



Programming Project
Project Summary
COMPX576

STUDENT NAME: SUMEET
STUDENT ID: 1652187

Table of Contents

1. Introduction.....	3
2. Architecture and Security.....	3
2.1. Technologies Used.....	3
2.2. Security Implementation	6
3. Implemented Functionalities and Application Workflow.....	6
4. Database Architecture	16
5. Challenges and Resolutions.....	17
6. Future Improvements.....	18
7. Conclusion.....	19
References.....	20

1. Introduction

FinTrack is a web-based account ledger system that is designed to simplify the management of financial transactions. It provides users with a platform to track and organise their financial activities by maintaining accurate records. Both businesses and individuals can benefit by utilising its intuitive and user-friendly interface of the application to manage their financial activities efficiently and effectively.

The core objective behind developing this web application is to assist users in keeping their detailed records of all financial transactions up to date so that it can give them the power to make better financial decisions through interactive reports. Moreover, this application is based on secure data management, which ensures the safe storage of all financial transactions by preventing unauthorised access to the application.

2. Architecture and Security

FinTrack uses modern technologies to ensure efficient financial management and secure data handling. Below is a summary of the key technologies and security features used in the application.

2.1 Technologies Used

The FinTrack system is built using the following technologies:

- **Frontend (React):**

The frontend of the application is developed using React, a JavaScript library that enables the creation of dynamic and interactive user interfaces. It handles the state management of the application and ensures smooth navigation between different views, such as account creation and transaction history. Axios is used for making HTTP requests to the backend, facilitating communication between the frontend and server for data retrieval and submission (GeeksforGeeks, n.d.). React's component-based architecture also allows users to customise logic and appearance more easily across multiple pages (React, n.d.).

- **Backend (FastAPI):**

The backend is built using FastAPI, which is a Python-based web framework known for its high performance because it supports asynchronous programming (FastAPI, n.d.). FastAPI is also based on the two key components:

- **Starlette:** It is a lightweight ASGI (Asynchronous Server Gateway Interface) framework that allows users to create async web services in Python. It also provides with the efficient handling of web requests, routing, and sessions, which is a very important feature used in our application during transaction management (Starlette, n.d.).

- **Pydantic:** It is a validation library that handles data validation and serialisation in Python by making use of type hints so that if any user enters any kind of input that is not of the correct type, it will throw a respective error if it is not defined as per schema (Pydantic, n.d.).
- **Database (MySQL):**

For the development of the application, the database used is MySQL for storing all user, account, and transaction-related information. The database used ensures the integrity of financial data in a structured and relational way by creating tables for users, accounts, transactions, and balances and preserving the ACID (Atomicity, Consistency, Isolation and Durability) properties in linking the multiple tables, which is required during the processing part. Moreover, each transaction updates the payer and payee account balances in real time, ensuring that financial activities are recorded accurately (Oracle Corporation, n.d.).
- **API Testing (Postman):**

During the development of FinTrack, Postman was used extensively for API testing. Postman allows developers to test API endpoints, send requests, and inspect responses in a user-friendly interface. It was essential to make sure the backend API functions worked as expected and that the communication between the frontend and backend systems flowed smoothly (Postman, n.d.).
- **API Documentation (Swagger UI):**

Swagger UI was used for API documentation in the application that allows clear and interactive documentation of all available API endpoints in the project. This tool allows developers and users to easily explore the API right in their browser, see the request and response formats, and interact with the API directly. This helps streamline the development process and provides a clearer understanding of how the backend services work. (SmartBear Software, n.d.).

Figure 1 represents all the technologies used in building the application.

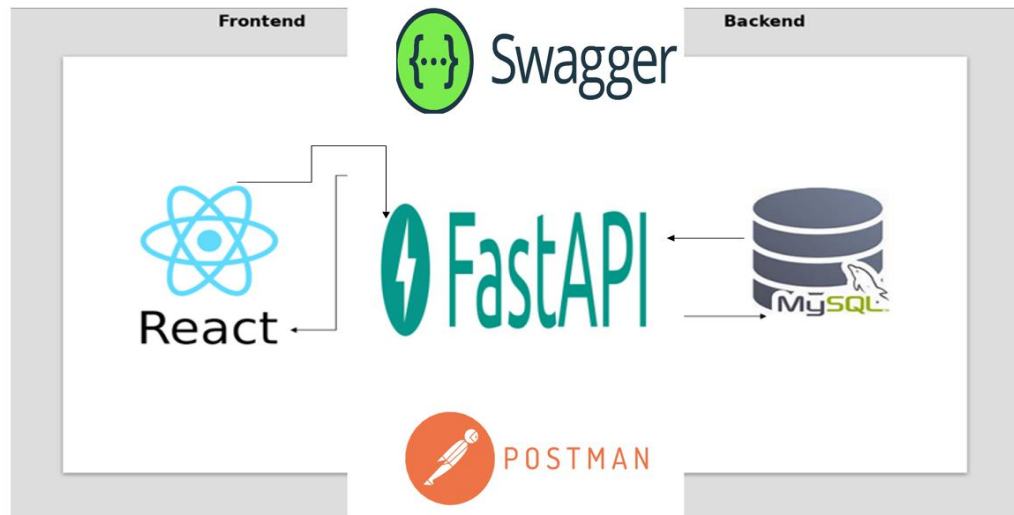


Figure 1: Technologies Diagram

Credits - https://tactless7.github.io/cv/img/icons/react_logo_2.png
<https://imgur.com/p0NuJin>
<https://static-00.iconduck.com/assets.00/database-mysql-icon-1954x2048-08uox8qu.png>
<https://www.scottbrady91.com/img/logos/swagger-banner.png>
<https://assets.apidog.com/blog/2023/04/postman-logo.png>

Figure 2 shows the architecture of FinTrack, where the React frontend handles routing, components, and services. Axios manages HTTP requests between the frontend and FastAPI backend that handles core routes, middleware, and connects to the database using a DB adapter. SQLAlchemy here in the backend, serving as the DB adapter, abstracts the database(MySQL) operations, allowing easy interaction with the database through Python models. The system communicates through a REST interface for smooth data exchange.

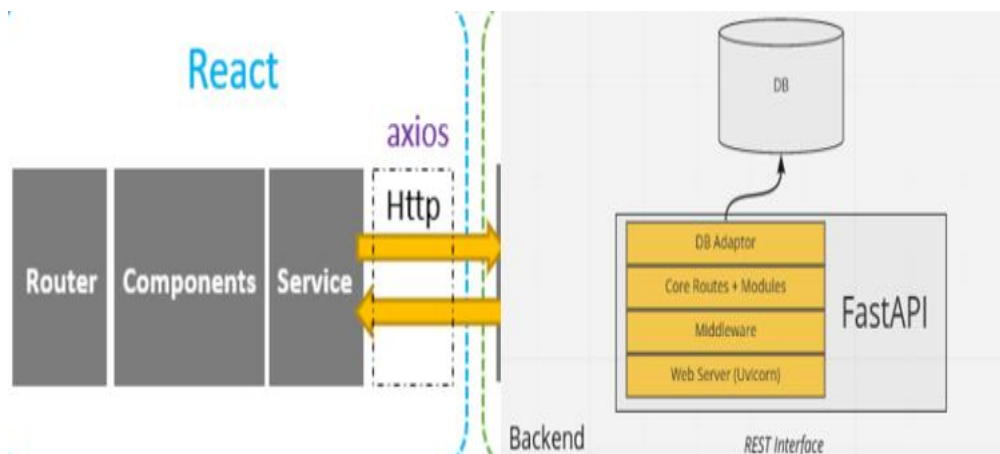


Figure 2: Architecture Diagram

2.2 Security Implementation

To ensure data security and user protection, FinTrack incorporates the following security features:


