MAP Project

Team Members:

Nour Habra - 202010515 ,
Mustafa Al Hassny - 202010474

Actors:

- User
- System (Backend Server)

Use Cases:

- Authenticate (Login)
- Register (Signup)
- View Dashboard (Home)
- View Profile (Profile)
- View Transactions (Transactions)

==========

◈ Actor: User

- Use Case: Authenticate (Login)

- User enters username and password
- User submits login form
- System validates credentials through post method

- Use Case: Register (Signup)

- User provides personal details
- User submits registration form
- System creates a new user account

- Use Case: View Dashboard (Home)

- User views account balance
- User views latest transactions
- User interacts with credit card widget

- Use Case: View Profile (Profile)

- User views personal information
- User updates personal information
- System saves updated information

- Use Case: View Transactions (Transactions)

1. User views list of all transactions
2. User selects a transaction to view details

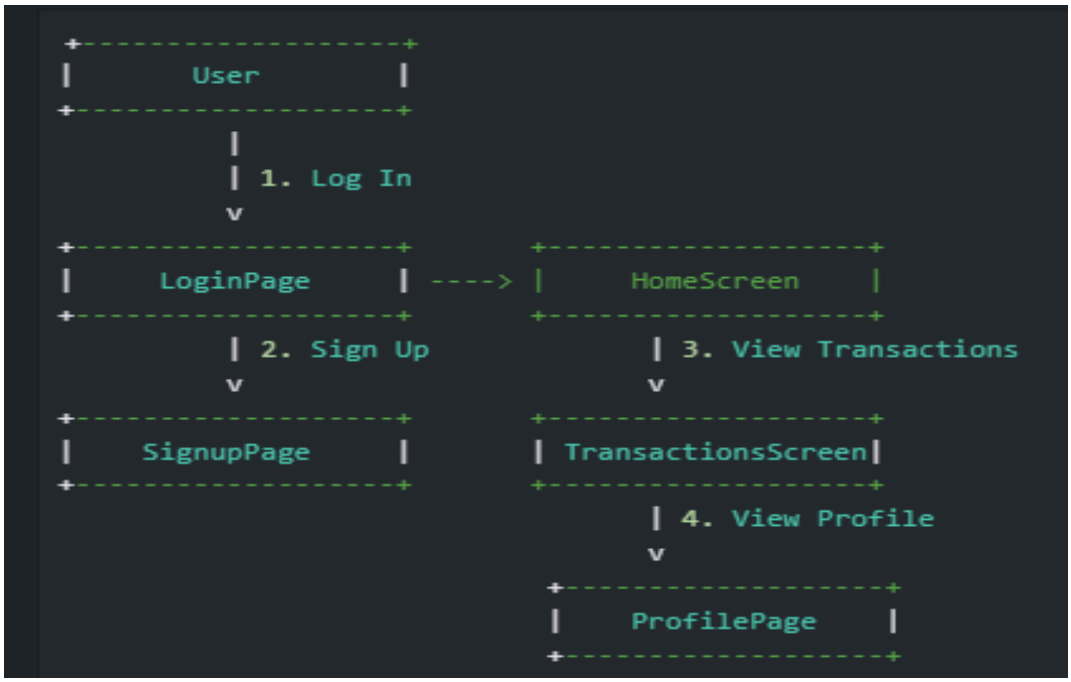
==========

==========

◈ Actor: System (Backend Server)

- Use Case: Authenticate (Login)
    - System receives login request
    - System verifies credentials
    - System responds with success or error message
- Use Case: Register (Signup)
    - System receives registration request
    - System validates and stores user data
    - System responds with success or error message
- Use Case: View Dashboard (Home)
    - System provides account balance
    - System provides latest transactions
- Use Case: View Profile (Profile)
    - System provides user's personal information
    - System updates and saves user's personal information
- Use Case: View Transactions (Transactions)
    - System provides a list of all user's transactions

==========



- Technology Stack

    - Programming Language:

      Dart: The primary programming language used for developing the mobile application, leveraging its reactive programming features and focus on UI.
      Framework:
      Flutter: A UI toolkit from Google for crafting natively compiled applications for mobile, web, and desktop from a single codebase. Flutter is known for its fast development cycles, expressive UIs, and great performance.

    - State Management:
      Provider (assumed): A popular state management technique in Flutter applications, which allows for efficient data flow and state sharing across the app.

