**FINANCE MANAGEMENT APPLICATION**

***A***

***Project Report***

*submitted in partial fulfillment of the*

*requirements for the award of the degree of*

**BACHELOR OF COMPUTER APPLICATIONS**

**by**

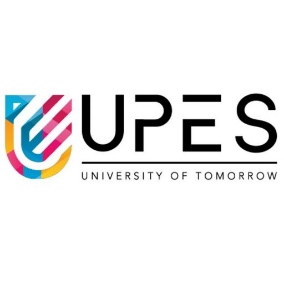
|  |  |
| --- | --- |
| **Name** | **Roll No.** |
| **Aviral Birthare** | **R252222132** |
| **Munendra Yaduvnshi** | **R252222136** |
| **Harsh Sharma** | **R252222137** |
| **A Teja** | **R252222158** |

***under the guidance of***

J Dhiviya Rose

Assistant Professor – Selection Grade

AI Cluster, School of Computer Science



**School of Computer Science**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, Uttarakhand**

**December – 2023**

**CANDIDATE’S DECLARATION**

We hereby certify that the project work entitled **“FINANCE MANAGEMENT APPLICATION”** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF COMPUTER APPLICATIONS with specialization in AI/ML and submitted to the AI Cluster, School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **August 2024** to **November 2024** under the supervision of **J Dhiviya Rose, Assistant Professor-Selection Grade, AI Cluster, SOCS.**

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

|  |  |
| --- | --- |
| **Name** | **Roll No.** |
| **Aviral Birthare** | **R252222132** |
| **Munendra Yaduvnshi** | **R252222136** |
| **Harsh Sharma** | **R252222137** |
| **A Teja** | **R252222158** |

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 25-Nov-24 **J Dhiviya Rose**

Project Guide

**ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our guide Prof. J Dhiviya Rose for all advice, encouragement and constant support she has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thanks to our respected Cluster Head, Prof. Anil Kumar, AI Cluster, for his great support in doing our project in **FINANCE MANAGEMENT APPLICATION.**

We are also grateful to Dean, Prof. Vijaysekhar Chellaboina, SoCS for giving us the necessary facilities to carry out our project work successfully. We also thanks to our Course Coordinator & Activity Coordinator, Prof. J Dhiviya Rose for providing timely support and information during the completion of this project.

We would like to thank all our friends for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our parents who have shown us this world and for every support they have given us.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Aviral Birthare** | **Munendra Yaduvanshi** | **Harsh Sharma** | **A teja** |
| **Roll No.** | **R252222132** | **R252222136** | **R252222137** | **R252222158** |

**ABSTRACT**

The Financial Management Application is designed to help users manage their personal finances easily and securely. It includes features like real-time tracking of transactions, secure money transfers, and managing multiple bank accounts from one platform. The app is built using modern technologies like React.js, Next.js, and Node.js to ensure a smooth user experience and a strong backend. By integrating with APIs like Plaid, it fetches real-time banking data securely, making it a reliable tool for financial management.

This application aims to offer a seamless experience, helping users manage all their financial needs through one platform. Whether transferring money between accounts or tracking daily expenses, the app will simplify the process, providing a secure and intuitive interface. Users will be able to view their overall financial status, connect and manage multiple bank accounts, and track real-time updates on their transactions.

The app’s features include an account overview that displays the total balance across all connected banks, a detailed transaction history, and secure fund transfers. Additional functionalities such as user authentication will enhance security, while a responsive design will ensure the app is accessible and user-friendly on both mobile devices and desktop screens. By integrating Next.js for server-side rendering, Tailwind CSS for a sleek user interface, and Plaid for secure bank connections, the application will empower users to make well-informed financial decisions, ultimately helping them reach their financial goals [2].

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Topic** | | **Page No** |
| Table of Content | |  |
| 1 | Introduction | 6 |
|  | 1.1 Prologue | 6 |
|  | 1.2 Problem Statement | 6 |
|  | 1.3 Objectives | 6 |
|  | 1.4 System Requirements | 6 |
| 2 | SYSTEM ANALYSIS | 9 |
|  | 2.1 Motivation | 9 |
|  | 2.2 Proposed System | 10 |
| 3 | SYSTEM DESIGN | 11 |
|  | 3.1 Flow Chart Diagrams | 11 |
| 4 | SYSTEM IMPLEMENTATION | 17 |
|  | 4.1 Methodology | 17 |
|  | 4.2 Algorithms | 18 |
|  | 4.3 Sample Output | 20 |
| 5 | LIMITATIONS AND FUTURE ENHANCEMENTS | 22 |
| 6 | CONCLUSIONS | 23 |
| 7 | REFERENCES | 24 |

**Chapter 1**

**INTRODUCTION**

**1.1Prologue**  
The Finance Management Application we are developing will leverage modern technologies to provide a user-friendly and secure platform for managing various financial activities. The app will be built using React.js, Next.js, Tailwind CSS, and other essential tools for front-end development. On the back-end, Node.js and TypeScript will be used, along with other supporting software. For deployment, we will utilize Vercel, ensuring a smooth and efficient hosting process.

This application aims to offer a seamless experience, helping users manage all their financial needs through one platform. Whether transferring money between accounts or tracking daily expenses, the app will simplify the process, providing a secure and intuitive interface. Users will be able to view their overall financial status, connect and manage multiple bank accounts, and track real-time updates on their transactions. [1]

**1.2Problem Statement**

Lack of a secure and user-friendly finance management application that enables users to efficiently track, manage, and transfer funds across multiple bank accounts in real-time. Also, applications that provids features such as transaction history, account overview, and financial insights in a single integrated application which can be accessible on both mobile and desktop devices is also not available.

**1.3Project Objective**

The main and primary objective of this project is to create a user-friendly Finance Management App that simplify the personal finance management of the user by using comprehensive tools for effective management enabling decision-making and financial stability. The app will allow user to project focuses on building a fully functional online banking platform using modern web technologies. The platform will allow users to manage their finances efficiently and securely. Key features include:

1. **Real-Time Transaction Displays**: Users can see their recent transactions and account balances instantly.
2. **Secure Money Transfers**: Transfer funds between accounts with enhanced security measures.
3. **Multi-Bank Integration**: Connect and manage multiple bank accounts from a single interface.

