

Finance Tracking App

Rebof Katwal

Rebofkatwal7@gmail.com

Table of Contents

1.	Project Title	1
1.1	Purpose.....	1
1.2	Scope.....	1
1.3	Objectives	1
2.	Project Features and Functionalities	2
2.1	Registration and Login Feature	2
2.2	Transaction History Page	5
2.2.1	Sorting by Date	6
2.2.2	Filtering by Tag	6
2.2.3	Filtering by Type.....	7
2.2.4	Filtering by Date Range	8
2.2	Dashboard.....	9
2.3.1	Clearing the Debts	10
2.3	Adding Transaction	11
2.3.1	Adding Credit	11
2.3.2	Adding Debit.....	12
2.3.3	Adding Debts.....	13
2.4	Custom Tags	14
2.5	Final Dashboard and Transaction History Page after all the Transactions.....	17
3.	Proof of Work.....	18
3.1	Entity relationship Diagram	18
3.2	Wireframes.....	19
3.3	Testing	25
3.3.1	Test 1.....	25
3.3.2	Test 2.....	28
3.3.3	Test 3.....	30
4.1	Roles and Responsibilities	32
4.2	Personal Insights.....	32
4.3	Challenges	32
4.4	Learnings and Growth.....	32
4.5	Impact on Future Work.....	32
4.6	Personal Evaluation	33
5.	Conclusion	34

5.1	Implications	34
5.2	Recommendations	34
5.3	Findings	34
5.4	Limitations	34
5.5	Future Research and Development	34

Table of Figures

Figure 1: Home Page	2
Figure 2: Registration Page.....	3
Figure 3: Login Page	4
Figure 4: Transaction History Page	5
Figure 5: Sorting by date feature	6
Figure 6: Filtering by tag feature	6
Figure 7: Filtering by type feature.....	7
Figure 8: Filtering by date range feature	8
Figure 9: Dashboard Page	9
Figure 10: Clearing the debt feature.....	10
Figure 11: Adding the credit feature	11
Figure 12: Balance after credit transaction.....	11
Figure 13: Adding debit feature	12
Figure 14: Balance after debit transaction.....	12
Figure 15: Adding debt feature	13
Figure 16: Showing debt in dashboard.....	13
Figure 17: Custom Tags feature.....	14
Figure 18: Added the custom tag	15
Figure 19: Showing custom tag in a transaction.....	15
Figure 20: Filtering by custom tag	16
Figure 21: Using the delete button	16
Figure 22: Final Dashboard.....	17
Figure 23: Final History Page.....	17
Figure 24: ERD	18
Figure 32: Registration Wireframe	19
Figure 33: Login Wireframe.....	20
Figure 34: Dashboard Wireframe	21
Figure 35: Settings Wireframe.....	22
Figure 36: Add Transaction Wireframe.....	23
Figure 37: Transaction History Wireframe.....	24
Figure 38: Validation for email.....	26
Figure 39: Validation for password creation	26

Figure 40: Checking for correct credentials	27
Figure 41: Successful Login	27
Figure 42: Amount Validation	28
Figure 43: Successful transaction	29
Figure 44: Debt transaction Validation	30
Figure 45: Successful debt transaction	31

Table of Tables

Table 1: Test 1 Registration	25
Table 2: Test 2 Credit/Debit Transaction.....	28
Table 3: Test 3 Debt Transaction	30

1. Project Title

The Expense Tracker is a cross-platform desktop application intended to help users easily manage personal finances. Users are also able to track inflows and outflows in cash, besides debts, without many hustles. This application supports custom tags, comprehensive financial summary dashboards, and transaction searching, filtering, and sorting by type, date, and tag. It was designed to be friendly for the user and ensured that debts were managed efficiently and that pending payments came into view. The data will be kept safe but shared in JSON format for easy and structured handling. This application is meant to make financial management a lesser hassle by providing users an intuitive and efficient way of tracking.

1.1 Purpose

This application is designed to enable users in tracking expenses by registering information and providing a secure and interactive environment. It ensures data security by offering SHA256 password hashing and facilitates good financial management.

1.2 Scope

It will also provide the functionality to the user in managing their money, tracking transactions, showing highlights, and managing debts. Some of the important features are user registration, analytics on the dashboard, transaction history, and user-configurable settings. Abstraction for JSON storage usage and a modular architecture for scalability reasons are some of the key technical concepts taken into consideration during its development.

1.3 Objectives

The main objective is to provide an efficient and user-friendly application for expense management that enables easy filtering, and provide insight into one's finance. This project aims at showing how code should be organized using interfaces, service layers, and model structures effectively.

2. Project Features and Functionalities

2.1 Registration and Login Feature

Displaying the Home page, which provides options to either register a new user or log in.

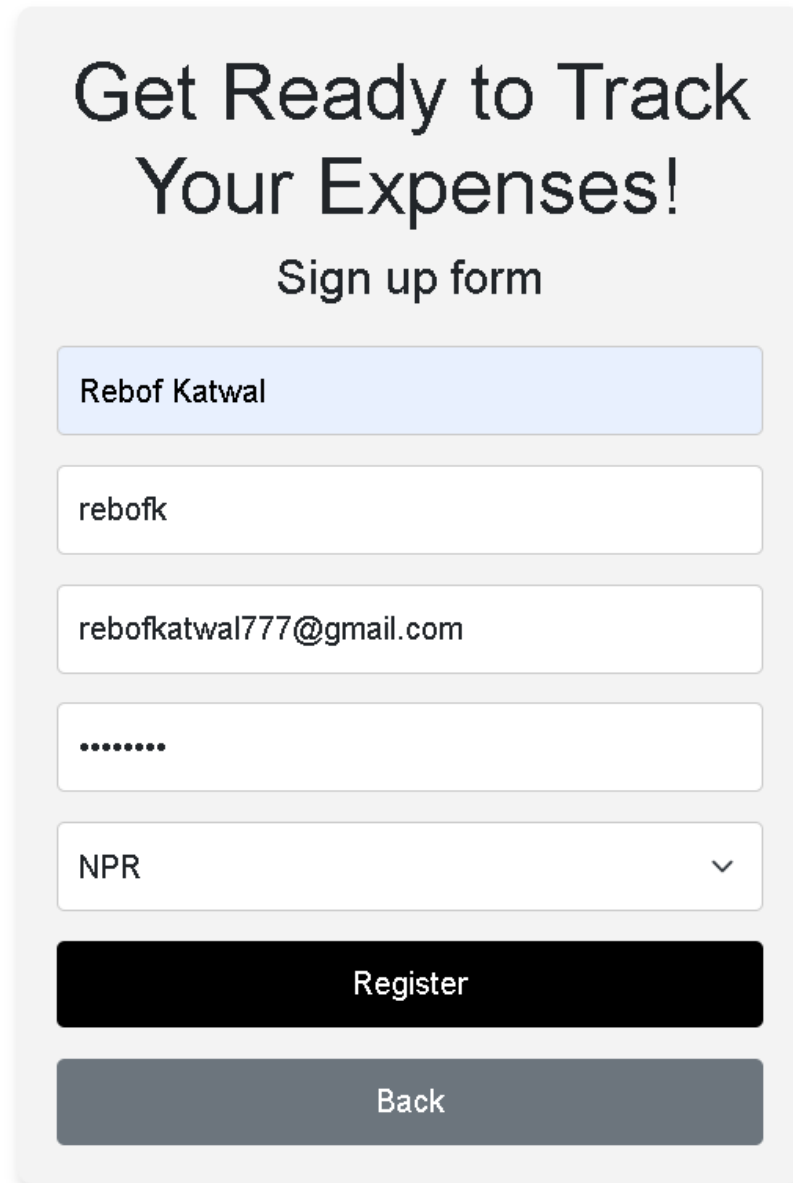
Welcome to Expense Tracker

Would you like to sign up or login?



Figure 1: Home Page

Registering a user with the following details.



The image shows a registration form with a light gray background. At the top, the text "Get Ready to Track Your Expenses!" is displayed in a large, bold, black font. Below this, the text "Sign up form" is centered in a smaller, regular black font. The form consists of several input fields: a name field containing "Rebof Katwal", an email field containing "rebofk", a password field containing "rebofkatwal777@gmail.com", a password confirmation field containing seven dots, and a dropdown menu currently showing "NPR". At the bottom of the form are two buttons: a black "Register" button and a gray "Back" button.

Get Ready to Track
Your Expenses!

Sign up form

Rebof Katwal

rebofk

rebofkatwal777@gmail.com

.....

NPR

Register

Back

Figure 2: Registration Page

Logging in with same username and password after registration

Get Ready to Track Your Expenses!



A login form titled "Login" is displayed within a light gray rounded rectangle. It features two input fields: the first contains the username "rebofk" and the second contains a masked password "*****". Below the fields are two buttons: an orange "Login" button and a blue "Back" button.

Figure 3: Login Page

2.2 Transaction History Page

After logging we can see the transaction history page where we can filter through the transactions as well.