- **JWT-Based Authentication:**
JSON Web Tokens (JWT) are used to handle user authentication. Upon successful login, the user is issued a JWT, which is included in every subsequent request to authenticate the user. This method ensures that sensitive operations are performed only by authenticated users, enhancing security (JWT.io, n.d.).
- **Middleware for Secure Request Handling:**
Authentication and Request middleware in the backend are used which is responsible for intercepting incoming requests, verifying the validity of JWT tokens, and routing them to the appropriate services. If the token is invalid or expired, access is denied. This step prevents unauthorised access to user data and secures communication between the frontend and backend.
- **Email Validation and Password Hashing (bcrypt):**
When users register in the application, their email addresses are validated using regex patterns to ensure that they meet security and format standards, avoiding common vulnerabilities. Passwords are hashed using bcrypt, a robust hashing algorithm that adds a salt to each password, ensuring that even if the database is compromised, passwords remain secure (FastAPI Tutorial, n.d.).
- **Password Storage and Authentication:**
In the database, all user passwords are stored as bcrypt hashes. When a user logs in, the system compares the hash of the entered password with the stored hash to authenticate the user. This method ensures that plain-text passwords are never stored in the system, mitigating the risk of password exposure. Moreover, their login time stamp is also being captured in the application (FastAPI Tutorial, n.d.).

3. Implemented Functionalities and Application Workflow

FinTrack is a web-based application system that offers multiple key functionalities that allow users to manage their expenses efficiently. The following sections explain each feature and its role in the system:

- **User Registration and Login:** Firstly, the user will be allowed to first register on the platform using an e-mail address and password, where email ID and password will be validated against the regex check that is developed in the application. After being registered successfully, it will be redirected to the login page, where a user must enter his e-mail address and password provided during the registration process. After successful login, their


secure session will be maintained using the concept of JWT token, and it will be redirected to the home page where users can perform the further activities. Figures 3.1 and 3.2 shows the respective Register and Login forms.



Register

Already have an account? [Login](#)

Figure 3.1: Register View of FinTrack



Log in

Don't have an account? [Register](#)

Figure 3.2: Login View of FinTrack

- **Home View:** After successful login, the user will be taken to the home page of the application (as shown in Figure 4), from where navigation to other functionalities of the application will be possible. The homepage would provide the user with the following options to interact with the system and manage financial data seamlessly:

- **Create a financial account**
- **Initiate/Create a financial transaction**
- **Fetch account details**
- **Fetch transaction statements**

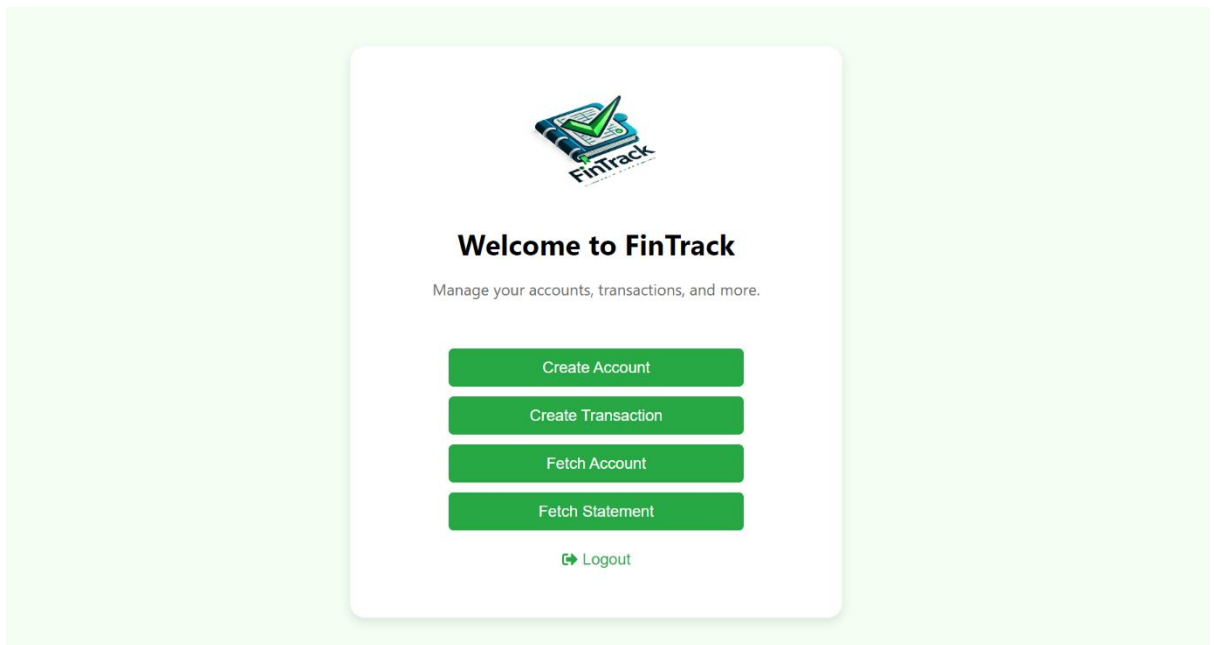



Figure 4: Home View of FinTrack

- **Create Account:** This view allows a new user to create a new financial account (according to Figure 5.1), or it also allows users to create multiple accounts with different account names, enabling them to make transactions in different currencies by entering details such as:
 - **Account Name:** The name of the account.
 - **Purpose of the account:** The reason for opening the account (e.g., funds transfer) which user can enter.
 - **Currency:** User can also select from multiple available currencies like INR, NZD, or USD and so on from the drop-down list available.

Once this account is created successfully, it redirects the user to the home page. In the database, the “account” table stores details regarding an account, with a user ID mapped to every account. Moreover, an email (Figure 5.2) will be sent to the user’s registered email address with all the account information.

Note: This is the entry point of the application, which means that the user won’t be able to access other services like creating transactions, fetching accounts, or fetching statements unless they create their first account supporting any currency for transactions after registration.



Create Account

☐ I consent to the creation of this account.

