A close-up of a logo

Description automatically generated

**Web Development Fundamentals Project**

**CMPS 350 L51**

**Spring 2024**

**Dr. Mucahid Kutlu**

A light bulb with a brain inside

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Student Name | QID | Email |
| Yosra Elshayeb | 201907974 | ye1907974@qu.edu.qa |
| Nashwa AlShamasi | 201901556 | na190556@qu.edu.qa |

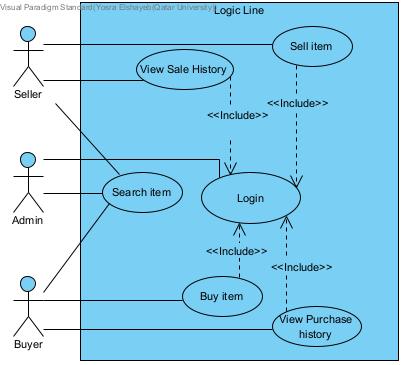
**Project GitHub Link:** [**https://github.com/yosra-ye1907974/cmps350-spring24-project.git**](https://github.com/yosra-ye1907974/cmps350-spring24-project.git)

**Phase 2:**

**Project objectives:**

The main objective of this project is to utilize the knowledge and skills acquired in the CMPS350 course to create an E-commerce platform. This platform will facilitate the buying and selling of items, as well as provide users with the ability to track their transaction history, which will record all activities carried out on the platform.

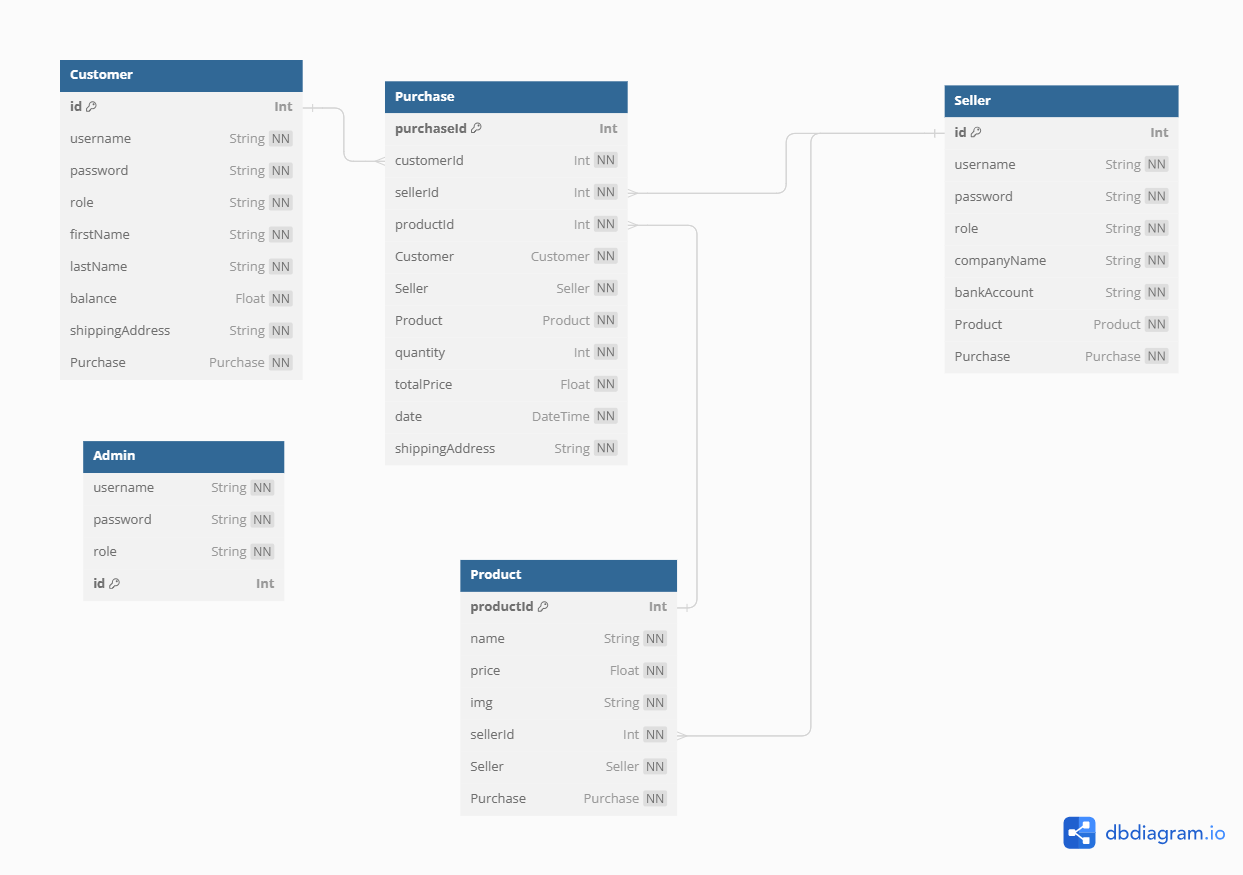
**System Use Case Diagram**



**Database Model (Database Schema)**

The second phase of our project includes transitioning from using JSON files or local storage to managing data in an actual database. This process involved several steps:

1. **Designing and modelling our Data**

****We spent time brainstorming and planning the organization of our application's data. We recognized the entities (products, buyers, sellers, purchases) and how they were related. We developed a theoretical model, which took the shape of an entity-relationship diagram (ERD) or a comparable visual representation tool. This model was intended to show the connections between various entities.