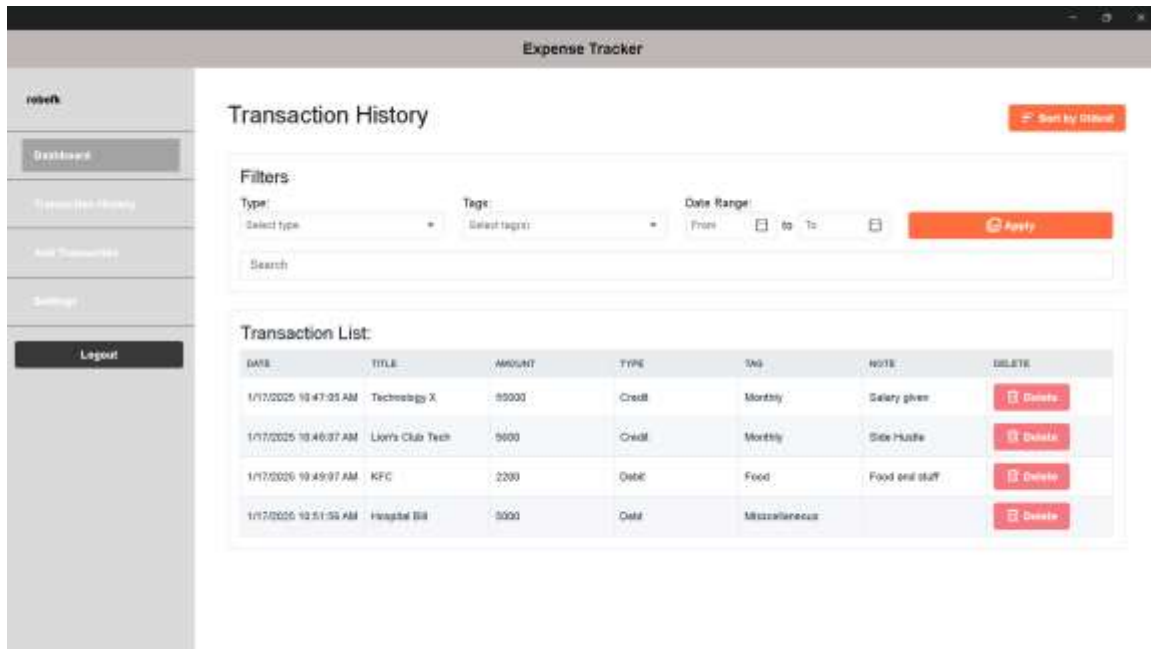


Figure 4: Transaction History Page

2.2.1 Sorting by Date

Using the “Sort by Latest” button

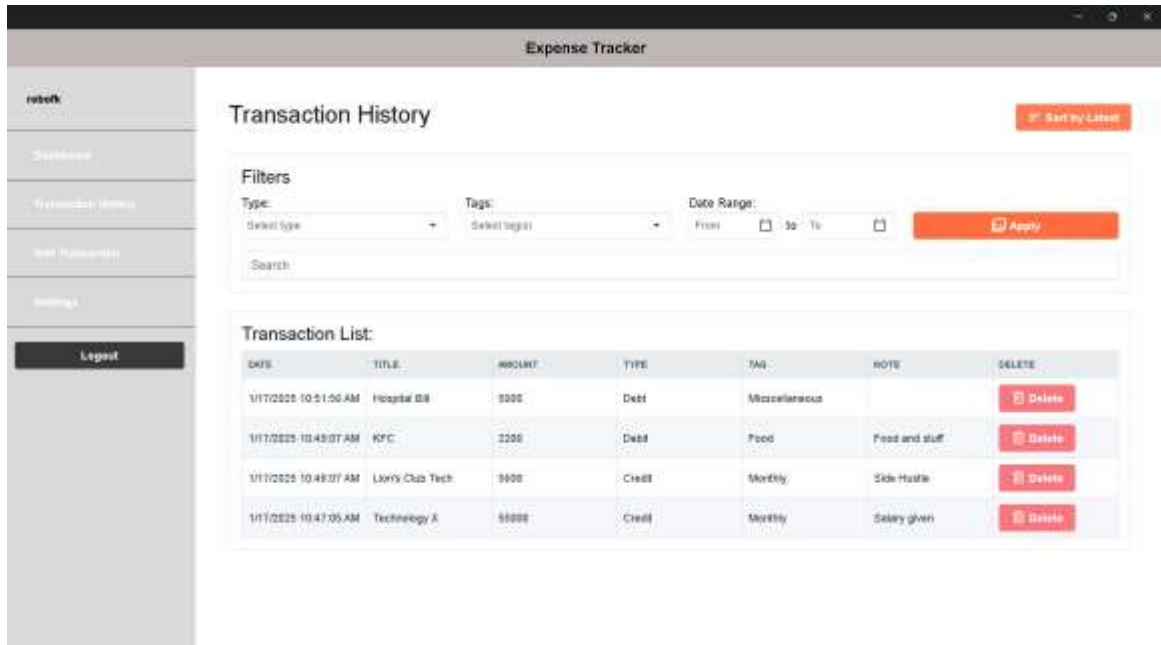


Figure 5: Sorting by date feature

2.2.2 Filtering by Tag

Searching for Monthly Tags.

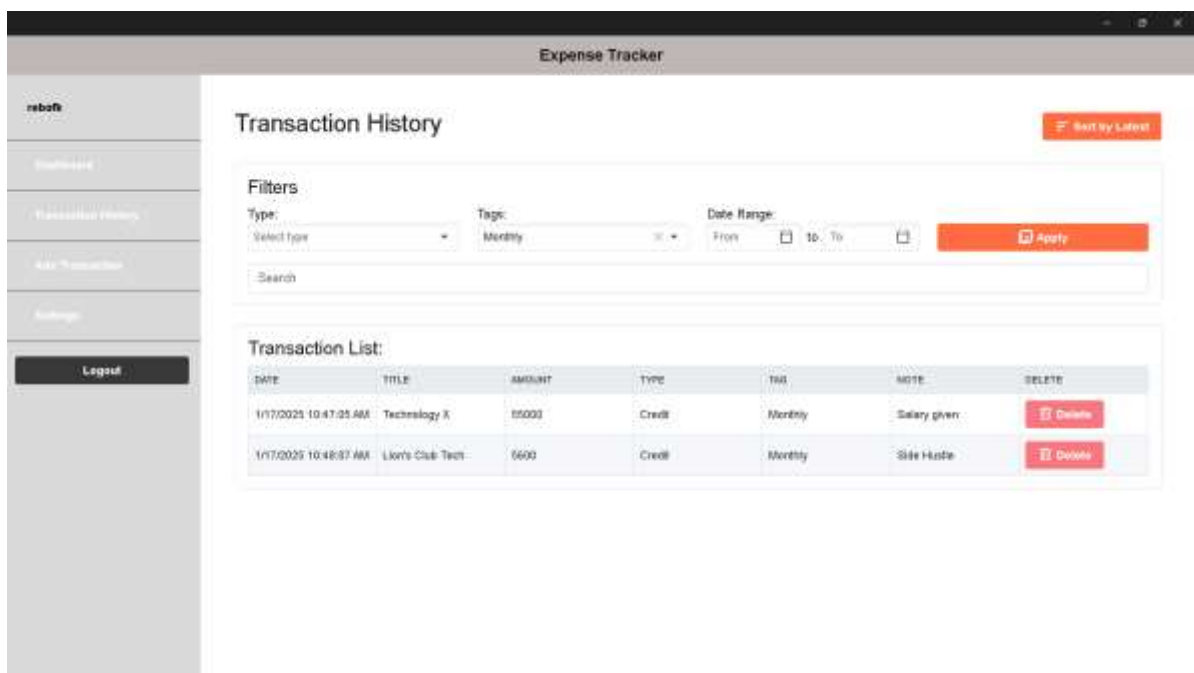


Figure 6: Filtering by tag feature

2.2.3 Filtering by Type

Searching for Debit Type transaction.

The image shows a web application titled "Expense Tracker". On the left is a sidebar with the username "rebof" and a "Logout" button. The main content area is titled "Transaction History" and includes a "Sort by Latest" button. Below the title is a "Filters" section with three dropdown menus: "Type" (set to "Debit"), "Tags" (set to "Select tags"), and "Date Range" (set to "From" to "To"). An "Apply" button is next to the filters. Below the filters is a "Transaction List" table with columns: DATE, TITLE, AMOUNT, TYPE, TAG, NOTE, and DELETE. The table contains one transaction: 1/17/2025 10:49:07 AM, KFC, 220\$, Debit, Food, Food and stuff. A "Delete" button is next to the transaction.

DATE	TITLE	AMOUNT	TYPE	TAG	NOTE	DELETE
1/17/2025 10:49:07 AM	KFC	220\$	Debit	Food	Food and stuff	Delete

Figure 7: Filtering by type feature

2.2.4 Filtering by Date Range

Applying a date range filter; however, no records are displayed as there is no data within the selected range.

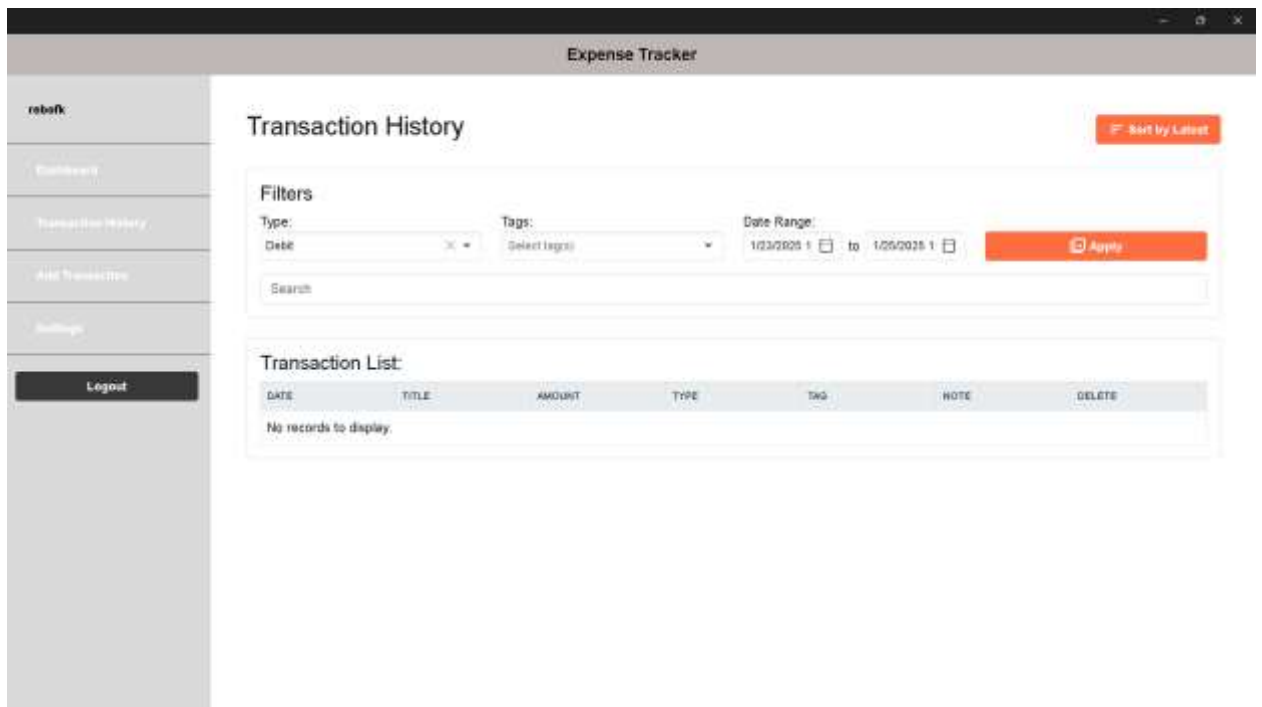


Figure 8: Filtering by date range feature

2.2 Dashboard

The dashboard page displays statistics and highlights the lowest and highest transactions for each type.