**1.4System Requirements**

**Software Requirements**

1. **Operating System**: Windows 10/11, macOS, or Linux (for local development and deployment).
2. **Frontend Framework**:

**-React.js**: For building the user interface.

**-Tailwind CSS**: For styling and creating a responsive design.

1. **Backend Framework**:

**-Node.js**: For handling server-side logic.

**-Appwrite**: For database management and user authentication.

1. **APIs**:

**-Plaid API**: For multi-bank integration and fetching real-time financial data.

**-Dwolla API**: For secure fund transfers.

1. **Database**:

**-Supabase**: To store user data, transactions, and account details.

1. **Deployment Platform**:

**-Vercel**: For deploying the application with high availability.

1. **Development Tools**:

-Code Editor: VS Code or equivalent.

-Version Control: Git and GitHub for collaboration and source control.

**Hardware Requirements**

1. **For Development**:

**-Processor**: Intel Core i5 (or equivalent) or higher.

**-RAM**: Minimum 8 GB (16 GB recommended for smoother performance).

**-Storage**: At least 50 GB free space for code, dependencies, and databases.

**-Display**: Full HD (1920x1080 resolution) or higher.

1. **For Users**:

-Any device capable of running a modern web browser (Google Chrome, Mozilla -Firefox, etc.).

-Stable internet connection for real-time updates and secure data transfers.

-Compatible with both mobile devices and desktops.

**Chapter 2**

**SYSTEM ANALYSIS**

**2.1Motivation**

Existing financial management tools either lack essential features like real-time updates or fail to provide a user-friendly experience, leaving users frustrated and financially disorganized. Additionally, concerns about data security and privacy further discourage users from adopting digital finance solutions.

The motivation behind developing the **Financial Management Application** stems from the need to address these challenges by creating a comprehensive, secure, and intuitive platform. By leveraging modern technologies, this application aims to empower users to manage their finances effortlessly, gain real-time insights, and ensure data protection. It is designed to simplify financial management for individuals, enabling them to achieve better financial stability and confidence.

**2.2Proposed System**

The proposed system is a comprehensive **Financial Management Application** that simplifies personal finance management by integrating real-time tracking, secure transactions, and multi-bank account management into a single, user-friendly platform. This system leverages modern web technologies and secure API integrations to ensure seamless and efficient operation.

**Key Features:**

1. **Real-Time Transaction Tracking**:  
   Users can monitor their recent transactions and account balances instantly, enabling them to stay updated with their financial activities.
2. **Secure Fund Transfers**:  
   The application ensures safe and encrypted fund transfers between accounts, with robust authentication mechanisms like multi-factor authentication.
3. **Multi-Bank Integration**:  
   Users can connect and manage multiple bank accounts from a single platform, reducing the hassle of switching between banking apps.

**Technologies Used:**

* **Frontend**: React.js and Tailwind CSS for creating an intuitive and responsive user interface.
* **Backend**: Node.js and Appwrite for server-side logic, database management, and user authentication.
* **APIs**: Plaid for bank account integration and Dwolla for secure payment processing.
* **Deployment**: Vercel for hosting the application with high availability and scalability.

**Chapter 3**

**SYSTEM DESIGN**

**3.1 Design Diagrams**

* **Use-Case Diagram:** Illustrates interactions between the user and the application's primary features, such as User Authentication, generating Financial Insights, Multi-Bank Integration, Transfer Fund, Track Transactions in Real-time, and View Account. This diagram ensures all user actions and Administrator action are accounted for and aligned with system functionalities.

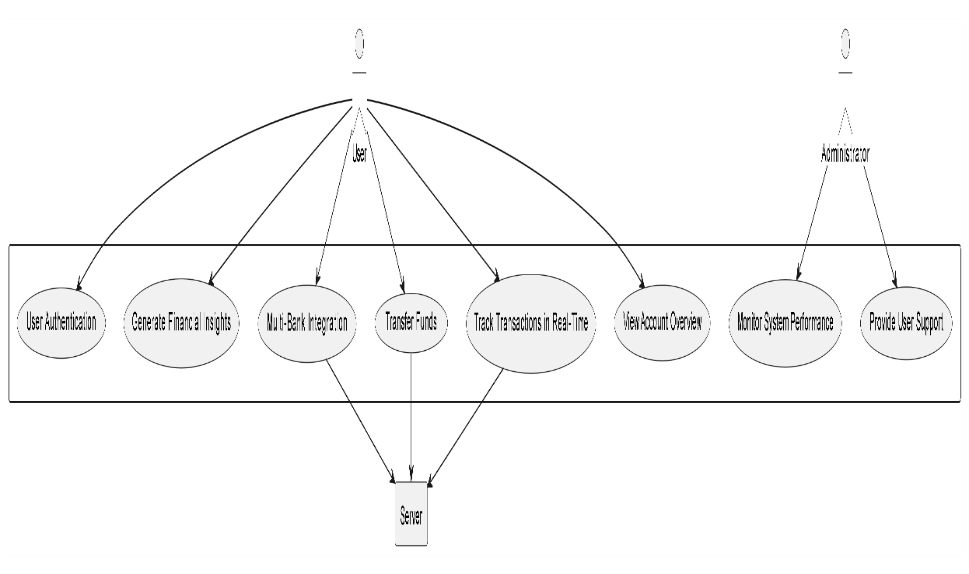
****

Figure 1: Use Case Diagram

* **Class Diagram:** Represents the relationships between core components of the system, including User, Account, application, and Transaction entities. Each class includes attributes and methods that define its behavior, such as view Balance () for the Account class or initiate Transaction () for the Transaction class.