[Logout](#)

Figure 5.1: Create Account View of FinTrack

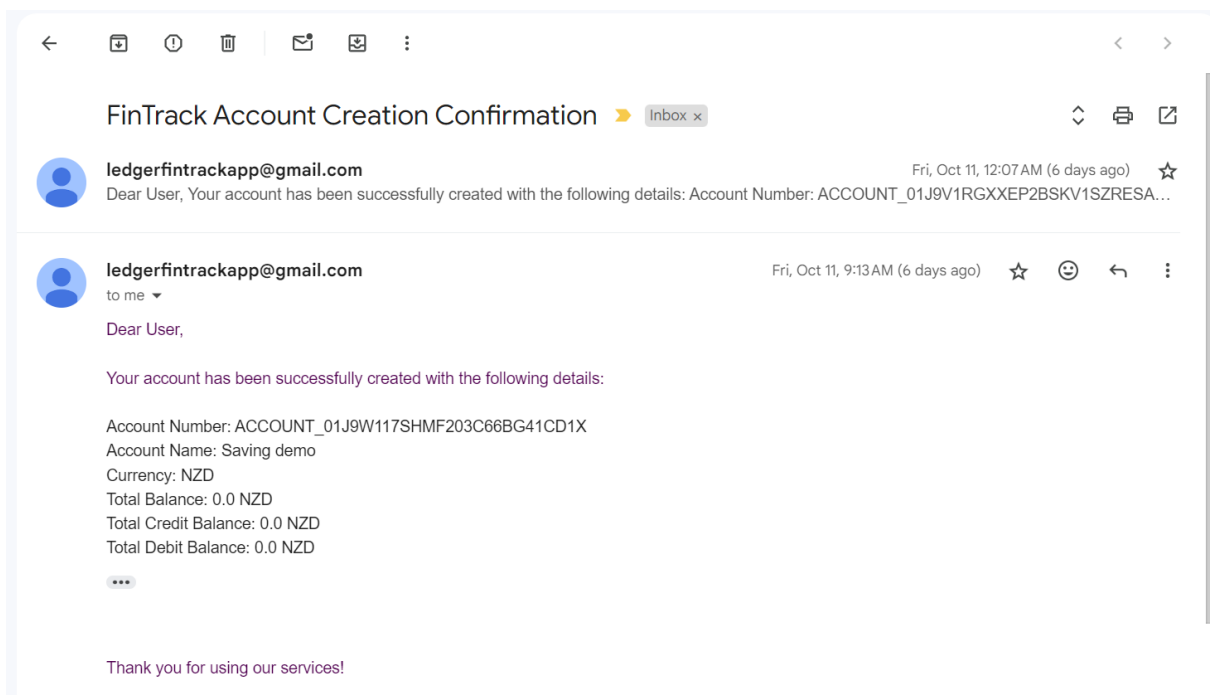


Figure 5.2: Account Opening Email Notification

- **Create Transaction:** Users can create a financial transaction (as shown in Figure 6.1) by entering the following details:
 - **Payer account:** The account from which money will be deducted.
 - **Payee account:** The account that will receive the money.
 - **Amount:** The transaction amount that needed to be transferred or received.

- **Purpose of the transaction:** User can specify the reason for the transaction.

Once the transaction is created, it is recorded in the “transaction” table. The balances for both the payer and payee accounts are updated in real time to reflect the debit and credit transactions. Post successful transactions, email confirmation (Figure 6.2) will be sent to both payer’s as well as payee’s registered email address stating the debit and credit details.

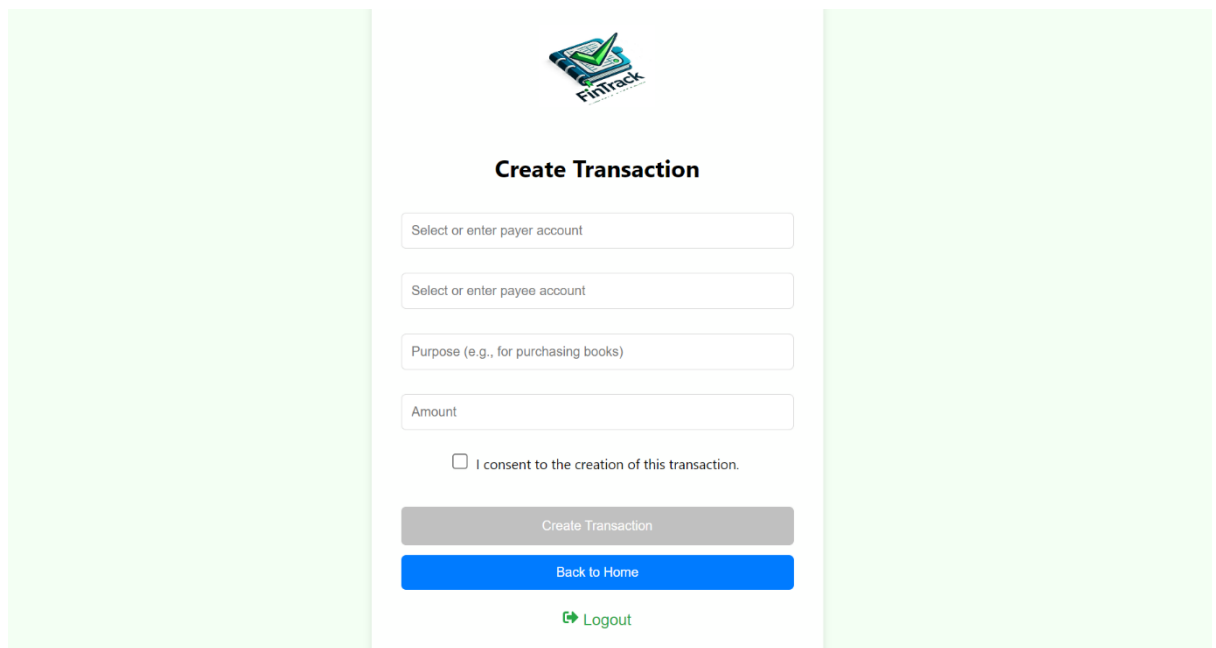
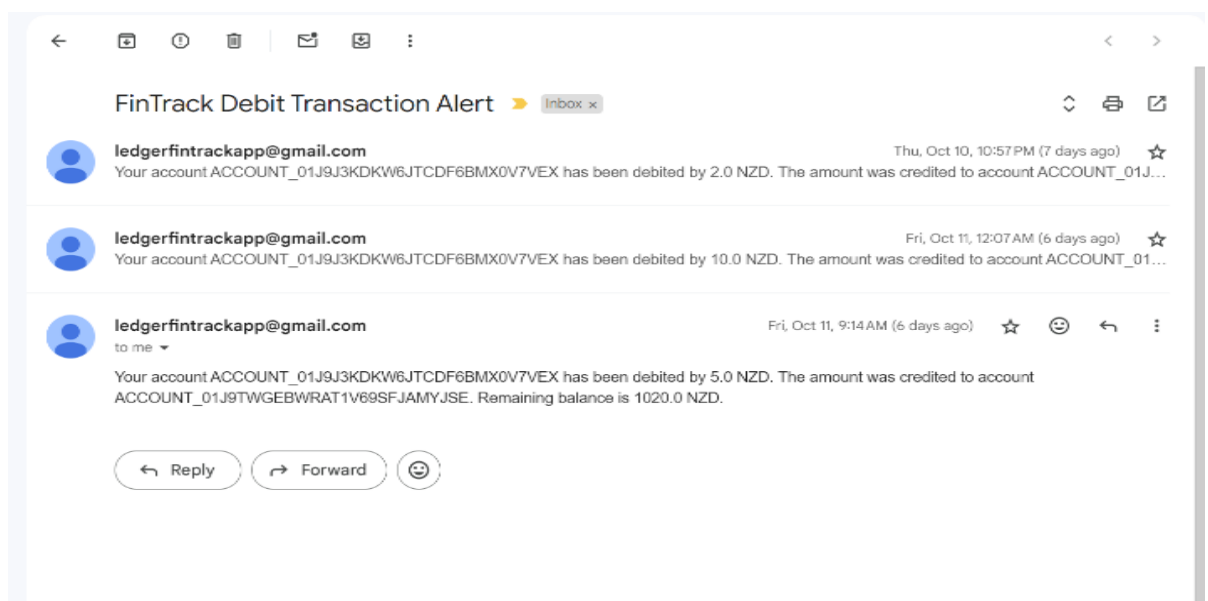


