``

CS6004NP

Information Systems

****

**Module Code & Module Title**

**CS6004NP Application Development**

**Assessment Type**

**30% Individual Coursework**

**Semester- 5**

**2024 Autumn**

**Student Name: Aman Gurung**

**London Met ID: 22068950**

**College ID: NP04CP4A220019**

**Assignment Due Date: Sunday, January 5, 2025**

**Assignment Submission Date: Sunday, January 5, 2025**

**Submitted To: Mr. Mahesh Dhungana**

**Git:** <https://github.com/amangrx/DigiDhan>

**Declaration**

*I confirm that I understand my coursework needs to be submitted online via My second teacher under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submission will be treated as non-submission and a mark of zero will be awarded.*

**Table of Contents:**

[1. Introduction: 1](#_Toc187980544)

[1.1 Aims and Objectives: 2](#_Toc187980545)

[2 Feature and functionalities: 3](#_Toc187980546)

[2.1 Prototype: 3](#_Toc187980547)

[2.1.1 Prototype of Login page: 3](#_Toc187980548)

[2.1.2 Prototype of Home page: 4](#_Toc187980549)

[2.1.3 Prototype of Income page: 5](#_Toc187980550)

[2.1.4 Prototype of Expenses page: 6](#_Toc187980551)

[2.1.5 Prototype of Debt page: 7](#_Toc187980552)

[2.2 Developed Application: 8](#_Toc187980553)

[2.2.1 Login page: 8](#_Toc187980554)

[2.2.2 Home page: 9](#_Toc187980555)

[2.2.3 Income page: 10](#_Toc187980556)

[2.2.4 Expense page: 11](#_Toc187980557)

[2.2.5 Debt page: 12](#_Toc187980558)

[2.2.6 Tags page: 13](#_Toc187980559)

[2.3 Test cases: 14](#_Toc187980560)

[2.3.1 Test 1: Logging in 14](#_Toc187980561)

[3. Development and Logical solution: 15](#_Toc187980562)

[3.1 Entity Relationship Diagram: 15](#_Toc187980563)

[3.2 Use Case Diagram: 16](#_Toc187980564)

[3.3 Flowchart: 17](#_Toc187980565)

[3.4 Version Control (Git): 18](#_Toc187980566)

[3.5 Technology stack: 19](#_Toc187980567)

[3.5.1 Framework: 19](#_Toc187980568)

[3.5.2 External Libraries: 19](#_Toc187980569)

[3.5.3 Persistence Mechanism: 19](#_Toc187980570)

[3.6 Class, properties and methods: 20](#_Toc187980571)

[3.6.1 Classes and their properties: 20](#_Toc187980572)

[3.6.2 Classes and their methods: 21](#_Toc187980573)

[4 Conclusion: 26](#_Toc187980574)

[Bibliography 27](#_Toc187980575)

**Table of Figures:**

[Figure 1: Prototype for login page. 3](#_Toc187994702)

[Figure 2: Prototype for Home page 4](#_Toc187994703)

[Figure 3: Prototype for income page 5](#_Toc187994704)

[Figure 4: Prototype for expenses page 6](#_Toc187994705)

[Figure 5: Prototype for Debt page. 7](#_Toc187994706)

[Figure 6: Developed Login page. 8](#_Toc187994707)

[Figure 7: Developed Home page. 9](#_Toc187994708)

[Figure 8: Developed Income page. 10](#_Toc187994709)

[Figure 9: Developed Expense page 11](#_Toc187994710)

[Figure 10: Developed Debt page. 12](#_Toc187994711)

[Figure 11: Developed tags page. 13](#_Toc187994712)

[Figure 12: Entity relationship diagram of Digi Dhan 15](#_Toc187994713)

[Figure 13: Use case diagram of Digi Dhan 16](#_Toc187994714)

[Figure 14: Flowchart of Digi Dhan 17](#_Toc187994715)

[Figure 15: Version Control in Git. 18](#_Toc187994716)

**Table of Table:**

[Table 1: Transaction class and its properties. 20](#_Toc187994687)

[Table 2: DatabaseConnection.cs and its methods. 21](#_Toc187994688)

[Table 3: DatabaseValueInsertion.cs and its methods. 22](#_Toc187994689)

[Table 4: DatabaseExtraction.cs and its methods. 23](#_Toc187994690)

[Table 5: DebtManagement.cs and its methods. 25](#_Toc187994691)

[Table 6: TagManagement.cs and its methods. 25](#_Toc187994692)

# Introduction:

This is the project report for our coursework of Application Development. In this coursework, we are required to develop a desktop application for personal expense tracking. We are required to develop the application in .NET Core framework. The application includes some major features like tracking of cash income, cash expense, debts, search transactions, filter and sort by title, type etc.

My project is titled ‘Digi Dhan’. Digi Dhan provides user friendly interfaces and working environment to the end user. The user is prompted to the login page at the start of the application at which the user will be able to provide their credentials along with their preferred currency type. Digi Dhan allows for seamless tracking of income and expenses as well as highlighting the debts incurred. The user can add the amount of income/expenses, include the date of transaction and additionally tag transactions with labels as well as descriptions. Similarly, the users can access detailed statistics such as total income, expense, debt as well as search the transactions and sort them out by titles, transaction type, date range and tags. Additionally, the user can also create their own custom tags.

In this report, we will go through the user interface design of the application along with the ERD. This report also provides the details regarding the technology stack such as framework, libraries used, persistence mechanism etc. Similarly, this report entails all the working of the developed application. Various test cases are also conducted for the developed application to ensure the proper functionality. Additionally, the flowchart, use case diagram and classes along with their methods and properties are given.

## 1.1 Aims and Objectives:

The major aim of this project is to ensure the smooth tracking of an individual’s financial transactions. Some of the other objectives of the project are given below:

* To ensure the proper inflows, outflows and debt tracking.
* To handle exceptions and check for sufficient balance before making expenses.
* To search for transactions based on titles along with sorting and filtering.
* To properly list and highlight pending debts.
* To display the transactions performed along with the balance, highest transactions performed, lowest transaction etc.
* Have a complete understanding of the .NET framework and database medium used.
* To ensure version control during the development of the application.

# Feature and functionalities:

## 2.1 Prototype:

The prototype created for each page of the application is given below:

### 2.1.1 Prototype of Login page:

The UI for login page of Digi Dhan is given below:



Figure 1: Prototype for login page.

### 2.1.2 Prototype of Home page:

The UI for home page of Digi Dhan is given below:

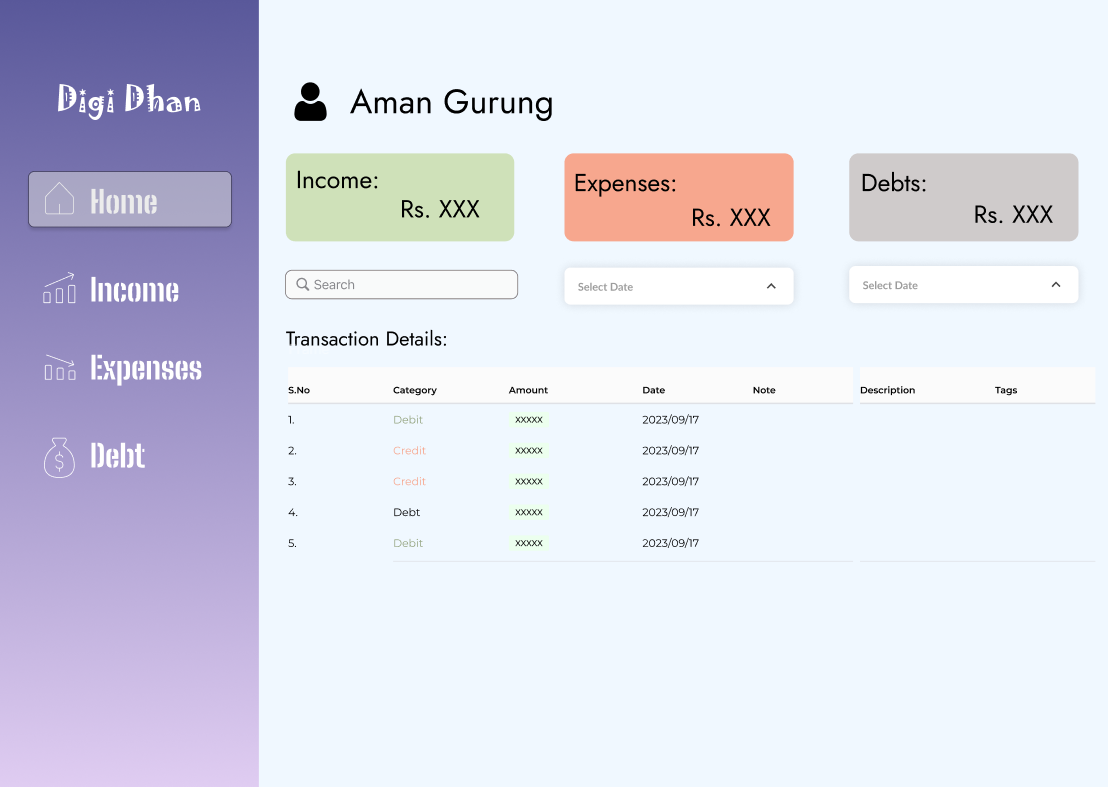


Figure 2: Prototype for Home page

### 2.1.3 Prototype of Income page:

The UI for income page of Digi Dhan is given below:

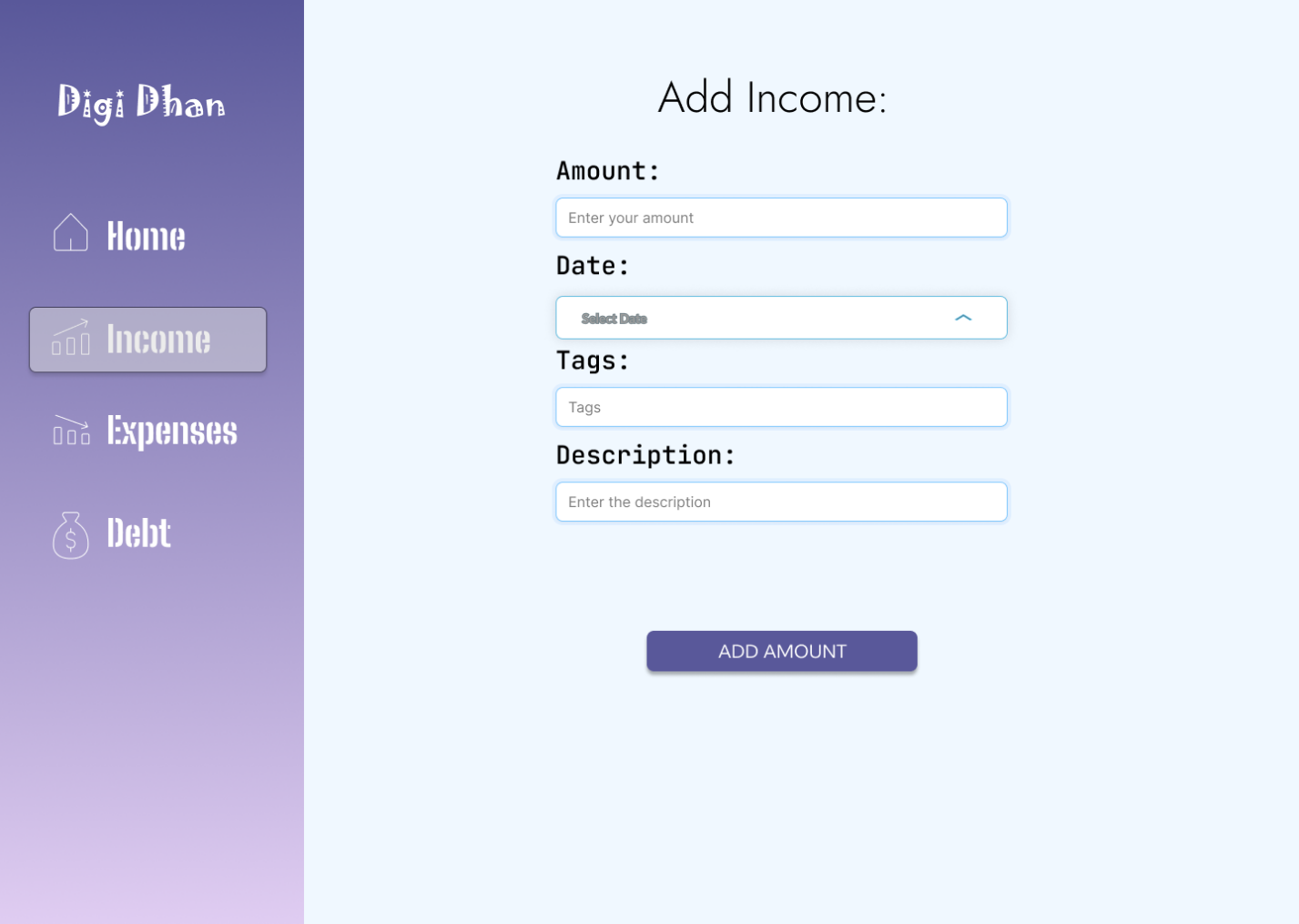


Figure 3: Prototype for income page

### 2.1.4 Prototype of Expenses page:

The UI for expenses page of Digi Dhan is given below:

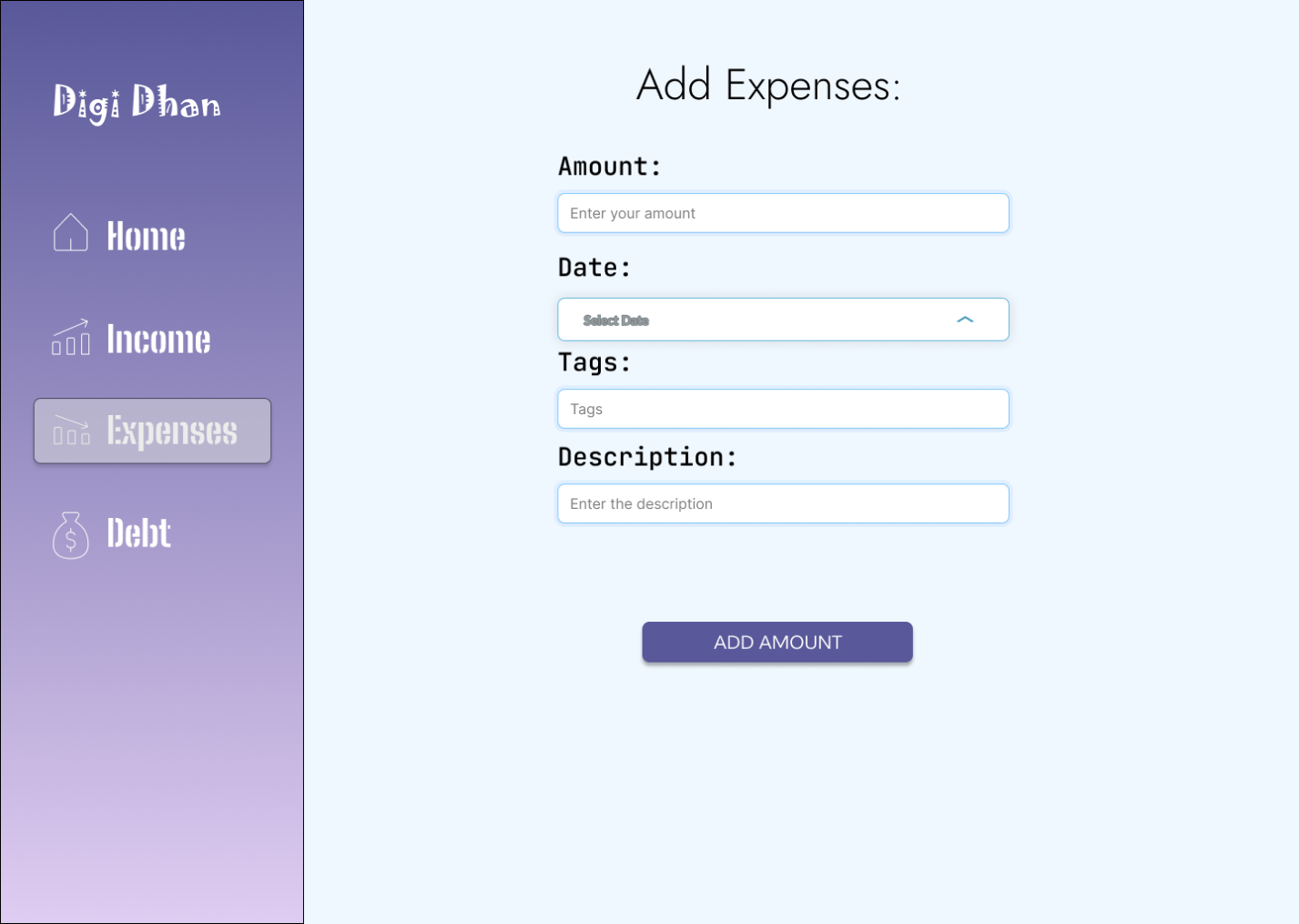


Figure 4: Prototype for expenses page

### 2.1.5 Prototype of Debt page:

The UI for debt page of Digi Dhan is given below:

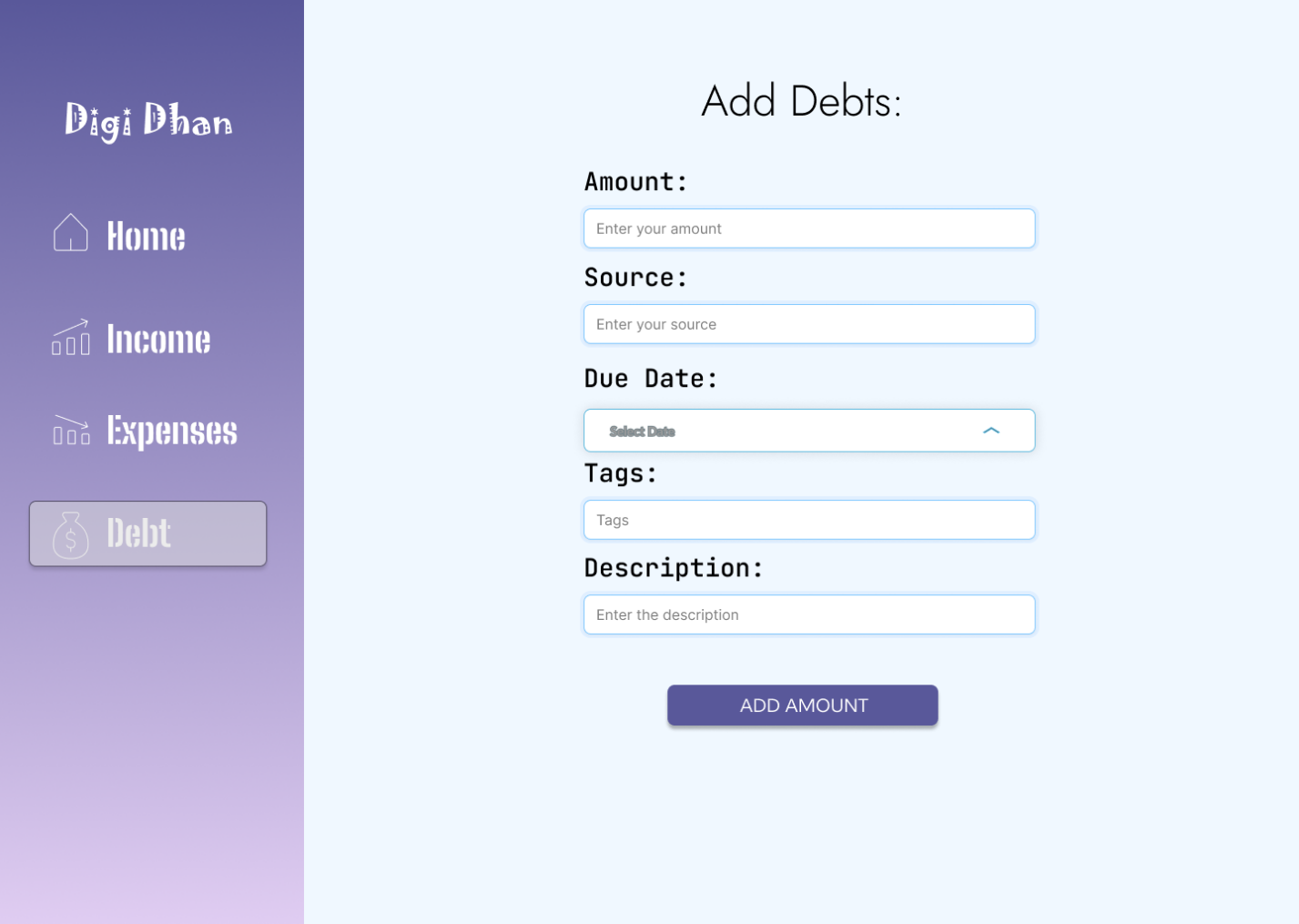


Figure 5: Prototype for Debt page.

## 2.2 Developed Application:

The developed application ‘DigiDhan’ consists of a total of 6 pages namely: Login page, Home, Add Income, Add Expenses, Add Debt and Create Tag page. The description for each page along with their workings is given below:

### 2.2.1 Login page:

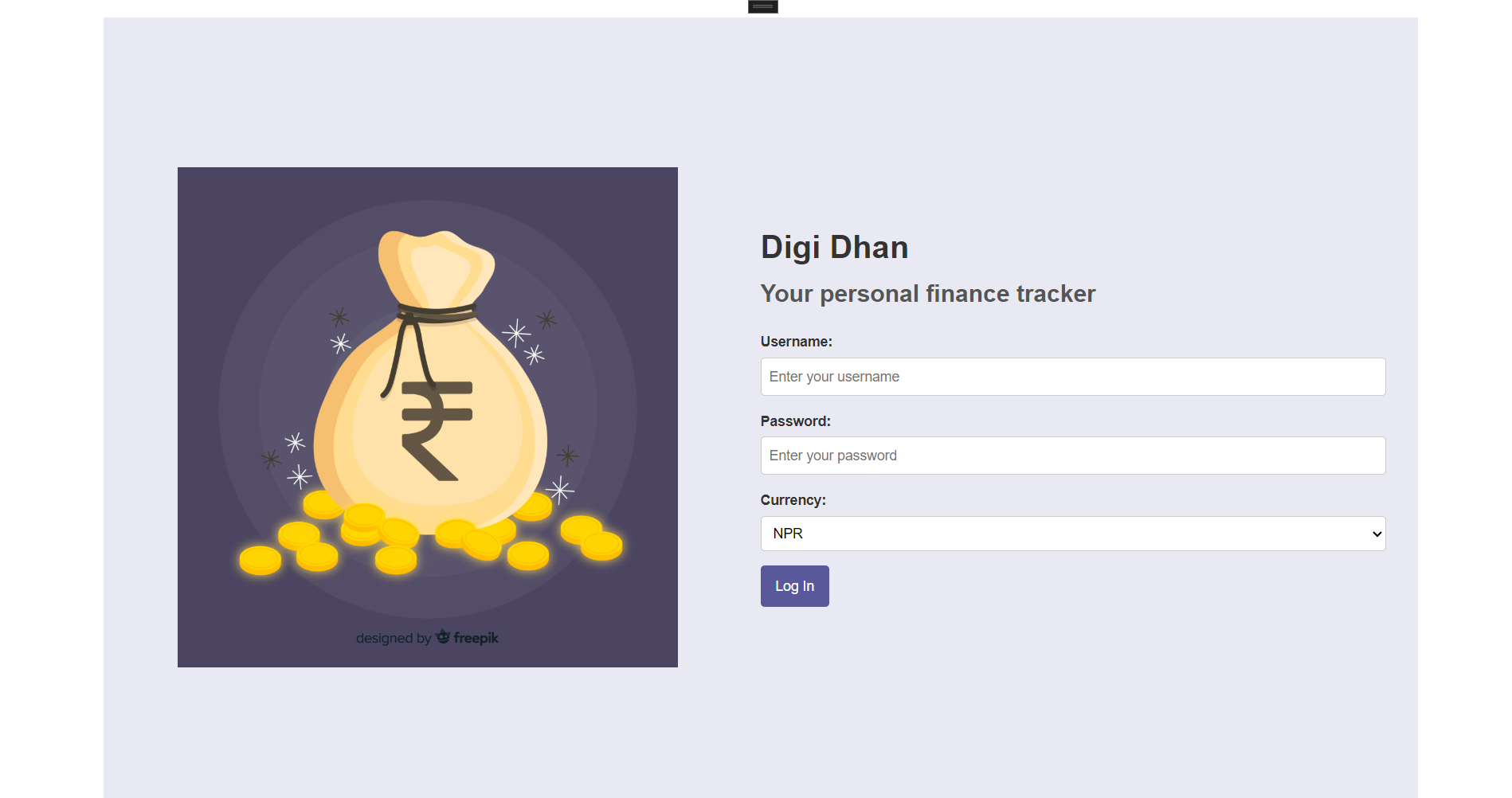


Figure 6: Developed Login page.

This is the first page that is loaded at the start of the application. From this page, the user can log in to the application using the correct credentials. Upon successful login, the user is redirected to the home page.

### 2.2.2 Home page:

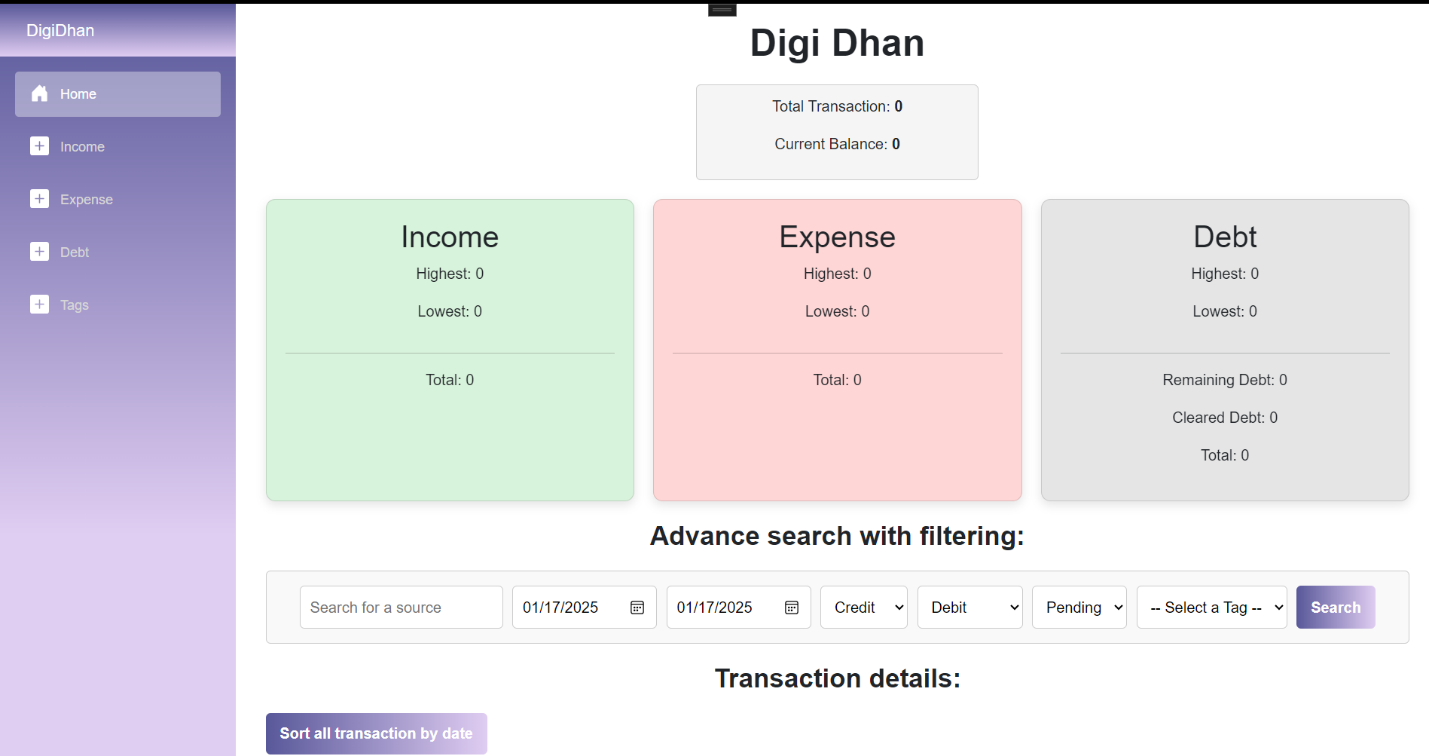


Figure 7: Developed Home page.