1. **Implementing the Data Model using Prisma**

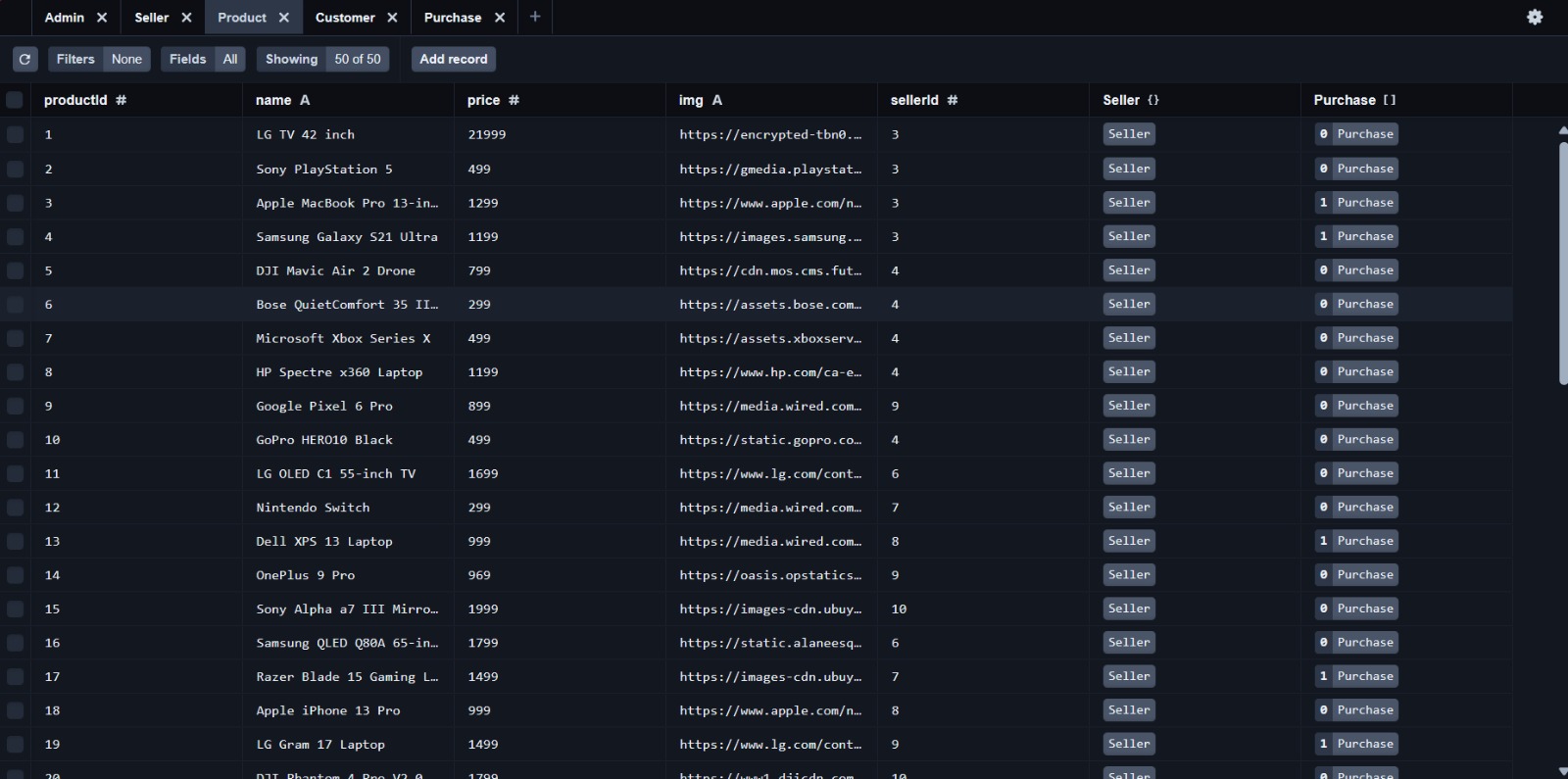
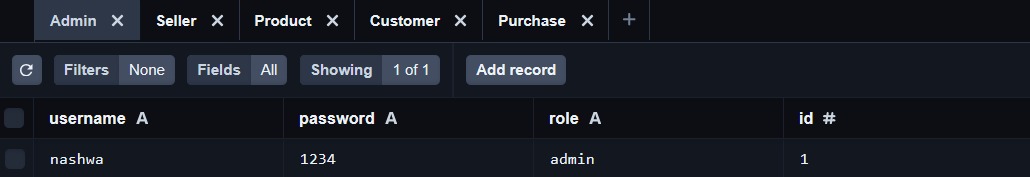
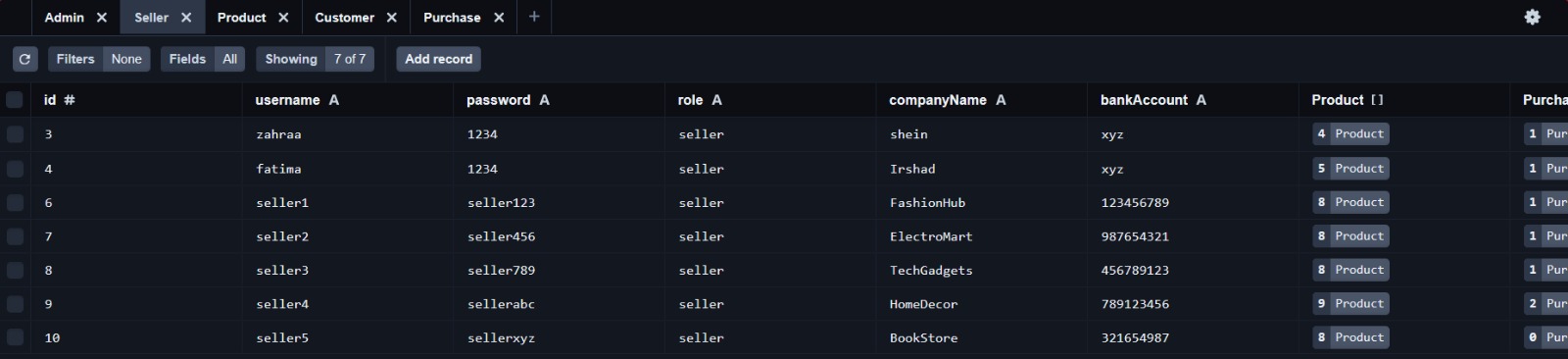
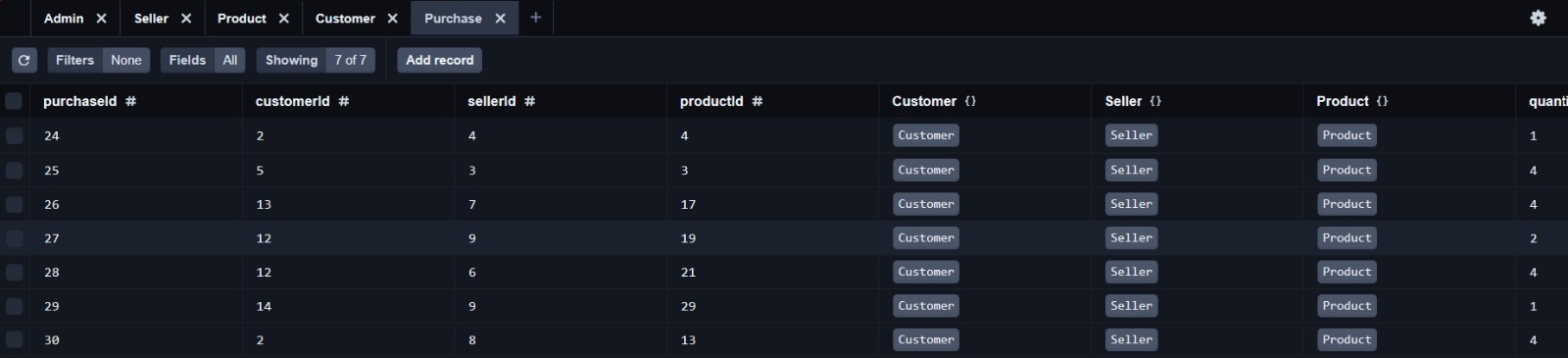
Prisma proved to be an excellent tool for ORM (Object-Relational Mapping) and helped us define our data models using a declarative syntax. We defined our data models in the Prisma schema file (**schema.prisma**). This file describes our application's data structures, including entities, their fields, and relationships**.**