Figure 6.1: Create Transaction View of FinTrack



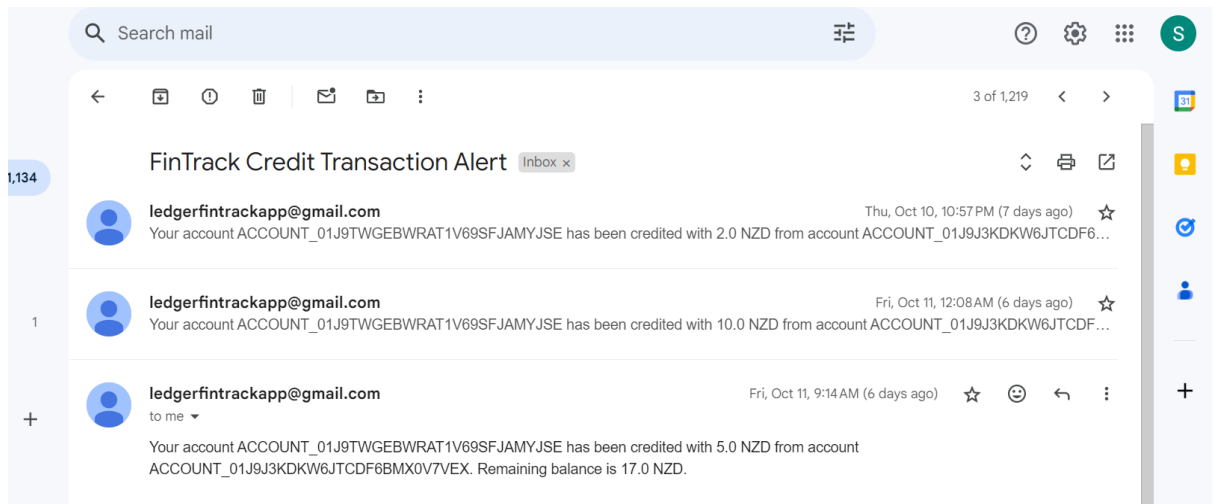


Figure 6.2: Debit/Credit Transactions Email Notification

- **Fetch Account:** Using this view, users can view the details of their accounts by selecting an account from a dropdown list. The system fetches and displays information such as:
 - Account name
 - Account URN (Unique Reference Number)
 - Currency in use (which is used at the time of account creation)
 - Total balance remaining, along with the credit and debit balances.

This functionality provides users with an overview of their financial position (as illustrated in Figures 7.1 and 7.2).