Upon successful login, the user lands on the home page. The home consists of a number of features and components. All the performed transactions are shown in a table in the home page. Sorting of the transactions by date, searching for transactions based on specific date range, searching for transactions by title and filtering can also be done. Similarly, the current user balance along with the highest, lowest and total amount of income, expense and debt is also displayed on home page. The pending and cleared debts are also highlighted in home page.

### 2.2.3 Income page:

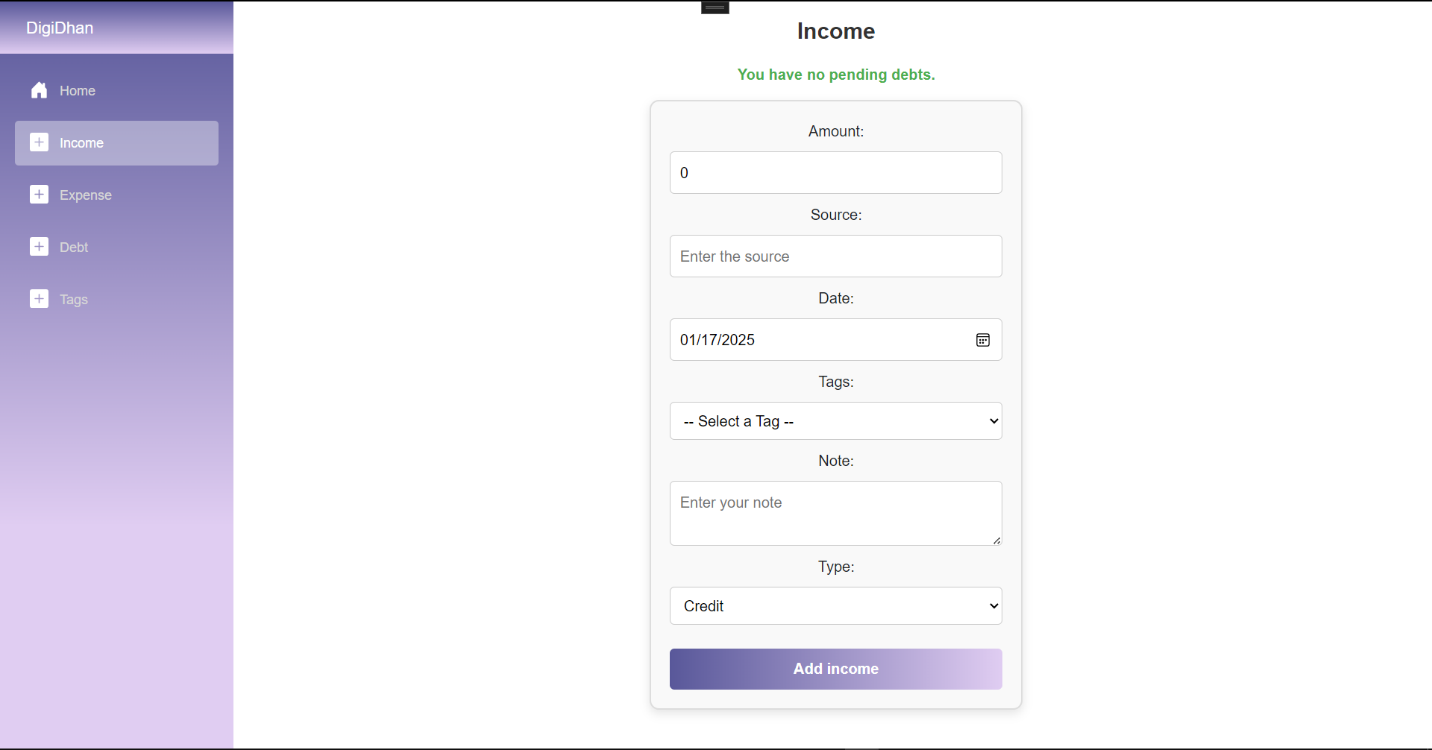


Figure 8: Developed Income page.

Upon clicking on income on the side navbar, the user is directed to the add income page. The users can add the income from this page providing various details such as amount, source, date, tags, note and type. Before adding income, pending debts are checked. Incase of pending debt, the added income is used to clear the pending debt.

### 2.2.4 Expense page:

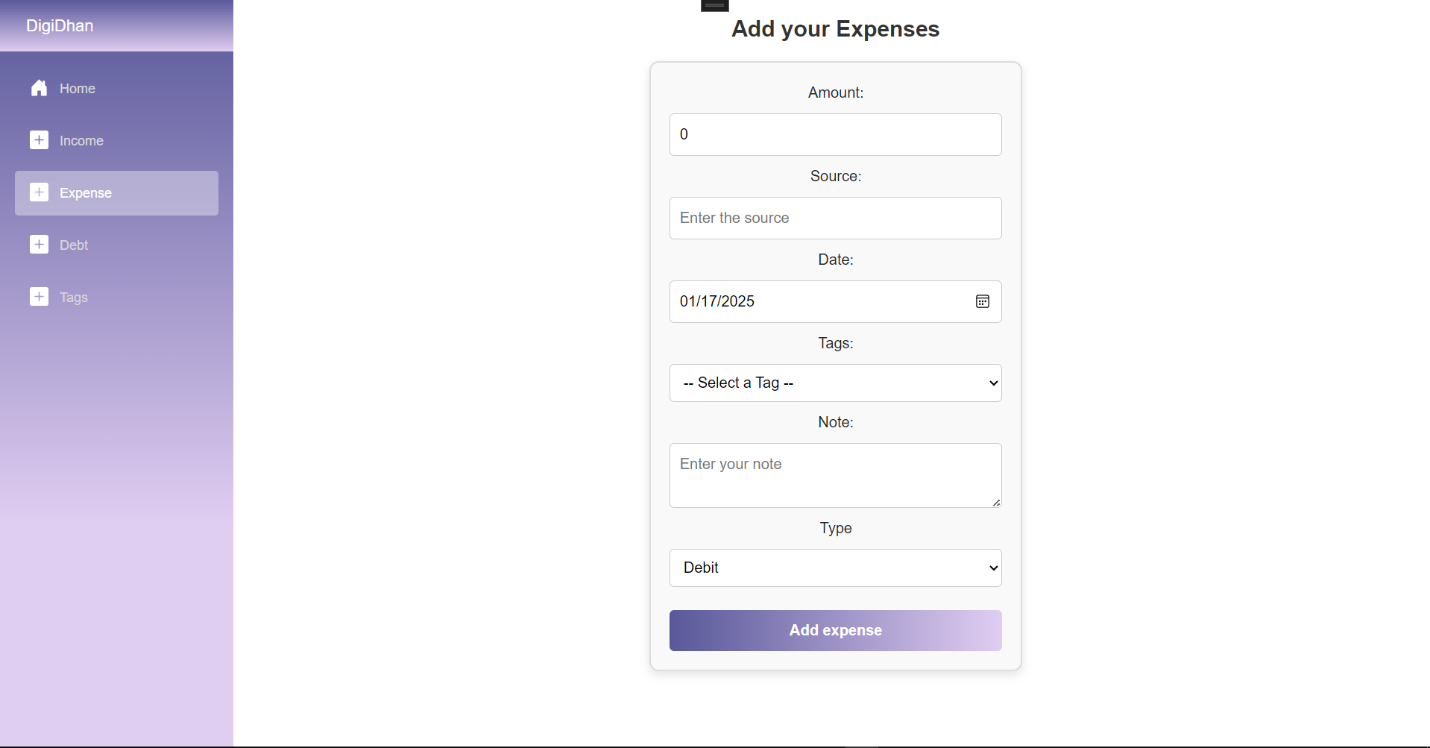


Figure 9: Developed Expense page

The expense page is used to add any expenses made by the user. The page contains input field for amount, source, date, tags, note and type. The expenses made is deducted form the balance of user. Incase of insufficient balance, an appropriate message is shown.

### 2.2.5 Debt page:

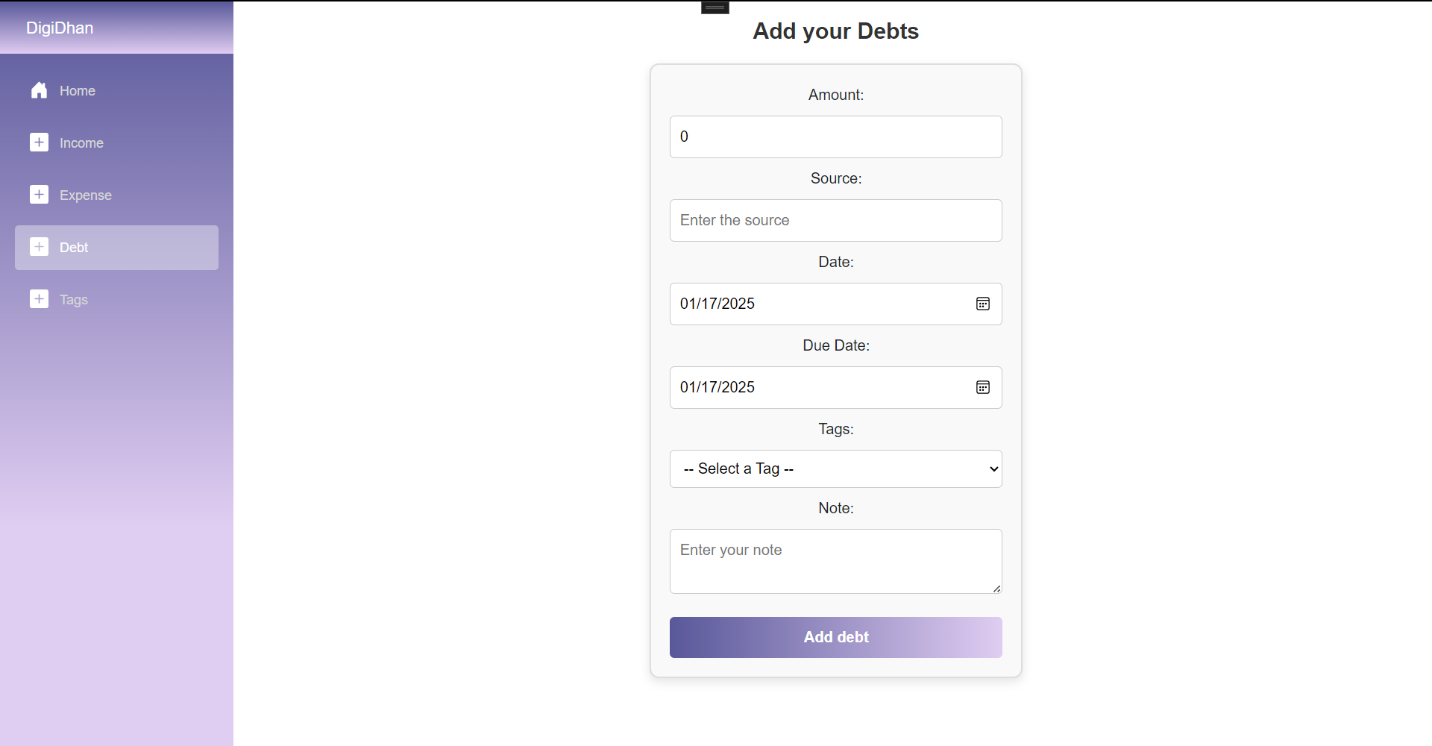


Figure 10: Developed Debt page.

The debt page is used to handle the debt transactions. The debt page contains input field for amount, source, date, due date, tags and note. The pending debts is added to the current balance of the user. The pending debts are cleared from income.