We ensured that our data model accurately reflected the conceptual model we had designed in the previous step. We chose a relational database (SQLite) supported by Prisma and configured our project to use it.

**3. Data Initialization (seeding)**

We created a seed.js file or similar script to populate the database with initial data. We extracted data from our existing JSON files and wrote scripts to insert this data into the corresponding tables in our database.

We ensured that the data we populated included a diverse set of products, buyers, sellers, purchases, etc., to adequately represent the scope of our application. Finally, we tested the seeding process to verify that the data was inserted correctly into the database tables.

1. **Product Table**
2. **Admin Table**
3. **Seller Table**
4. **Customer Table**
5. **Purchase Table**

**Database queries**

Our project aimed to enhance efficiency and reduce unnecessary data exchange by developing a Data Repository that includes all essential features for accessing and updating database information through Prisma Client queries. This method made sure that only the necessary information was extracted from the database. We highlighted the importance of using database server functionalities for data filtering, sorting, and aggregation, instead of depending on application code.

A screen shot of a computer program

Description automatically generated

Additionally, we developed APIs using Next.js to serve as endpoints callable from our application, which was developed in phase 1 of the project. These APIs seamlessly utilized the functions provided by our Data Repository, facilitating efficient data access and manipulation within our application. By adhering to these practices, we ensured optimal performance and scalability while maintaining a clean and organized codebase.