    - Database:
      MongoDB: A NoSQL document-oriented database used for storing application data. It provides a flexible schema and is designed for scalability and complex queries, making it suitable for the dynamic data handling required by a mobile banking application.

    - Backend Services:
      Node.js (assumed): A JavaScript runtime for building scalable network applications, likely used for creating backend services.
      Express.js (assumed): A web application framework for Node.js, designed for building web applications and APIs.

    - Package Management:
      Flutter Pub Dev: The package manager for Flutter, providing reusable libraries and packages for various functionalities.
      Version Control:
      Git: A distributed version control system used to track changes in the source code during development.

    - IDE:
      Android Studio or Visual Studio Code: Integrated development environments used to write Dart code and develop Flutter applications.

- Project Overview

  Our mobile banking application," EvilBank Mobile," allows users to perform various banking activities such as authenticating, registering, viewing their dashboard, profile, and transaction history.

- Existing Components

  - Login (lib/login.dart): Allows users to authenticate by entering their username and password. It also provides the functionality to remember the user's credentials and navigate to the signup page.

  - Signup (lib/signup.dart): Enables new users to create an account by providing their personal details and submitting a registration form.

  - Main (lib/main.dart): Serves as the entry point of the application, setting up the theme and home route, which currently points to the login page.

  - Home (lib/home.dart): Presents the user's dashboard, including a credit card widget, account balance, and latest transactions. It also includes navigation to other parts of the app through a custom bottom navigation bar.

  - Profile (lib/profile.dart): Although not provided, we can assume this component allows users to view and edit their personal information.

  - Transactions (lib/transactions.dart): Displays a list of all transactions made by the user, with the ability to fetch transaction data from a backend service.

- Functional Requirements

  - Authentication: Securely authenticate users and handle login sessions.

  - User Registration: Allow new users to register and create an account.

  - Dashboard: Show an overview of the user's financial status and recent transactions.

  - Profile Management: Enable users to view and update their personal information.

  - Transaction History: Provide a detailed list of past transactions with filtering and search capabilities.

- **Non-Functional Requirements :**

    - Usability: The app should have an intuitive and user-friendly interface.
    - Performance: The app should perform efficiently, with minimal latency in fetching and displaying data.
    - Security: Sensitive information such as passwords and financial data should be encrypted and securely transmitted.
    - Scalability: The app should be able to handle an increasing number of users and transactions without degradation in performance.
    - Maintainability: The code should be well-organised and documented to facilitate updates and maintenance.

- **Potential Improvements**

    - Enhanced Security: Implement multi-factor authentication and improve encryption methods.
    - User Experience: Improve the UI/UX design for a more engaging and modern look.
    - Feature Expansion: Add features such as bill payments, fund transfers, and notifications for account activity.
    - Analytics: Integrate user analytics to understand behaviour and improve services.
    - Testing: Establish a comprehensive testing framework to ensure the reliability of the application.

- **Resources and Tools**

  In the development of the "EvilBank Mobile" application, several resources and tools have been instrumental in shaping the project's design and functionality. These resources have provided the necessary components and design frameworks to create a robust and user-friendly mobile banking application.

    - Design Tool : Figma

      Usage: Figma has been utilised for UI/UX design, allowing designers to create high-fidelity prototypes and collaborate in real-time. The design team has used Figma to map out the user interface, ensuring a modern and intuitive experience.
      Impact: The use of Figma has streamlined the design process, enabling quick iterations and immediate feedback. This has resulted in a cohesive and visually appealing interface that aligns with the application's usability goals.

    - Component Library : Flutter Pub Dev

Usage: Flutter Pub Dev, the package manager for Flutter, has been a source for third-party packages and components. Developers have leveraged a variety of packages to implement features such as form validation, network requests, state management, and more.
Impact: By incorporating packages from Flutter Pub Dev, the development team has been able to accelerate the development process, reduce the need for custom code, and ensure that the application adheres to best practices.

- **Market Analysis and Competitive Landscape** :

  In the realm of digital banking and financial services, "EvilBank Mobile" enters a market with well-established players. Services such as PayPal, Google Pay, and Wise have set high standards for convenience, security, and user experience. To position "EvilBank Mobile" effectively, it is crucial to analyse these services and identify areas for differentiation and innovation.

  - **PayPal** : PayPal is a global leader in online payment solutions, offering users the ability to make online transactions, send and receive money, and access a range of financial services. It is known for its ease of use, extensive user base, and robust security measures.