The pie chart illustrates the inflow and outflow of cash and debts.

The donut chart shows the transaction count.

Pending debts are also displayed, which can be paid off if there is enough balance available.

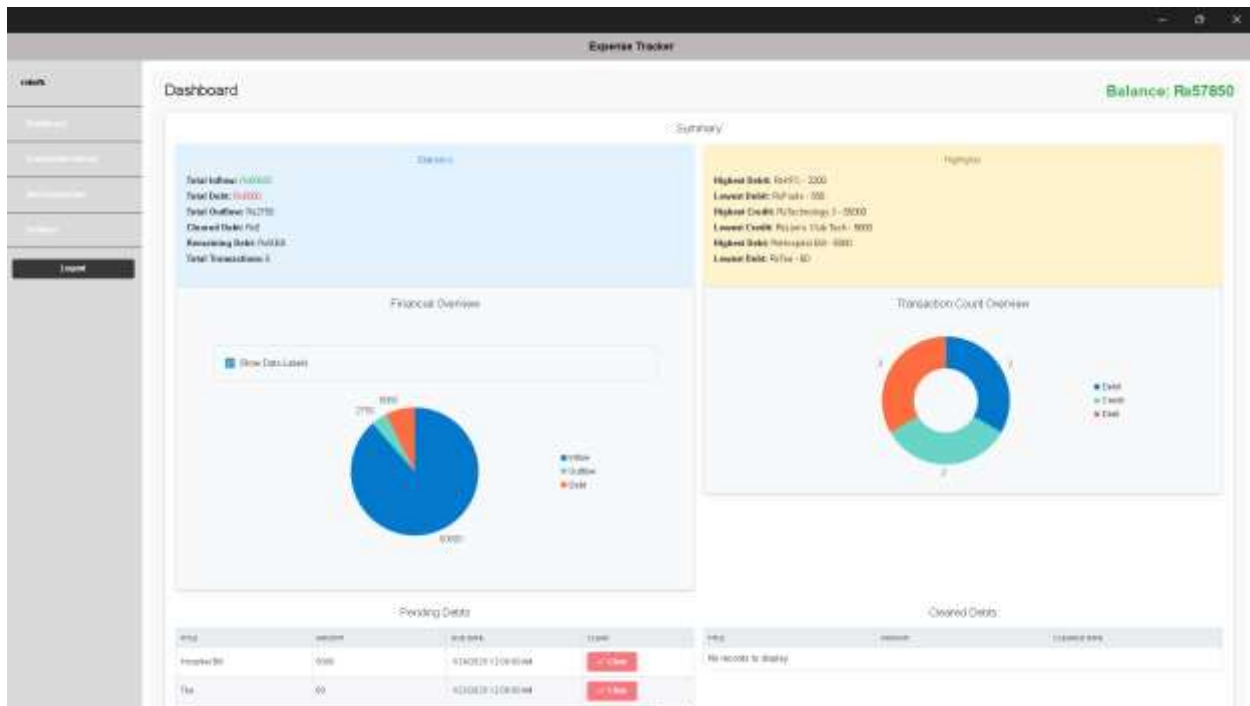


Figure 9: Dashboard Page

2.3.1 Clearing the Debts

After clearing the debts, there is a deduction in the balance as well.

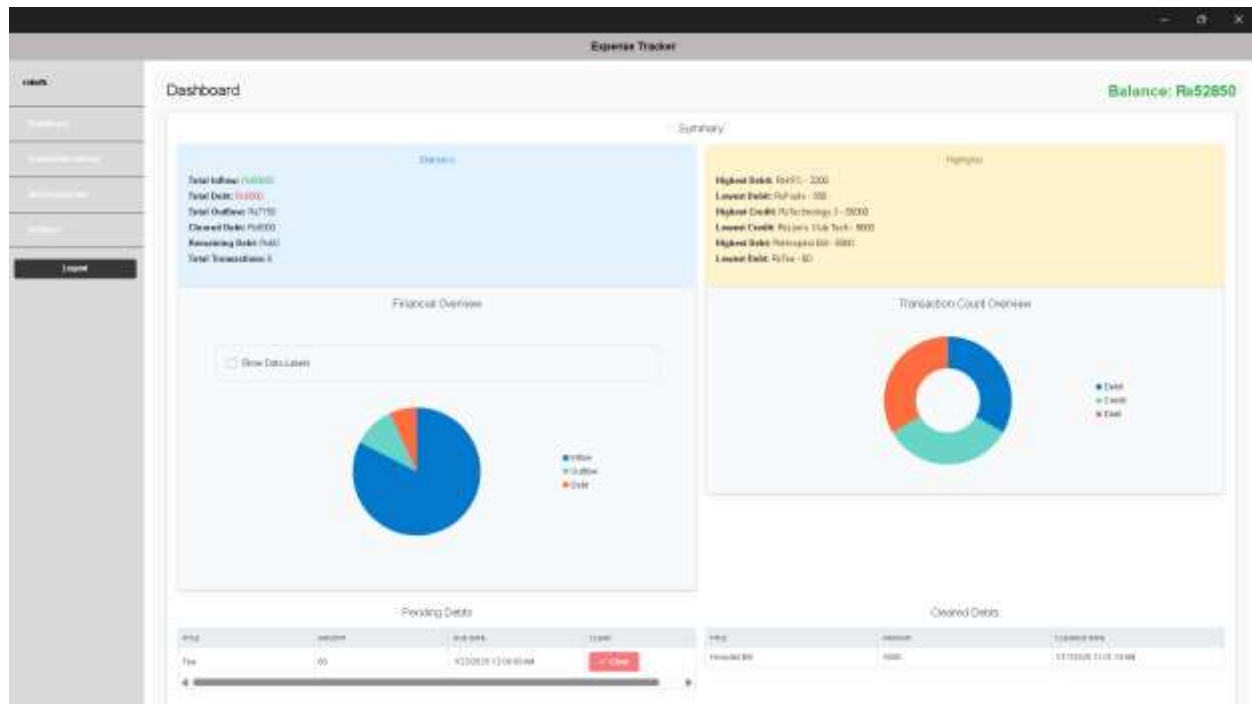
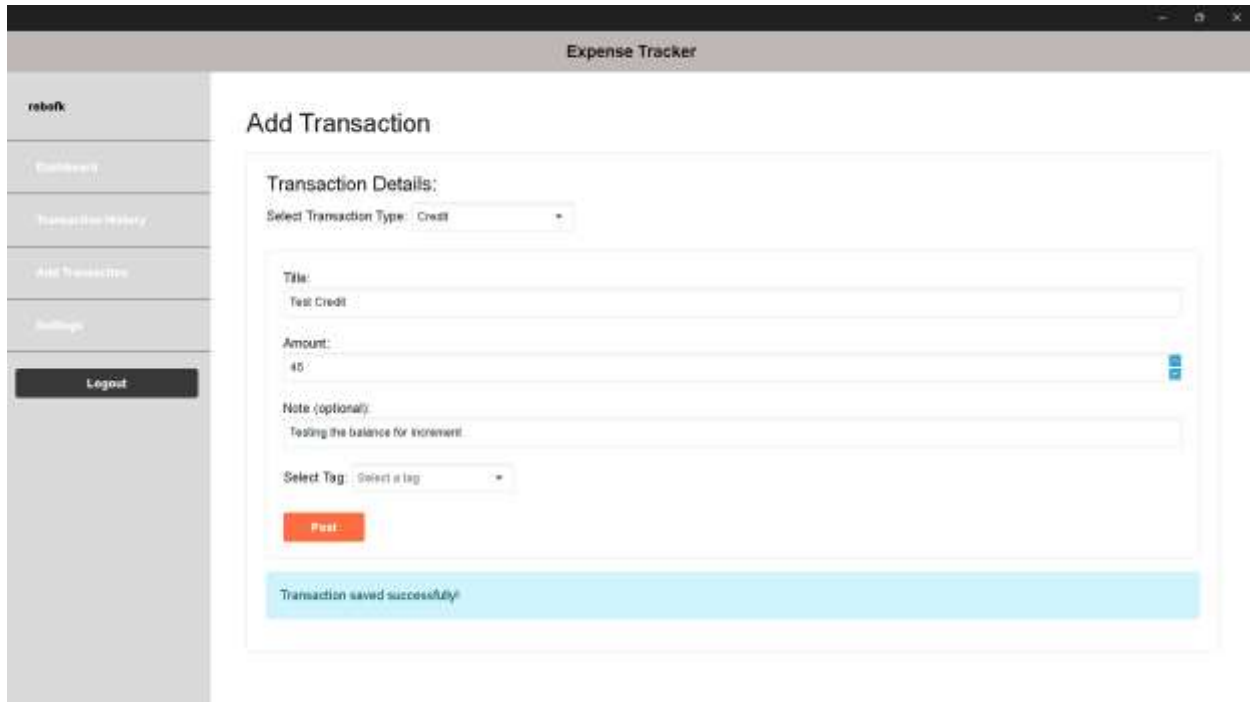


Figure 10: Clearing the debt feature

2.3 Adding Transaction

2.3.1 Adding Credit

I have combined debit, credit, and debt on the same page for user convenience. This way, all three transaction types are accessible without the need for additional navigation pages. Here we are adding a credit transaction.



