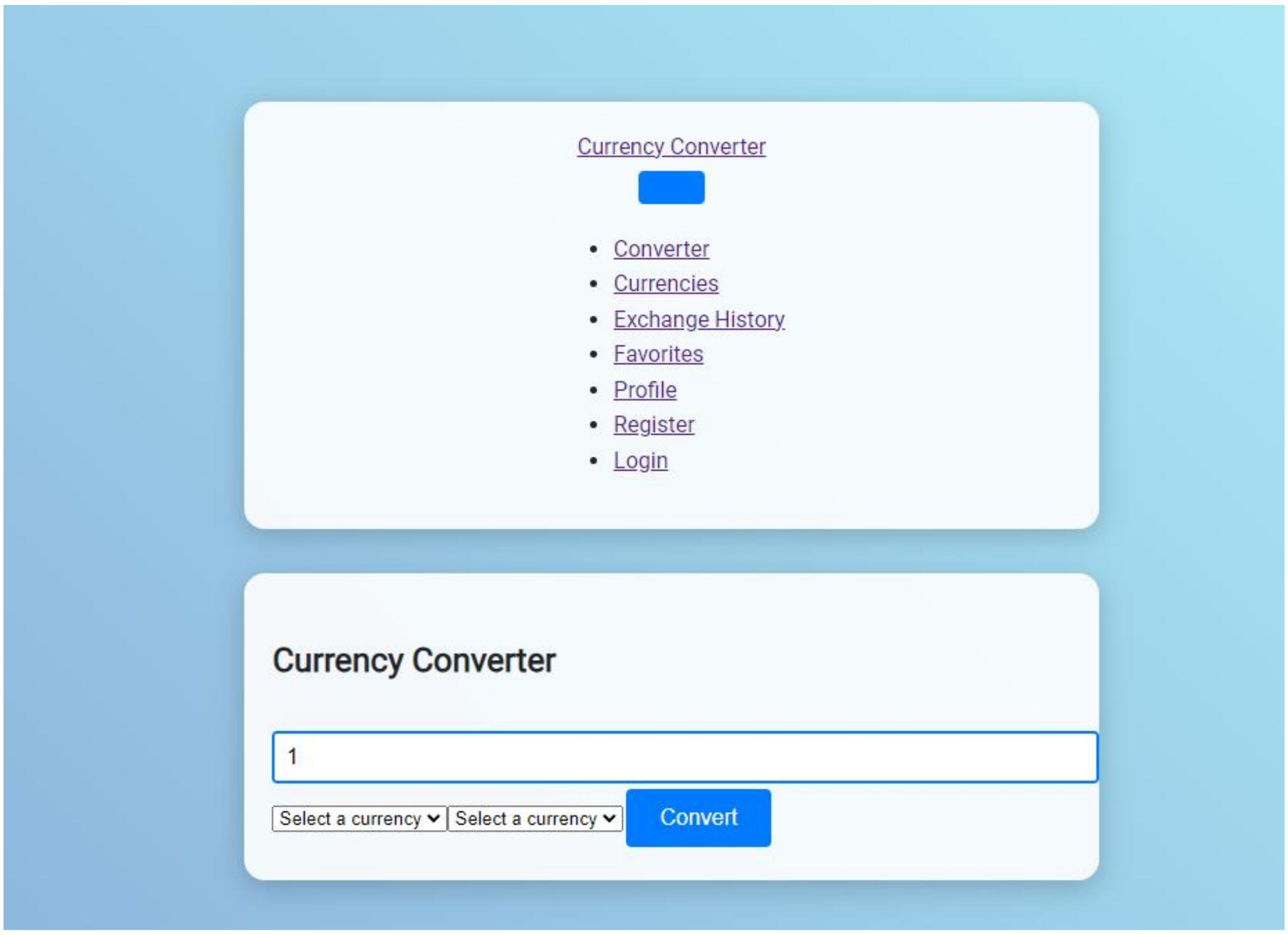
7. **About Converter:** This is just a simple and classic “about us” page, that basically explains the website and concept of it.



About the Converter

This is a currency converter that uses the latest exchange rates to calculate conversions between different currencies.

8. **Homepage:** That’s basically it when it comes to the functionalities and options of the website as of now. Here’s a graphical representation of how the UI and all these options look like:



2. Project Structure

2.1. Technologies

The breakdown of my project technologies is as following:

Backend

As already mentioned previously, the backend part of application is built with Node.js. For me personally it is great for the development of efficient and scalable applications. Besides Node.js, I used Express.js to create the server side logic of this project. Express.js is a web application framework for Node.js and it's very handy to use.