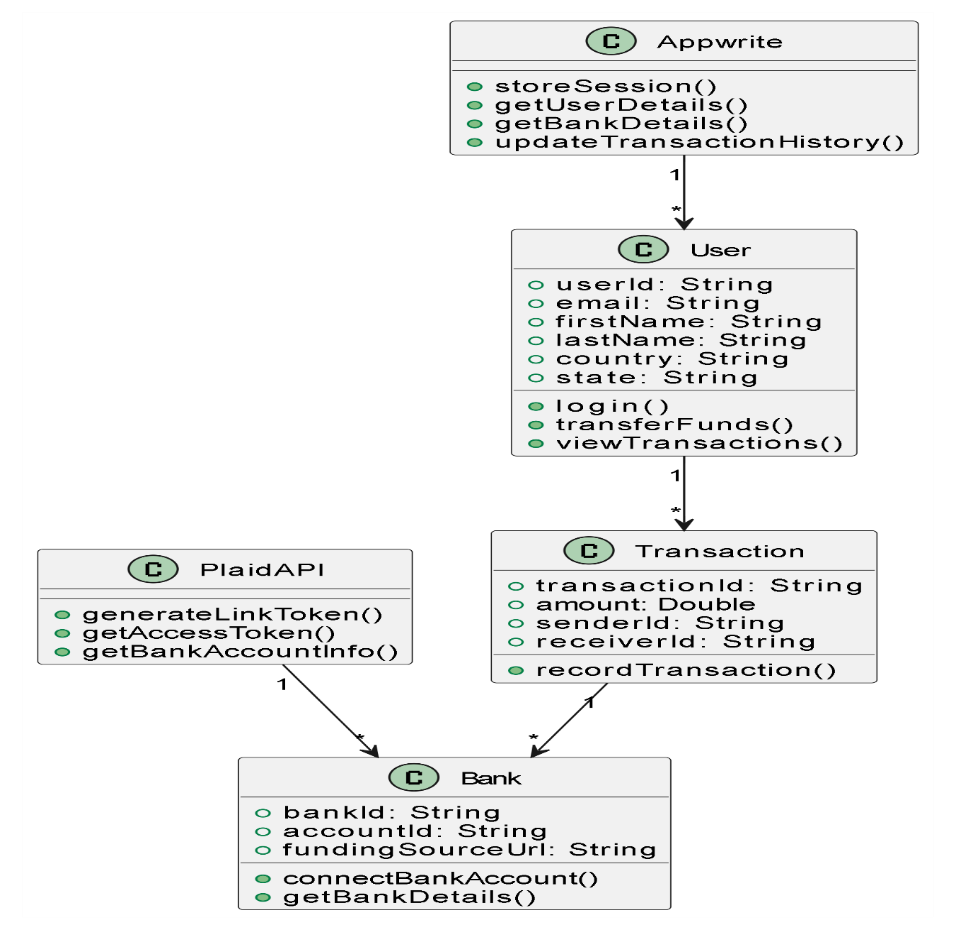


Figure 2: Class Diagram

* **Activity Diagram:** Maps out workflows like logging in, authenticating users, and executing fund transfers. It provides a step-by-step view of how users interact with the application.

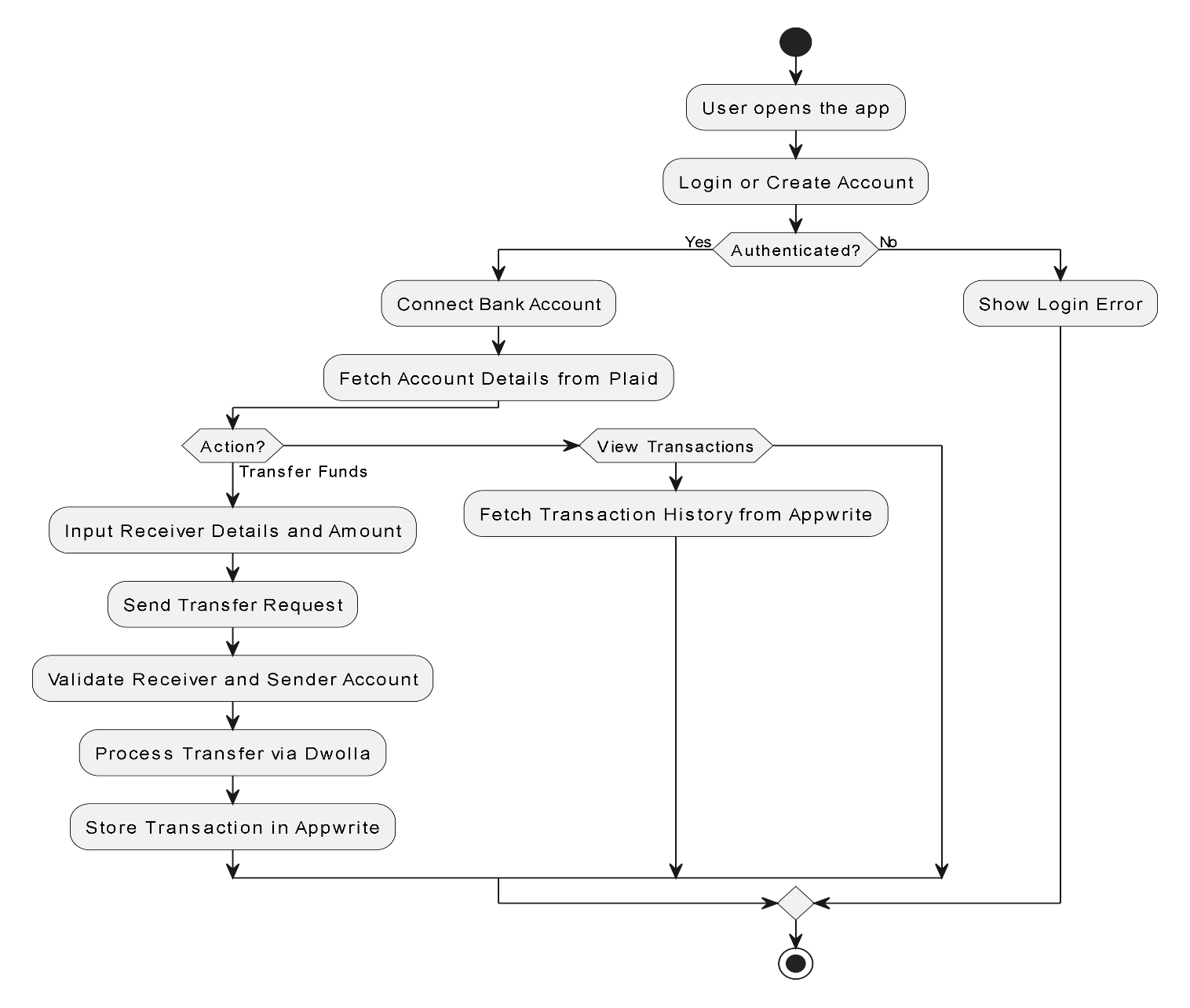


Figure 3: Activity Diagram

* **Sequence Diagram:** The sequence of interactions happing between the user interface, backend, and external APIs during a transaction process, ensuring synchronization across components.