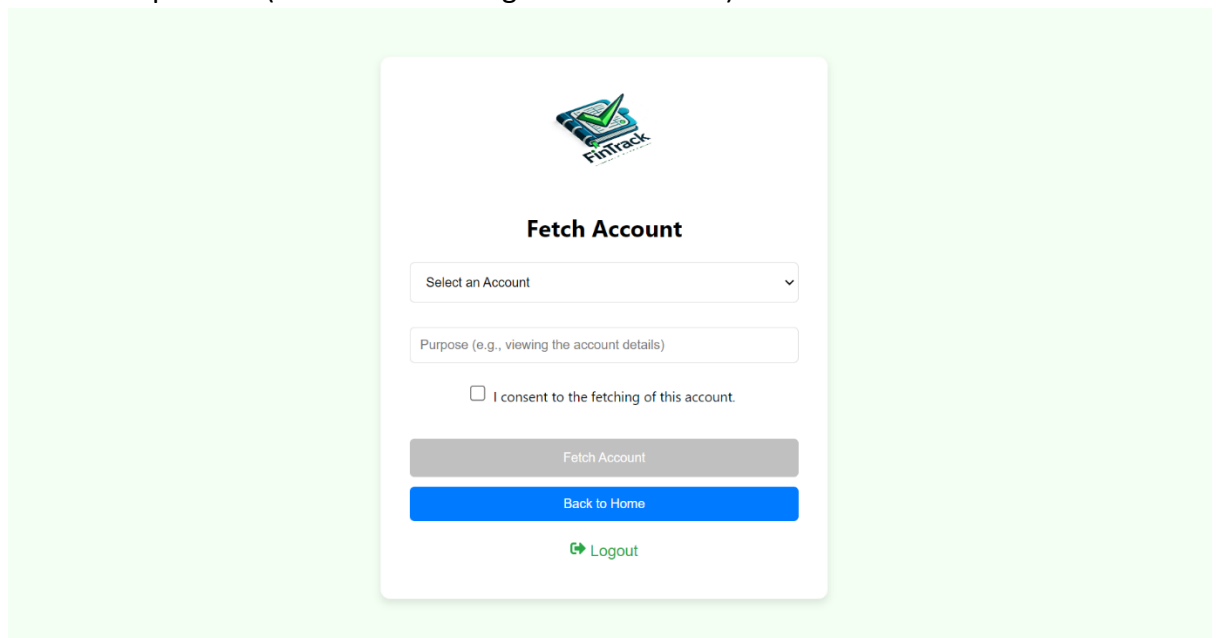


Figure 7.1: Fetch Account View of FinTrack

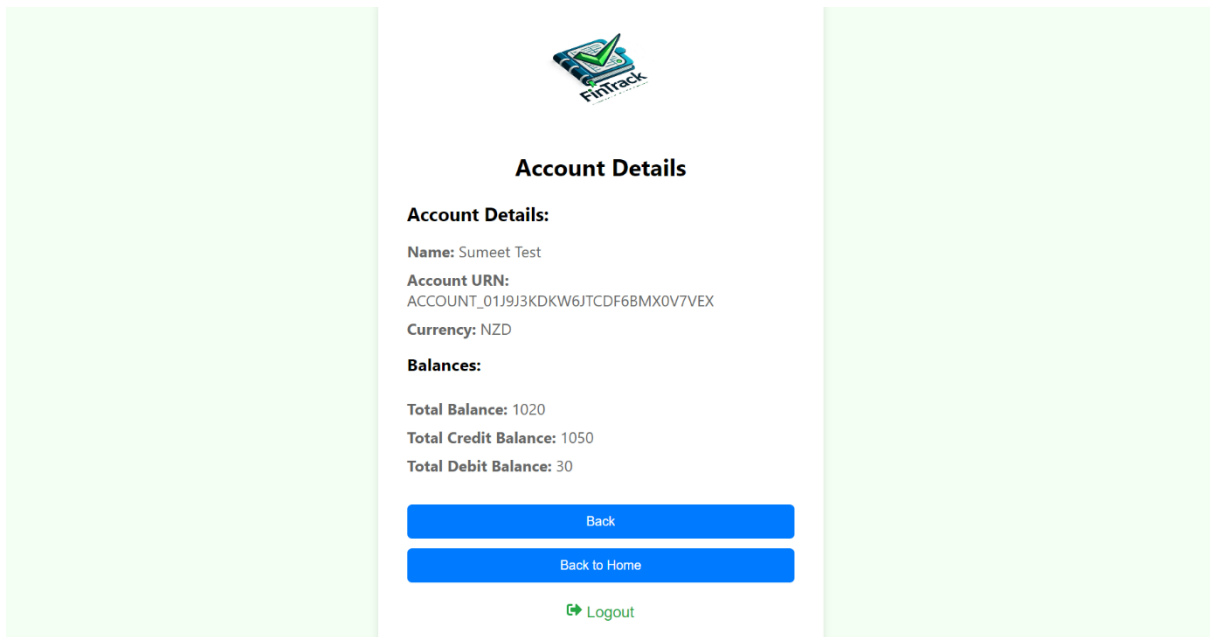



Figure 7.2: Account Details View of FinTrack

- **Fetch Statement:** In this view, users can view their past transactions of any account by filtering them based on a date range, i.e., From Date and To Date. Post correctly entering all the details by the user, the system retrieves detailed information for each transaction, including:
 - Transaction URN
 - Payer and payee accounts
 - Transaction amount
 - Purpose
 - Transaction type (credit or debit)
 - Transaction Timestamp


This feature helps users track their financial activities over a specific period and stay informed about their transactions (as shown in Figures 8.1 and 8.2).



Fetch Statement

☐ I consent to the fetching of this statement.

Figure 8.1: Fetch Statement View of FinTrack



Statement Details

Transaction URN	Payer URN	Payer Name	Payee URN	Payee Name	Amount	Purpose	Type	Timestamp
01/9/W13HGPD/PS2C/1A81M1FV	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	ACCOUNT_01/9TWGEBWRAT1V69SFJAMVSE	Saving Test	5	purchase pen	DEBIT	2024-10-11 09:14:44
01/9/V1T6M2S9GVVXX1GJ887E0	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	ACCOUNT_01/9TWGEBWRAT1V69SFJAMVSE	Saving Test	10	coffee	DEBIT	2024-10-11 00:07:52
01/9/V15UBMRP886GY92KPTPYM	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	1000	loan	CREDIT	2024-10-10 23:56:45
01/9TXRFD80CTD5GEWY14K7JMW	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	ACCOUNT_01/9TWGEBWRAT1V69SFJAMVSE	Saving Test	2	pen	DEBIT	2024-10-10 22:57:08
01/9/PPV7YF5GZQGMKHTPN0P62	ACCOUNT_01/9/3J3T31Z3YKMYT34H4NVEAE	Sumeet	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	5	books	CREDIT	2024-10-07 18:19:53
01/9/8TCV4CFHEQQCCYR5693C4	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test			10	test	DEBIT	2024-10-07 14:17:08
01/9/8JJCQ9AD46E6KSZD01848			ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	25	books	CREDIT	2024-10-07 14:16:08
01/9/725WTRH7EDTHS7T7BEHJ	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	ACCOUNT_01/9/3J3T31Z3YKMYT34H4NVEAE	Sumeet	1	item	DEBIT	2024-10-07 13:46:47
01/9/6X6P556825B997MFKY35	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	ACCOUNT_01/9/3J3T31Z3YKMYT34H4NVEAE	Sumeet	2	pen	DEBIT	2024-10-07 13:43:43
01/9/6ND6PWRQMAHPT29FR6G	ACCOUNT_01/9/3J3T31Z3YKMYT34H4NVEAE	Sumeet	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	5	books	CREDIT	2024-10-07 13:35:14
01/9/610ER159XB21WHAG0N4JN	ACCOUNT_01/9/3J3T31Z3YKMYT34H4NVEAE	Sumeet	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	5	for purchasing book	CREDIT	2024-10-07 13:28:19
01/9/3Q0G2HMS6RVYDSAG519J	ACCOUNT_01/9/3J3T31Z3YKMYT34H4NVEAE	Sumeet	ACCOUNT_01/9/3KDKW6/TCDF6BMXOV7VEX	Sumeet Test	10	For Purchasing pen	CREDIT	2024-10-07 12:47:55
Total Balance - Filtered							NZD 1020.00	
Overall Total Balance							NZD 1020.00	

Figure 8.2: Statement Details View of FinTrack

- Transaction Report:** In this view, users can analyse their past transactions with the support of interactive charts as per Figures 9.1 and 9.2. They can also filter transactions by selecting a date range (From Date and To Date) and clicking the "Filter" button. The system then displays two types of charts for better insights as mentioned below:
 - Credit and Debit Bar Chart:** This bar chart shows the credit and debit amounts for each transaction date. Green bars represent credit transactions, and red bars represent debit transactions which makes it easy to distinguish between the two.

- **Balance Line Chart:** The line chart displays how the user's account balance changes over the selected period. It provides a visual representation of the balance after each transaction the user make with the blue line indicating the balance trend over time.

At the bottom of the graphs, the application provides a summary with the Total Credit and Total Debit amounts for the selected period. Moreover, Chart.js, which is a free, open-source JavaScript library, is being used for data visualisation (Chart.js, n.d.).

