

2024

APDS7311 POE

Task 2

Prepared By:

The Non-Risk Rangers

Mpho Ndlela	--	ST10203065
Anele Siwela	--	ST10084586
Lucian Young	--	ST10039287
Alec Cowan	--	ST10049180



Contents

Introduction to Customer-Portal by Risk Rangers Installation of React.....	3
Setting Up the Backend.....	3
MongoDB Setup and SSL Certificate.....	3
JWT Authentication and Session Management.....	3
Password Security: Hashing and Salting	3
Preventing SQL Injection with MongoDB	4
Postman Testing	4
UI Component Library.....	4
Project Images	4
Setting up our development environment	5
Installation of react.....	5
Setting up the backend	5
Installation of mongoose	6
Generating ssl certificate	7
Testing on Postman.....	7
Ui component library	8
The React Website.....	9
The Landing page.....	9
The registration page	9
The Connected Mongo Db.....	10
User specific Page after logging in	10
The populated MongoDB NOSQL database.....	11
Transaction Page	11
Currency converter api was implemented	12
User makes payment.....	12

Introduction to Customer-Portal by Risk Rangers

Installation of React

We began by installing React to serve as the front-end framework for the application. React was chosen for its modular architecture, efficient rendering capabilities, and active community support, making it an ideal choice for building modern, dynamic web applications.

Setting Up the Backend

The backend was set up using Node.js along with MongoDB as the database. We utilized Mongoose as the ODM (Object Data Modeling) library to simplify the interaction between the application and MongoDB. With Mongoose, we defined schemas and handled data associations, which significantly streamlined the process of converting between code objects and their MongoDB representations.

Mongoose also provides built-in data validation and query building features, which enhanced our development efficiency. We referred to tutorials and resources, including freeCodeCamp's Introduction to Mongoose for MongoDB for setup and configuration.

MongoDB Setup and SSL Certificate

After configuring the MongoDB database and obtaining the connection string, we moved on to generating an SSL certificate for secure data transfer. This step ensures all communication between the client and server is encrypted, adding a layer of protection against man-in-the-middle attacks. Securing the connection was an important step toward achieving the project's security goals.

JWT Authentication and Session Management

For authentication, we implemented JWT (JSON Web Tokens). JWTs are employed to securely transmit information between the client and server. Once a user logs in successfully, a JWT token is issued, containing encrypted user data such as the user ID and roles. This token is then attached to requests that require authentication, ensuring only authorized users can access specific routes or services.

JWT also supports session expiration, meaning tokens have a defined lifespan. When the token expires, the user must log in again, providing an added level of security. This approach helps mitigate the risk of session hijacking.

Password Security: Hashing and Salting

To securely manage passwords, we incorporated hashing and salting techniques. Hashing is the process of converting a password into a fixed-length string using a cryptographic hash function. Rather than storing passwords in plaintext, we hashed them using a secure hashing algorithm, preventing exposure even if the database were compromised.

We also employed salting to further strengthen security. A unique random salt was added to each password before hashing, ensuring that even if two users have the same password, their stored hashes will differ. This makes it impossible for an attacker to exploit precomputed attacks like rainbow tables.

Preventing SQL Injection with MongoDB

Using MongoDB as our NoSQL database inherently helped us avoid SQL injection attacks, which are more common with relational databases. NoSQL injection vectors are also addressed by using Mongoose, which sanitizes inputs to protect against unauthorized data manipulation.

Postman Testing

We conducted rigorous testing of the backend API using Postman. Postman allowed us to test various endpoints, verify authentication flows, and validate data handling to ensure the backend services were working correctly.

UI Component Library

For the front-end, we installed and utilized a UI component library to accelerate the development of visually appealing and consistent UI components. This provided a cohesive look and feel to the application while allowing us to leverage pre-built components for features such as buttons, form elements, and layout grids.

Project Images

The following images document the key aspects of our development process, showcasing the UI design, data flows, backend setup, and application testing results:

Setting up our development environment

Installation of react

```
PS D:\Downloads\Customer_portal> npx create-react-app customer-portal
>> cd customer-portal
>> code .
>>

Creating a new React app in D:\Downloads\Customer_portal\customer-portal.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1481 packages in 6m

262 packages are looking for funding
  run `npm fund` for details

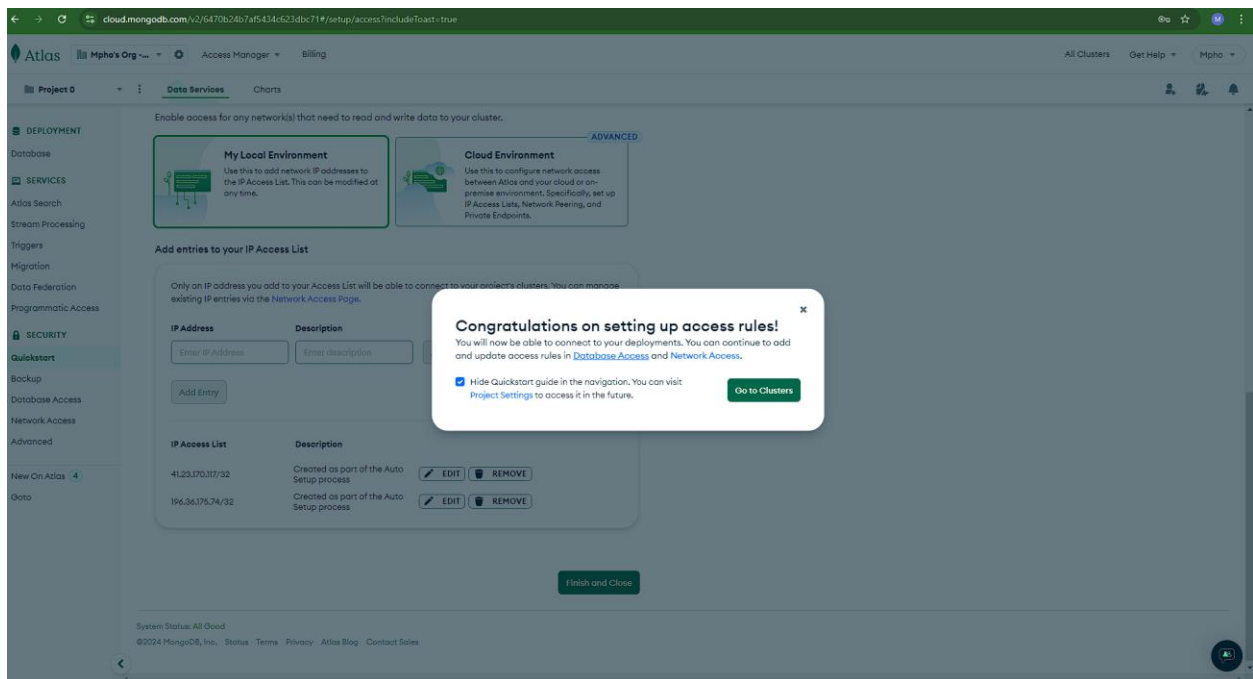
Initialized a git repository.

Installing template dependencies using npm...

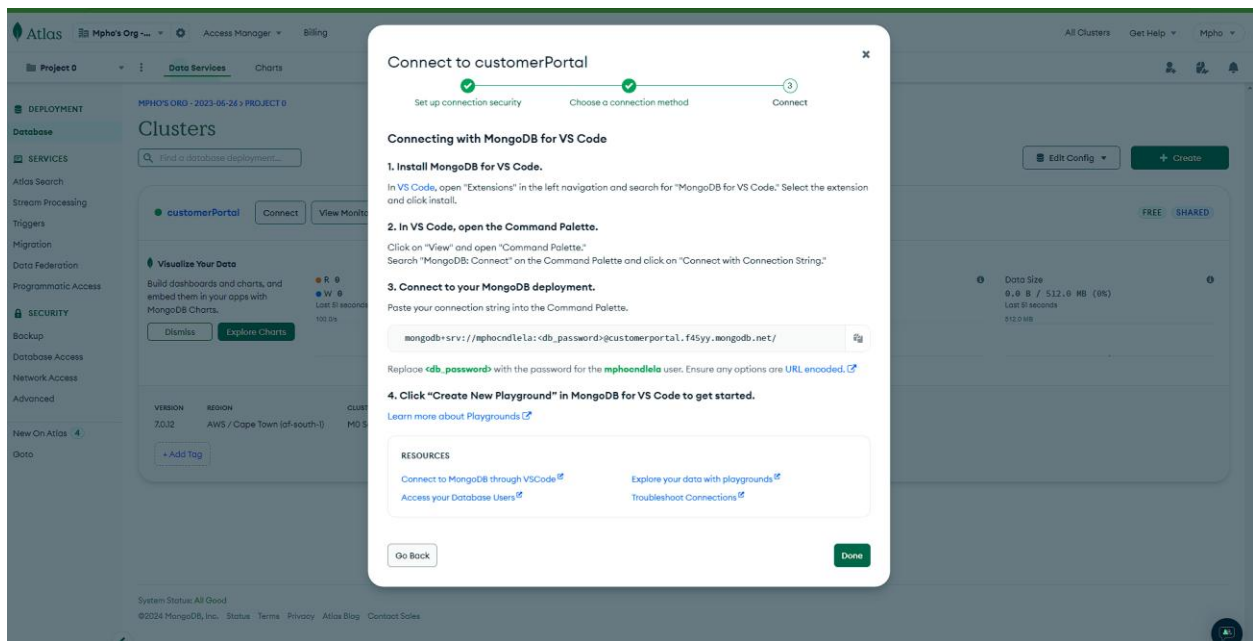
added 63 packages, and changed 1 package in 15s
```

Busy installing react for the application.