### 2.2.6 Tags page:

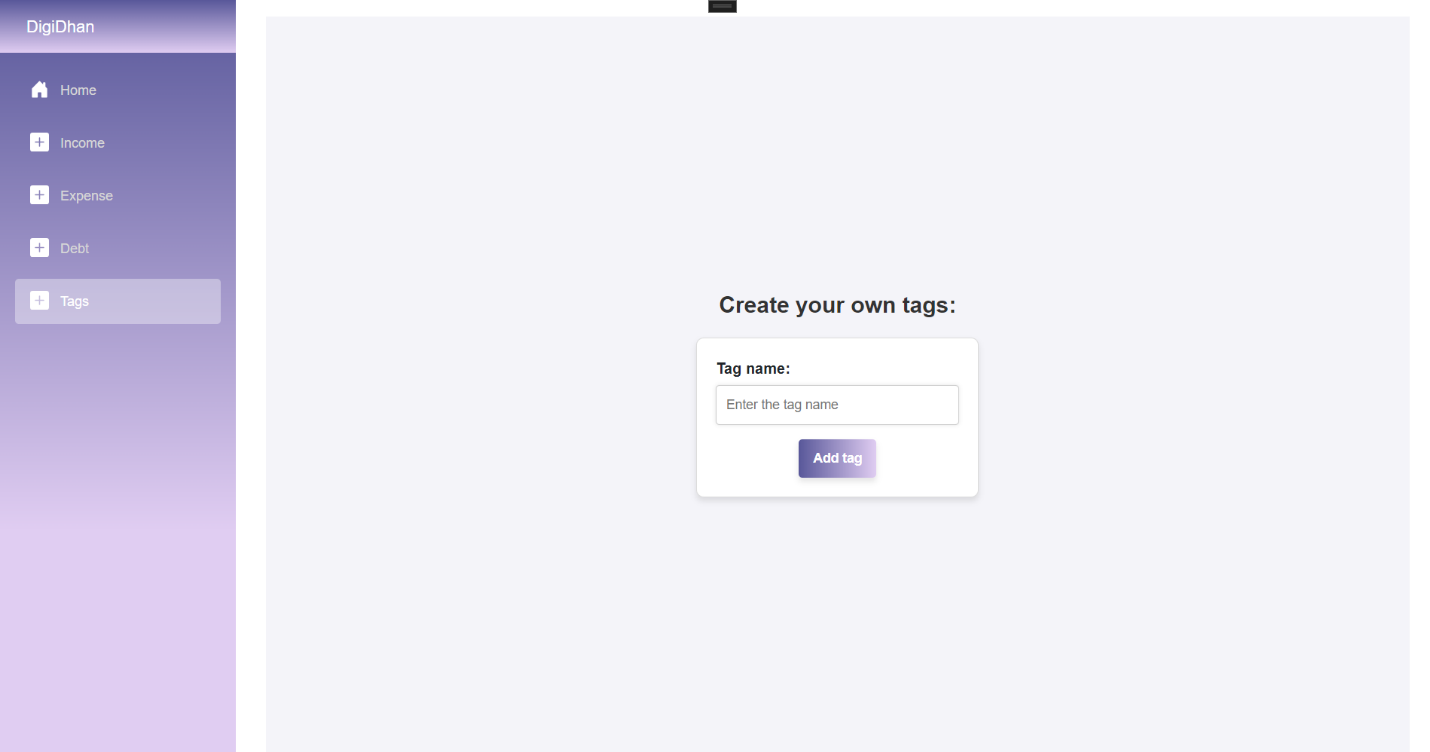


Figure 11: Developed tags page.

The tag page is used to create custom tags. It takes only one input tag name.

## 2.3 Test cases:

The test cases for each feature and functionalities of the developed application Digi Dhan are given below:

### 2.3.1 Test 1: Logging in

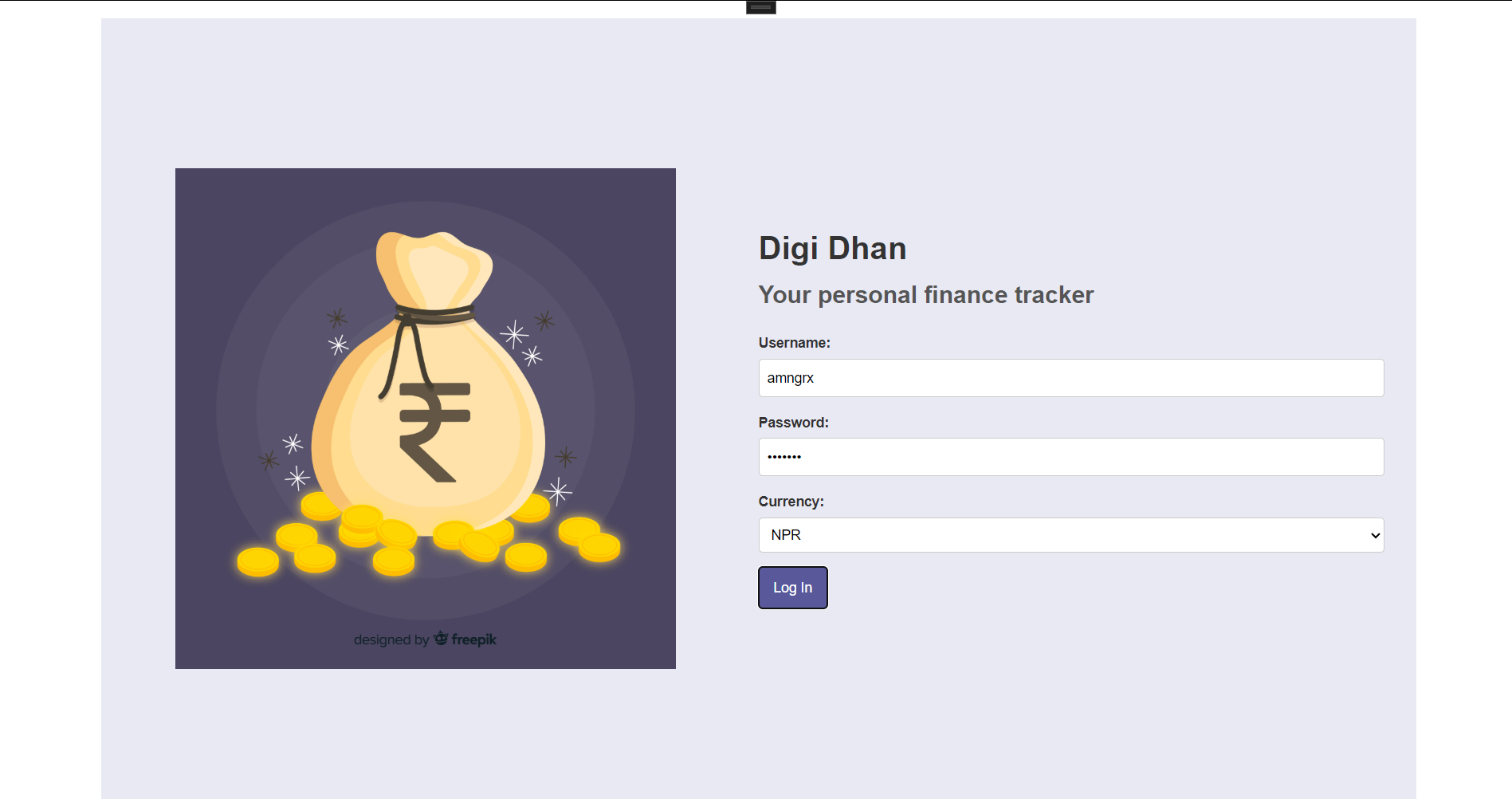


Figure 12: Providing correct credentials.

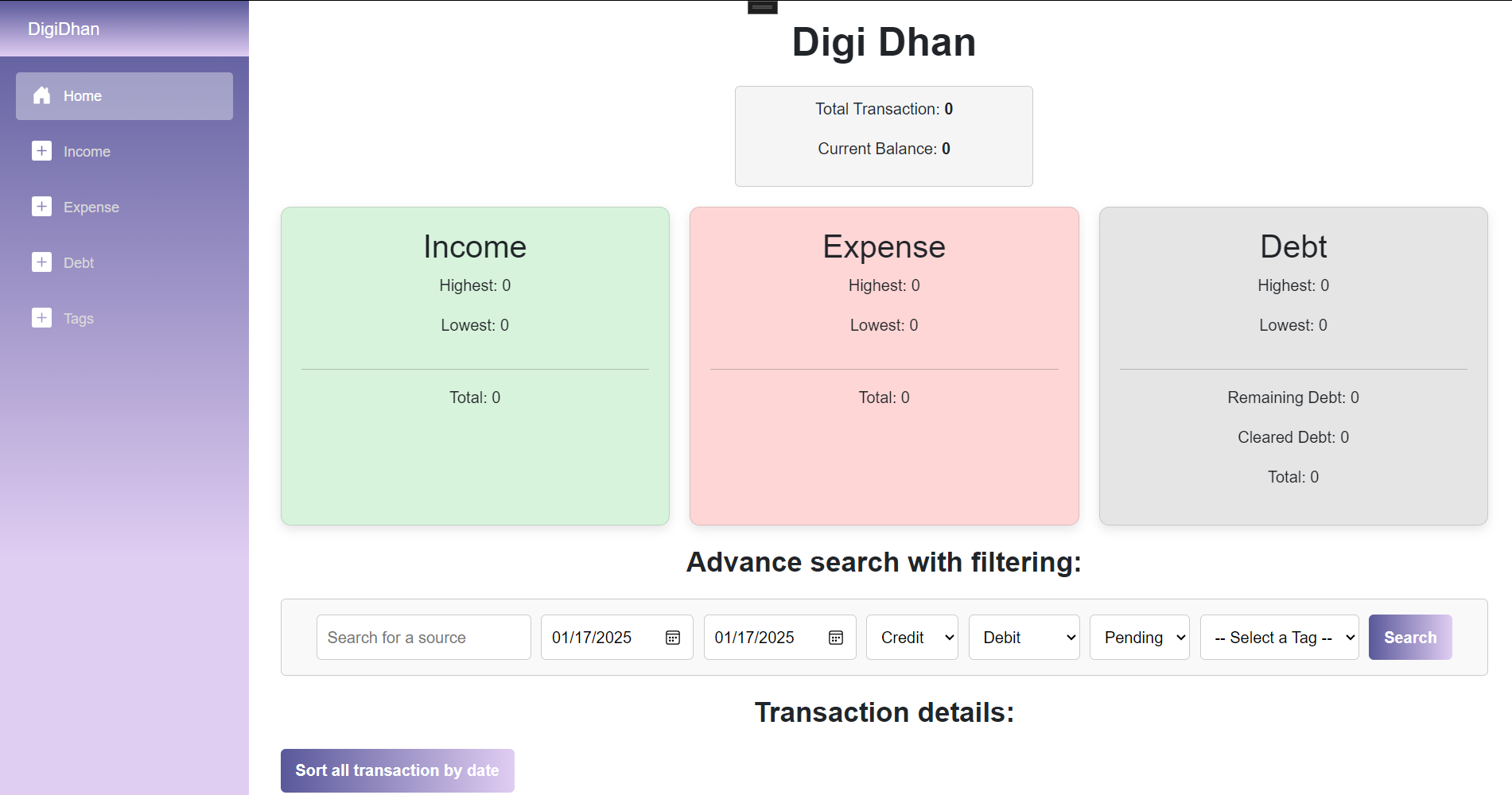


Figure 13: Redirection to home page after successful login.

Table 1: Successful login.

|  |  |
| --- | --- |
| Objective | To ensure the log in of the user. |
| Action | The user inputs the correct credentials during login. |
| Expected Outcome | The user should be redirected to home page. |
| Actual Result | The user is redirected to home page. |
| Conclusion | Test successful |

### 2.3.2 Test 2: Input wrong credentials during login:

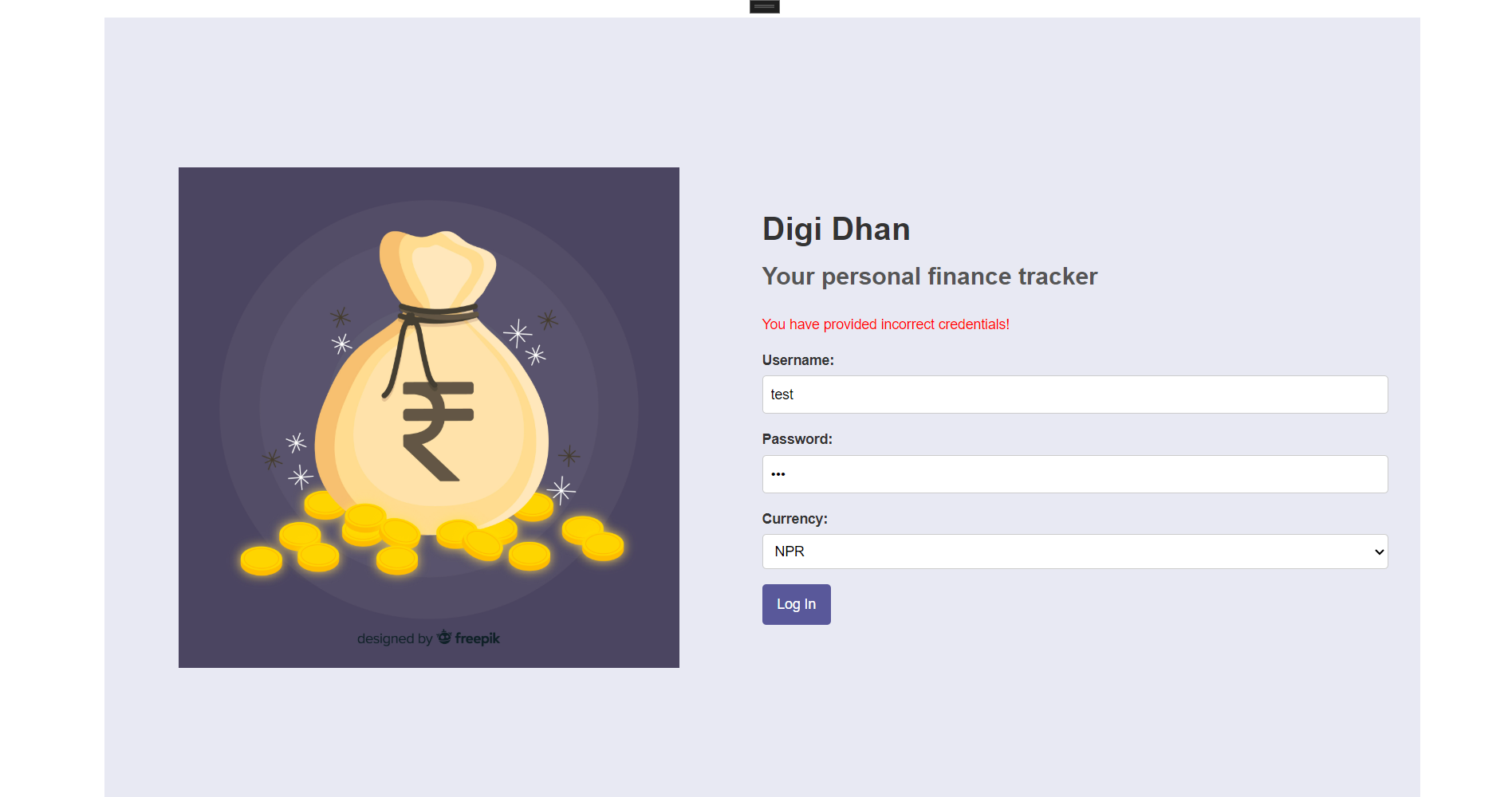


Figure 14: Message to show invalid credentials.