Frontend

For frontend I chose React.js which is a popular JavaScript library used for building UIs. In my opinion it's very easy to work with since it allows me to create reusable UI components which makes the code more maintainable and efficient.

Database

My choice for the DB was MongoDB. It's the most popular NoSQL database and it offers a lot of flexibility. It's document oriented and it's usually the go to when working with Node.js and JavaScript.

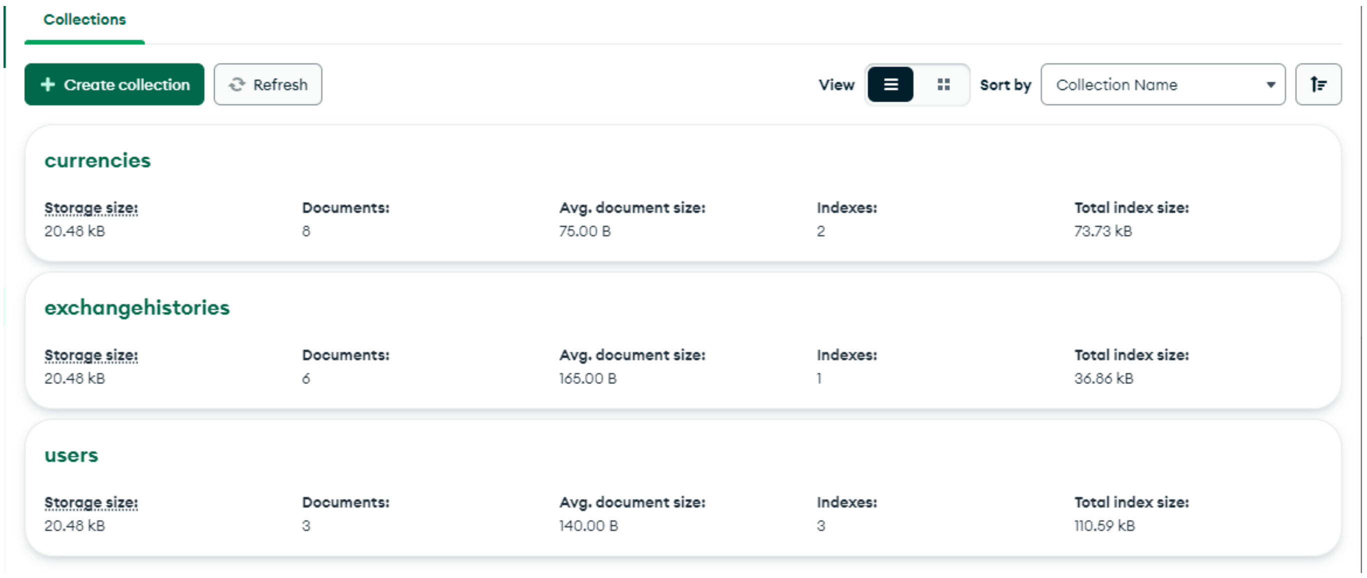
When it comes to other technologies I used Axios for handling HTTP requests. It's very useful as it provides a consistent API, and it enables the possibility to interact with the rest of my REST API efficiently. Besides that, I also used Mongoose - this is a MongoDB object tool with which I manage to keep relationship between data and translate between objects in code and objects represented in MongoDB.

Coding standards:

I followed ES6 coding standard in both my backend and frontend. Usually when I program in JavaScript I use ES6 since it includes a lot of features. It gives me the **possibility** to write more modern code and to use feature like 'let' or 'const' for variable declarations and more.
I also used ESLint with Airbnb JavaScript Style Guide to make sure that the code is clean.
To keep the code **consistent,** I used "Prettier" (it's very handy since it supports a lot of **languages**, including all versions of JavaScript).

2.2. Database Entities

The list of table and entities I used in my database are currencies, exchangehistories and users.