Setting up the backend



Setting up the backend and database with MongoDB.



Setting up the MongoDB database and getting the connection string.

Installation of mongoose

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE
P5 D:\Downloads\Customer_portal> cd customer_portal
P5 D:\Downloads\Customer_portal(customer_portal)> npm install mongoose
P5 D:\Downloads\Customer_portal(customer_portal)> npm install mongoose
P5 D:\Downloads\Customer_portal(customer_portal)>

added 16 packages, and audited 1902 packages in 28s

122 packages are looking for funding
  run `npm fund` for details

162 vulnerabilities (1 low, 122 moderate, 36 high, 3 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
P5 D:\Downloads\Customer_portal(customer_portal)>

```

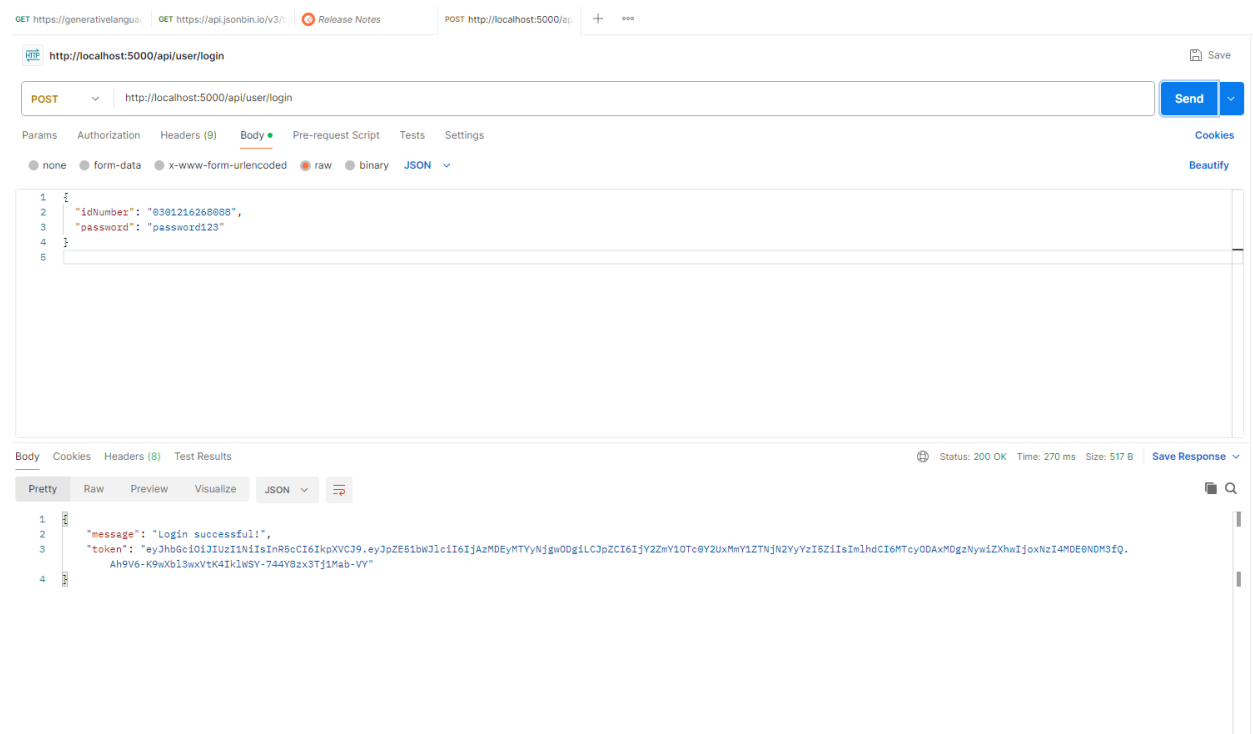
Installing Mongoose - Mongoose is an Object Data Modeling (ODM) library that works with MongoDB and Node.js. It handles data associations, performs schema validation, and is used to convert between code objects and their MongoDB representations. (freeCodeCamp (2018) Introduction to mongoose for mongodb, freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/> (Accessed: 08 October 2024).)

Generating ssl certificate

[illegible]

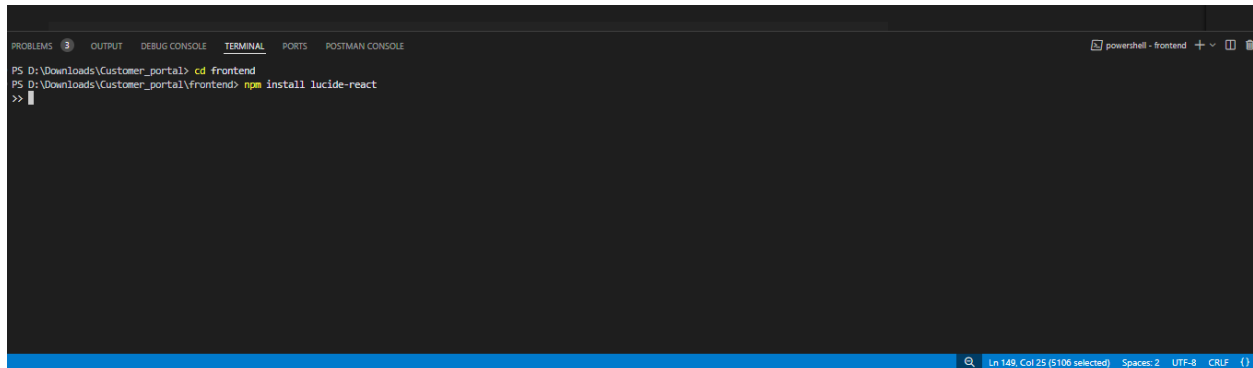
Generating the ssl certificate.

Testing on Postman



Testing on postman to make sure the application backend is working.

Ui component library



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE
PS D:\Downloads\Customer_portal> cd frontend
PS D:\Downloads\Customer_portal\frontend> npm install lucide-react
>>
```