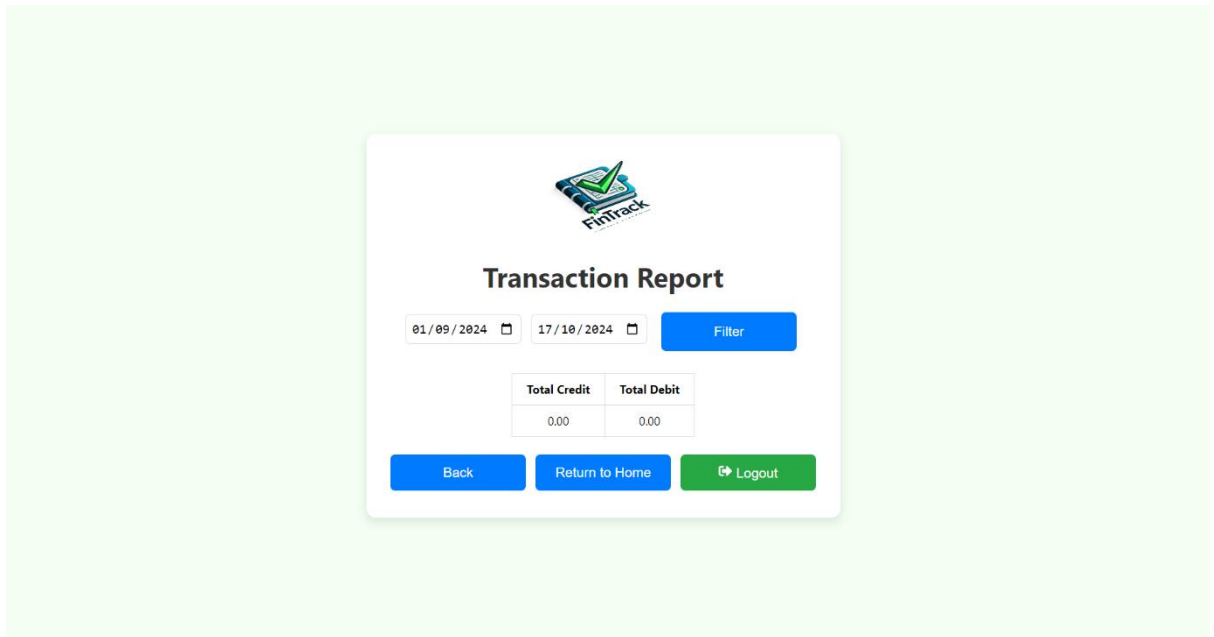


Figure 9.1: Transaction Report View of FinTrack - For User Entry

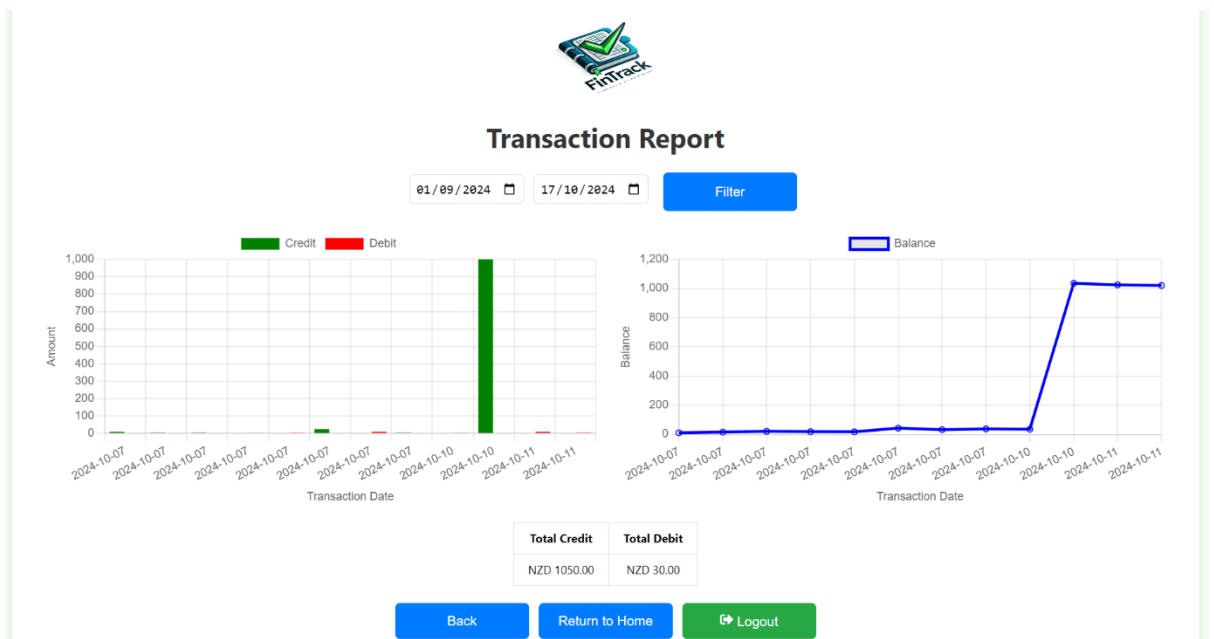


Figure 9.2: Transaction Report View of FinTrack

- **User Logout:** When the user decides to log out, the system will securely terminate their session. The application will also allow the capture the last login timestamp details of the individual to prevent any unauthorised access to sensitive data after logout.
- **Swagger Documentation:** Moreover, the Swagger Documentation (Figure 10) has been done for the developers which provides the flexibility to explore the API in a browser, view request and response formats, and interact with the API directly (SmartBear Software, n.d.).

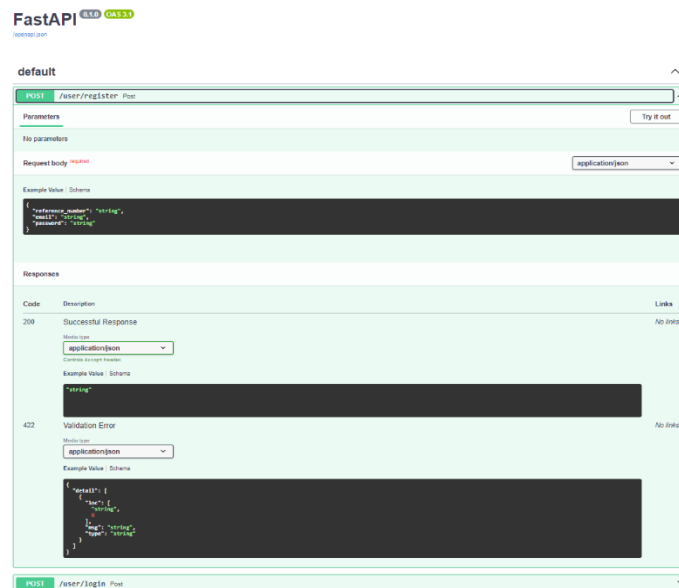


Figure 10: Swagger UI Documentation

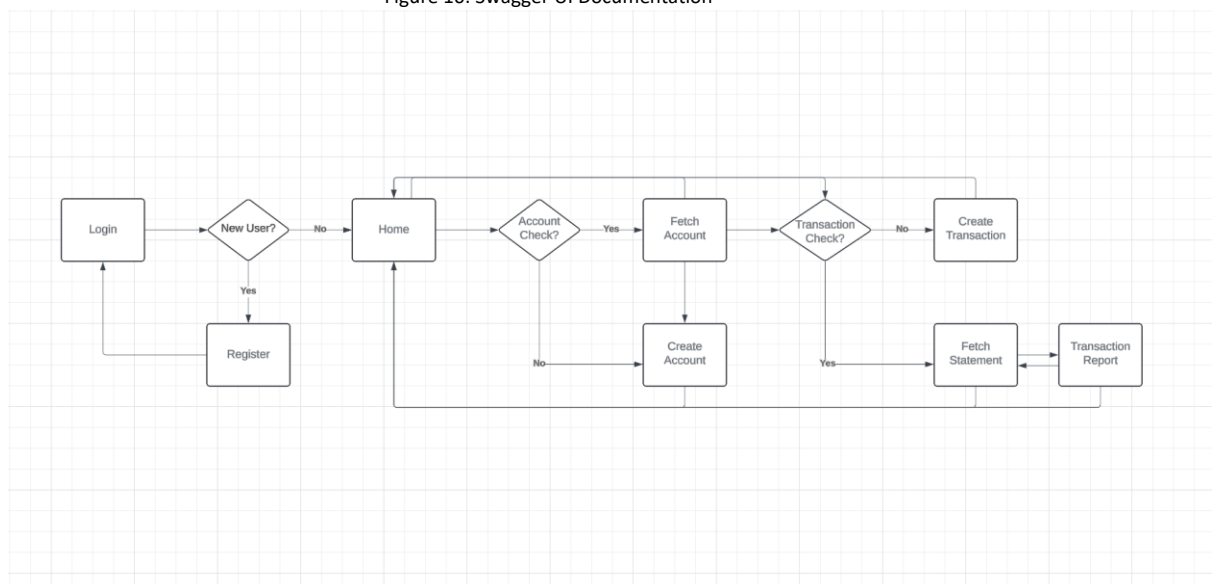


Figure 11: Systematic Flow of the Application

Therefore, the FinTrack application flow (as shown above in Figure 11) makes it easy for users to manage their financial accounts. It guides them through each step, ensuring a smooth and secure experience. Here's the basic flow of the application:

- **Login or Register:** Users start by logging in or signing up as a new user.
- **Create an Account:** After logging in, users can create a financial account to manage.
- **Make Transactions:** Users can add, or view transactions in their accounts.
- **View Transaction Reports:** The system generates detailed transaction reports, helping users track their financial activities.