  - **Google Pay** : Google Pay, developed by Google, is a digital wallet platform and online payment system that enables users to make payments with Android phones, tablets, or watches. It integrates with other Google services and offers a seamless payment experience across various Google platforms.

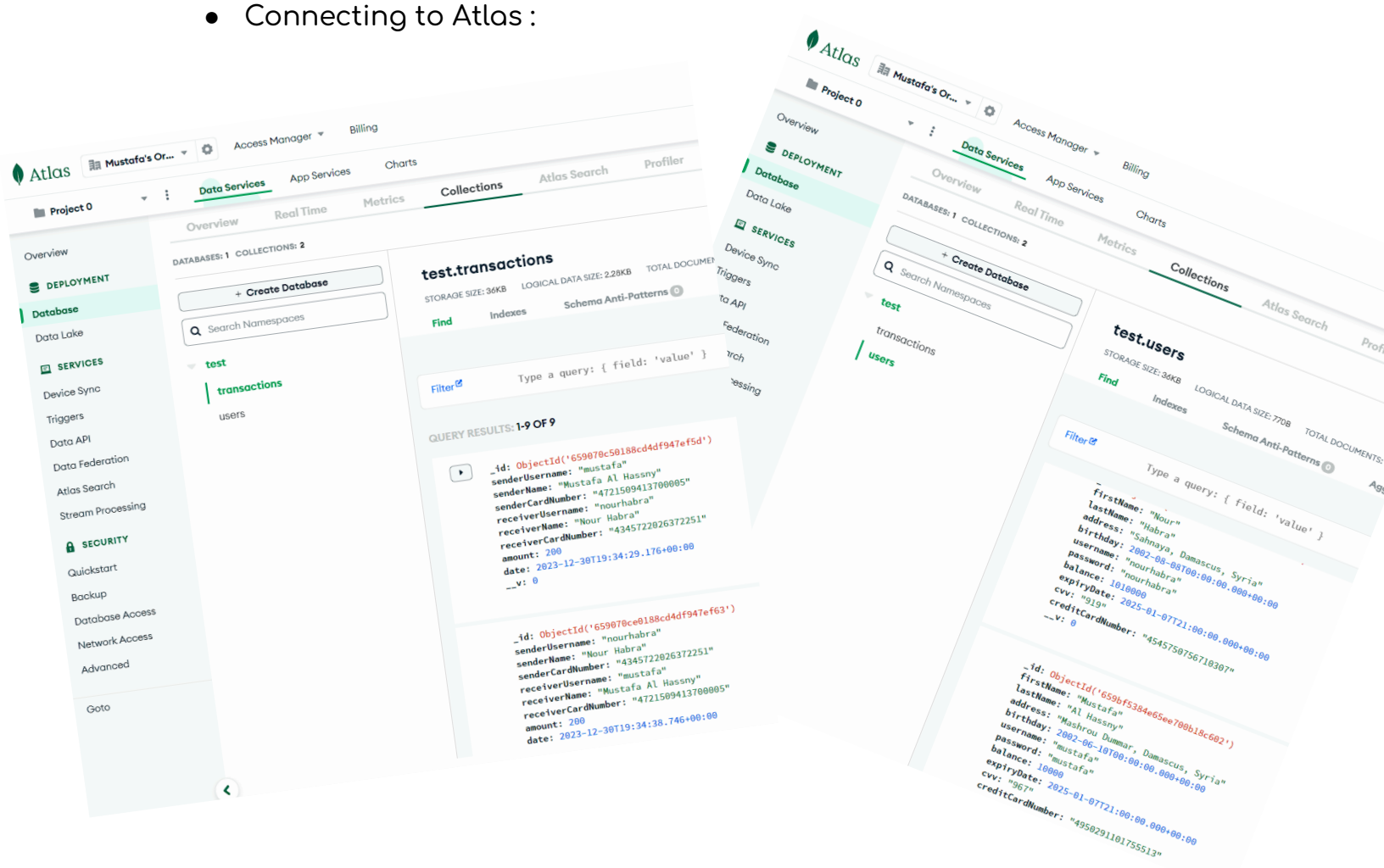
  - **Wise** : Wise is a financial technology company that provides international money transfer and currency exchange services. It is celebrated for its transparent fee structure, real exchange rates, and efficient cross-border transactions.