**A screenshot of a computer program

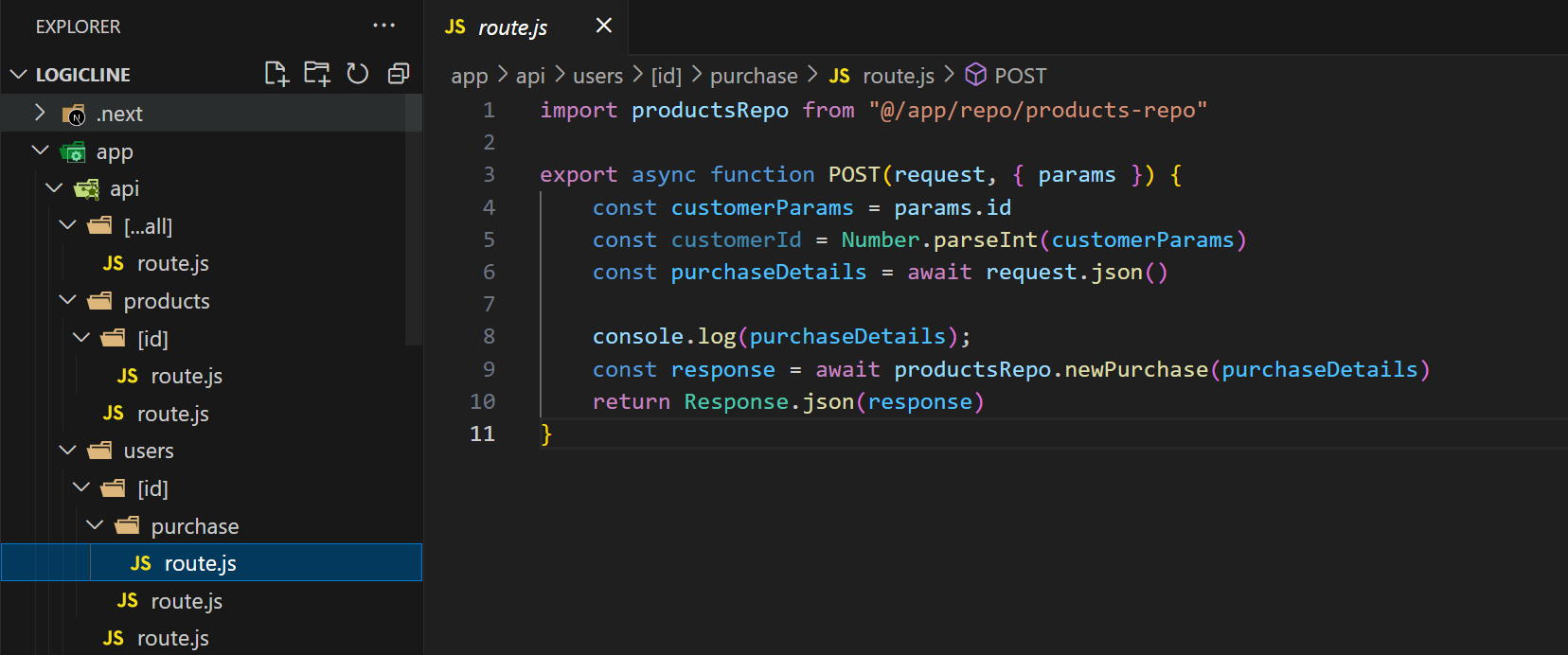
Description automatically generated**

As the figure above shows, we have implemented the API routes under app/api folder. It is notable that we have 6 routes**.**