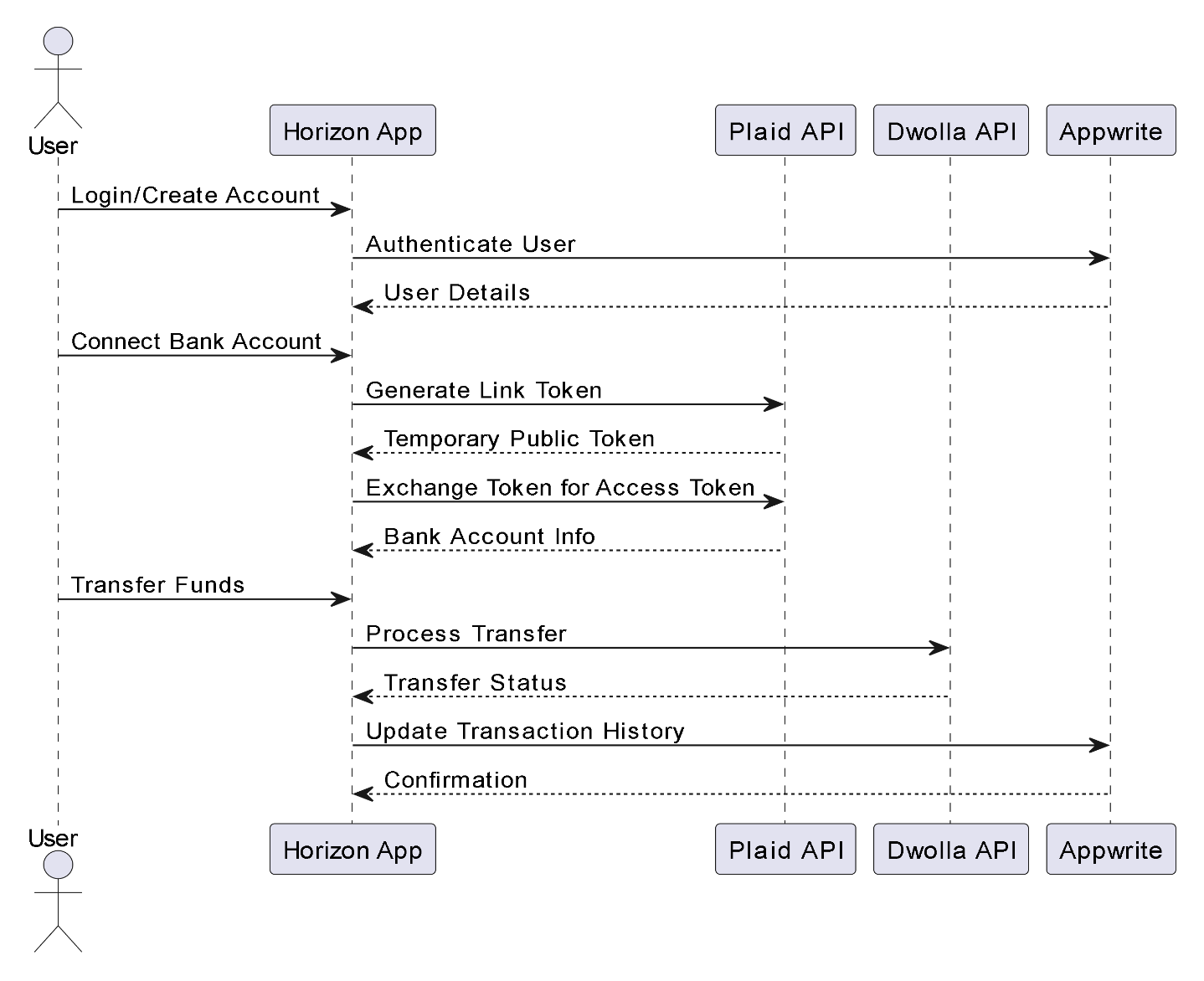


Figure 4: Sequence Diagram

* **Data Flow Diagram (DFD):** Demonstrates the movement of data, starting from user input (e.g., fund transfer request) to data storage in the database and retrieval via APIs.