- **Conclusion**

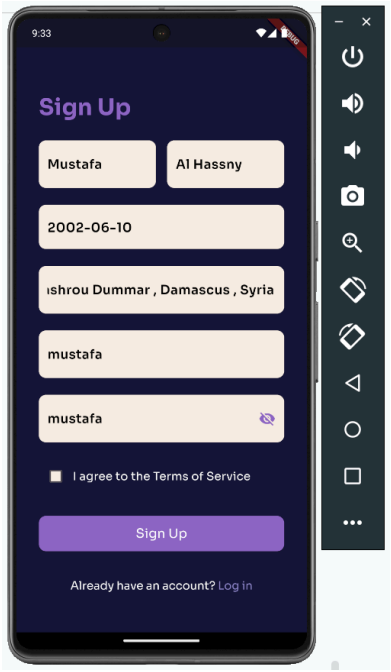
  The "EvilBank Mobile" project provides a solid foundation for a mobile banking application. However, there is room for improvement in terms of security, user experience, and feature set. A detailed analysis of each component and user feedback can guide the development of future iterations of the application.

-

- Testing
    - Connecting to Atlas :



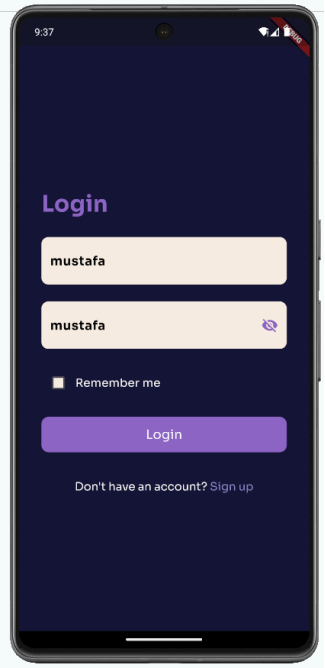
We find that our user
is being initialised
correctly using out
Node.js server ,
And all data are being correctly
Provided (CardNumber, Cvv, ExpiryDate)
or generated (FirstName, LastName,
BirthDate, Username, Password).



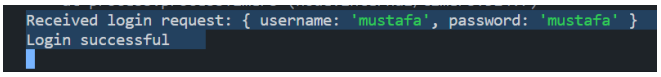
Notice how the node.js server is generating
Needed data which indicates
for a successful signup

```
Generated Expiry date 2025-01-13T21:00:00.000Z
Generated CVV 504
Generated Card Number 4930157506790370
```

Now after a Successful register we are
Redirected to the login page
which demands for a
username and
a password



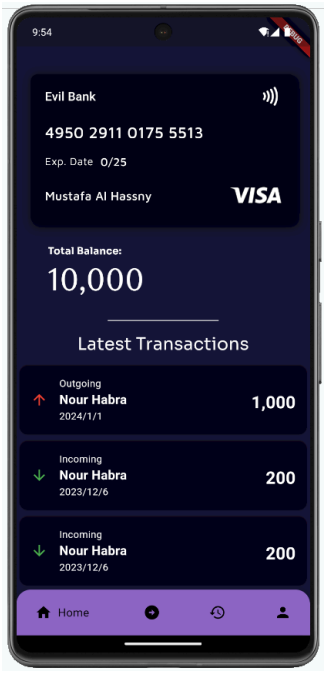
Notice how the node.js does its role
by indicating that the login
was successful



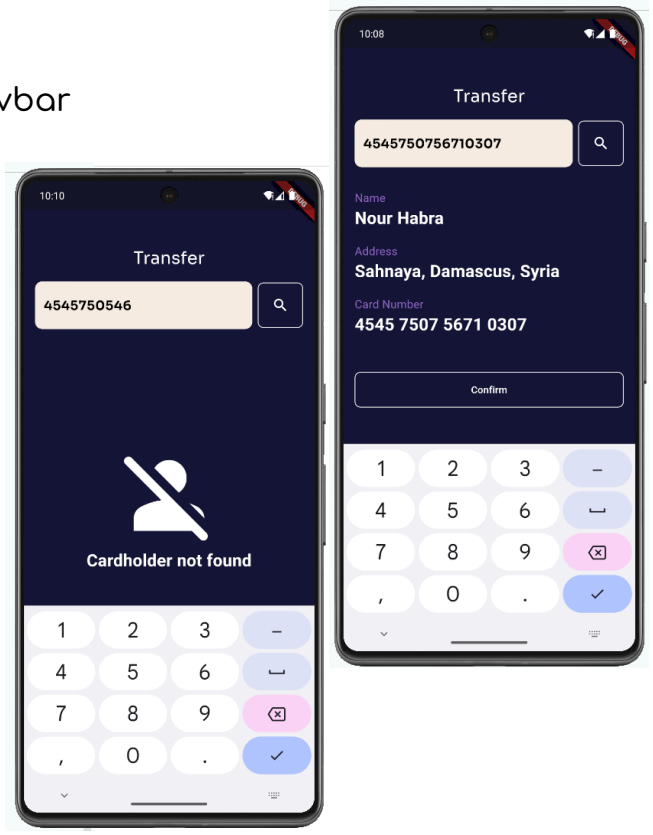
After this the user is navigated into
the HomeScreen which
should look like this :