Expense Tracker

rebof

Dashboard

Transaction History

Add Transaction

Settings

Logout

Add Transaction

Transaction Details:

Select Transaction Type: Credit

Title: Test Credit

Amount: 45

Note (optional): Testing the balance for increment

Select Tag: Select a tag

Post

Transaction saved successfully!

Figure 11: Adding the credit feature

Before the credit transaction, the balance was 57,850 (as shown in the previous dashboard screenshot). After adding 45, the balance has increased to 52,895.

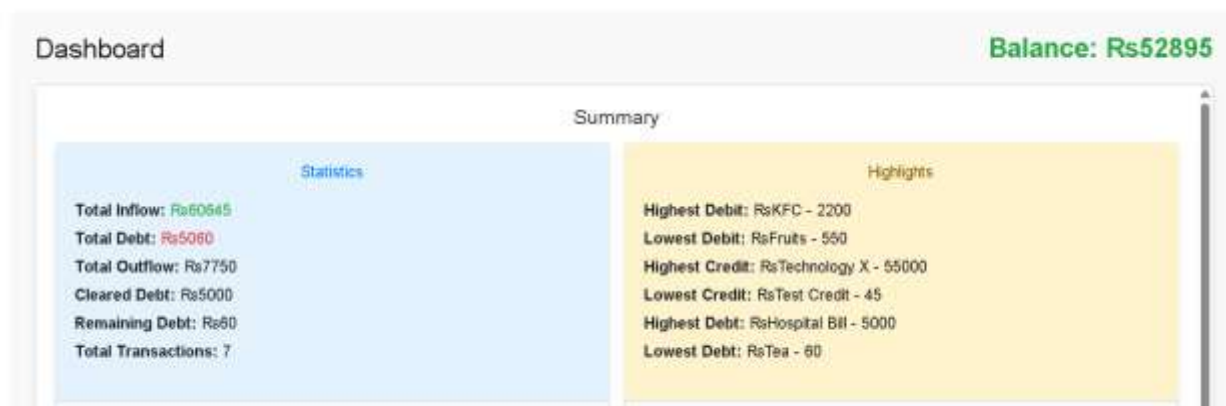


Figure 12: Balance after credit transaction

2.3.2 Adding Debit

Now adding a debit transaction

The screenshot shows the 'Add Transaction' form in the Expense Tracker application. The form is titled 'Add Transaction' and contains the following fields:

- Transaction Details:**
 - Select Transaction Type:** A dropdown menu with 'Debit' selected.
 - Title:** A text input field containing 'Test Debit'.
 - Amount:** A text input field containing '33'.
 - Note (optional):** A text input field containing 'Testing the balance for decrement'.
 - Select Tag:** A dropdown menu with 'Select a tag' selected.
- Post:** A red button to submit the transaction.
- Transaction saved successfully:** A light blue confirmation message at the bottom of the form.

Figure 13: Adding debit feature

The balance was initially 52,895, and after deducting 33 for the debit transaction, the balance is now 52,862.

The screenshot shows the Dashboard of the Expense Tracker application. The dashboard displays the following information:

- Balance:** Rs52862 (in green text at the top right).
- Summary:**
 - Statistics:**
 - Total Inflow: Rs60845
 - Total Debt: Rs5060
 - Total Outflow: Rs7783
 - Cleared Debt: Rs5000
 - Remaining Debt: Rs60
 - Total Transactions: 8
 - Highlights:**
 - Highest Debit: RsKFC - 2200
 - Lowest Debit: RsTest Debit - 33
 - Highest Credit: RsTechnology X - 55000
 - Lowest Credit: RsTest Credit - 45
 - Highest Debt: RsHospital Bill - 5000
 - Lowest Debt: RsTea - 60

Figure 14: Balance after debit transaction

2.3.3 Adding Debts

A debt of 1000 was added.

Figure 15: Adding debt feature

Shown in the dashboard as well

Pending Debts			
TITLE	AMOUNT	DUE DATE	CLEAR
Tea	60	1/22/2025 12:00:00 AM	✓ Clear
Movie Tickets	1000	1/18/2025 12:00:00 AM	✓ Clear

Figure 16: Showing debt in dashboard

2.4 Custom Tags

Now we will be adding a custom tag.

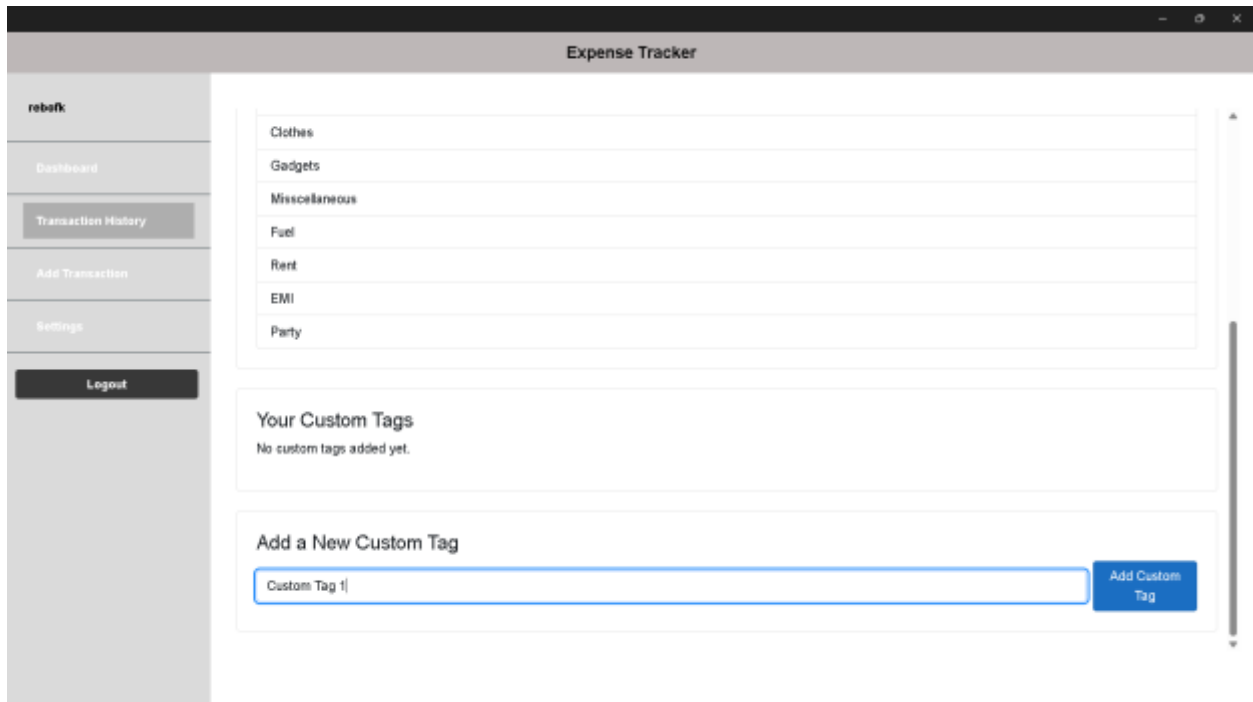


Figure 17: Custom Tags feature

Custom tag was successfully added.

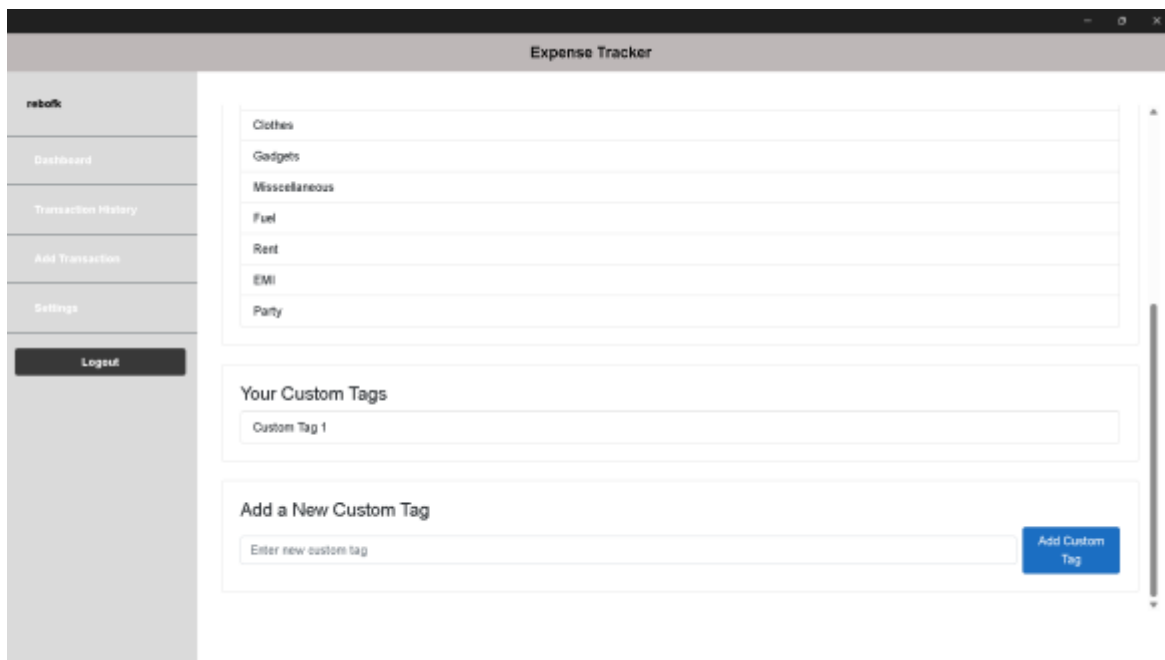


Figure 18: Added the custom tag

We can use the custom tag while adding any type of transaction.