currencies

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	8	75.00 B	2	73.73 kB

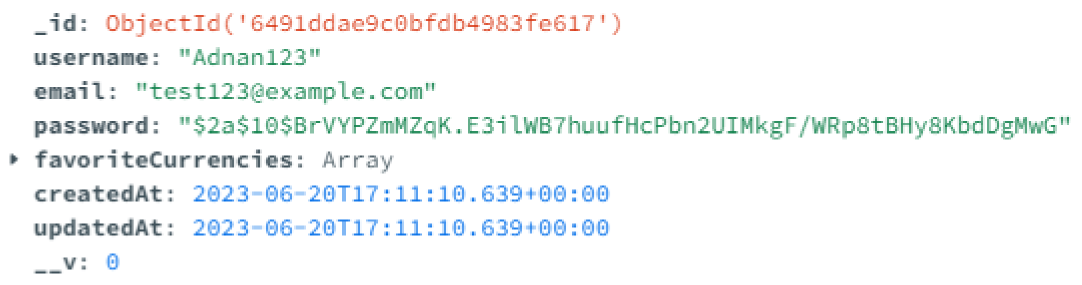
exchangehistories

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	6	165.00 B	1	36.86 kB

users

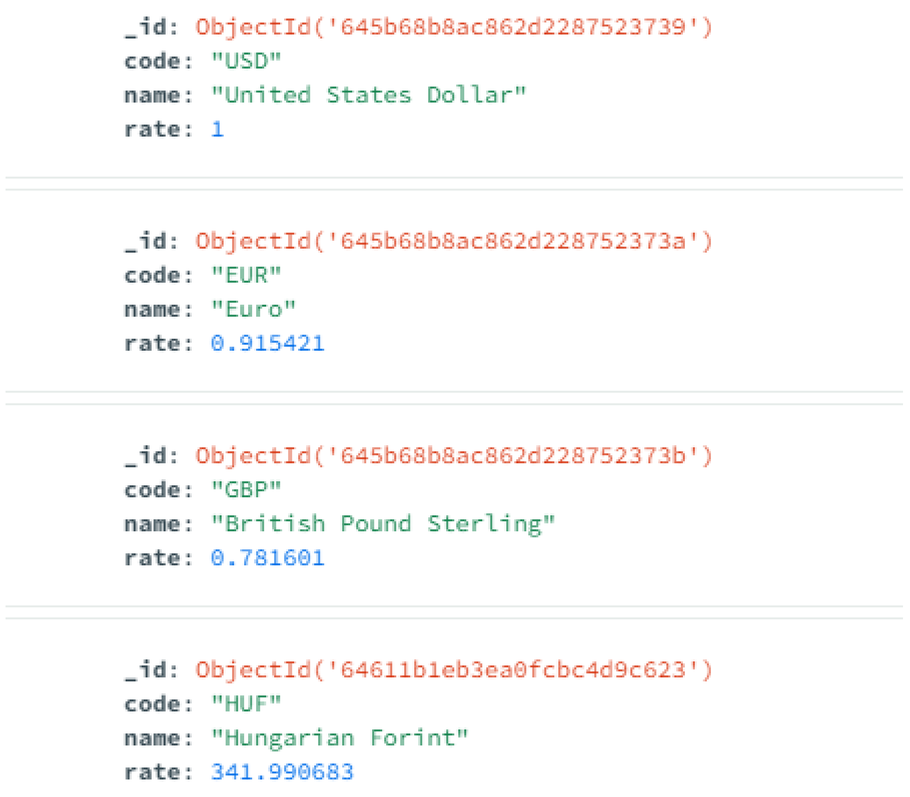
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	3	140.00 B	3	110.59 kB

I think it's pretty self-explanatory. Users is used for registration and login purposes.



```
  _id: ObjectId('6491ddae9c0bfdb4983fe617')
  username: "Adnan123"
  email: "test123@example.com"
  password: "$2a$10$BrVYPZmMZqK.E3ilWB7huufHcPbn2UIMkgF/WRp8tBHy8KbdDgMwG"
▸ favoriteCurrencies: Array
  createdAt: 2023-06-20T17:11:10.639+00:00
  updatedAt: 2023-06-20T17:11:10.639+00:00
  __v: 0
```

Currencies is the list of currencies stored in the DB.



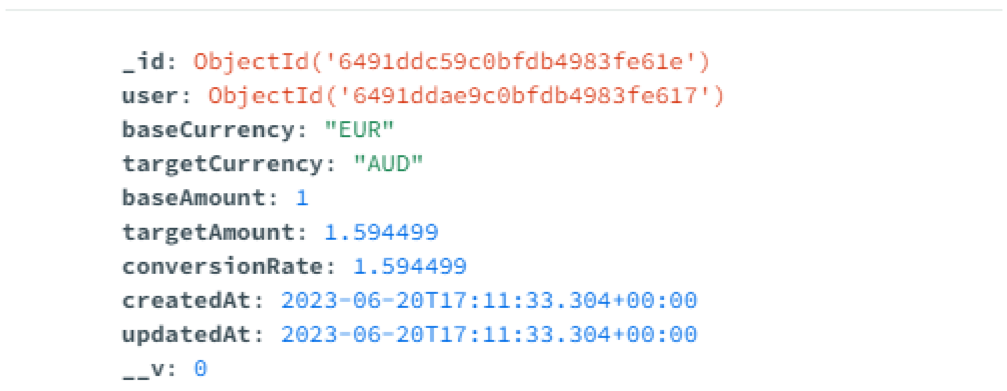
```
_id: ObjectId('645b68b8ac862d2287523739')
code: "USD"
name: "United States Dollar"
rate: 1
```

```
_id: ObjectId('645b68b8ac862d228752373a')
code: "EUR"
name: "Euro"
rate: 0.915421
```

```
_id: ObjectId('645b68b8ac862d228752373b')
code: "GBP"
name: "British Pound Sterling"
rate: 0.781601
```

```
_id: ObjectId('64611b1eb3ea0fcbc4d9c623')
code: "HUF"
name: "Hungarian Forint"
rate: 341.990683
```