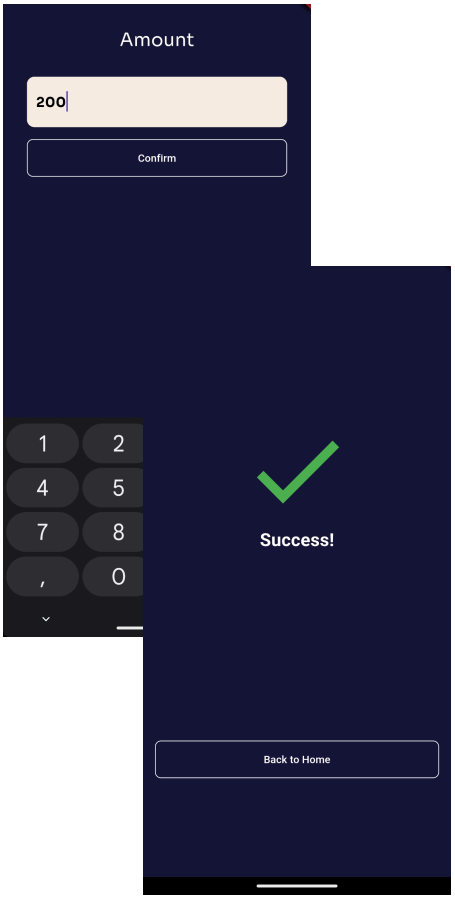
Components of home screen are :

- Credit card component with it being clickable to show the cvv
- Total user balance
- Last 5 transactions showing if the transaction is outgoing or incoming and the amount of it
- Bottom NavBar



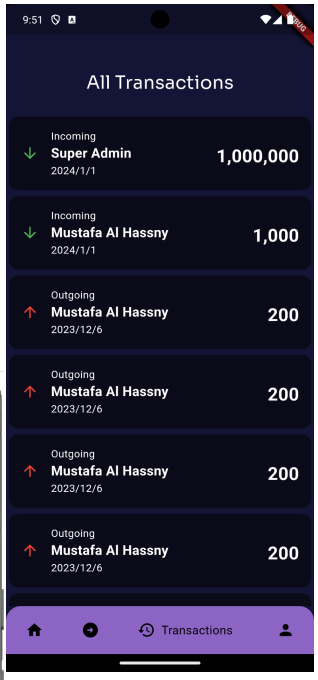
Now the second icon in the bottom navbar navigates user to a transaction page where the user inputs the recipient card number and a search button to search of there is an existing user



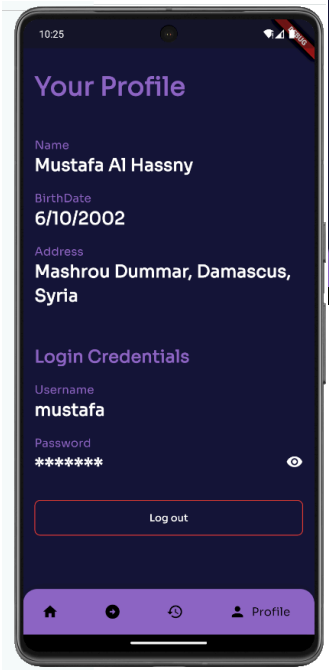


After a valid number it prompts to give the amount to transfer and if everything checks out like balance of first account is lower than the amount to be transferred a successful sign is shown for confirmation

Third Icon in the BottomNavBar navigates to a screen where all transaction that the user has ever made are displayed



And the fourth and last icon in the BottomNavBar navigates to a profile page where all user personal data are shown and a button to log out of the logged in account



==========

For better convenience we used Render.com to deploy the backend on it instead of running on a local server