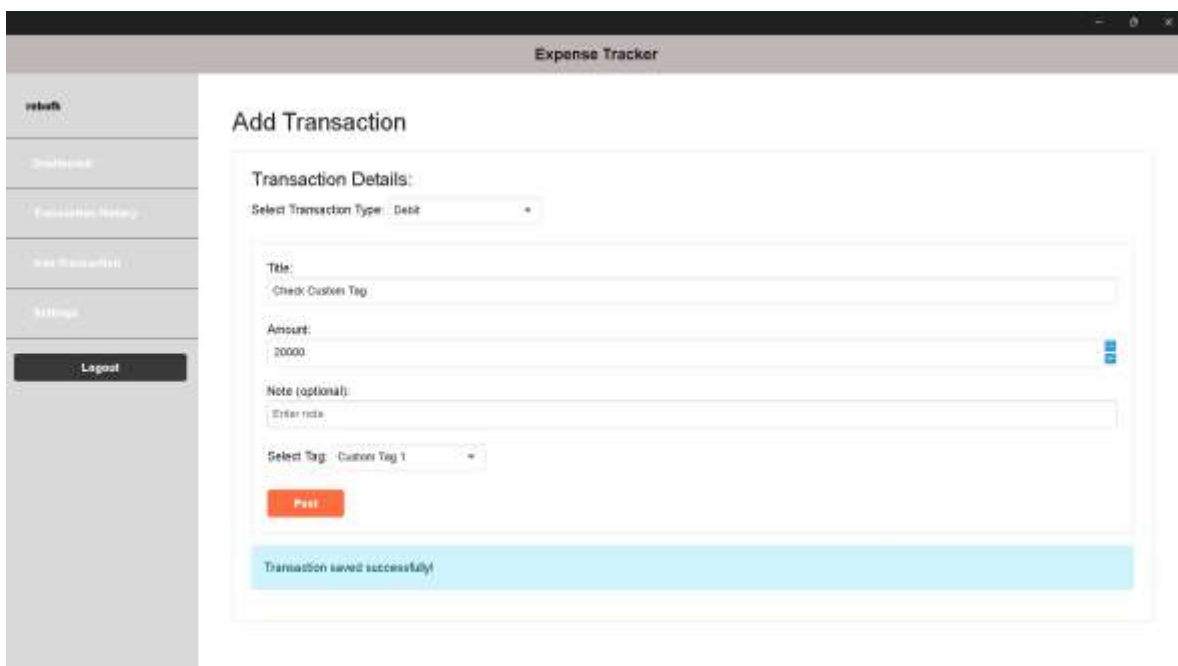


Figure 19: Showing custom tag in a transaction

We can use the custom tag for filtering as well

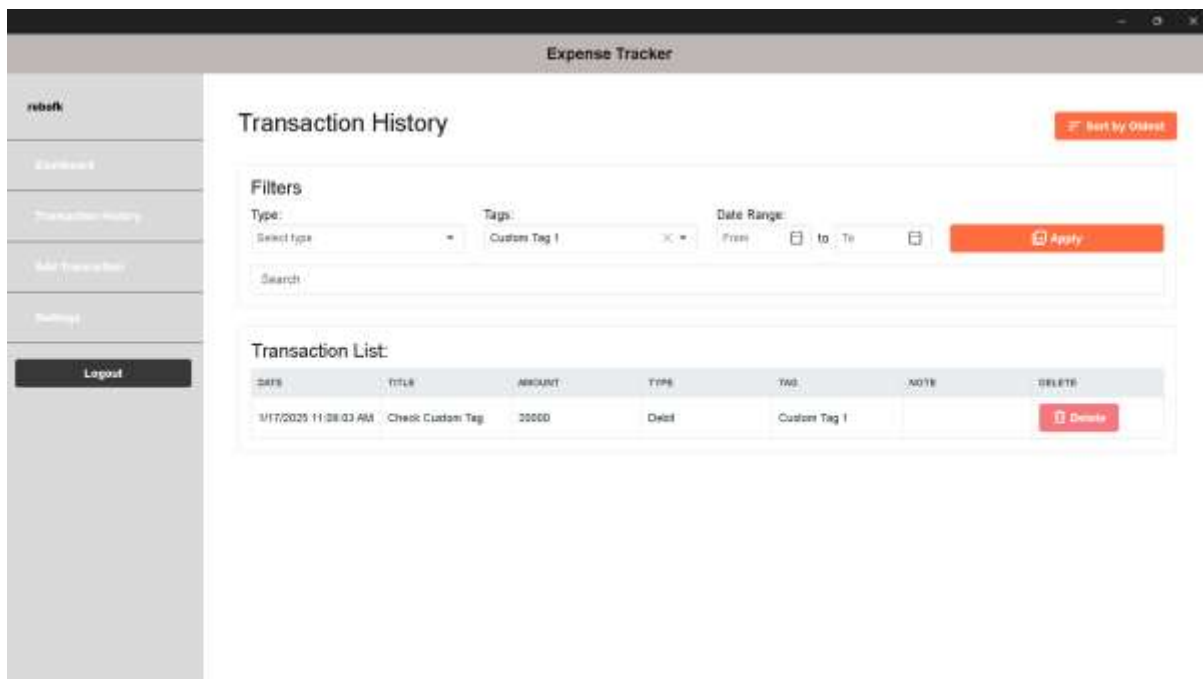


Figure 20: Filtering by custom tag

Using the delete button, I have deleted the previously shown transaction.