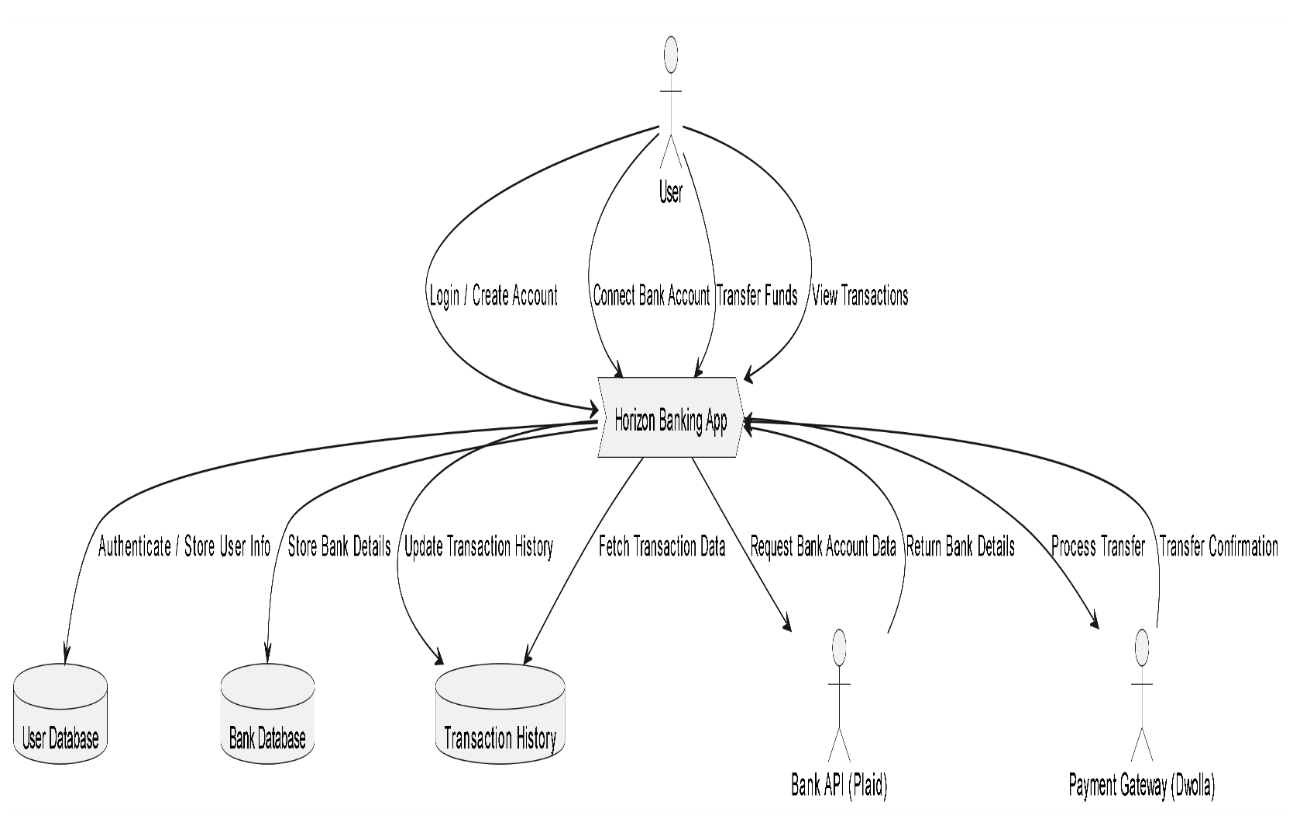


Figure 5: Data Flow Diagram

* **State Diagram:** Shows the various states of a transaction, when the user initiation to processing, and eventual completion or failure, ensuring that the system handles all possible scenarios.,

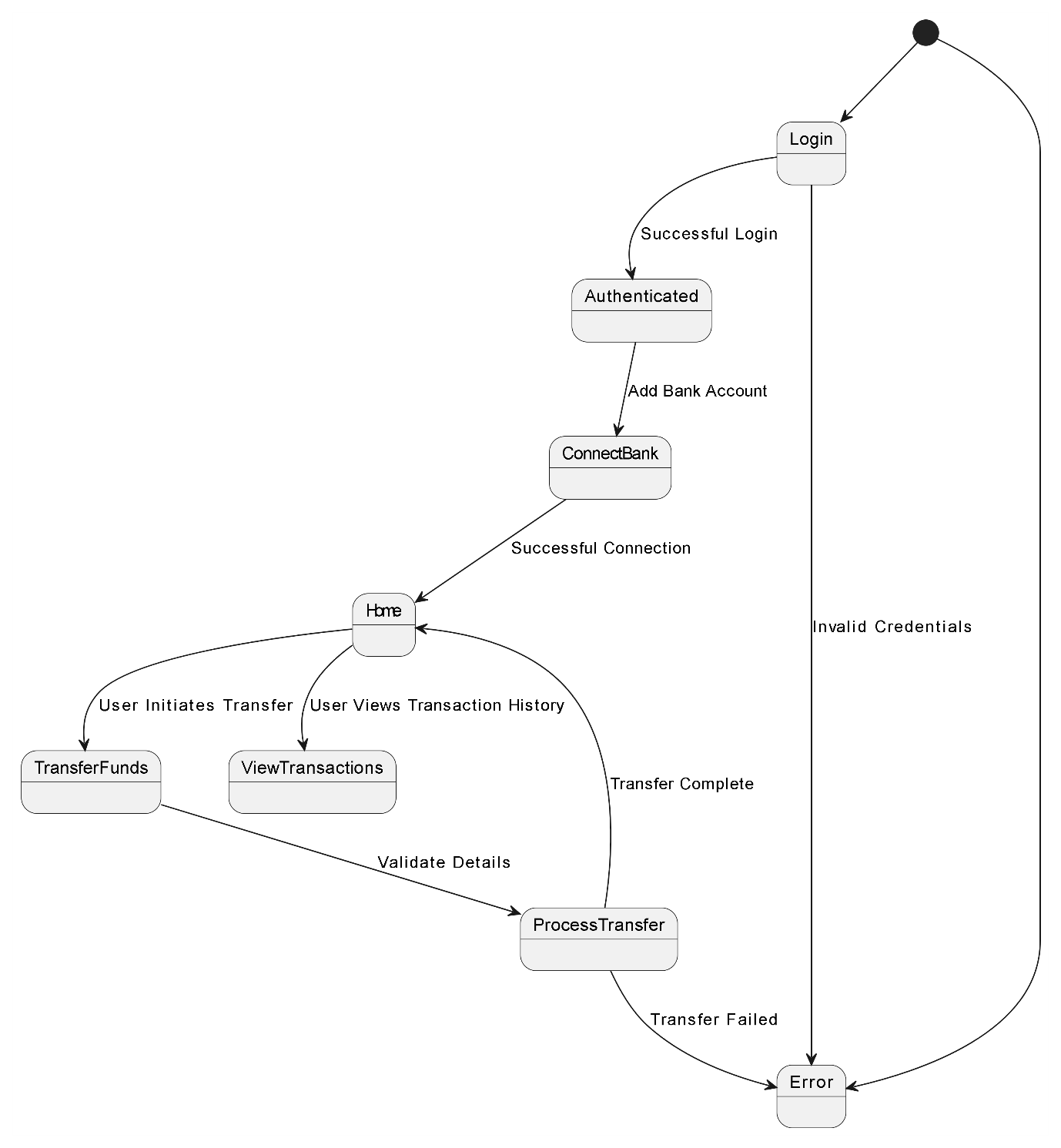


Figure 6: State Diagram

**Chapter 4**

**SYSTEM IMPLEMENTATION**

**4.1Methodology**

The Development of the Finance Management App will have many phases, that ensure a systematic approach building a good and reliable and user-friendly product. The several steps include:

1. Planning and Requirements of the App**:**

* Planning all the things that we need for the creation of the application like the features, tech stacks for the effective financial management.
* Define functional and non-functional requirements, ensuring the app meets the user requirements and expectation for performance, and security.

1. Frontend Development:

* **React.js**: The core library for building the user interface. It will be used for creating interactive components like transaction tables, forms, and charts.
* **Next.js**: A React framework for building the frontend. It supports server-side rendering (SSR), which ensures fast loading times.
* **Tailwind CSS**: A utility-first CSS framework that helps design a responsive and modern UI. It simplifies styling by allowing you to quickly apply CSS classes without writing custom styles.