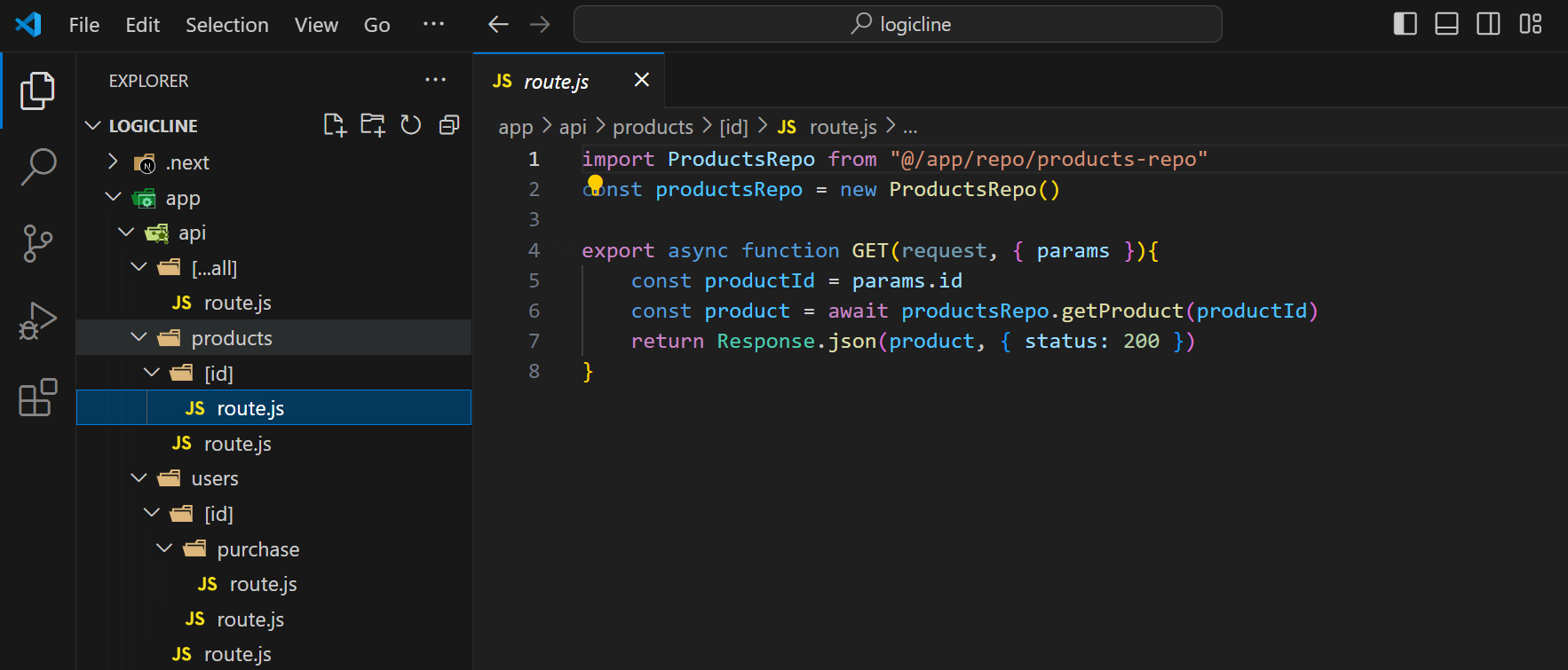
**A screen shot of a computer

Description automatically generated**

Catch-all for detecting unsupported routes.



POST: method

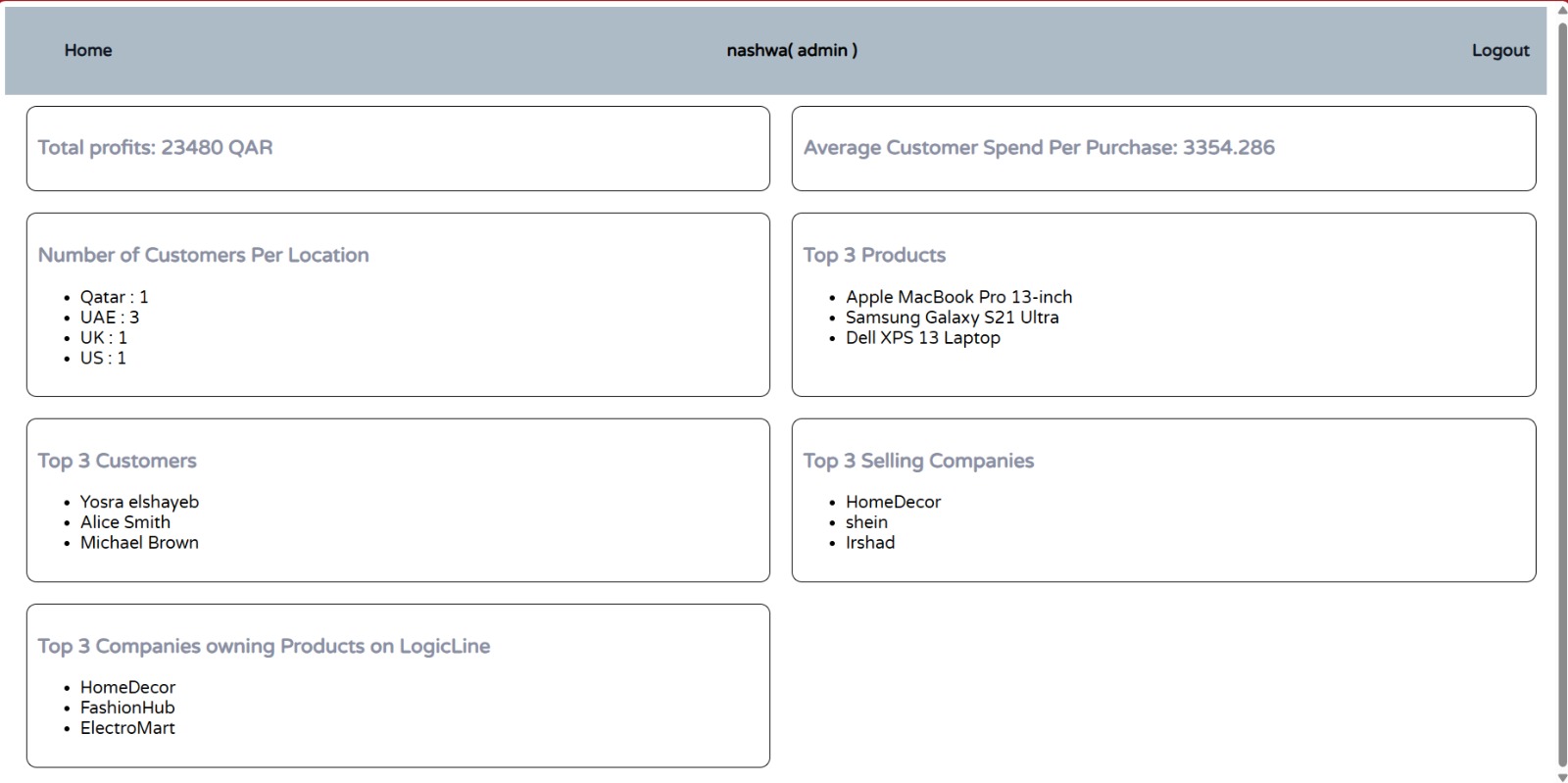
****

GET: method

**UI Design with screenshots and description**.

We developed a new use-case solely utilizing Next.js and React to provide helpful statistics on our online e-commerce platform. These statistics involve:

1. Top 3 customers purchasing from our website.
2. Top 3 selling companies
3. Total profits
4. Average customers spend per purchase.
5. Number of customers per location
6. Top 3 products (most purchased)

**UI in big screen**:

**A screenshot of a phone

Description automatically generatedUI in small screen:**

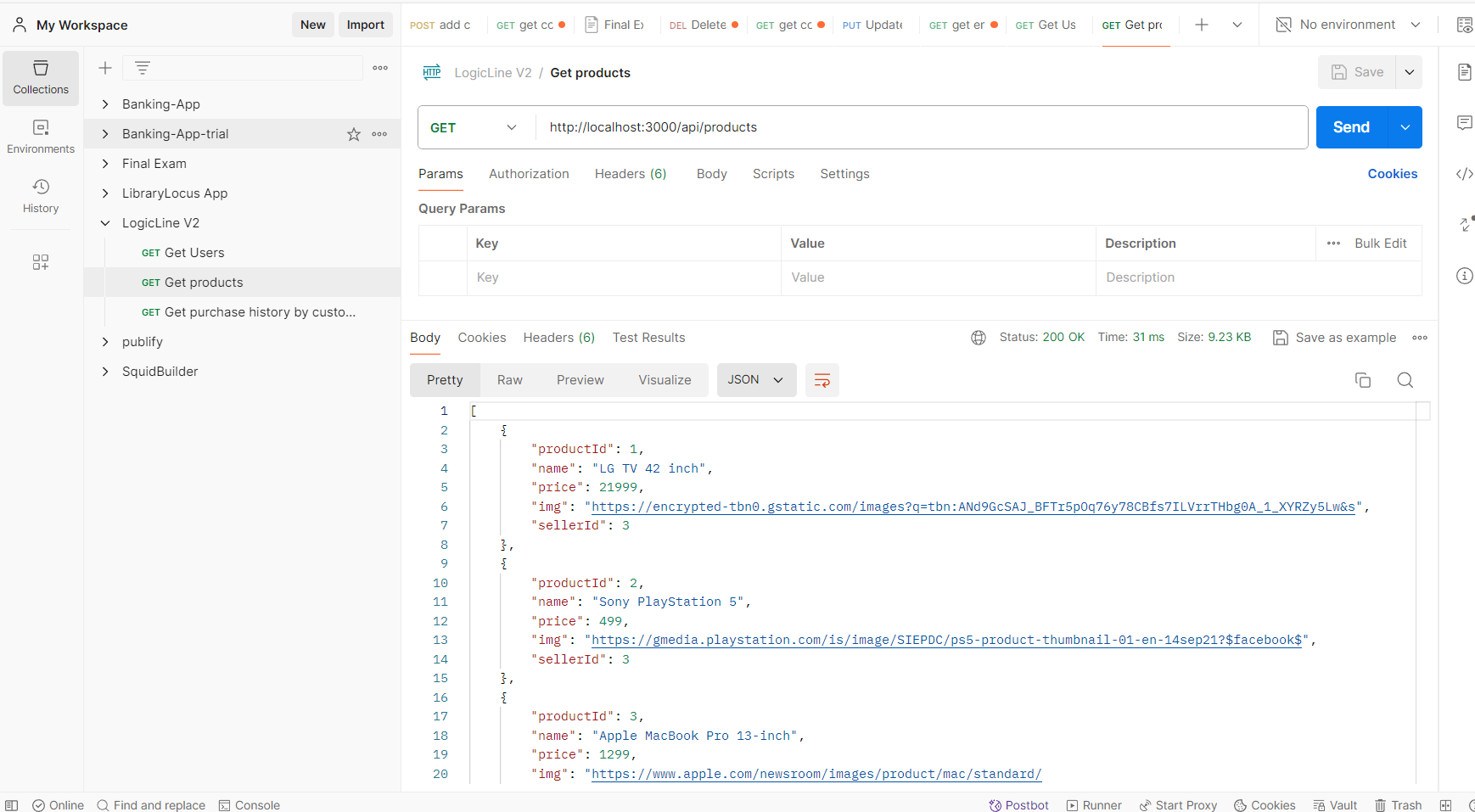
**Conducted tests and evidence.**

To validate the functionality of the implemented routes and methods, we conducted thorough testing using Postman, a robust API testing tool. Postman allowed us to send requests to the API endpoints and observe the responses, ensuring that the routes operated as intended and returned the expected data.