Table 2: Appropriate message shown when invalid credentials.

|  |  |
| --- | --- |
| Objective | To display appropriate message during invalid input credentials. |
| Action | The user inputs wrong credentials. |
| Expected Outcome | An appropriate message should be shown. |
| Actual Result | An appropriate message is shown. |
| Conclusion | Test successful |

### 2.3.3 Test 3: Adding Income

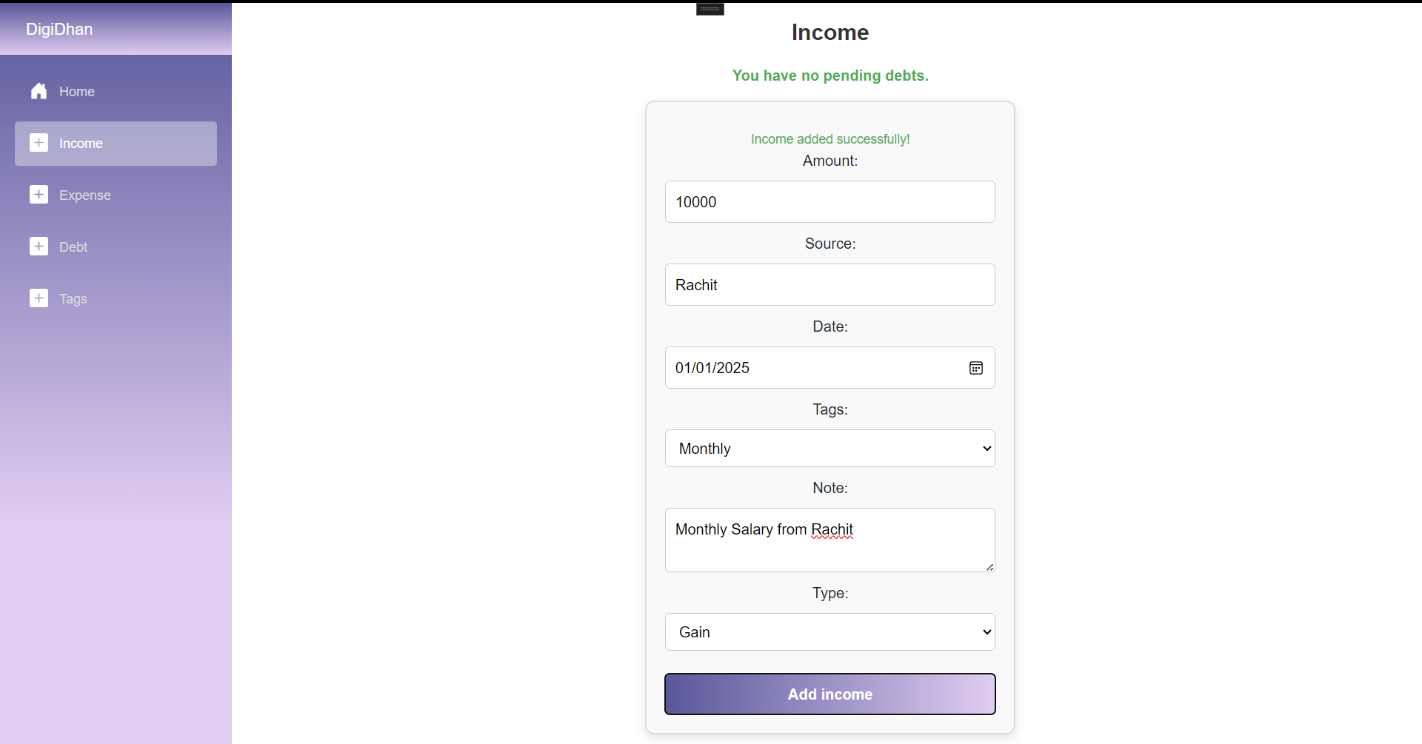


Figure 15: Adding Income.

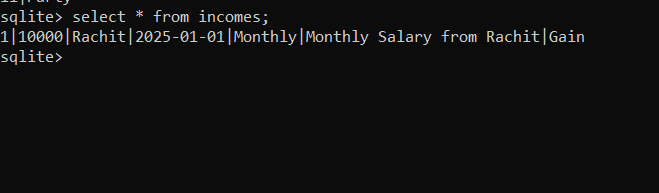


Figure 16: Income added to incomes table.

|  |  |
| --- | --- |
| Objective | To add income in the income table. |
| Action | The user inputs the required fields and adds the income. |
| Expected Outcome | The income transaction should be added to the income table. |
| Actual Result | The income transaction is added to the income table. |
| Conclusion | Test successful |

### 2.3.4 Test 4: Adding Expenses

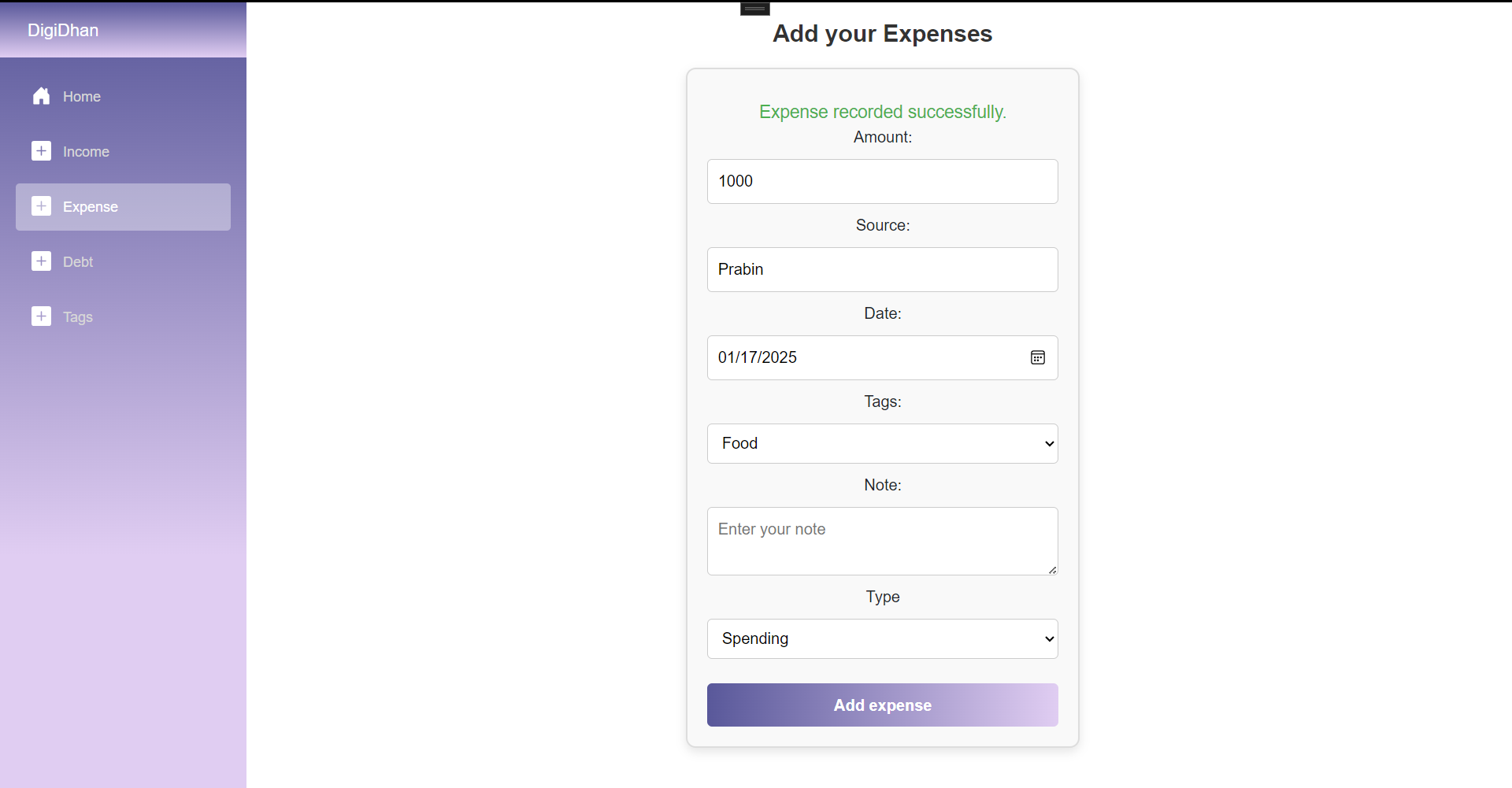


Figure 17: Adding Expense transaction

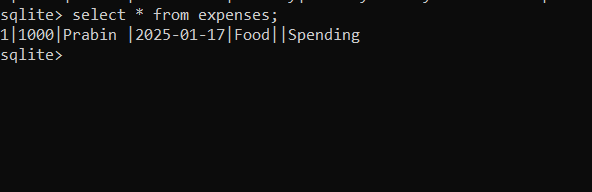


Figure 18: Expense transaction added to table.

|  |  |
| --- | --- |
| Objective | To add the expenses made by the user. |
| Action | The user inputs the required fields and adds the expenses. |
| Expected Outcome | The expense should be added to the expenses table. |
| Actual Result | The expense is added to the expenses table. |
| Conclusion | Test successful |

### 2.3.5 Test 5: Adding excessive expense

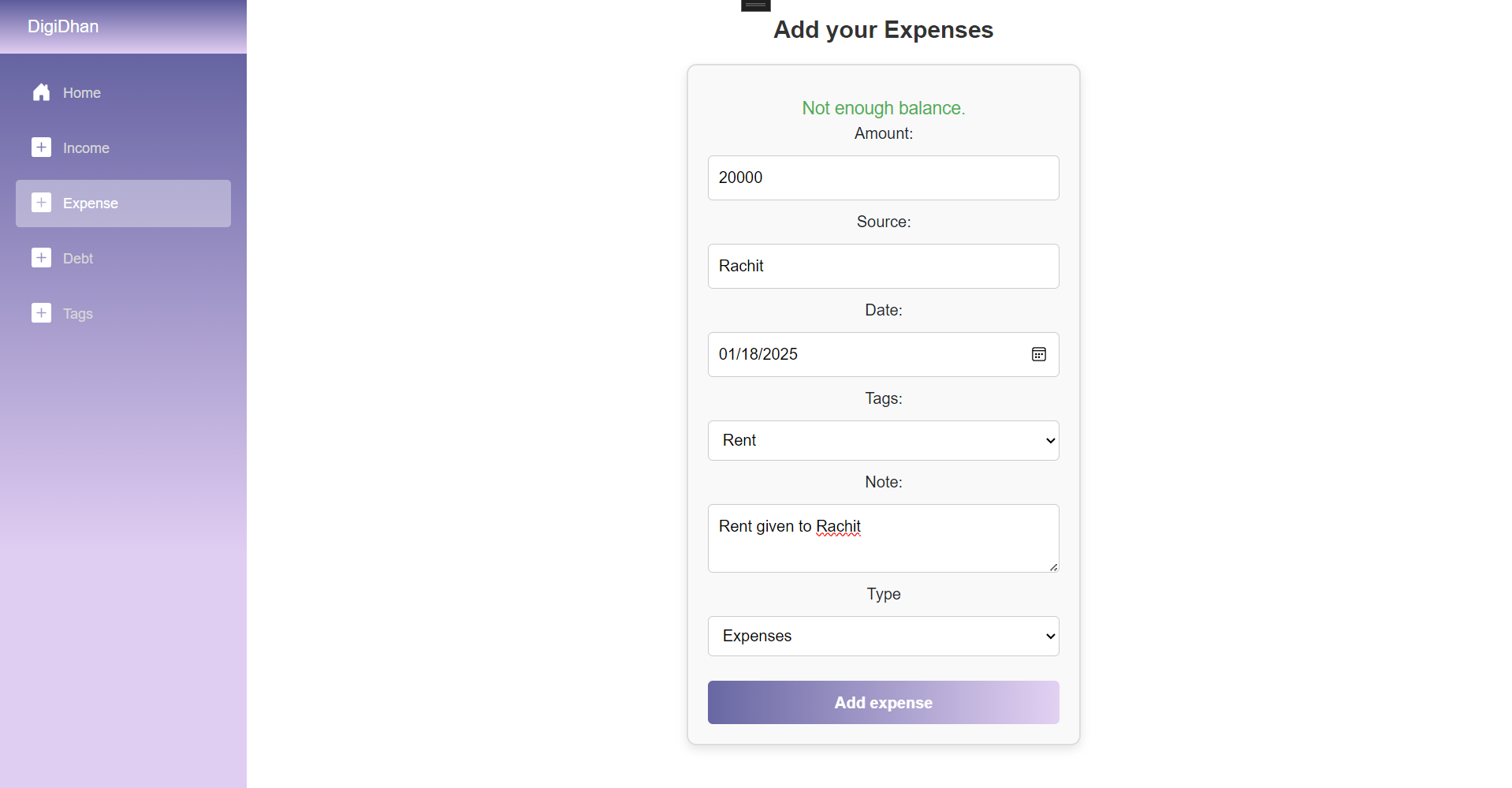


Figure 19: Adding excessive expense.