1. Backend Development:

* **Appwrite**: This is the backend service responsible for handling authentication, database management, and server-side actions. It helps store and manage user accounts, transactions, and bank data securely.[3]
* **Node.js**: The server-side technology used to build APIs and handle backend logic for managing bank account data, transaction history, and secure transfers.
* **Supabase**: A backend-as-a-service that can act as the database to store application data such as user profiles, transaction history, and bank account information. It integrates smoothly with Appwrite and Next.js.
* **Sentry**: A monitoring tool for tracking errors and debugging, helping ensure the app's stability.
* Deployment: Deploy the banking application to vercel (free web hosting application), through GitHub.

**4.2Algorithms**

The project will use many different algorithms for authentication, data encryption, and transaction management. Queue-based algorithms will be used for handling asynchronous tasks like processing transactions in real time, ensuring users receive updates without delays. The use of JWT (JSON Web Tokens) for authentication ensures that user sessions are secure, while OAuth integration with third-party APIs like Plaid allows for secure access to financial data. Data structure that we are going to use for this project are as mentioned below:

**Arrays/Lists:**

* Used to store multiple transactions, account balances, or bank accounts.
* Example: A list of recent transactions or connected bank accounts.

**Dictionaries/Maps:**

* Used to store key-value pairs, such as user details (name, email), account information (account number, balance), or transaction metadata (transaction ID, amount, date).
* Example: {“account\_id": 123, "balance": 5000, "currency": "USD”}

**Queues:**

* Used for handling asynchronous tasks like processing transactions or real-time updates. Queues help manage the order of tasks efficiently.
* Example: A queue for pending financial transactions waiting for server validation.

**Trees:**

* Used in hierarchical structures like categories of expenses or financial goals (e.g., Budget → Savings → Investment).
* Example: A tree structure can categorize transactions (e.g., Food → Groceries → Supermarket).

**Graphs:**

* Used for managing connections between different accounts or relationships between entities like users and banks.
* Example: A graph could represent user relationships with multiple bank accounts and their respective balances.

**Stacks:**

* Used in undo/redo functionalities, especially for user actions such as editing transactions or updating financial goals.
* Example: Stack structure for tracking recent user actions.

**Hash Tables:**

* Used for quick lookups of user data, transaction history, or account details.
* Example: A hash table to quickly access transaction details using transaction IDs.

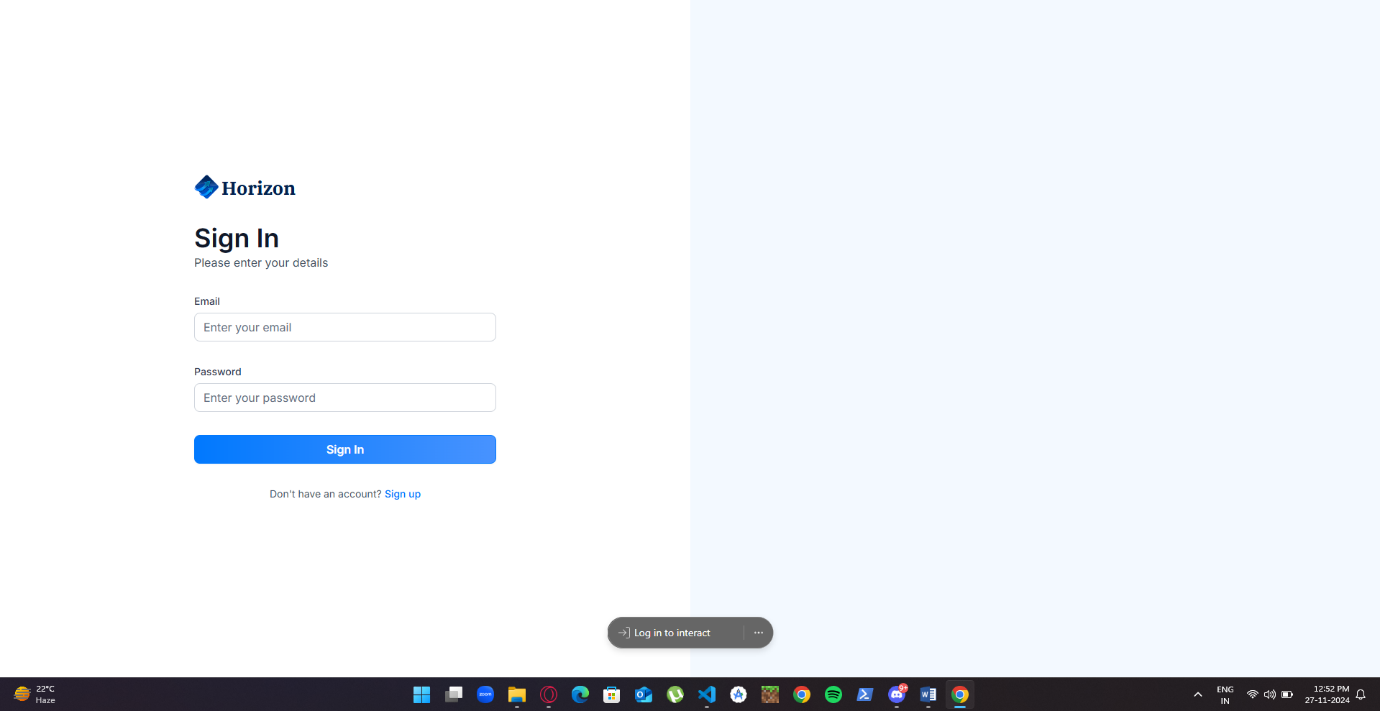
Most of these data structures are abstracted away by libraries and frameworks, so in most cases, we will be using higher-level abstractions like arrays and objective.

-React.js will handle much of the UI state using virtual DOM, arrays, and objects.