The following figures illustrates the conducted tests:

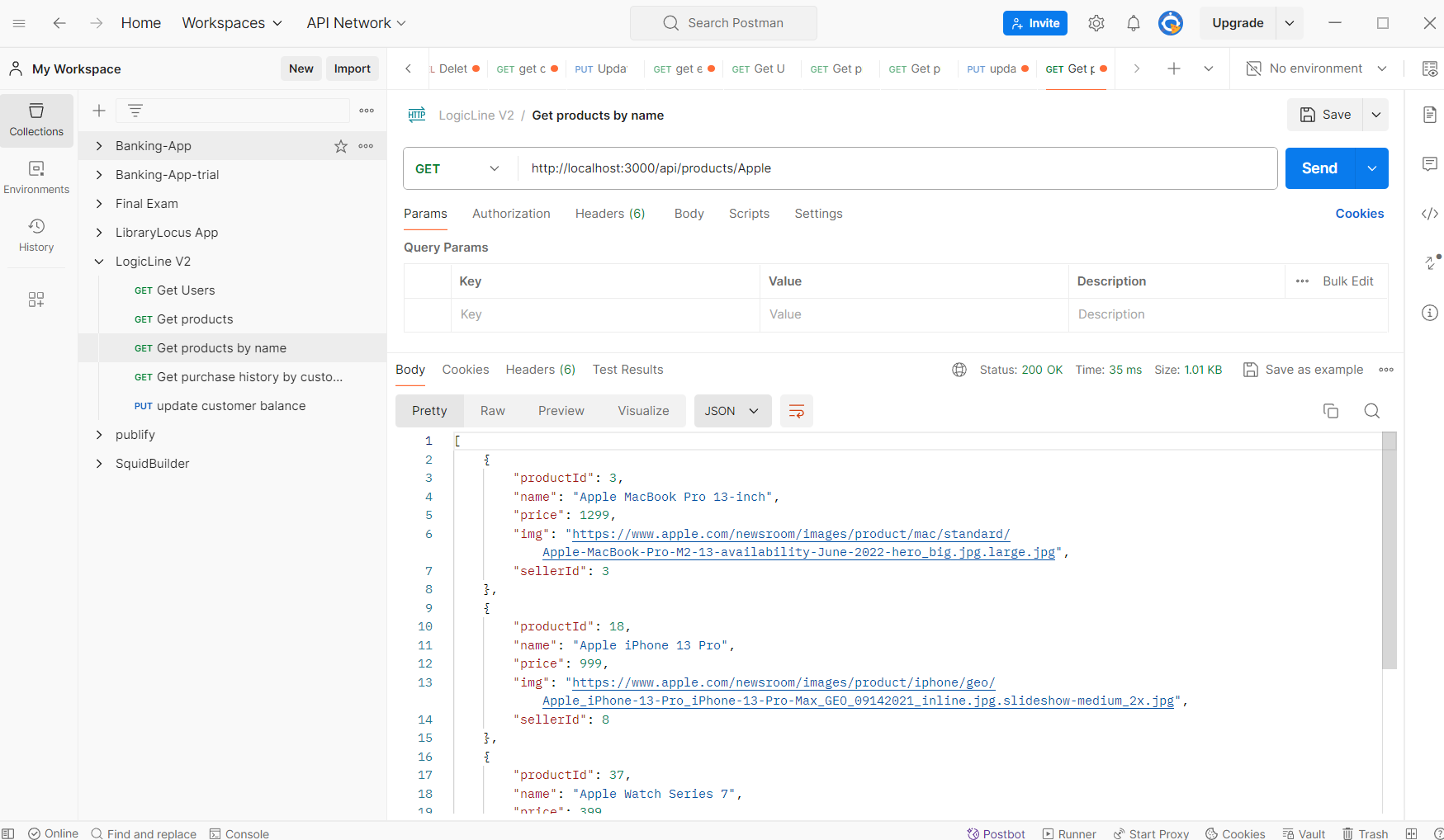
**A screenshot of a computer

Description automatically generatedtesting GET method to retrieve all users from the database:**

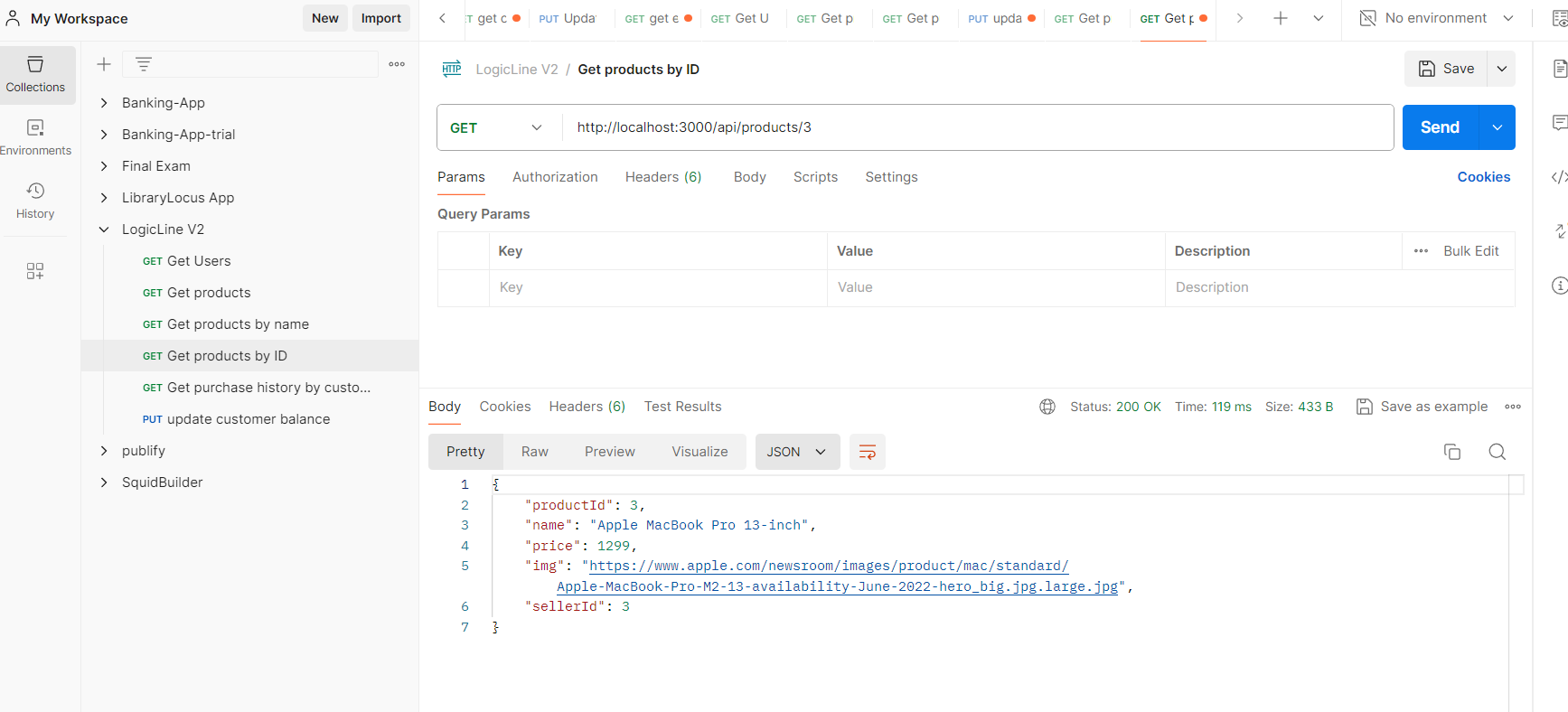
** testing GET method to retrieve all products from the database:**

