Report | Project

CS349 - Database and Information Systems

Project Name: Zepto Clone

Aryaman Angurana (22B1043)
Krish Rakholiya (22B0927)
Omkar Shirpure (22B0910)
Tanmay Mandaliya (22B1037)

git-repo: https://github.com/sudo-boo/dbis-proj/tree/main

Contents

1 Overall Plan of the Project 1

2 Data 2

2.1 Database Setup 2

3 Backend Development 3

4 Customer Interface 3

5 Vendor Interface 3

6 Delivery Interface 6

7 Integration 6

8 How we did the implementation 6

9 Future Plans and Tasks 7

1 Overall Plan of the Project

We have developed a scalable and efficient delivery platform similar to Zepto. While the exact workings of their system are not publicly disclosed, we hypothesized that their operations function as follows:

Multiple vendors are strategically located in various areas—some in dense urban regions, others in more sparse locations, depending on demand predictions. These vendors serve as localized warehouses for groceries and other items. Delivery agents are stationed near these vendor warehouses and remain idle until an order is assigned to them. Customers are only shown products available from the nearest vendor within a specific range (evident from the Seller License pattern obtained while scraping). When a customer places an order, it is assigned to the closest vendor, and a delivery agent is either automatically assigned or can choose to accept the delivery task.

In our implementation, we are adopting a similar approach. The system will consist of three main components, each with its respective sub-components:

- **Client Interface**
  - Customer
  - Vendor
  - Delivery Agents
- **Backend/Server Interface**
- **Database Management**

Tech Stack Chosen

- Frontend : **Flutter**
- Backend : **Node.js**
- Database : **PostgreSQL**
- Routing : **ngrok**

For the frontend, we chose Flutter for its ease of development and reliability. It allows us to build apps for both Android and iOS simultaneously, ensuring faster development and consistency across platforms.

For the backend, we selected Node.js due to its high performance, scalability, and non-blocking architecture, which is well-suited for real-time applications. Node.js also provides a rich ecosystem of libraries, facilitating rapid development.

We opted for PostgreSQL as our database due to its robustness, scalability, and support for complex queries. Its reliability and ACID compliance are crucial for managing transactions and ensuring data integrity within the platform.

We chose ngrok for its simplicity in exposing local servers to the internet, making it easy to test and develop in real-time during the development phase.

Below are the major milestones we have achieved:

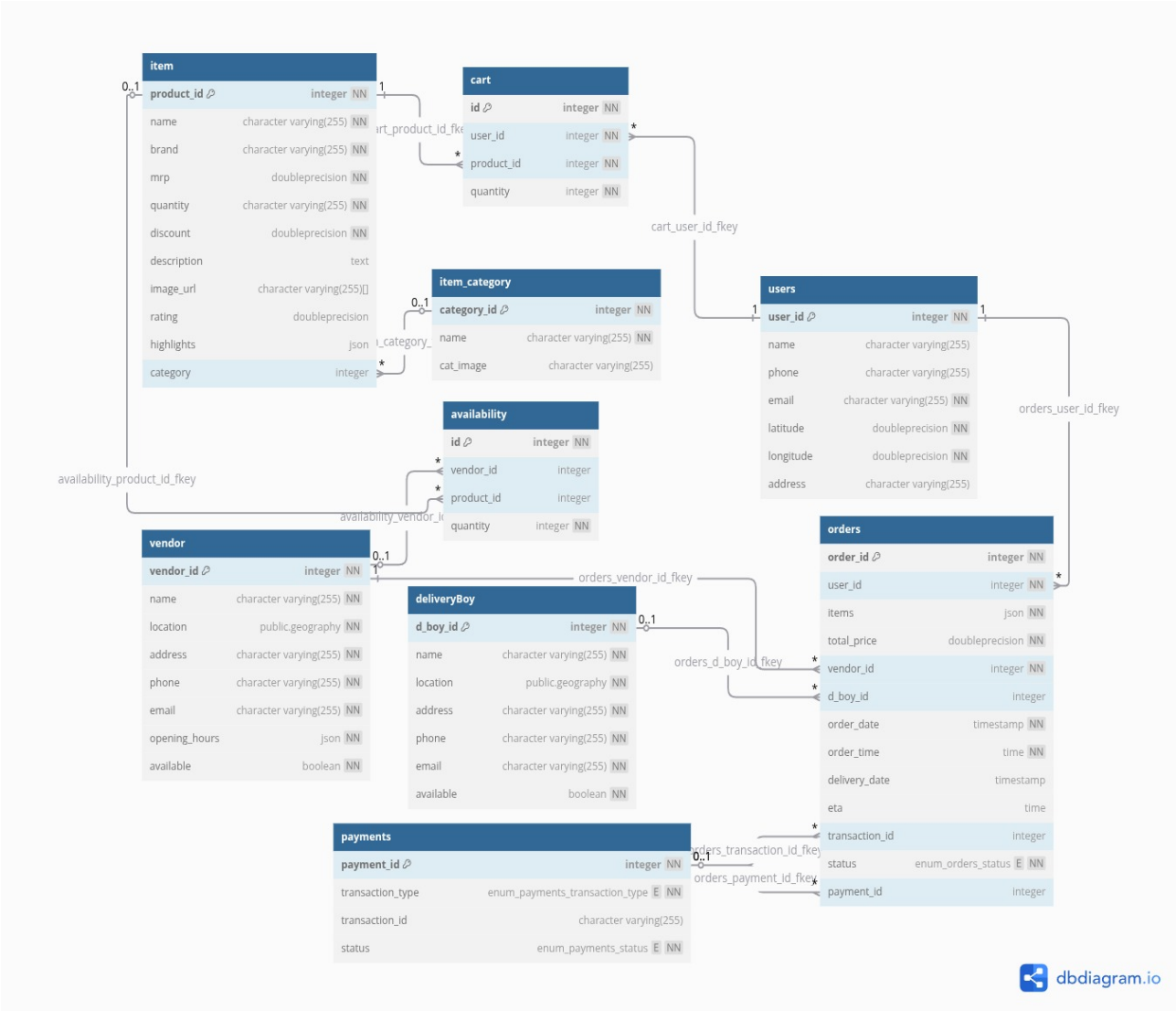
2 Data

- We scraped over 3000 products from Zepto, gathering information about their prices, descriptions, properties, categories, suppliers, etc.
- We successfully obtained detailed item data, including their prices, descriptions, properties, and categories.
- Additionally, we processed the data and made it ready to be put into our database through .sql files

2.1 Database Setup

We have designed and created the database schema, covering entities such as Items, Vendors, Delivery Agents, and several other relations, based on the overall project structure. The PostgreSQL database has been configured, and essential tables for users, products, and orders have been set up. Additionally, we have established multiple relationships between tables to enable faster access to frequently needed data.

Below is the schematic diagram of the schema that we used in our database.



3 Backend Development

We have developed and deployed several core backend functionalities, including:

- APIs for email-based login with OTP verification.
- User creation and management functionalities.
- APIs to fetch products and filter them by category and update products(called from vendor side).
- Cart management functionality (add to cart, remove from cart, get from cart).
- Manage the placing and efficient handling of orders (like updating eta when api is called or changing status of orders by vendors and delivery boys when needed).
- We have set up **ngrok** for routing and exposing local servers to the internet during development.
- All this is done with efficient authorization access with the help of jwt tokens.

4 Customer Interface

The entire customer interface has been fully built using Flutter, including key pages such as:

- Login and user registration pages.
- Home page displaying product categories.
- Product category and individual product pages.
- Cart page for viewing and managing the cart.
- Profile editing and updating features.
- Order confirmation and history page.
- Several reusable components for consistent UI/UX across the app.

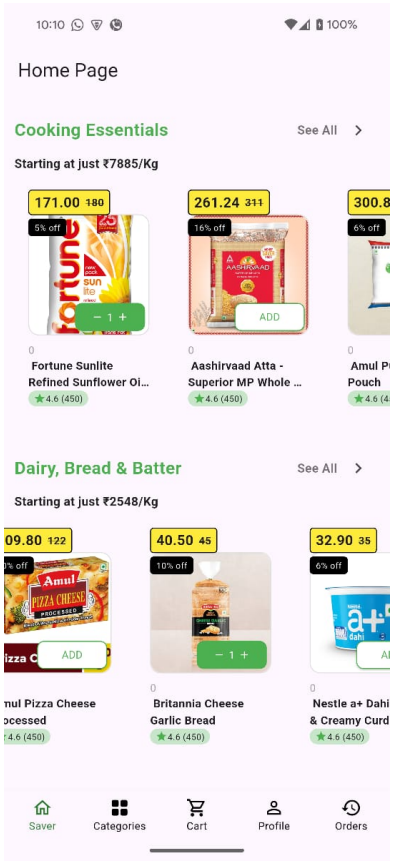
A small glimpse from our customer interface is shown in the figures on the next pages.