And exchangehistories is what I mentioned earlier, it stores the history of currency conversion the user preformed, so for example:



```
_id: ObjectId('6491ddc59c0bfdb4983fe61e')
user: ObjectId('6491ddae9c0bfdb4983fe617')
baseCurrency: "EUR"
targetCurrency: "AUD"
baseAmount: 1
targetAmount: 1.594499
conversionRate: 1.594499
createdAt: 2023-06-20T17:11:33.304+00:00
updatedAt: 2023-06-20T17:11:33.304+00:00
__v: 0
```

2.3. Design Patterns

In the process of building my project, I have used several design patterns to help me maintain and organize the code more efficiently.

Module Pattern: I used it extensively through the whole project, both on frontend and packed. This pattern allows me to encapsulate functions and variables into single modules, which makes the code easier to manage. By using Node.js, every file is treated as a separate module, so by using this pattern I can organize the server side code into separate modules for different options. On frontend, I organized the React components into modules.

Singleton Pattern: I used singleton for my database connections. By using this pattern, I make sure that a class has only one instance and that it provides a global point of access to it. I used this when working with mongoose in my code.

Middleware patterns: In my Express.js server-side code I used middleware functions to handle requests and responses. With the usage of this pattern, I can break down the Serer into smaller reusable parts. I use this pattern is server setup, specifically in the App.js file.

Component Pattern: While storing the components in fronted/src/components directory I used this pattern by breaking down my project's UI into smaller composable objects to be able to write cleaner and more maintainable code.

By using each of these patterns I made my decisions based on the requirements of my projects (looking at the perspective how this pattern can benefit me). By using them I was able to organize my code so that it's more understandable, clean and scalable.

2.4. Tests

During the development of my Currency Converter project, I used multiple testing strategies to ensure that the code runs properly.

Unit and Integration Tests:

So basically, I covered the backend of my application with various unit and integration tests. The goal of these tests is to verify the correctness of functions and how they interact with other objects in the code. I used Jest testing framework and Supertest for making HTTP requests to my Express.js server. To be more specific:

1) **currency.test.js** (located in tests directory in backend) - focuses on testing the functionality of the currency conversion API. What it does it - it checks the /api/currencies endpoint and then verifies that the response contains an array of those currencies.

2) **exchangeHistory.test.js** - this one verifies the functionality of the exchange history feature. It includes tests for both authenticated and unauthenticated access to the /api/exchangehistory

3) **user.test.js and userController.test.js** - both these tests focus on user management. In more simple words, they test the user registration and logion functions to make sure that api/users/register and api/users/login are working correctly. They check that the users are correctly saved in the database and that they are able to successfully log.

Testing setup and mock data:

To test data, I have a "fixtures" directory within my "tests" directory. In here I made the db.js file that helps manage test data for users. I define a test user, create a JWT token and provide a function to set up the database state before each test runs.

3. Conclusion

The creation of this project was very interesting, it gave me the opportunity to expand my knowledge and apply it more practically. There were a lot of challenges and obstacles. It was an exhausting process to implement real time currency exchange rates and keeping track of managing all user data securely.

While I'm generally happy with the project, there are definitely some areas I would want to improve in the future. For example, I want to expand the test coverage on frontend as it would increase my confidence in the stability and reliability of the app. Maybe adding some features and analytics would be a lot better for user experience.

Also, I had a lot of trouble with the deployment. I tried more than 10 websites to deploy (Vercel, Heroku, Cloudflare) - but there always seemed an issue. For example, Vercel would give me an error message of "not available, try again later." While Heroku, build log would not ever finish - it would seem like it's stuck in an infinite loop. I even tried recreating the repository and linking that one from GitHub to deployments, but it didn't help. I contacted Heroku support about build log file being stuck and just saying the project is in build phase (for more than 40 minutes), but I didn't receive a response. Therefore, this was an issue I sadly couldn't resolve.

Overall, it was a great experience building a full stack application and working with these languages. The course in general was very insightful and I feel like I learned a lot. I'm looking forward to future projects, to apply my knowledge and skills I obtained by doing this one.