Installing UI component library.

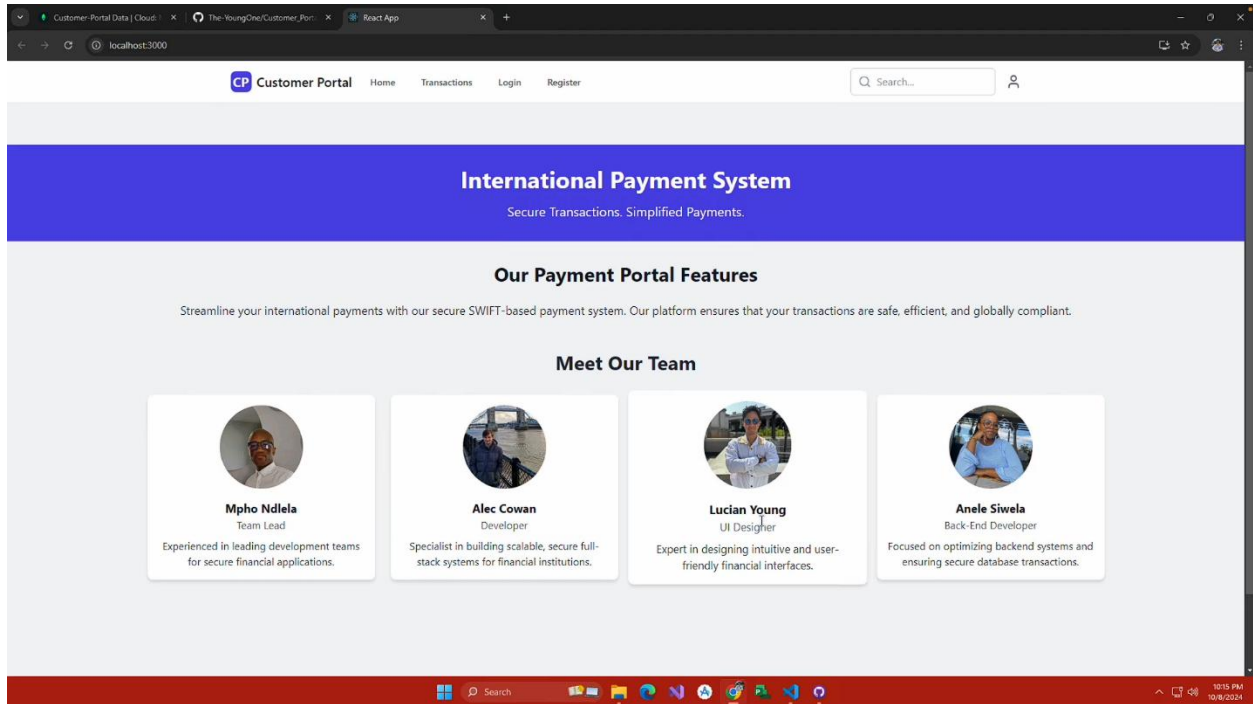
```
Token verification failed or expired: TokenExpiredError: jwt expired
    at C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\jsonwebtoken\verify.js:190:21
    at getSecret (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\jsonwebtoken\verify.js:97:14)
    at module.exports [as verify] (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\jsonwebtoken\verify.js:101:10)
    at authenticateToken (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\backend\middleware\authenticateToken.js:15:7)
    at Layer.handle [as handle_request] (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\express\lib\router\layer.js:95:5)
    at next (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\express\lib\router\route.js:149:13)
    at Route.dispatch (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\express\lib\router\route.js:119:3)
    at Layer.handle [as handle_request] (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\express\lib\router\layer.js:95:5)
    at C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\express\lib\router\index.js:284:15
    at Function.process_params (C:\Users\lab_services_student\Desktop\Customer_portal\Customer_portal\node_modules\express\lib\router\index.js:346:12) {
  expiredAt: 2024-10-08T18:54:41.000Z
}
```