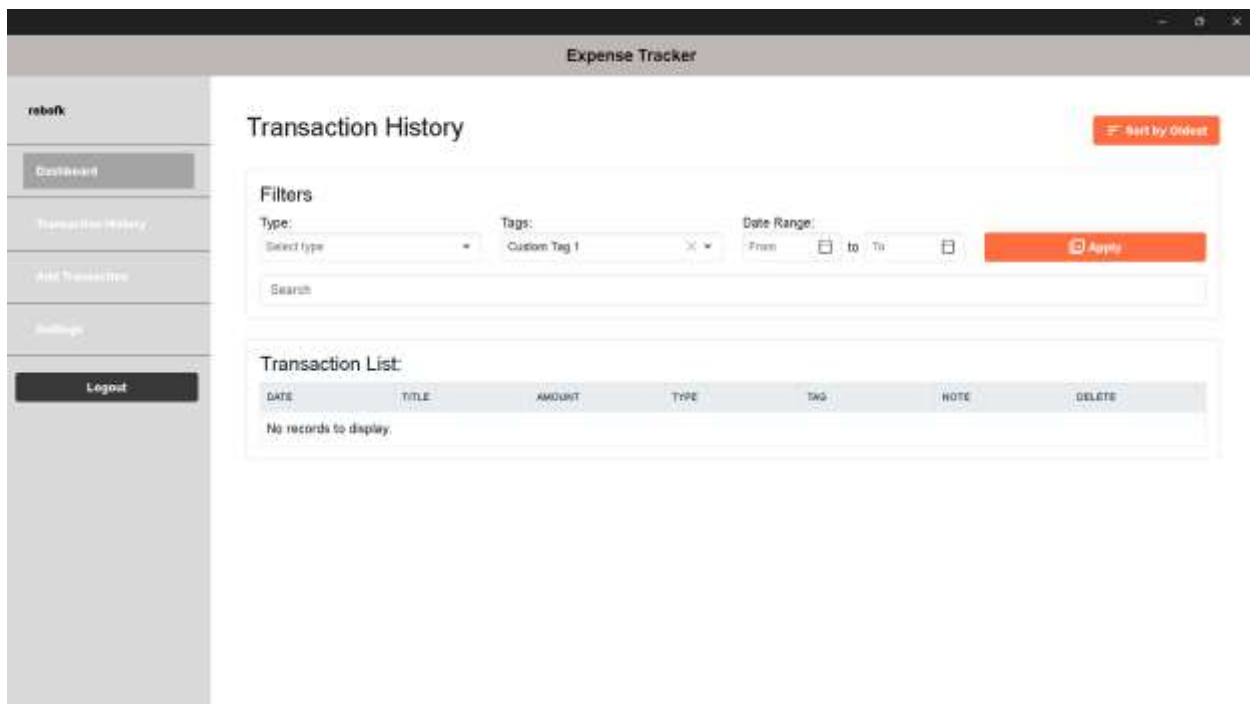


Figure 21: Using the delete button

2.5 Final Dashboard and Transaction History Page after all the Transactions.

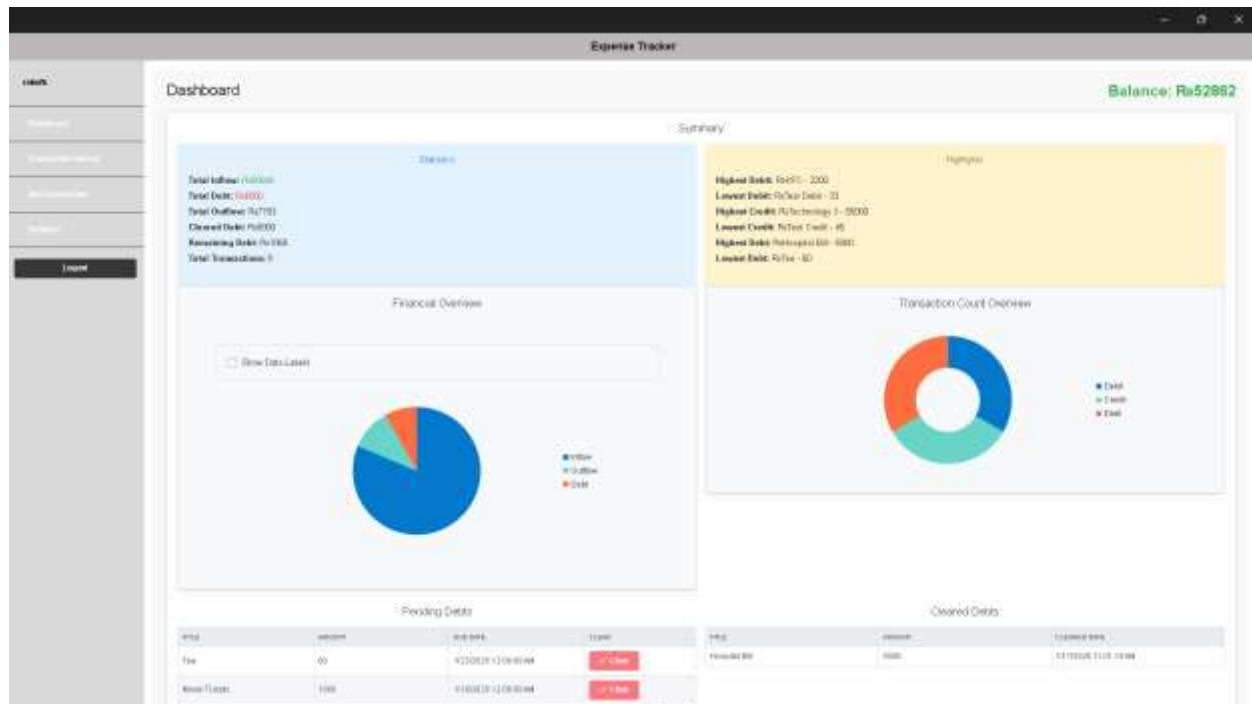


Figure 22: Final Dashboard

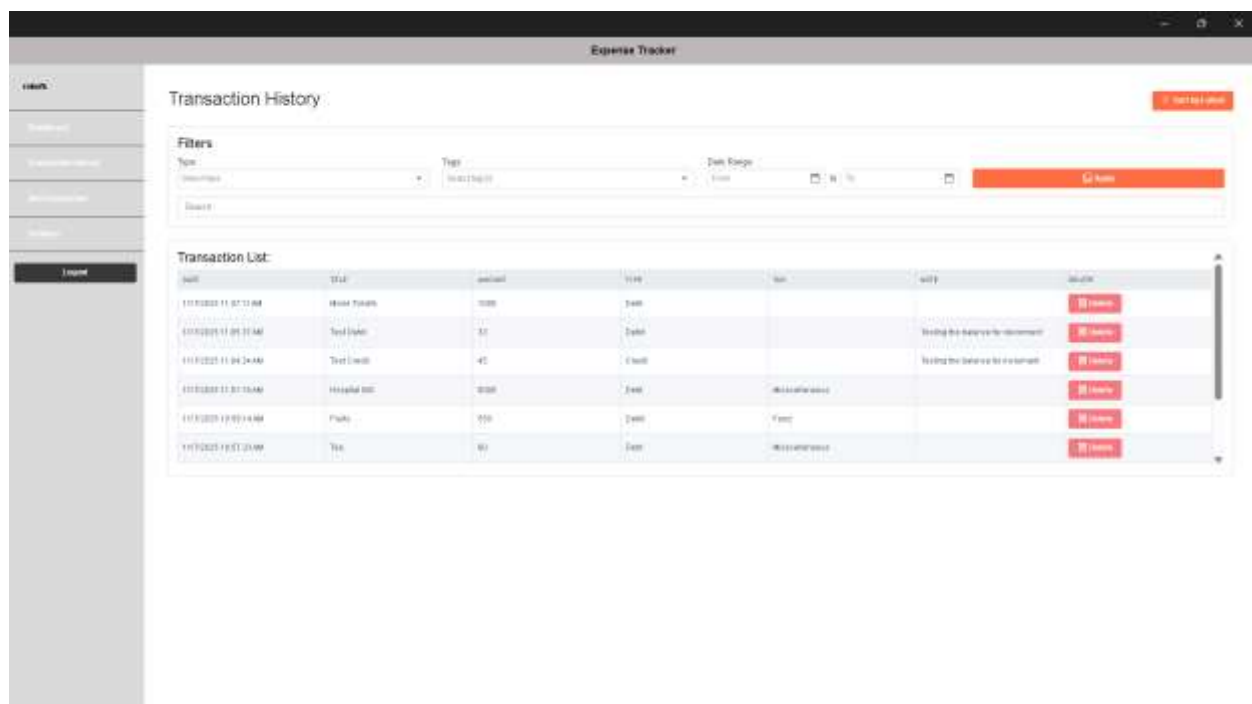


Figure 23: Final History Page

3. Proof of Work

3.1 Entity relationship Diagram

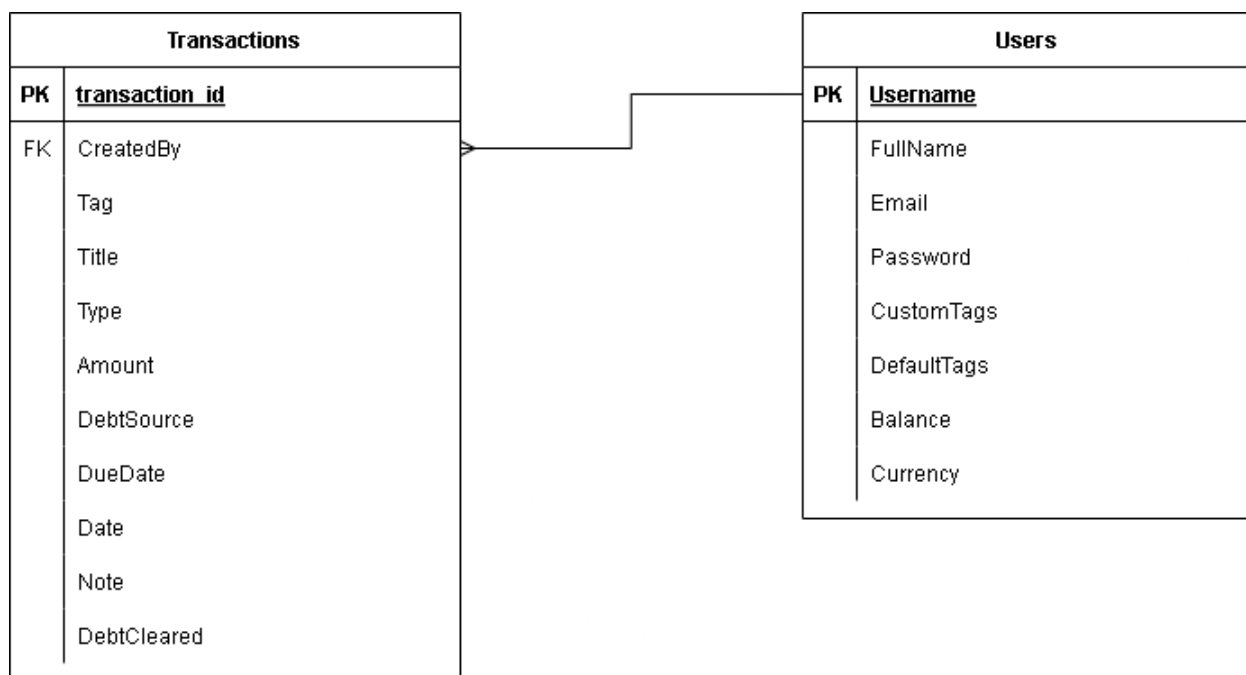
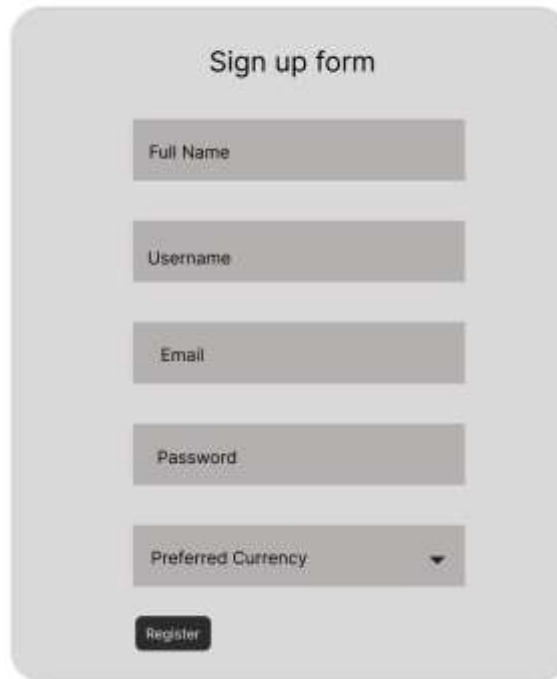


Figure 24: ERD

3.2 Wireframes

Get Ready to Track Your Expenses!



A wireframe for a registration form titled "Sign up form". The form is contained within a light gray rounded rectangle. It features five input fields stacked vertically: "Full Name", "Username", "Email", "Password", and "Preferred Currency" (which is a dropdown menu with a downward arrow). Below these fields is a dark gray "Register" button.

Figure 25: Registration Wireframe

Get Ready to Track Your
Expenses!



A wireframe for a login form. It features a light gray rounded rectangle containing the text "Login" at the top. Below this are two input fields: the first is labeled "Username" and the second is labeled "Password". At the bottom of the form is a dark gray button with the text "Login" in white.