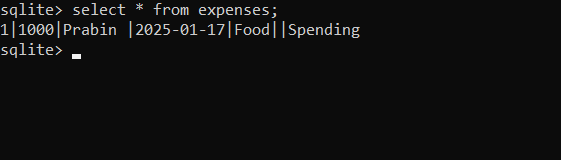


Figure 20: Expense is not added to the table due to insufficient balance.

|  |  |
| --- | --- |
| Objective | To display appropriate message, in case of insufficient balance. |
| Action | The user should input expense greater than balance. |
| Expected Outcome | The message should be displayed and the transaction should not be recorded. |
| Actual Result | The message is displayed and the transaction is not recorded. |
| Conclusion | Test successful |

### 2.3.6 Test 6: Adding Debt

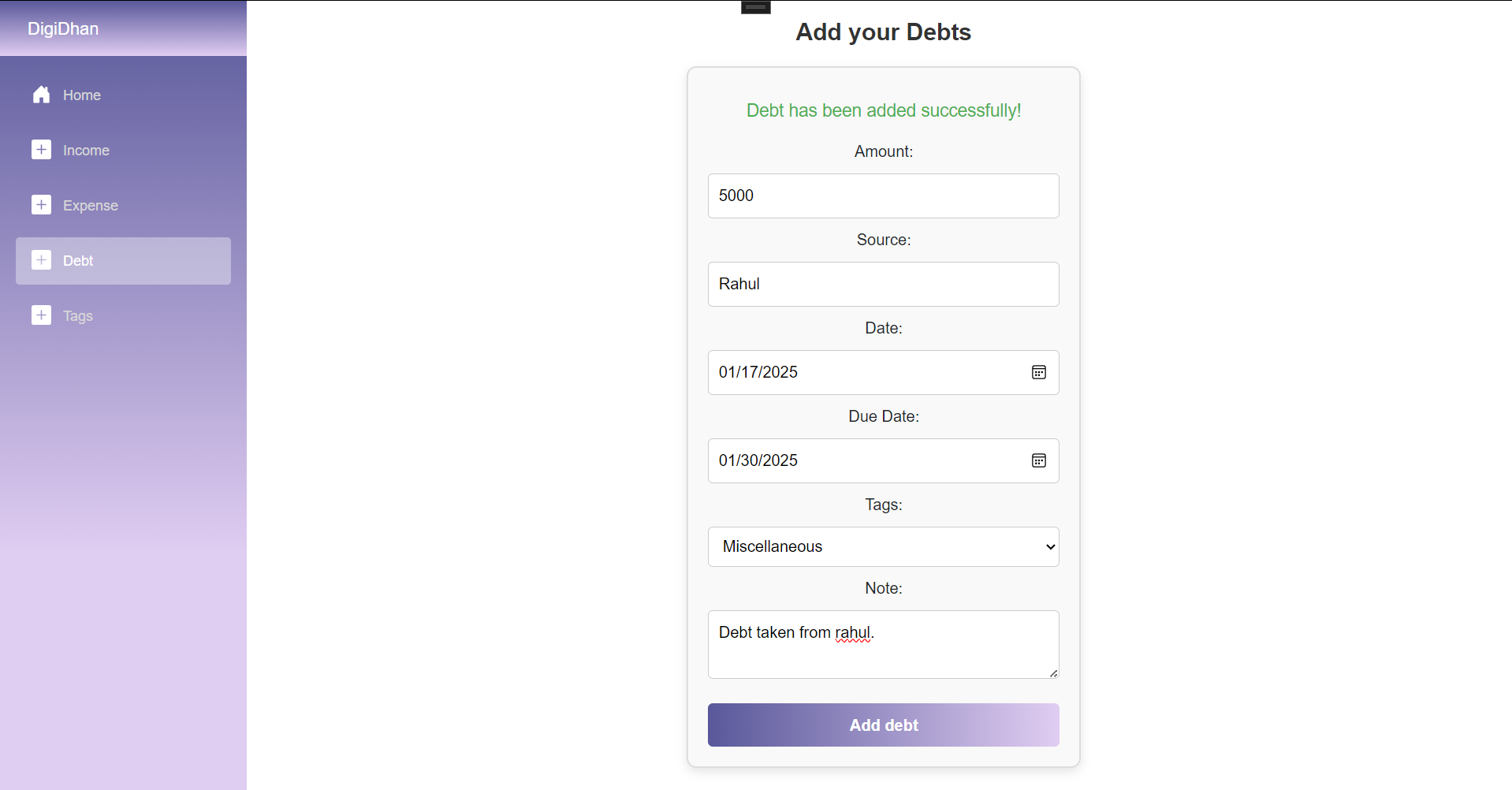


Figure 21: Adding debt.

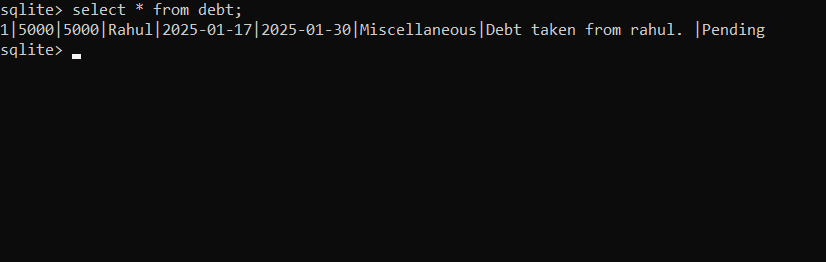


Figure 22: Debt is added to the table.

|  |  |
| --- | --- |
| Objective | To add the debt transaction. |
| Action | The user inputs required field. |
| Expected Outcome | The transaction should be added to the respective table. |
| Actual Result | The transaction is added to the respective table. |
| Conclusion | Test successful |

### 2.3.7 Test 7: Adding Custom Tags:

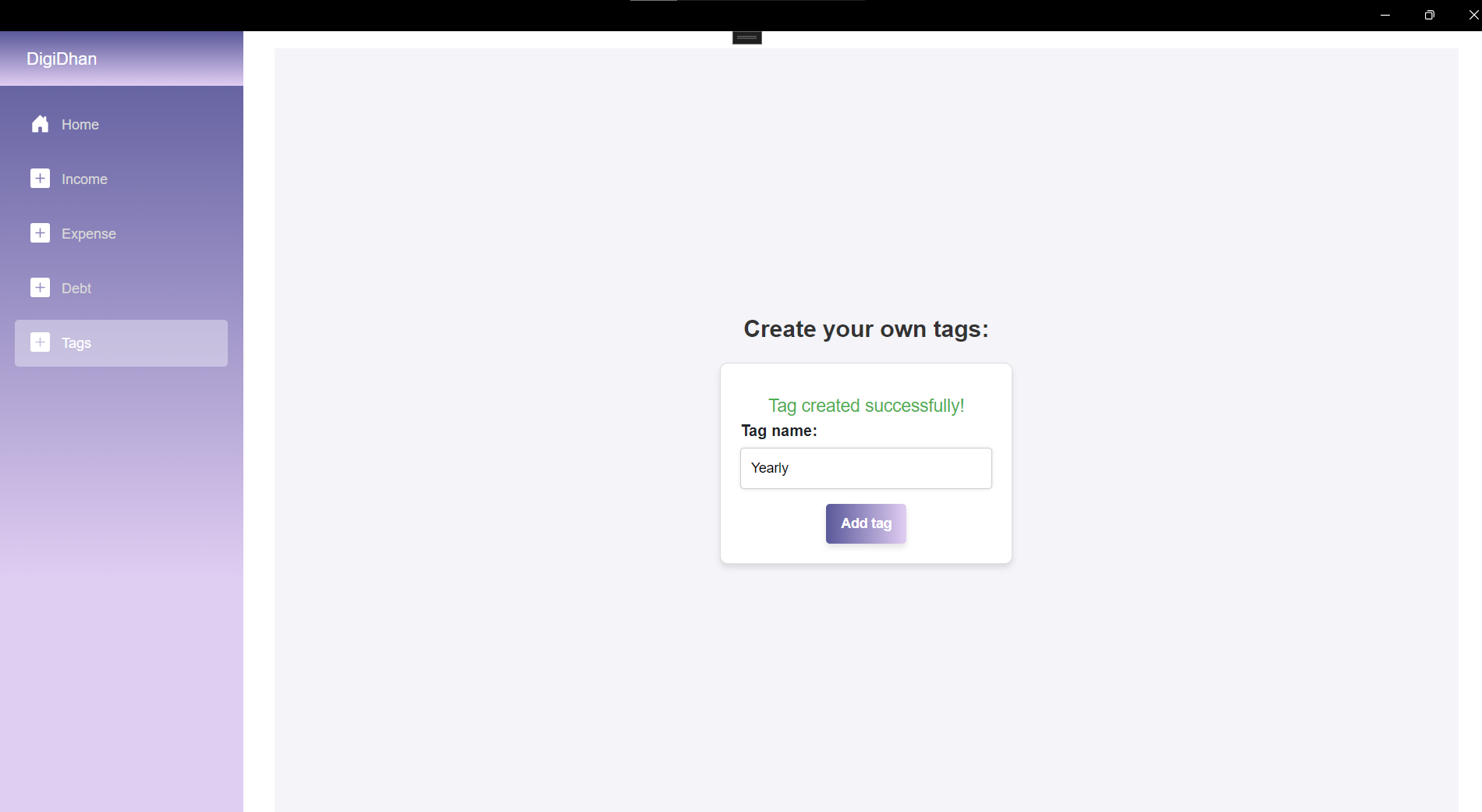


Figure 23: Adding custom tag.

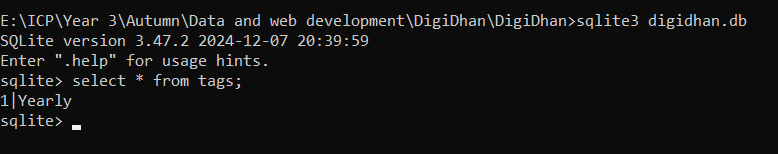


Figure 24: Custom tag is added

|  |  |
| --- | --- |
| Objective | To create a custom tag. |
| Action | The user inputs the tag name. |
| Expected Outcome | The tag name should be added to the tag table. |
| Actual Result | The tag name is added to tag table. |
| Conclusion | Test successful. |

### 2.3.8 Test 8: Clearing pending debt:

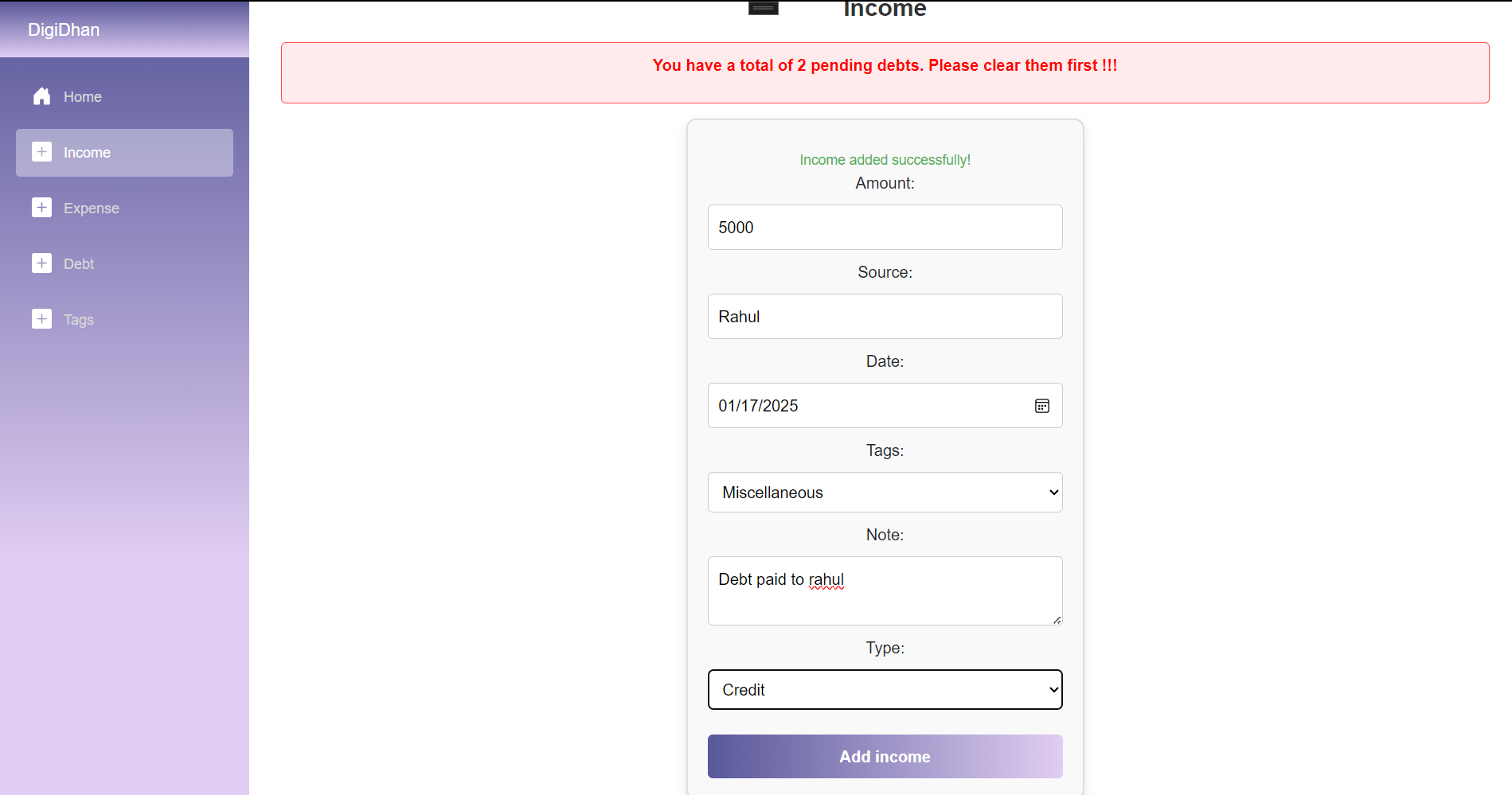


Figure 25: Income added to clear debt.