This shows a session timeout with JWT tokens expiring.

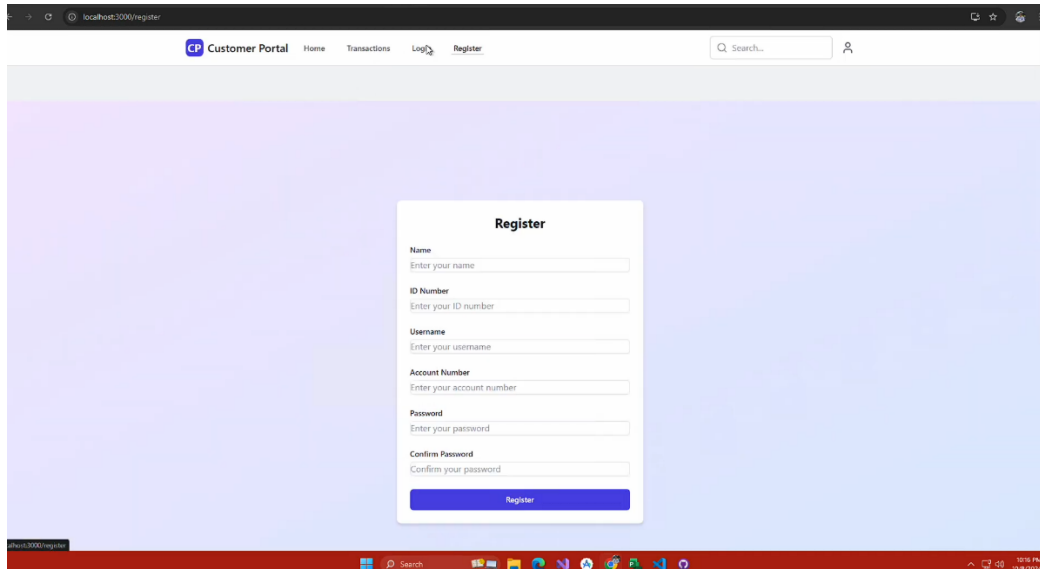
We are also protecting against SQL injection attacks through the use of a NoSQL database with MongoDB.