This step-by-step process ensures that users can easily manage their finances and stay informed about their transactions.

4. Database Architecture

The database design illustrated below in Figure 12 is used to manage user accounts, financial transactions, and account balances for the FinTrack application. It is implemented using a relational database model, which makes MySQL an ideal choice due to its ability to handle structured data and implement relationships between multiple tables efficiently using the concept of primary and foreign keys. Following key tables used in the development of the application:

- **User Table:** Stores information about users, including email, password, and login status. It provides a means for secure authentication, hence capturing the timestamp of every login made by the users in the table.
- **Account Table:** Holds information about an account, such as the name of the account, the currency in use, and the current balance held by an individual. It is associated with the user table by one-to-many mapping, meaning one user can have many accounts with different currencies, and each account should have an owner mapped to it.
- **Balances Table:** Tracks the financial status of each account by recording total balance remaining till date, credit, and debit information.
- **Transaction Table:** It helps in logging all financial transactions between accounts, storing details like payer, payee, transaction amount, and timestamps for the transaction.
- **Currency_1k Table:** Stores currency details used in various accounts, providing flexibility for users to manage accounts in different currencies. Moreover, the user can initiate the transaction to another account only if that account supports those same currencies otherwise, it will throw an error.

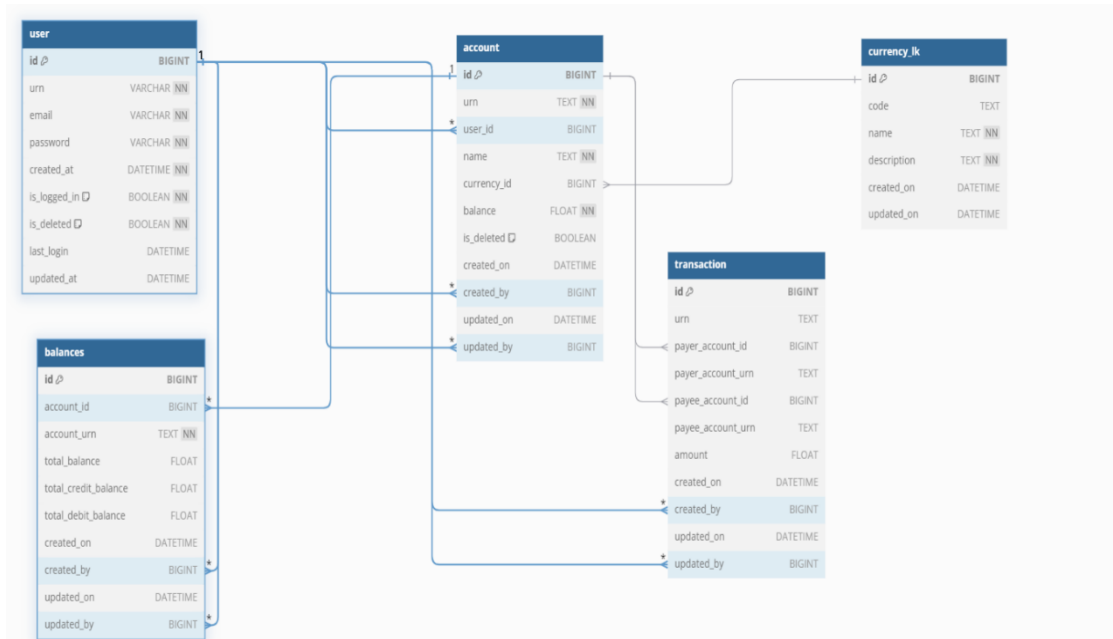


Figure 12: Database Diagram

5. Challenges and Resolutions

The development of FinTrack presented several challenges, which are as follows:

- Learning New Technologies:** Implementing FastAPI and React was a new experience for me. Understanding the concepts of how asynchronous programming works in FastAPI and how React manages the frontend state required a learning curve. However, these technologies provided the necessary performance and scalability for the system that I have developed.
- Database Architecture:** Another challenge that I faced during the development of the application was designing the database so that the transactions were accurately recorded. The main concept of this application is to handle each transaction, which involved a debit from the payer's account and a credit to the payee's account leading to a double entry method, meaning every transaction affects at least two accounts. The database structure ensured that the correct amounts were debited and credited in real time. This was achieved through well-defined relations between the User, Account, Transaction, and Balances tables.
- Integration Issues (CORS Handling):** Initially, there were many CORS (cross-origin resource sharing) issues I faced when the frontend tried to communicate with the backend APIs, resulting in the server error. These issues were resolved by configuring the backend to bypass authentication for OPTIONS HTTP method in authentication middleware so that smooth communication between the frontend and backend can be established (Murgesh, 2023).
- Secure Data Handling:** Since the FinTrack application deals with sensitive financial data, security for the application was a top priority. JWT token-based authentication and middleware were implemented to ensure secure access to user accounts (JWT.io, n.d.). Furthermore, the Users table also tracks the login

status of each user to ensure proper session management and prevent unauthorised access after logout. Initially, I faced some issues in setting up the flow of requests and responses handling, but it got resolved by referring the documentation.

6. Future Improvements

As FinTrack continues to evolve, several enhancements are planned to improve its functionality, scalability, and user experience. These future improvements will ensure the system meets the growing needs of its users and offers advanced features for financial management.

- **Hosting on Microsoft Azure:** FinTrack will be hosted on Microsoft Azure to ensure scalability and availability. Currently, the application is running and fully functional on a local system to avoid high expenditure. Azure services like Azure Virtual Machines (VM) for hosting, Azure SQL Database for managing relational data, and Azure Blob Storage for secure file storage will be utilised for application deployment and data management. This will enhance the application's ability to handle more users and provide greater reliability (Microsoft, n.d. -b).
- **Google Sign-in and Multi-Factor Authentication (MFA):** The system will also integrate Google login for easier user access. Apart from tracking the user access currently, MFA will also be implemented to provide additional security for user accounts.
- **Power BI Integration:** Power BI will be integrated into the existing application, which provides users with interactive visualisations of their financial data. Users will also be able to view more different types of charts and graphs related to their expenses and savings, helping them understand their financial habits better. This step will involve the purchase of a Power BI Pro license (Microsoft, n.d. -a).
- **Q&A Visual Implementation using Power BI:** Power BI's Q&A feature will be implemented to allow users to ask questions about their financial data and receive answers in the form of charts or visual insights (Microsoft, n.d. -a).
- **Developed mobile version of FinTrack:** A mobile version of the FinTrack application will be developed to allow users to manage their financial activities on the go, providing more convenience and accessibility.

Moreover, below is the implementation of the Power BI and Q&A Visual (as shown in Figures 13.1 and 13.2) which will be integrated in near future.