**testing A screenshot of a computer

Description automatically generatedGET method to retrieve the purchase history by customer ID:**

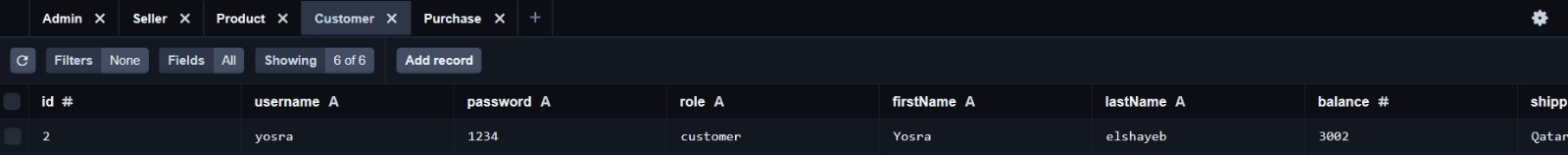
** testing GET method to retrieve a product by name:**

**testing GET method to retrieve a product by id:**

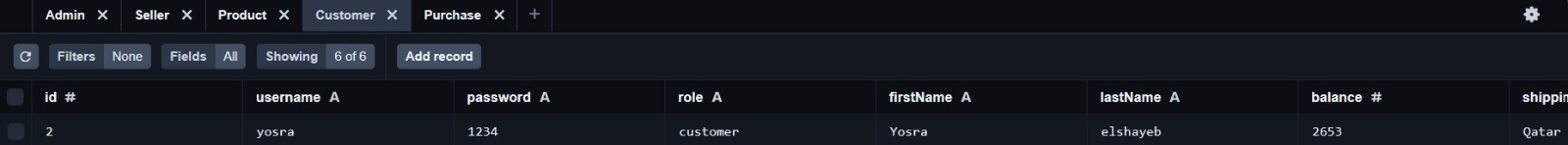
****

**testing updating the balance after purchase:**

* A screenshot of a video game

  Description automatically generated**before purchase:**
* A screen shot of a video game device

  Description automatically generated**after purchase:**



**Deployment:**

**Conclusion:**

In summary, this project served as a practical application of the theoretical concepts learned in class, providing us with an opportunity to self-learn and explore new concepts and resources. Despite encountering challenges such as time constraints, particularly when one team member dropped the course midway through, we persevered and collaborated to complete the remaining tasks, ensuring the project's fulfillment. One notable obstacle we faced was in phase 2 is dealing with the runtime errors of react and troubleshooting the source of the error. However, Prisma was easier to deal with and we had fun implementing the design of the data model.

**Grading Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **%** | **Functionality**\* | **Quality of the implementation** | **Grade** |
| Design and implement the Data Model. | 10 | 10% working and completed |  |  |
| Init DB: populate the database with the data from the Json files in seed.js | 5 | 5% working and completed |  |  |
| APIs and Repository Implementation to read/write data from the database | 25 | 25% working and completed |  |  |
| Statistics use-case with NextJS | 40 | 40% working and completed |  |  |
| **Documentation**   * Data Model diagram. * UI Design with screenshots and description. * Database queries. * Conducted tests and evidence. * **Contribution** of each team member [-10pts if not done] | 20 | 20% Done |  |  |
| **Total** | 100 |  |  |  |
| Bonus - successful deployment of the app and the Database to a cloud hosting service such as <https://vercel.com/> | 5 |  |  |  |
| Copying and/or plagiarism or not being able to explain or answer questions about the implementation. | 0 |  |  |  |