The React Website

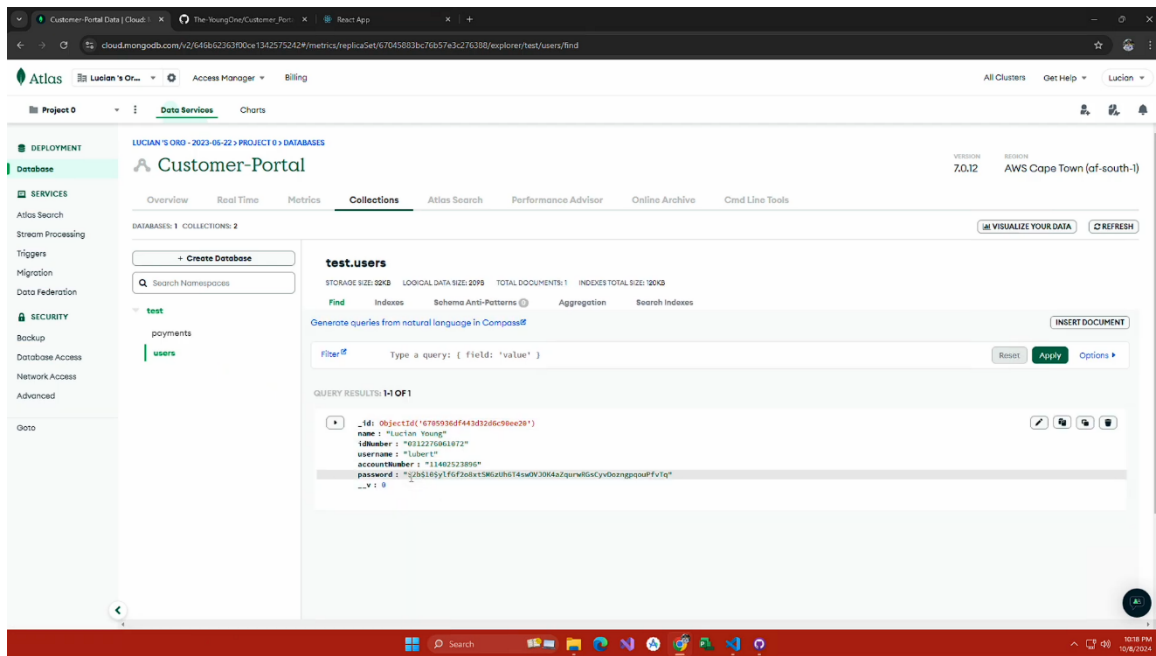
The Landing page



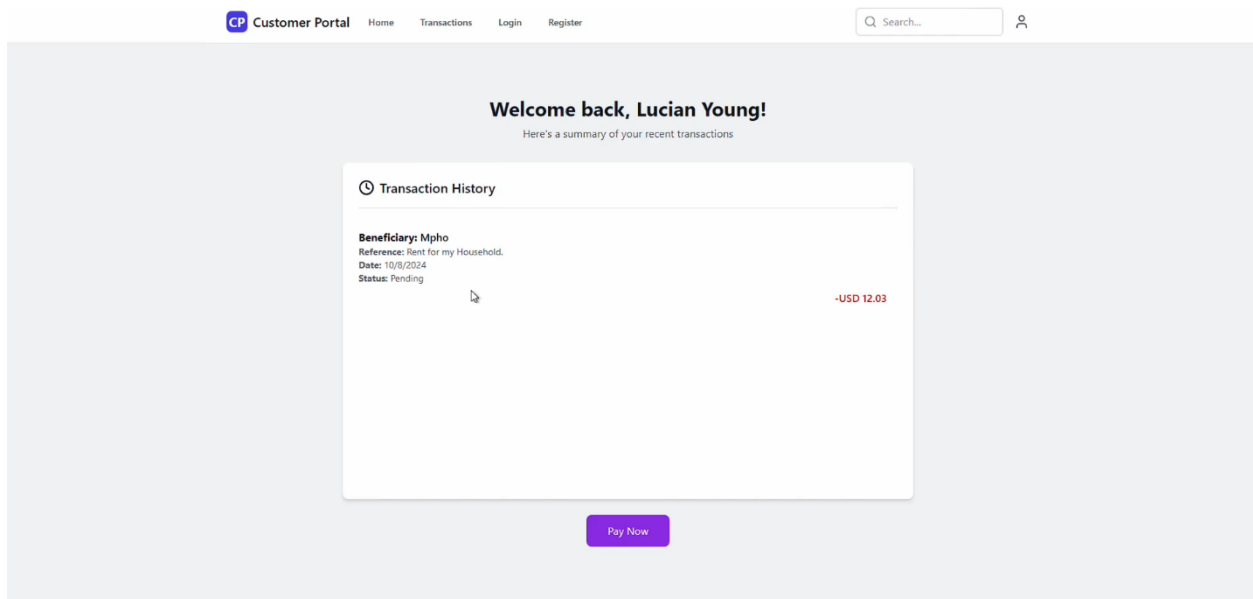
The registration page



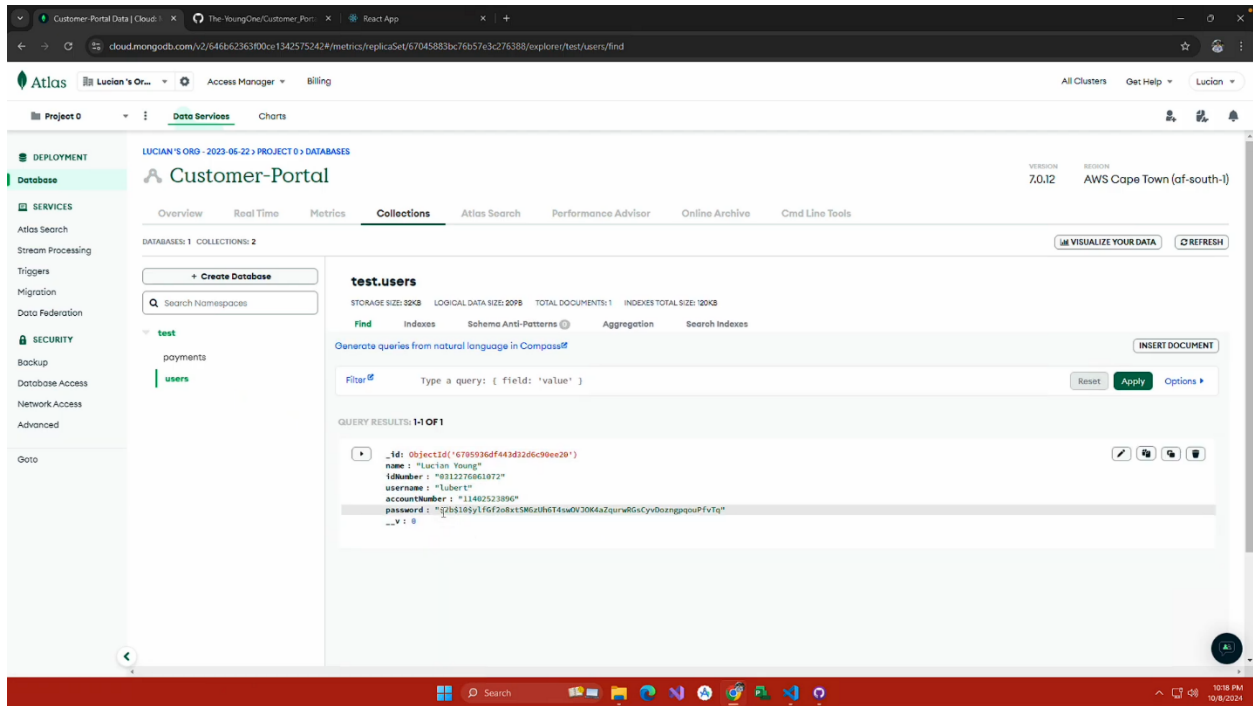
The Connected Mongo Db



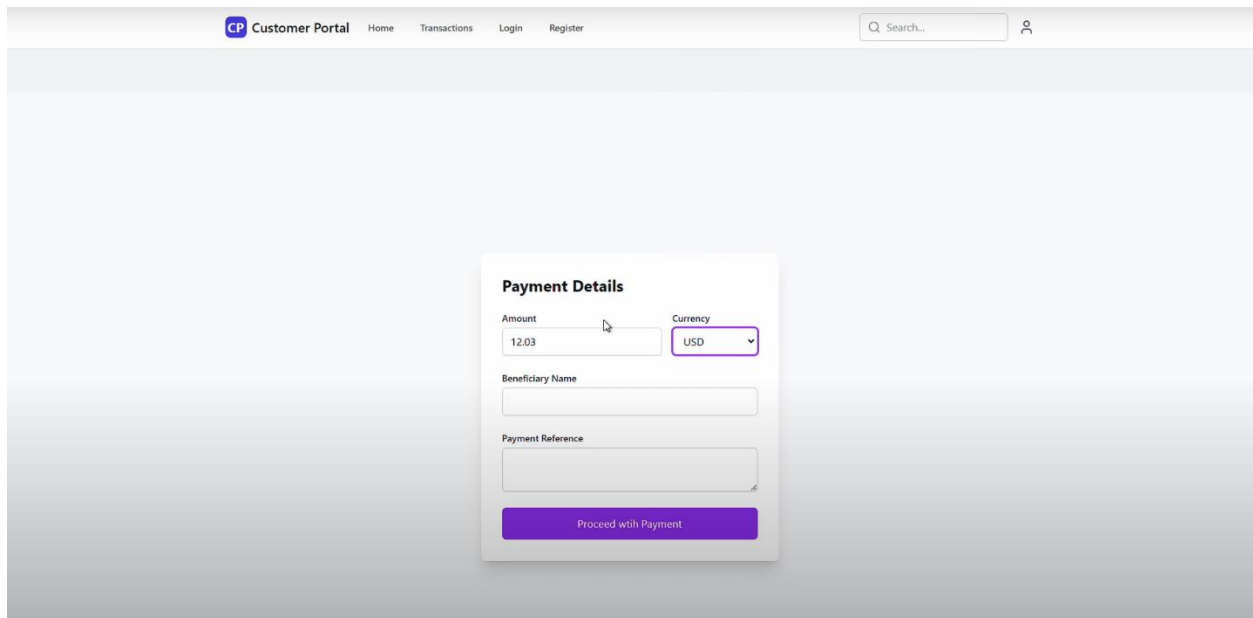
User specific Page after logging in



The populated MongoDB NOSQL database



Transaction Page



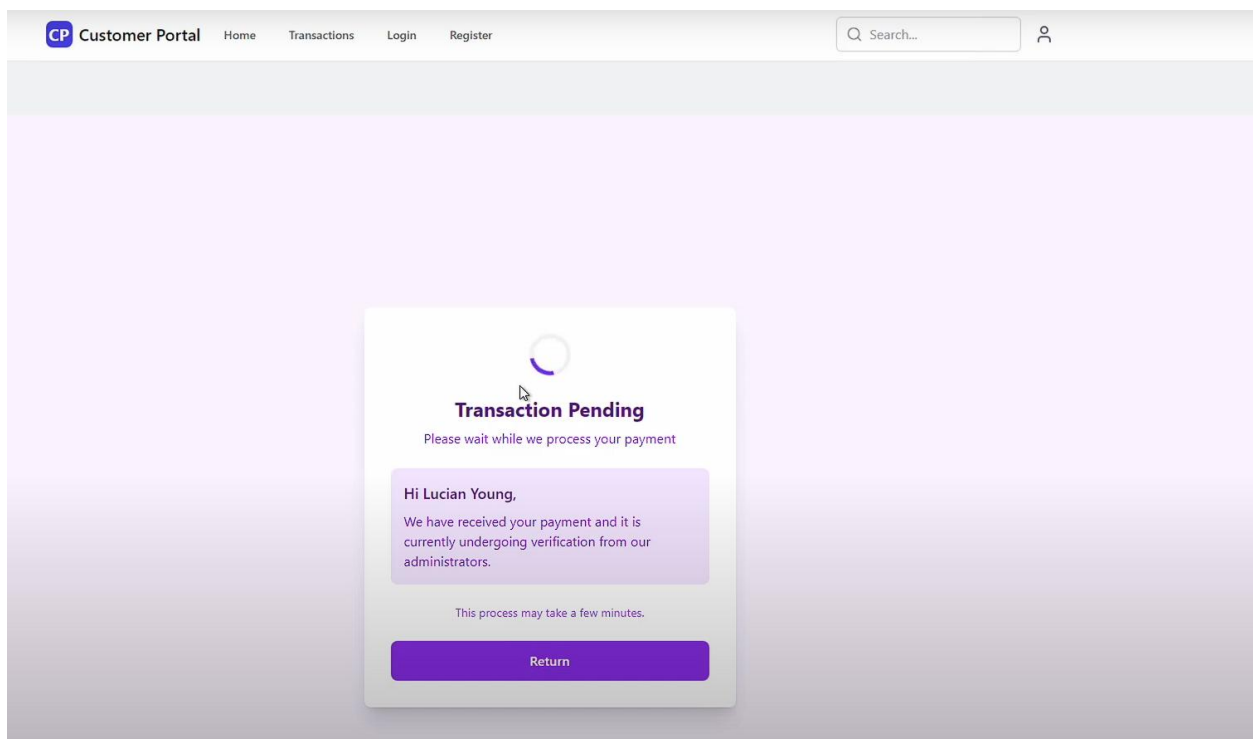
Currency converter api was implemented

Currency

USD

▼

User makes payment



Bibliography

freeCodeCamp. (2018). *Introduction to Mongoose for MongoDB*. Available at: <https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/> (Accessed: 05 October 2024).

Postman. (n.d.). *Postman API Testing*. Available at: <https://www.postman.com/> (Accessed: 05 October 2024).

MongoDB. (n.d.). *MongoDB Documentation*. Available at: <https://docs.mongodb.com/> (Accessed: 05 October 2024).

Node.js. (n.d.). *Node.js Documentation*. Available at: <https://nodejs.org/en/docs/> (Accessed: 05 October 2024).

JSON Web Tokens. (n.d.). *JWT Introduction*. Available at: <https://jwt.io/introduction/> (Accessed: 05 October 2024).

SSL.com. (n.d.). *SSL Certificate Guide*. Available at: <https://www.ssl.com/faqs/> (Accessed: 05 October 2024).

OWASP. (n.d.). *Password Storage Cheat Sheet*. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (Accessed: 05 October 2024).

React. (n.d.). *React Documentation*. Available at: <https://react.dev/> (Accessed: 05 October 2024).