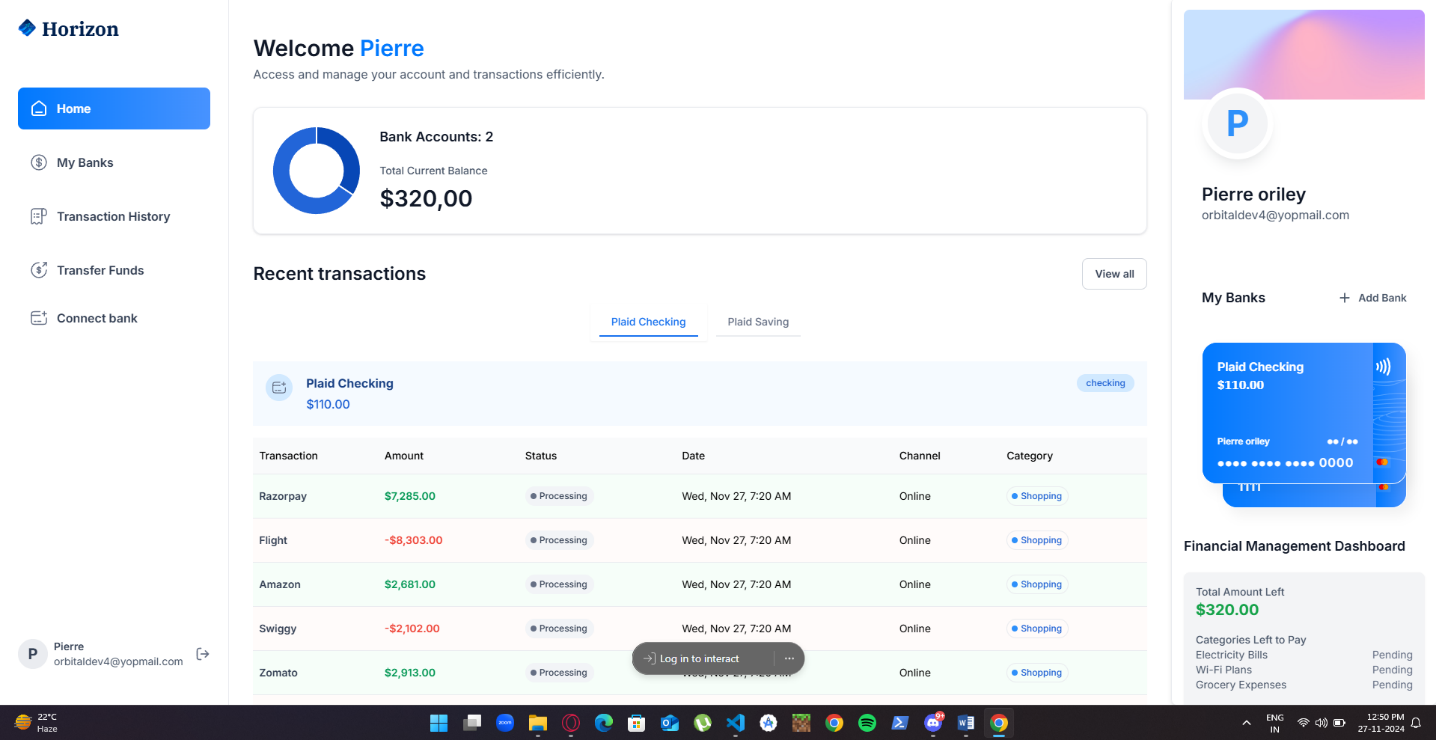
-Node.js will manage server-side data using arrays, objects, and sets for managing users, requests, and session data.