Figure 26: Login Wireframe

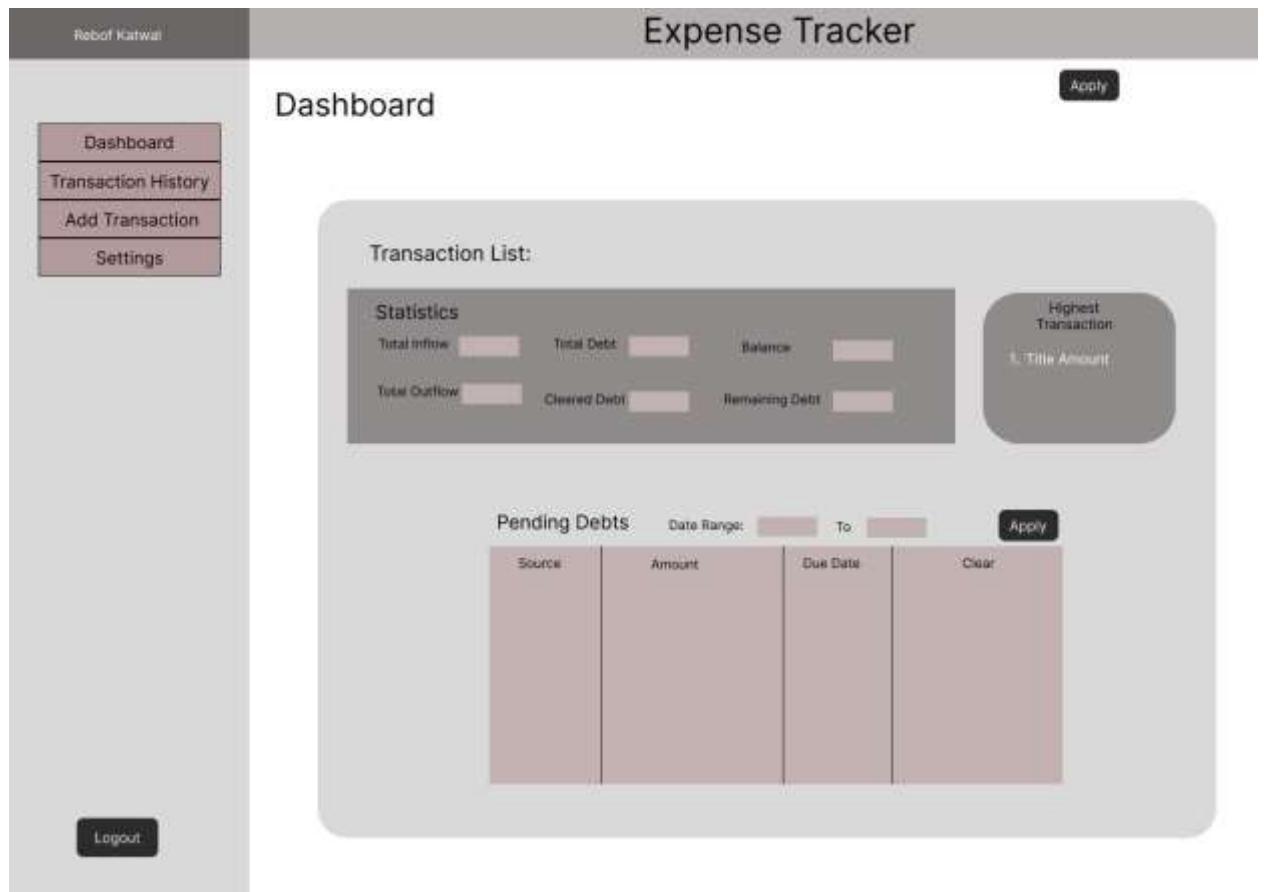
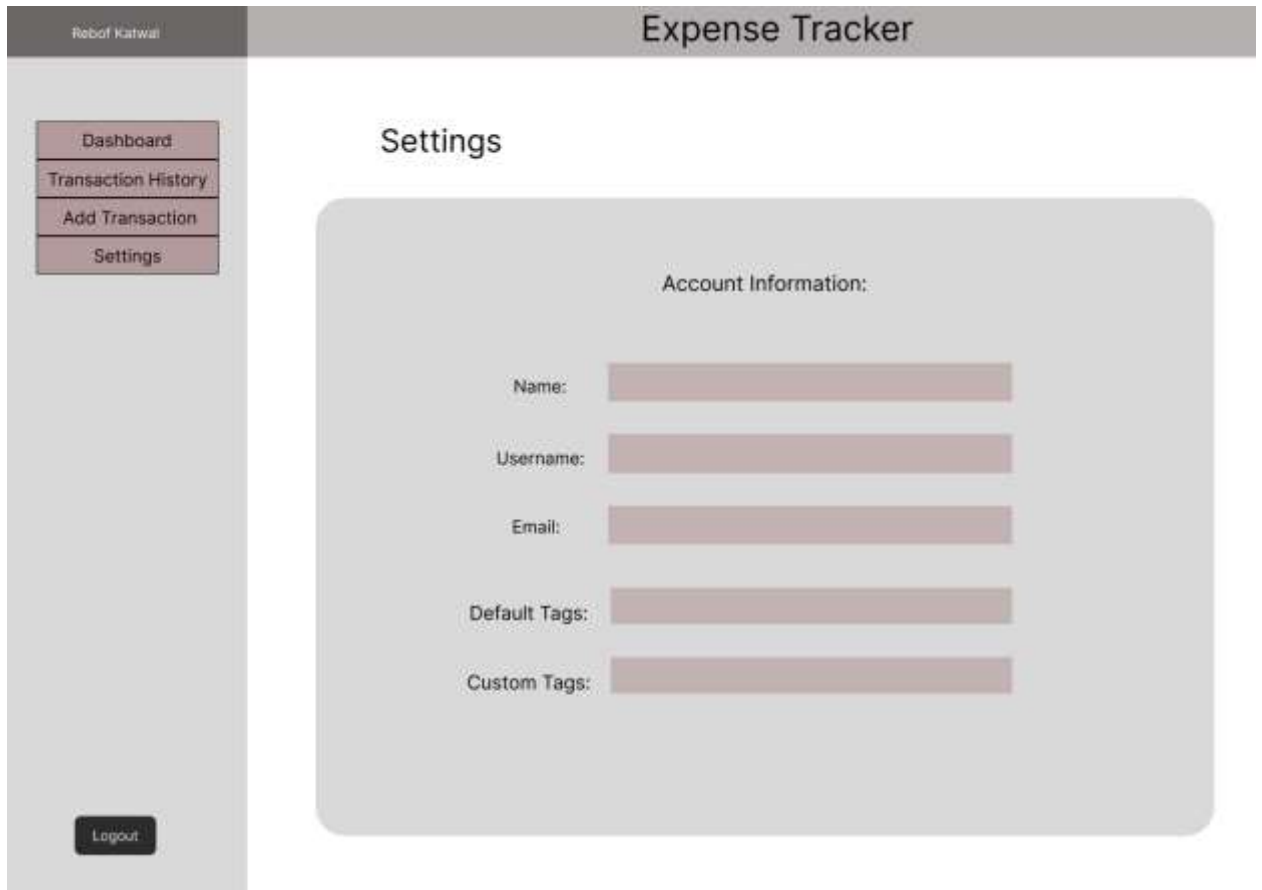


Figure 27: Dashboard Wireframe



The wireframe shows a web application interface for an "Expense Tracker". At the top, a dark header bar contains the user's name "Rebof Katwal" on the left and the application title "Expense Tracker" on the right. Below the header, the page is divided into two main sections. On the left is a vertical sidebar with a light gray background. It contains a menu with four items: "Dashboard", "Transaction History", "Add Transaction", and "Settings", each in a light red button. At the bottom of the sidebar is a dark "Logout" button. The main content area on the right has a white background and is titled "Settings". Below the title is a light gray rounded rectangle containing the "Account Information:" section. This section includes five labels with corresponding input fields: "Name:", "Username:", "Email:", "Default Tags:", and "Custom Tags:". Each label is followed by a light red rectangular input field.

Rebof Katwal

Expense Tracker

Settings

Account Information:

Name:

Username:

Email:

Default Tags:

Custom Tags:

Logout

Figure 28: Settings Wireframe

Rebof Katwal

Expense Tracker

Dashboard

Transaction History

Add Transaction

Settings

Logout

Add Transaction

Transaction Detail:

Title:

Amount:

Debit Source
(IF APPLICABLE):

Due Date
(IF APPLICABLE):

Note
(Optional):

Tags:

+

Type

☒ Debit

Post

Figure 29: Add Transaction Wireframe

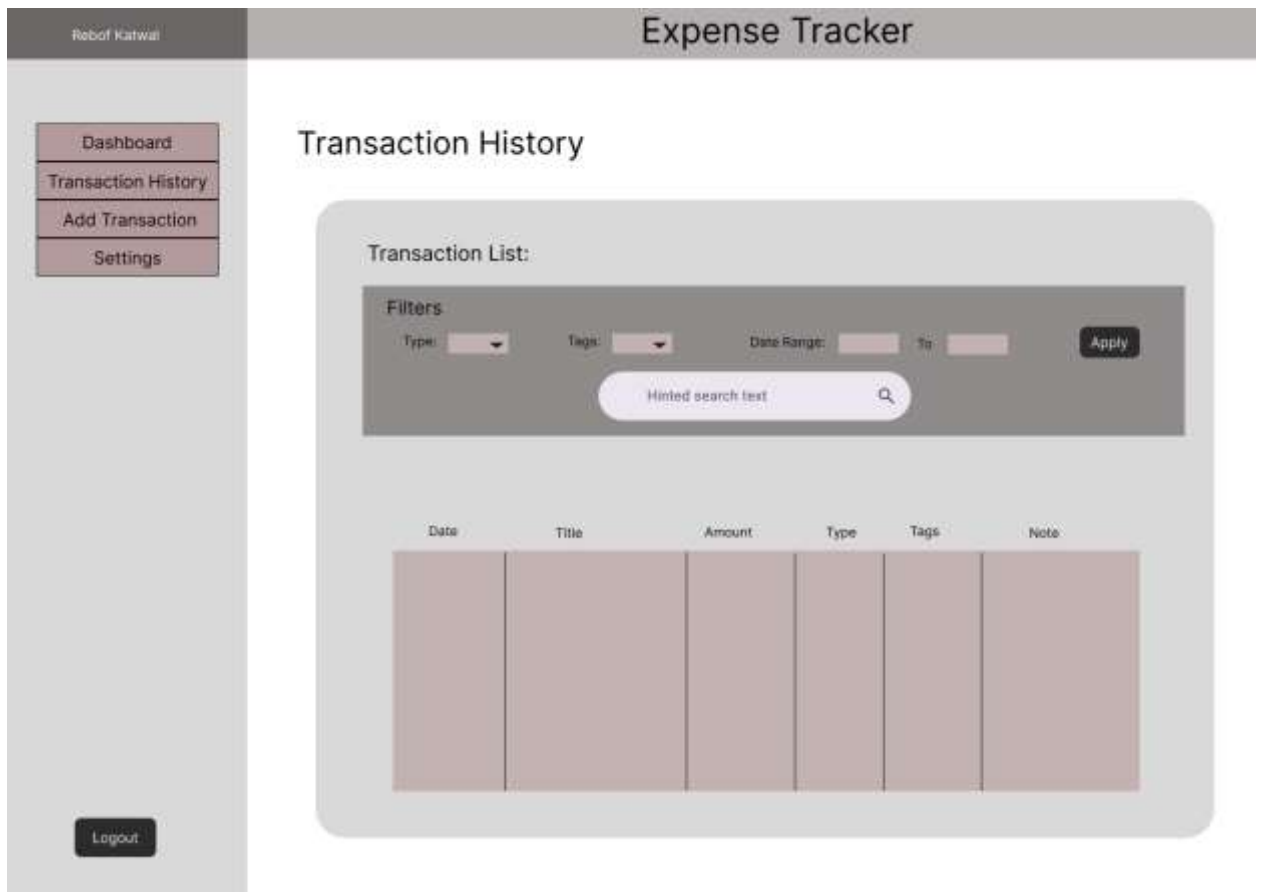


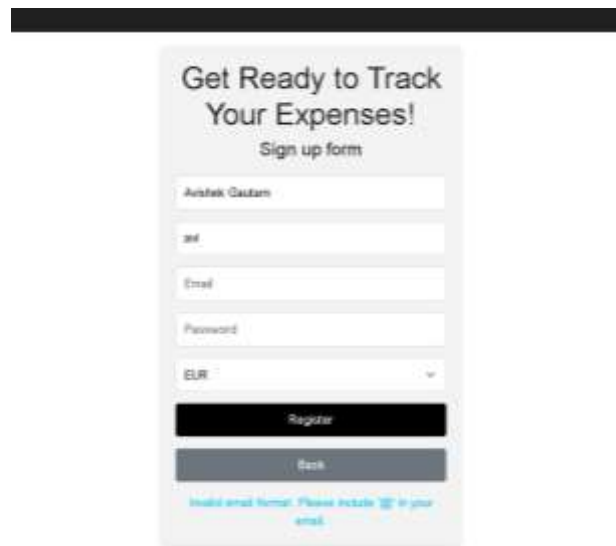
Figure 30: Transaction History Wireframe

3.3 Testing

3.3.1 Test 1

Test no:	1
Objective:	To check the registration and login validations.
Action	<ul style="list-style-type: none">• Tried to keep the email empty.• Then kept the password with only one character.• The validation says wo keep @ in the email and password should be 8 characters long.• After, redirected to login.• Wrong credentials were kept.• Could not log in.• With correct credentials successful login.
Conclusion:	The test was successful.

Table 1: Test 1 Registration



Get Ready to Track Your Expenses!

Sign up form

Avishek Gautam

avi

Email

Password

EUR

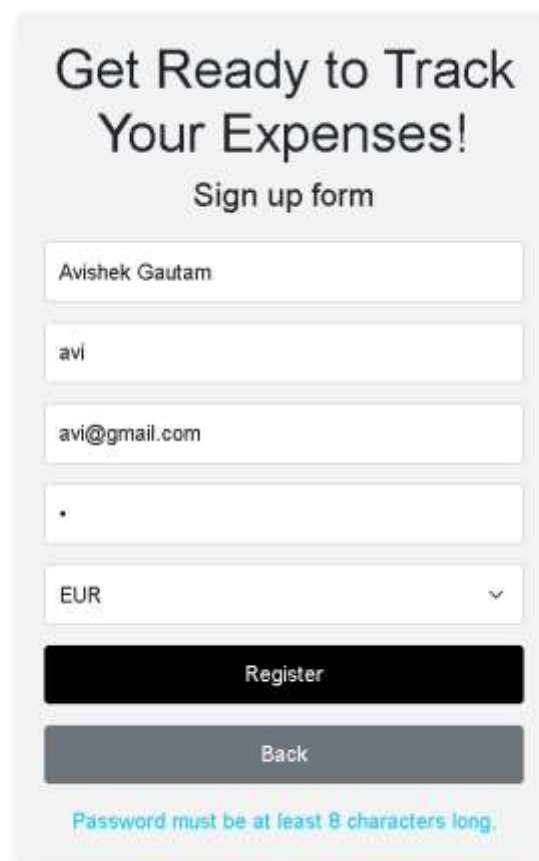
Register

Back

Invalid email format. Please include '@' in your email.

This figure shows a mobile app registration screen. The title is 'Get Ready to Track Your Expenses!' and the subtitle is 'Sign up form'. The form contains five input fields: a name field with 'Avishek Gautam', a username field with 'avi', an email field, a password field, and a currency dropdown menu showing 'EUR'. Below the fields are two buttons: 'Register' (black) and 'Back' (grey). A red error message at the bottom states 'Invalid email format. Please include @ in your email.'

Figure 31: Validation for email



Get Ready to Track Your Expenses!

Sign up form

Avishek Gautam

avi

avi@gmail.com

*

EUR

Register

Back

Password must be at least 8 characters long.

This figure shows the same mobile app registration screen as Figure 31, but with different data. The name field is 'Avishek Gautam', the username is 'avi', and the email is 'avi@gmail.com'. The password field now contains a single asterisk '*'. The currency dropdown is still 'EUR'. The 'Register' and 'Back' buttons are present. A red error message at the bottom states 'Password must be at least 8 characters long.'

Figure 32: Validation for password creation

Get Ready to Track Your Expenses!



A login form titled "Login" with a light gray background. It contains two input fields: the first contains the text "avi" and the second contains "wrongpassword" with a small eye icon to its right. Below the fields are two buttons: an orange "Login" button and a blue "Back" button. At the bottom, the text "Invalid username or password." is displayed in red.

Figure 33: Checking for correct credentials

Get Ready to Track Your Expenses!



A login form titled "Login" with a light gray background. It contains two input fields: the first contains the text "avi" and the second contains "*****". Below the fields are two buttons: an orange "Login" button and a blue "Back" button. At the bottom, the text "Login successful!" is displayed in green.

Figure 34: Successful Login

3.3.2 Test 2

Test no:	2
Objective:	To check for validation in credit/debit transaction
Action	The amount was kept at 0. Tried to post the transaction. It fails. Kept the amount at 10. It works.
Conclusion:	The test was successful.

Table 2: Test 2 Credit/Debit Transaction

Add Transaction

Transaction Details:

Select Transaction Type: Credit

Title:
Check validation

Amount:
0

Note (optional):
Enter note

Select Tag: Select a tag

Post

Error: Amount must be greater than zero.

Figure 35: Amount Validation

Add Transaction

Transaction Details:

Select Transaction Type: Credit

Title:

Check validation

Amount:

10

Note (optional):

Enter note

Select Tag: Select a tag

Post

Transaction saved successfully!

Figure 36: Successful transaction

3.3.3 Test 3

Test no:	3
Objective:	To check for validation in debt transaction
Action	<ul style="list-style-type: none"> The debt source and due date were left empty. Tried to post the transaction It fails. Then we enter the debt source and due date. It works.
Conclusion:	The test was successful.

Table 3: Test 3 Debt Transaction

Add Transaction

Select Transaction Type: Debt

Title: Check validation

Amount: 100

Debt Source (if applicable): Enter debt source

Due Date (if applicable): Select due date

Note (optional): Enter note

Select Tag: Select a tag

Post

Error: Debt Source and Due Date are required for a Debt transaction.

Figure 37: Debt transaction Validation

Add Transaction

Select Transaction Type: Debt

Title:
Check validation

Amount:
100

Debt Source (if applicable):
Prashant Shah

Due Date (if applicable):
1/25/2025 12:00:00 AM

Note (optional):
Enter note

Select Tag: Select a tag

Post

Transaction saved successfully!

Figure 38: Successful debt transaction

4. Individual Reflection

4.1 Roles and Responsibilities

My core role was to successfully implement and deliver the project in time with all the required functionalities. In addition to balancing other coursework deadlines, I had to effectively manage my time. I had to implement some key technical features, such as password hashing and modular design patterns.

4.2 Personal Insights

The project proved very insightful because I got to learn many concepts in programming that have made my understanding of C#, .NET, and most specifically MAUI so much richer. Working with both Radzen and JSON storage gave insight into abstraction and how it eases data handling.

4.3 Challenges

Given that this was my first time using this language and framework, the task proved to be quite challenging, given the time constraints too. The task was made achievable with daily practice over a month. The own share of issues with JSON parsing and third-party component integration tested my problem-solving capabilities.

4.4 Learnings and Growth

This project completion took my technical skills up a notch. I now fully understand concepts around .NET-especially how interfaces and service-model structures work-and feel confident using Radzen for UI components. It has strengthened my capabilities for complex development environments.

4.5 Impact on Future Work

The project added immense value to my resume. Now, I can develop .NET-based projects independently and create useful applications. It taught me the importance of clean architecture and code reusability, for which my work will be responsible in the future.

4.6 Personal Evaluation

That would be the kind of experience that leads one to massive changes in the skill set of a programmer. All the best features, like Razor syntax, `@code{}` blocks, dependency injection, and modular design, seemed to give me all the bases for making much larger projects and contributing to the development of teams.

5. Conclusion

The application is designed to be user-friendly, secure, and efficient in managing one's personal expenses. Integration of user registration, a dynamic dashboard with statistics, custom labels on transactions, and an intuitive history page within the application will be extremely valuable for the users. As for the advanced technical practices used in the application, these are Razor syntax, dependency injection, and modular design. Those were my efforts to make it scalable and easy to maintain. All was done in pursuit of the greater goal: to simplify expense tracking without taking so much from the user in terms of experience and data security.

5.1 Implications

This application showed the practical usefulness of the components of Radzen and how third-party libraries can be used to ease the process of developing something. It also proved that using service and model structures helps to keep codes clean and maintainable.

5.2 Recommendations

The future versions of the application will be able to include internet integration, sync with Google Calendar, and save data online through cloud services. Extended use of APIs can be used to extend this application.

5.3 Findings

The components of the Radzen used - charts, tables - reduced development time significantly and are highly functional. Apart from Radzen, I researched about MudBlazor and Winforms to see other alternatives and know where each one is useful.

5.4 Limitations

This application does not contain internet-based features, which limit users from creating expense records offline. Data synchronization in real-time and advanced analytics were not developed in the paper.

5.5 Future Research and Development

Future development may integrate the following: more integrations of applications for extended functionality, real-time data synchronization, and AI/ML-based financial

analysis. Advanced functionality of Radzen and integration with third-party APIs will also be a good addition.

This project wasn't just a serious technological challenge but also a very important learning experience. It laid the foundation for my growth as a programmer, enabling me to work confidently with advanced frameworks and design patterns. The knowledge and skills acquired will definitely influence and improve my contribution to software development projects in the future.