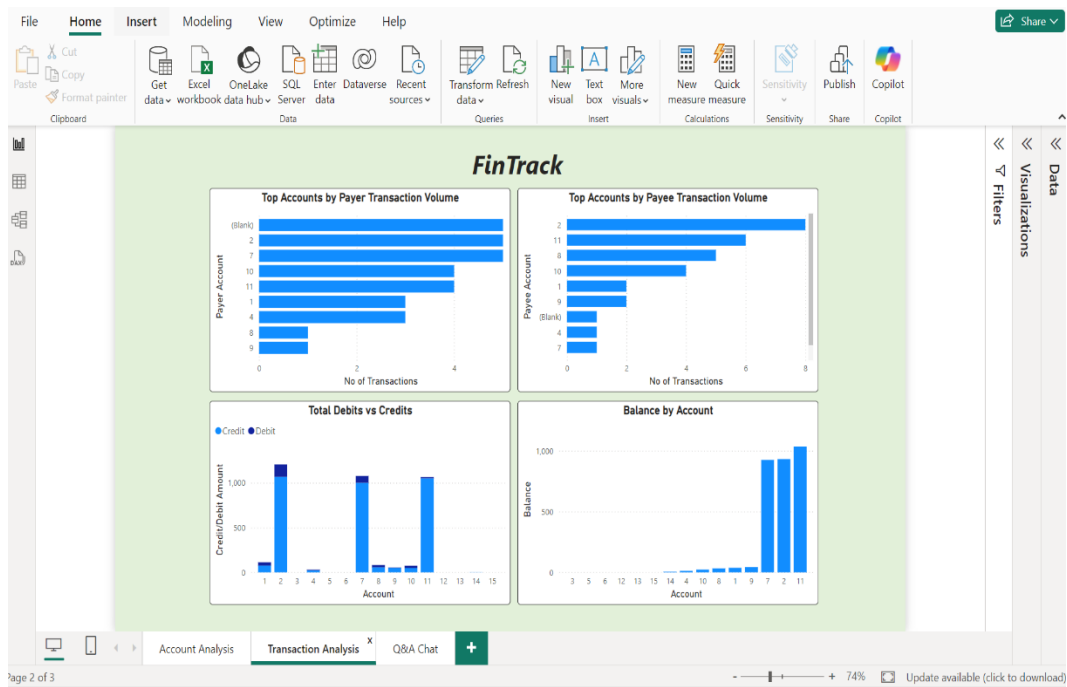


Figure 13.1: Transaction View

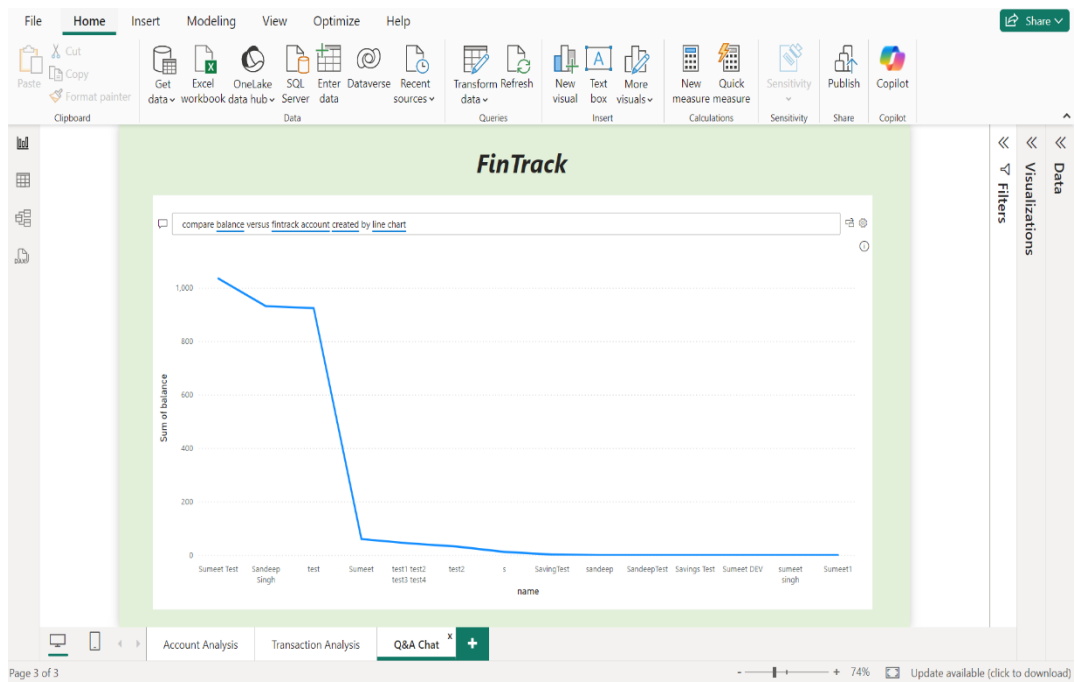


Figure 13.2: Q&A Chat View

7. Conclusion

To sum up, FinTrack is a comprehensive account ledger system designed to help users manage their finances, track transactions, and keep records securely and efficiently. Using technologies like React, FastAPI, and MySQL, the system offers strong account management, real-time transaction handling, and clear financial reporting. Future upgrades, such as hosting on Microsoft Azure, integrating Google login, multi-factor

authentication (MFA), and Power BI, will enhance its features and security. These improvements will make FinTrack a complete, scalable, and secure solution for managing personal and business finances and records.

References

- React. (n.d.). *React documentation: Learn React*. <https://react.dev/learn>
- FastAPI. (n.d.). *FastAPI documentation*. <https://fastapi.tiangolo.com/>
- Oracle Corporation. (n.d.). *MySQL documentation*. <https://dev.mysql.com/doc/>
- Postman. (n.d.). *Postman API platform*. <https://www.postman.com/>
- SmartBear Software. (n.d.). *Swagger: API documentation*. <https://swagger.io/>
- Pydantic. (n.d.). *Pydantic API documentation*. <https://docs.pydantic.dev/latest/>
- Starlette. (n.d.). *Starlette documentation*. <https://www.starlette.io/>
- Microsoft. (n.d. -a). *Power BI Desktop*. <https://www.microsoft.com/en-us/power-platform/products/power-bi/desktop>
- JWT.io. (n.d.). *JWT: JSON Web Tokens*. <https://jwt.io/>
- FastAPI Tutorial. (n.d.). *Password Hashing with Bcrypt*. <https://www.fastapitutorial.com/blog/password-hashing-fastapi/>
- Murgesh. (2023, March 29). *CORS errors: Understanding and fixing Access-Control-Allow-Origin and preflight request issues*. Medium. <https://medium.com/@murgesh.e/cors-errors-understanding-and-fixing-access-control-allow-origin-and-preflight-request-issues-5c89a040aeac>
- Microsoft. (n.d. -b). *Microsoft Azure*. <https://azure.microsoft.com/en-in/>
- Chart.js. (n.d.). *Chart.js documentation*. <https://www.chartjs.org/docs/latest/>
- GeeksforGeeks. (n.d.). *Axios in React – A guide for beginners*. <https://www.geeksforgeeks.org/axios-in-react-a-guide-for-beginners/>