**4.3Sample output**

**(a) Sign-Up page**

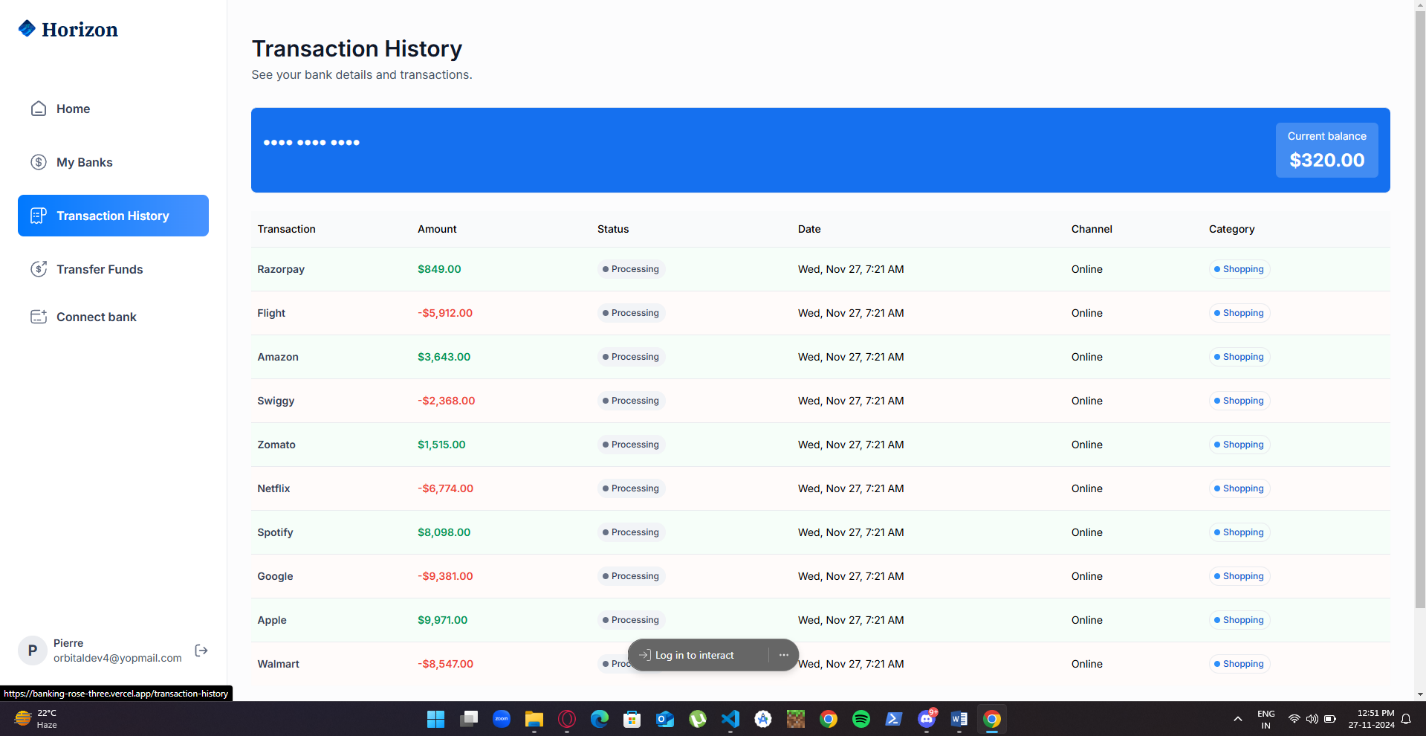
** Figure 7**

**(b) Homepage**



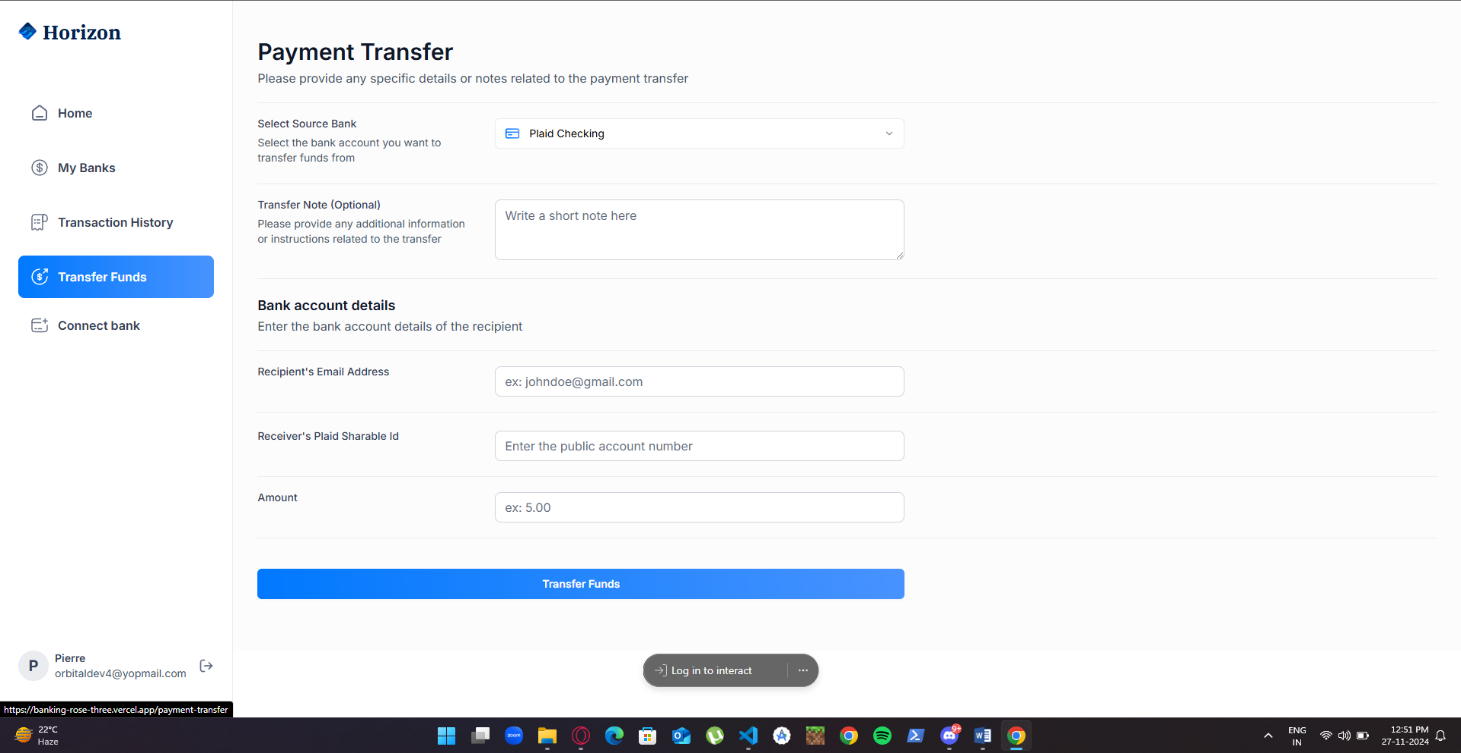
**Figure 8**

**(c) Transaction-History page**



**Figure 9**

**(d) Payment-Transfer page**



**Figure 10**

**Chapter 5**

**LIMITATIONS AND FUTURE ENHANCEMENTS**

**LIMITATIONS**

1, **Dependency on External APIs**:  
The system heavily relies on third-party APIs like Plaid and Dwolla for critical functionalities. Any downtime or service interruptions from these providers can affect the application’s usability.

2. **Scalability Challenges**:  
While the application is designed to handle up to 10,000 concurrent users, further scaling might require significant architectural adjustments.

3. **Limited Target Audience**:  
Currently, the application focuses on individual users and does not cater to businesses or advanced financial use cases like investment tracking.

4. **Mobile App Unavailability**:  
The system is accessible via a responsive web application but lacks a dedicated mobile app for enhanced user experience on smartphones.

5. **Data Privacy Concerns**:  
Despite implementing robust security measures, the application handles sensitive financial data, which might raise privacy concerns for some users.

**FUTURE ENHANCEMENTS**

1. **AI-Powered Financial Insights**:  
   Incorporate machine learning algorithms to provide personalized budgeting tips, saving strategies, and financial goal tracking based on user data. [4]
2. **Business Account Integration**:  
   Extend the application to support small businesses, offering features like payroll management, invoice tracking, and tax calculation.
3. **Mobile Application Development**:  
   Develop a dedicated mobile app for Android and iOS platforms to enhance accessibility and provide a better user experience.

**Chapter 6**

**CONCLUSIONS**

This online banking platform effectively combines modern technologies like Next.js, Tailwind CSS, and Plaid to deliver a secure, user-friendly financial management solution. With real-time transaction updates, secure transfers, and multi-bank integration, it provides a seamless and efficient banking experience across devices. The project demonstrates how new technologies can be used to create a user-friendly reliable application for managing finances.

**Chapter 7**

**REFERENCES**

1. Shahid, S., Islam, J. U., Malik, S., & Hasan, U. (2022). Examining consumer experience in using m-banking apps: A study of its antecedents and outcomes. Journal of Retailing and Consumer Services, 65, 102870.
2. Zhu W, Proksch S, German D, Godfrey M, Li LMcIntosh S(2024).An empirical assessment of security risks of global Android banking apps.
3. Cleidson R. B. de Souza, David Redmiles, Li-Te Cheng, David Millen, and John Patterson. 2004.
4. Gomber, Peter, Jascha-Alexander Koch, and Michael Siering. "Digital Finance and FinTech: current research and future research directions." *Journal of Business Economics* 87 (2017): 537-580.