5 Vendor Interface

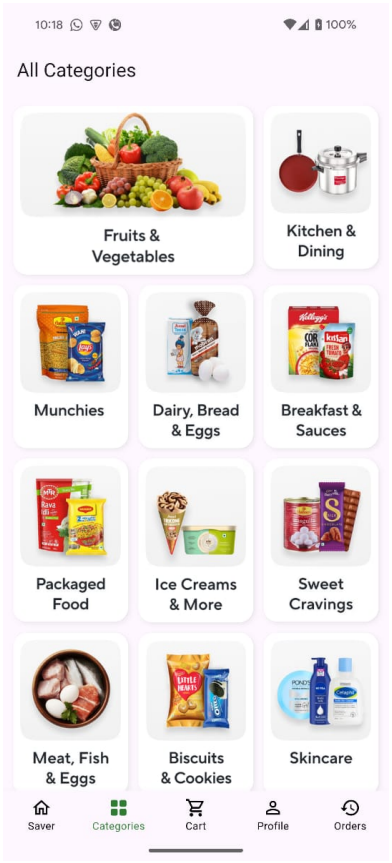
The entire customer interface has been fully built using Flutter, including key pages such as:

- Login and user registration pages.
- Home page displaying product categories.
- Product category and individual product pages, which also allow modifying products.
- Profile editing and updating features.
- Several reusable components for consistent UI/UX across the app.

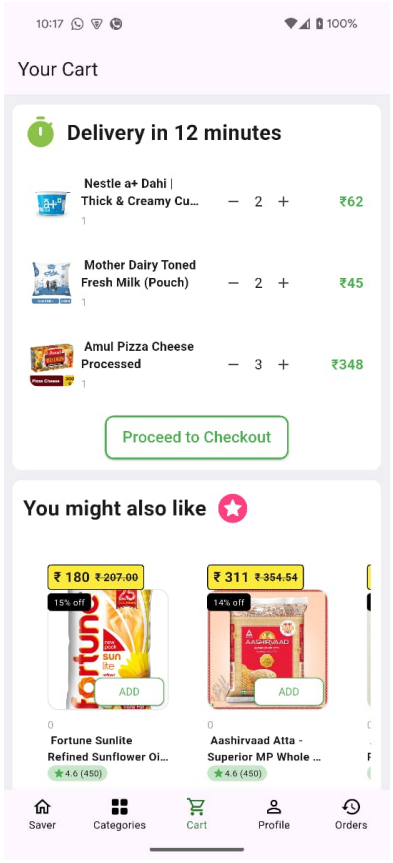
A small glimpse from our Vendor interface is shown on the next pages.