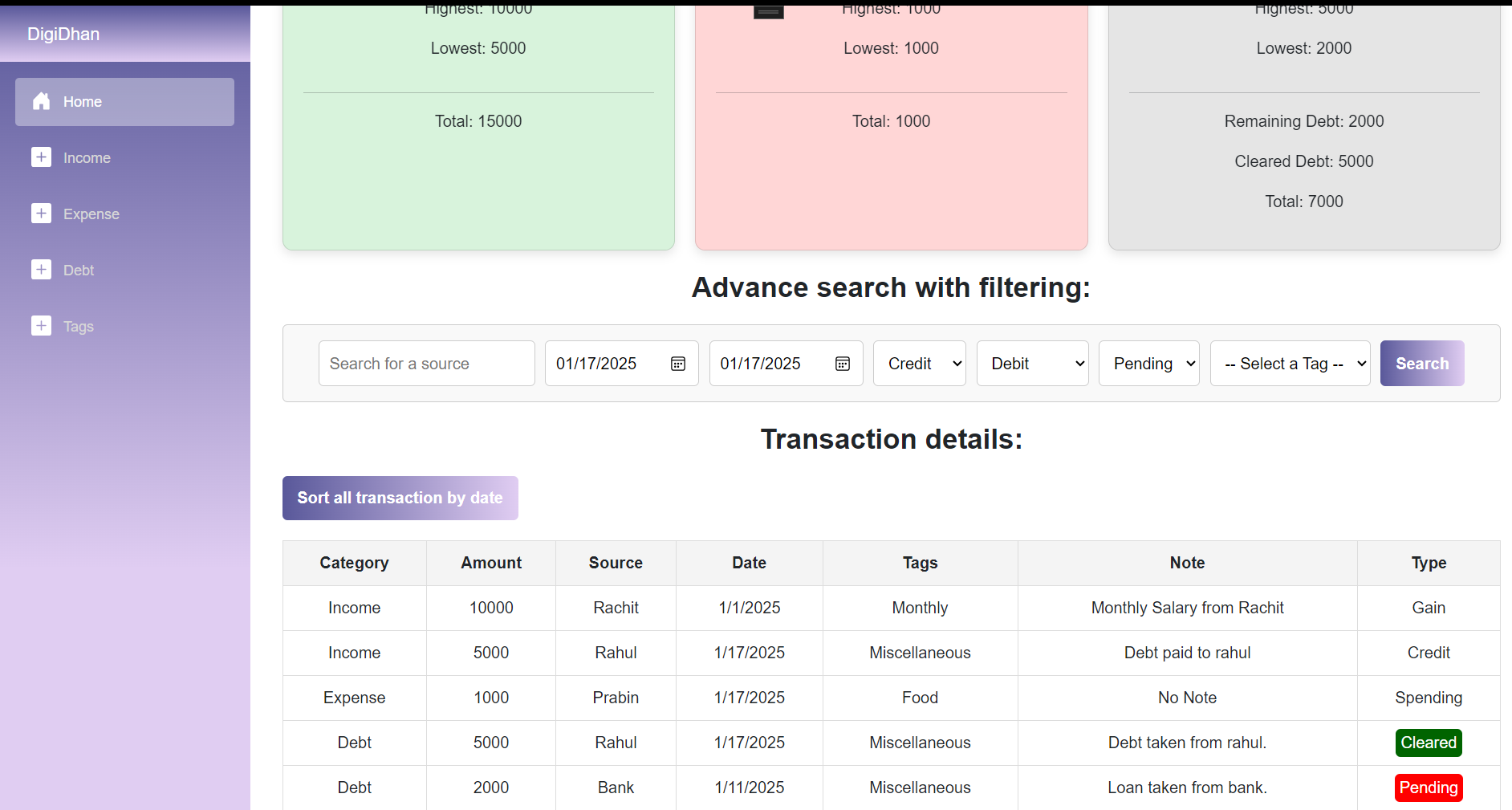


Figure 26: Cleared Debt.

|  |  |
| --- | --- |
| Objective | To clear the pending debt from income. |
| Action | The user inputs the debt amount to be cleared. |
| Expected Outcome | The debt should be cleared. |
| Actual Result | The debt is cleared. |
| Conclusion | Test successful |

### 2.3.9 Test 9: Get the total transaction and current balance

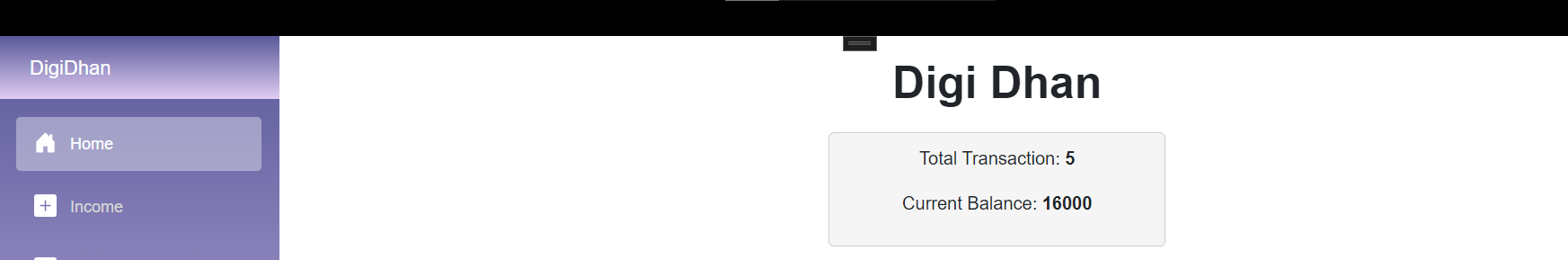


Figure 27: Displaying total number of transaction and balance.

|  |  |
| --- | --- |
| Objective | To display the total transaction and current balance. |
| Action | The user clicks on the home button in navbar. |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

### 2.3.10 Test 10: Get the highest, lowest and total of income, expense and debt

|  |  |
| --- | --- |
| Objective | To |
| Action | The user |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

### 2.3.11 Test 11: Display the remaining and cleared debt

|  |  |
| --- | --- |
| Objective | To |
| Action | The user |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

### 2.3.12 Test 12: Displaying the transactions:

|  |  |
| --- | --- |
| Objective | To |
| Action | The user |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

### 2.3.13 Test 13: Highlighting the debts pending and cleared.

|  |  |
| --- | --- |
| Objective | To |
| Action | The user |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

### 2.3.14 Test 14: Sort all transaction by date

|  |  |
| --- | --- |
| Objective | To |
| Action | The user |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

### 2.3.15 Test 15: Search transaction by title and adding filters

|  |  |
| --- | --- |
| Objective | To |
| Action | The user |
| Expected Outcome | The |
| Actual Result | The |
| Conclusion | Test successful |

# Development and Logical solution:

In this section, the logical solution for the application such as Entity Relationship Diagram, flowchart, use case as well as class, properties and their method Is given.

## 3.1 Entity Relationship Diagram:

The journal states, “The Entity-Relationship diagram has been widely used in structured analysis and conceptual modeling.”. It is implied to be easy to understand, powerful to model real-world problems and readily translated into a database schema. Similarly, the researcher states, “The ERD views that the real world consists of a collection of business entities, the relationships between them and the attributes used to describe them” (Il-Yeol Song, 1995).

The entity relationship diagram for my project Digi Dhan is given below:

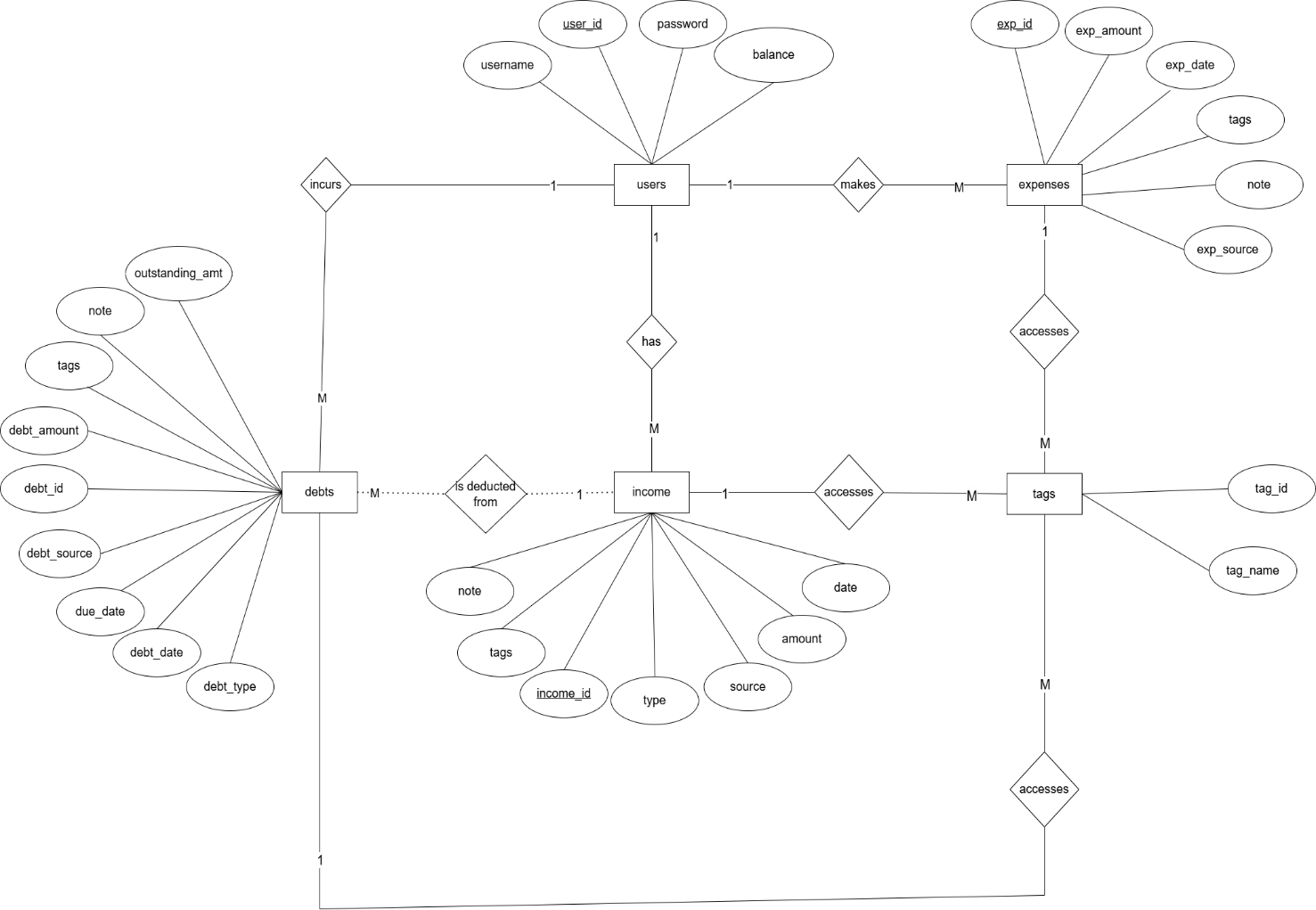


Figure 28: Entity relationship diagram of Digi Dhan

## 3.2 Use Case Diagram:

The use case diagram for my application Digi Dhan is given below:

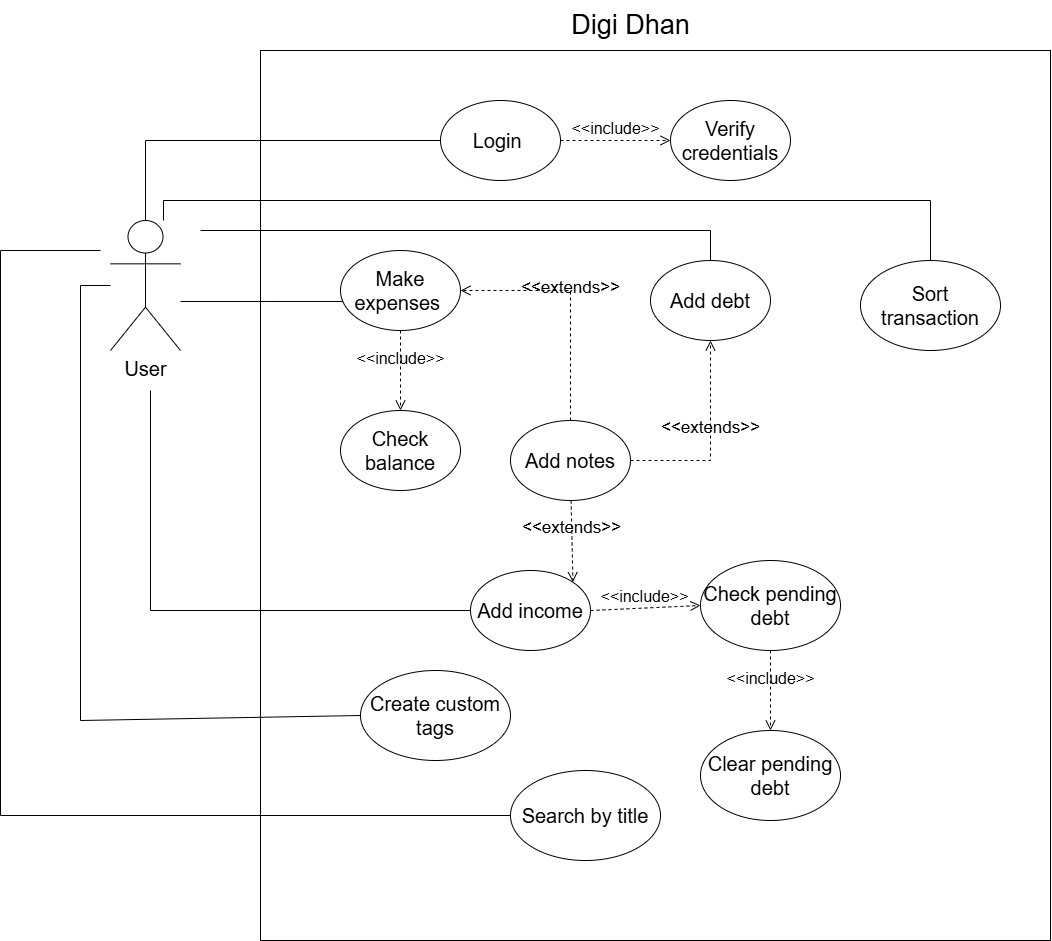


Figure 29: Use case diagram of Digi Dhan

## 3.3 Flowchart:

The flowchart showing the workflow of my project Digi Dhan is given below:

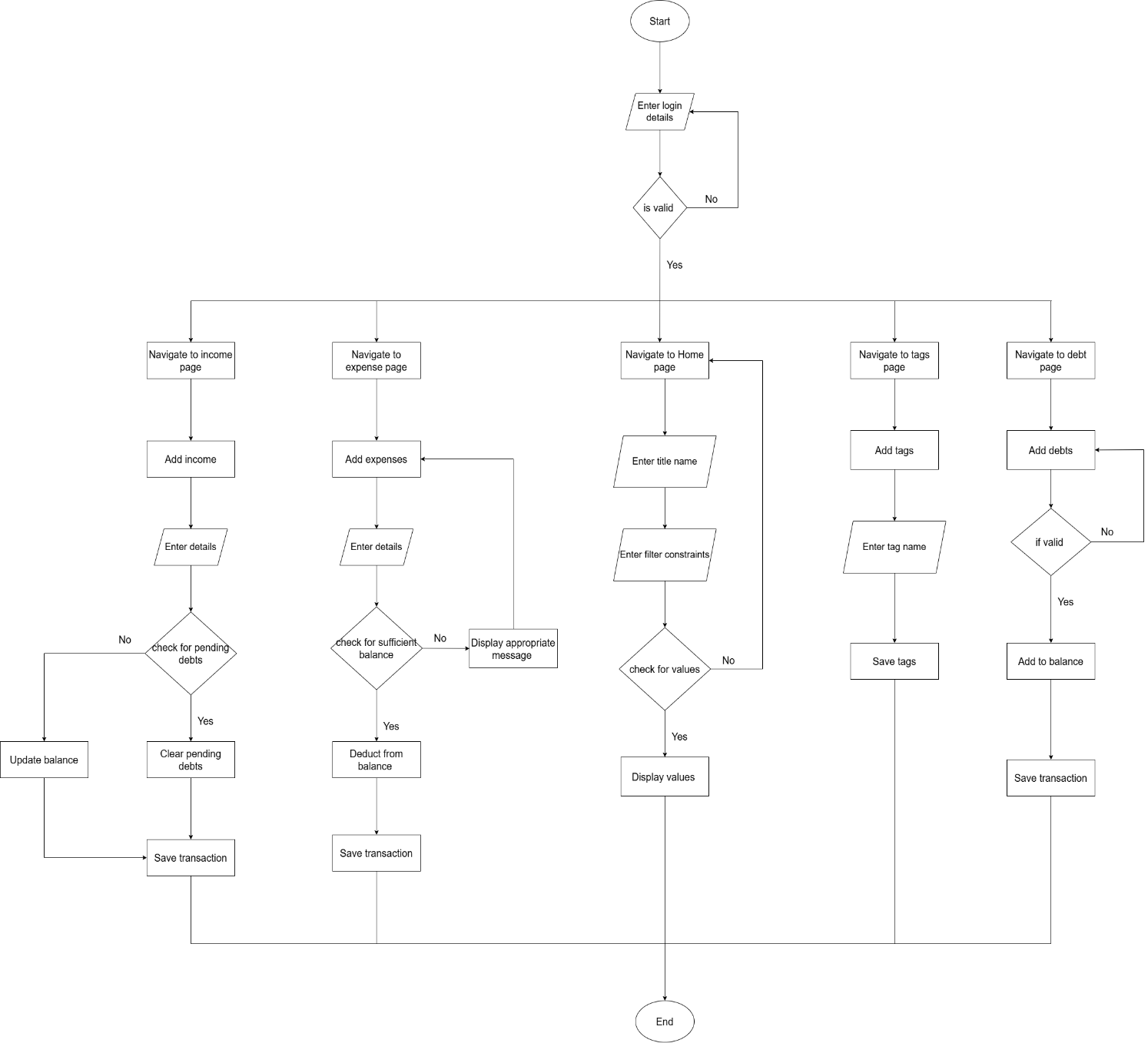


Figure 30: Flowchart of Digi Dhan

## 3.4 Version Control (Git):

The version control for the project is done with the help of GitHub. A private repository with the name of the application is created in GitHub.

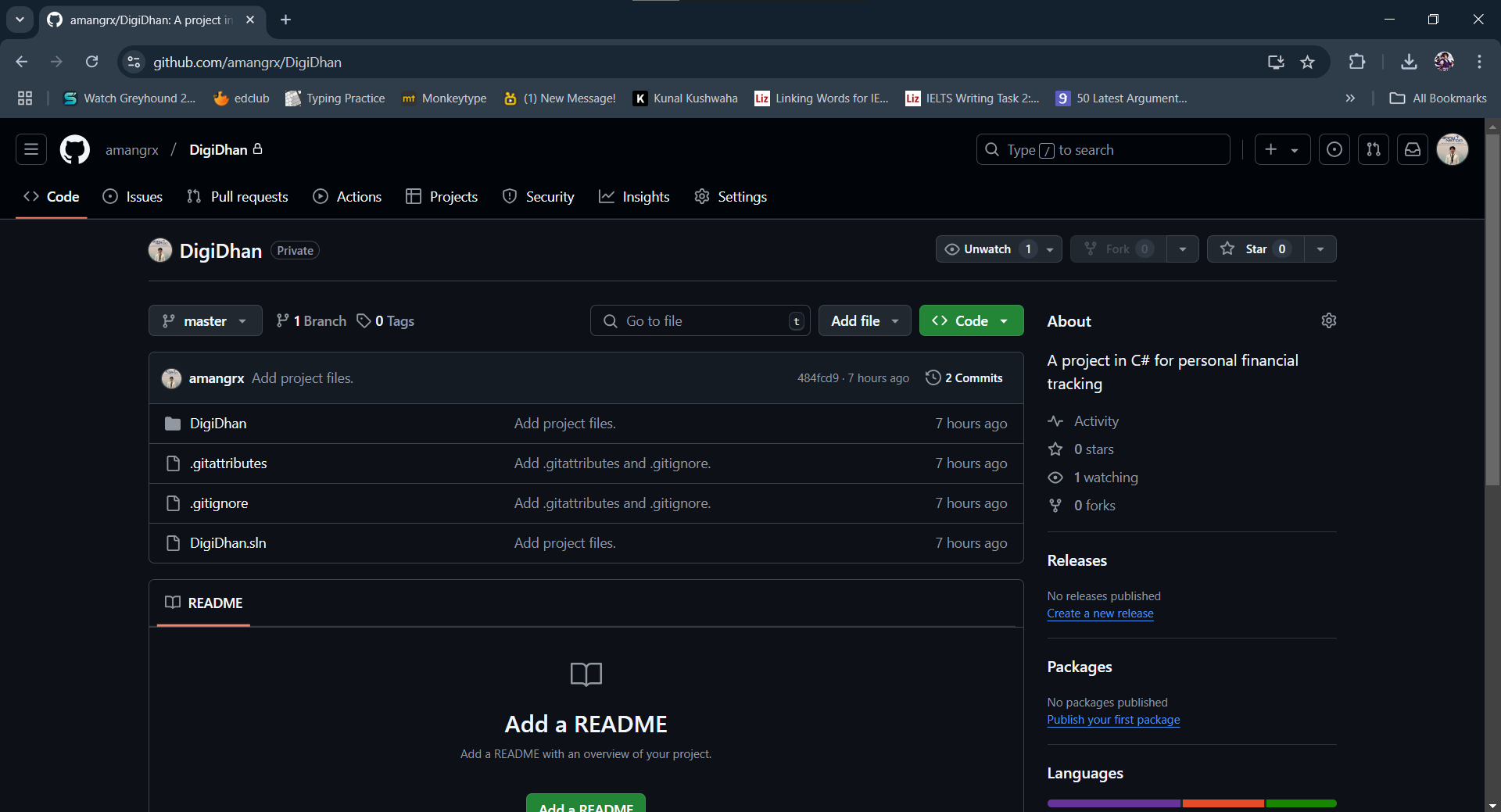


Figure 31: Version Control in Git.

Similarly, the screenshot for the commits of the repository is given below:

## 3.5 Technology stack:

The set of technologies to use in the development of Digi Dhan are as follows:

### 3.5.1 Framework:

Digi Dhan is created using MAUI Blazor Hybrid. .NET MAUI Blazor is a cross-platform framework capable of building hybrid apps.

### 3.5.2 External Libraries:

Some of the external libraries that will be used to develop the application are as follows:

* MudBlazor
* SQLite-net

### 3.5.3 Persistence Mechanism:

For persistence mechanism, local database using SQLite will be applied in the development of applications.

## 3.6 Class, properties and methods:

### 3.6.1 Classes and their properties:

The classes prevalent in my project along with their properties is given below:

* Transaction.cs

Table 3: Transaction class and its properties.

|  |  |  |
| --- | --- | --- |
| Properties | Description | Type |
| Category | Stores the category of the transaction. | String |
| Amount | Stores the amount of the transaction | Int |
| Source | Stores the source of the transaction. | String |
| DateOfTransaction | Store the transaction of the day performed. | DateOnly |
| Tags | Stores the tags associated with the transaction. | String |
| Note | Optionally stores the note of the transaction performed. | String |
| IncomeType | Store the type of income performed. | IncomeType |
| expenseType | Stores the type of the expense’s transaction performed. | ExpensesType |
| debtType | Store the type of the debt transaction performed. | DebtType |

### 3.6.2 Classes and their methods:

The classes relevant in my project along with their methods is given below:

* DatabaseConnection.cs:

Table 4: DatabaseConnection.cs and its methods.

|  |  |  |
| --- | --- | --- |
| Methods | Description | Return Type |
| CheckForExistingDatabase | This method checks for the database. If the database already exists, then it simply displays ‘Database already exists.’ In debug lin. Otherwise it calls the initializetable function and inserts the user details in the database. | Void |
| InitializeTable | This method runs the query to create the required table in the database. | Void |

* DatabaseValueInsertion.cs:

Table 5: DatabaseValueInsertion.cs and its methods.

|  |  |  |
| --- | --- | --- |
| Methods | Description | Return Type |
| InsertUsers | This method runs the query to insert the values of the users into the database. | Void |
| InsertIncome | This method runs the query to insert the values of the income into the database. | Void |
| InsertExpense | This method runs the query to insert the values of the expenses into the database. | Void |
| InsertDebt | This method runs the query to insert the values of the debt into the database. | Void |
| InsertTags | This method runs the query to insert the values of the tags into the database. | Void |
| UpdateUserBalance | This method runs the query to add the amount to the balance of the user. | Void |
| DeductUserBalance | This method runs the query to deduct the amount to the balance of the user. | Void |

* DatabaseExtraction.cs:

Table 6: DatabaseExtraction.cs and its methods.

|  |  |  |
| --- | --- | --- |
| Methods | Description | Return Type |
| GetUserBalance | This method runs the SQLite query to get the current balance of the user. | Int |
| GetNumberTransaction | This method counts the number of transactions performed and returns it. | Int |
| GetTransactions | This method runs the query to get the details from income, expense and debt table and returns the list of it. | List<Transaction> |
| SortByDate | This method runs the query to get the details from income, expense and debt table, sorts it by date and returns the list of it. | List<Transaction> |
| SearchByFilterAndSpecificDateRange | This method runs the query to get the details form income, expenses and debt table with constraints and sort it according to the user input and returns list. | List<Transaction> |
| GetHighest | This method returns the value of the highest income, expense and debt. | Int |
| GetLowest | This method returns the value of the lowest income, expense and debt. | Int |
| GetTotal | This method returns the total amount of income, expense and debt. | Int |
| GetPendingAndClearedTotal | This method runs the query to returns the pending and cleared debt amount. | Int |

* DebtManagement.cs:

Table 7: DebtManagement.cs and its methods.

|  |  |  |
| --- | --- | --- |
| Methods | Description | Return Type |
| CheckNumberOfPendingDebt | This method returns the number of pending debts in the debt table. | Int |
| ClearPendingDebt | This method clears the pending debt in fully paid and partial payment case and returns the amount. | Int |

* TagManagement:

Table 8: TagManagement.cs and its methods.

|  |  |  |
| --- | --- | --- |
| Methods | Description | Return Type |
| GetTagList | This method returns the tag name in the tag table in list. | List<string> |

# 4 Conclusion:

Digi Dhan has been completed with the specified features and functionalities. The completion of this project brings forth many learning things and experiences. Being completely new to C#, I had relatively no idea on how to develop this application and complete this coursework, however over the period of time, a constant amount of my time and effort was spent on completing this project.

This period of time consisted of numerous trials and errors, research as well as learning from the module leader. While developing the application, I encountered a number of problems such as during developing the sort by specific date range method, the values were not being displayed due to mismatch of format for date. Similarly, at first, I had no idea on how to connect the application with SQLite. After a through research and consultation with my friends, I was able to complete this task as well. I also gained familiarity in working with Visual Studio 2022 as the whole application is developed in Visual Studio. Various concepts of OOP such as overriding, overloading etc. was also used during the development of the application.

Overall, this coursework has been a very fruitful one for me. The completion of this coursework came with its own sense of huge relief. I have learned a number of skills at the completion of this project. From research skills to gaining proficiency in C#, this project has helped me develop skills and knowledge. These gained knowledge and skill will be useful in my future endeavors.

# Bibliography

Il-Yeol Song, M. E. ,. E. P., 1995. A Comparative Analysis of Entity-Relationship Diagrams1. *Journal of Computer and Software Engineering,* Volume 3, pp. 427-459.