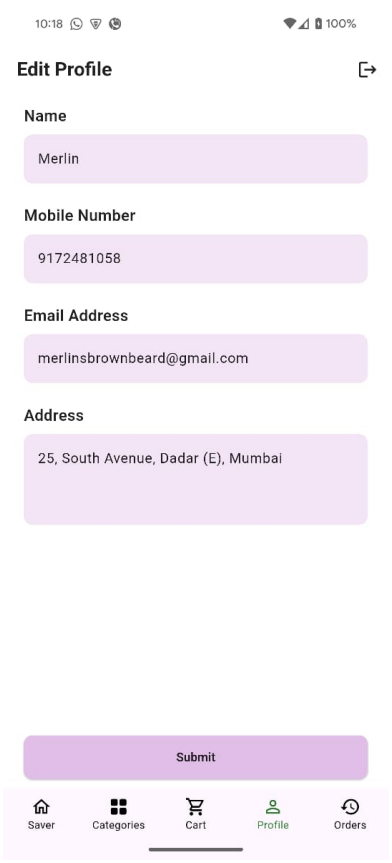
(a) Home Page



(b) Customer Page

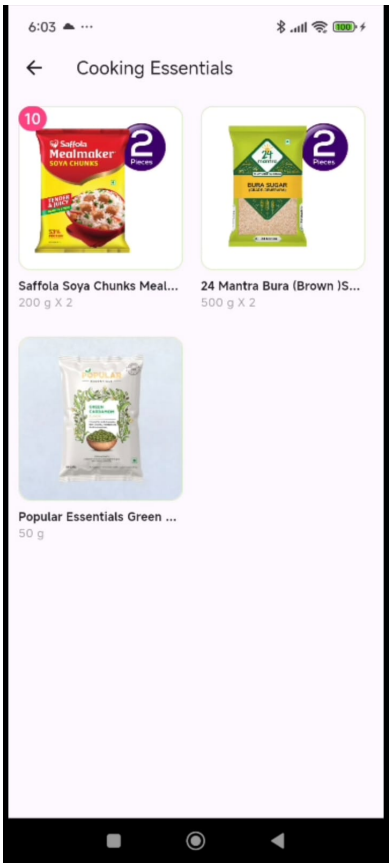


(c) Cart Page



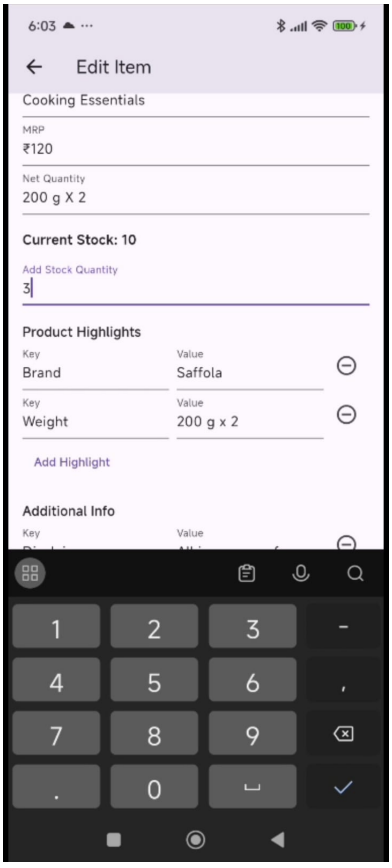
(d) Profile Editing

Figure 1: Customer Interface Screenshots

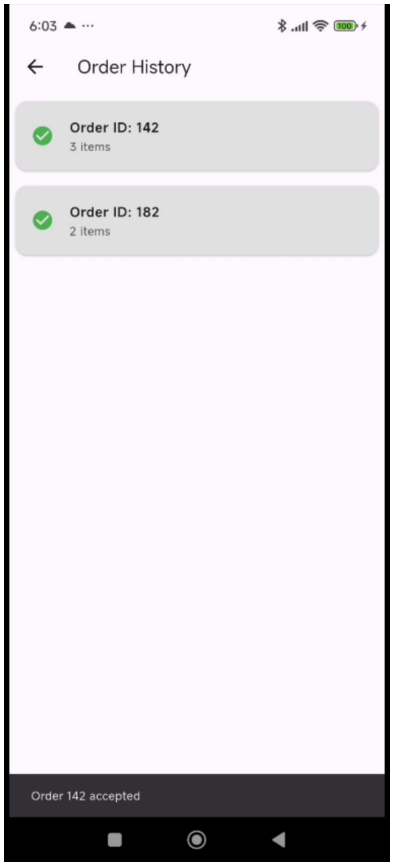


(a) Home Page

(b) Orders Page



(c) Editing Items



(d) Order History

Figure 2: Vendor Interface Screenshots

6 Delivery Interface

The entire delivery interface has been fully built using Flutter, including key pages such as:

- Login and user registration pages.
- Home page displaying the orders to be delivered.
- Profile editing and updating features.
- Order editing features.
- Several reusable components for consistent UI/UX across the app.

7 Integration

We have successfully integrated the frontend and backend for the features developed so far. This includes:

- Complete integration of user login, product browsing, cart management, and order confirmation functionalities.
- Real-time interaction with the live data stored in the database, ensuring the customer interface is fully operational.

8 How we did the implementation

We began by designing a modular and flexible database structure to ensure ease of future modifications. Our development process started with the front-end, followed by the implementation of corresponding backend APIs to support the required functionality. During development, we referred to online resources such as Stack Exchange and utilized AI-based tools to overcome challenges we encountered. While we successfully completed the customer-facing side of the application, we were unable to fully implement the delivery and vendor interfaces within the given timeframe.

9 Future Plans and Tasks

To further build our rapid delivery platform, we have identified several key areas of expansion and optimization.

1. Smarter Search and Product Discovery

We plan to implement an intelligent, AI-powered search system that enhances product discoverability and user convenience by monitoring the daily usage of the app across users. Key features will include:

- **Smart Suggestions**: The app will suggest relevant products in real time as users type in the search bar, based on past behavior, trending items, and contextual keywords.
- **Personalized Recommendations**: Leveraging user preferences and order history, the app will highlight frequently bought items, personalized deals, and restock reminders.

2. Integrated Chatbots for Customer Support

To further improve the user experience and reduce the dependency on manual customer support, we aim to integrate AI-driven chatbots capable of:

- Resolving common queries such as order status, refund policies, and delivery time estimates.
- Assisting users in navigating the app and completing purchases.
- Escalating complex issues to human support agents when necessary.

3. Live Delivery Tracking with Mapping Integration

To provide transparency and build trust with customers, we plan to incorporate a real-time mapping feature within the app. This will allow users to:

- **Track Delivery Agents Live**: View the current location of the delivery person on an interactive map once the order has been dispatched.
- **Get Accurate ETA Updates**: See estimated time of arrival that adjusts dynamically based on real-time traffic and routing.
- **Communicate Location Info Easily**: Allow delivery agents to pinpoint user locations using shared map links or GPS coordinates.

We were not able to implement this because most of the apis used for this are paid and we could not afford them for this particular project.

Conclusion

Ultimately, our goal is to build a robust and user-friendly platform that will contribute to improving the delivery experience for customers, vendors, and delivery agents. Through this process, we have gained invaluable knowledge and experience that will serve as a strong foundation for future projects in the field of database management, system architecture, and app development.