# Acknowledgement

I would like to take an opportunity to thank everybody for their direct and indirect support for this Computing Project. I am very thankful for the guidance that helped me built an Expense Tracker.

I would express my sincere gratitude to **Mr. Sudeep Lal Bajimaya**, for his continuous guidelines in this assignment. Also, would thank lecturer **Mr. Kiran Rana** and **Mr. Ritesh Singh** for their support to complete this assignment and explaining methodology.

Lastly, a big thanks to my friends their comments helped me improve the lacking parts of an Expense tracker.

# Abstract

Expense tracker keep track of how user are spending your money. This helps users manage his money, and take control of your finances. Users can use this app to manage their costs according to their budget. This app has numerous powerful tools like, Income/Expense, Bills, Accounts, Reports etc. It will keep users notified with their current situation of the budget. Recording transaction on daily basis with automatic features will reduce monotonous in users.

There are variety of expenses category, as per the expenses made users even have facility of making their own category. This app is not only about recording expenses, it also shows reports on daily, weekly, monthly, yearly basis. Through graphical representation of data, user can get better understanding.

Expense tracker also has feature of recording details regarding the nature of income. This reminds user to stick around their budget and obtain maximum utility and satisfaction through utilization of their limited budget.

Table of Contents

[Acknowledgement 1](#_Toc23514935)

[Abstract 2](#_Toc23514936)

[**Chapter 1 Introduction** 8](#_Toc23514937)

[1.1. Introduction to your system 8](#_Toc23514938)

[1.2. Background to the system 8](#_Toc23514939)

[1.2.1 Problem statement 8](#_Toc23514940)

[1.3. Overview of the proposed system Chapter 8](#_Toc23514941)

[1.4. Justification of the project 8](#_Toc23514942)

[1.5. Aims and objectives of the project 9](#_Toc23514943)

[**Chapter 2: Analysis** 10](#_Toc23514944)

[2.1 Introduction 10](#_Toc23514945)

[2.1.1. Analysis methodology, diagrams associated with methodology 10](#_Toc23514946)

[2.2 Information gathering technique 11](#_Toc23514947)

[2.3 Feasibility study 12](#_Toc23514948)

[2.4 SRS (System Requirement Specification) 13](#_Toc23514949)

[2.4.1 Functional requirement 13](#_Toc23514950)

[2.4.2 Non-functional requirement 15](#_Toc23514951)

[2.4.3 Prioritization 16](#_Toc23514952)

[2.4.4 Requirement specification 17](#_Toc23514953)

[2.5 Use case 18](#_Toc23514954)

[2.6 System architecture 20](#_Toc23514955)

[2.6.1 Natural Language Analysis (NLA) 21](#_Toc23514956)

[2.6.2 Initial class diagram 22](#_Toc23514957)

[**Chapter 3: Design** 23](#_Toc23514958)

[Introduction 23](#_Toc23514959)

[3.1. Structural Modeling 23](#_Toc23514960)

[3.1.1. Class diagram 23](#_Toc23514961)

[3.1.2. Flowchart diagram 24](#_Toc23514962)

[3.2. Behavioral Modeling 27](#_Toc23514963)

[3.2.1. Activity Diagram 27](#_Toc23514964)

[3.3.2. Sequence Diagram 29](#_Toc23514965)

[3.3. Database Modeling 32](#_Toc23514966)

[3.3.1. ER modelling 32](#_Toc23514967)

[3.3.2. Data dictionary 32](#_Toc23514968)

[3.4. UI Modeling 34](#_Toc23514969)

[Chapter 4: Implementation 42](#_Toc23514970)

[Chapter 5: Testing 43](#_Toc23514971)

[Unit Testing 64](#_Toc23514972)

[Chapter 6: Other project issues 68](#_Toc23514973)

[6.1. Limitation 68](#_Toc23514974)

[6.2. Future work 68](#_Toc23514975)

[6.3. Risk Management 68](#_Toc23514976)

[6.4. Configuration management 70](#_Toc23514977)

[6.5. Other project issues 71](#_Toc23514978)

[6.6. User manual 72](#_Toc23514979)

[Conclusion 84](#_Toc23514980)

[Appendix 85](#_Toc23514981)

**List of Figures**

[Figure 1: Use case diagram 20](#_Toc23514609)

[Figure 2: 3 tier architecture 22](#_Toc23514610)

[Figure 3: Initial class diagram 24](#_Toc23514611)

[Figure 4: Class Diagram 26](#_Toc23514612)

[Figure 5: Flow chart diagram 28](#_Toc23514613)

[Figure 6: Activity diagram 30](#_Toc23514614)

[Figure 7: sequence diagram of login 32](#_Toc23514615)

[Figure 8: sequence diagram of system 33](#_Toc23514616)

[Figure 9: ER Diagram 34](#_Toc23514617)

[Figure 10: UI for login 36](#_Toc23514618)

[Figure 11: UI for register 37](#_Toc23514619)

[Figure 12: UI for add category 38](#_Toc23514620)

[Figure 13: UI for category 39](#_Toc23514621)

[Figure 14: UI for add transaction 40](#_Toc23514622)

[Figure 15: UI for transaction 41](#_Toc23514623)

[Figure 16: UI for graph 42](#_Toc23514624)

[Figure 17: UI for report 43](#_Toc23514625)

[Figure 18: testing of login form 46](#_Toc23514626)

[Figure 19: testing of login form (2) 47](#_Toc23514627)

[Figure 20: testing of login form (3) 48](#_Toc23514628)

[Figure 21: testing of login form (4) 49](#_Toc23514629)

[Figure 22: testing of signup form 50](#_Toc23514630)

[Figure 23: testing of signup form (2) 51](#_Toc23514631)

[Figure 24: testing of signup form (3) 52](#_Toc23514632)

[Figure 25: testing of signup form (4) 53](#_Toc23514633)

[Figure 26: testing of signup form (5) 54](#_Toc23514634)

[Figure 27: testing of signup form (6) 55](#_Toc23514635)

[Figure 28: testing of signup form (7) 56](#_Toc23514636)

[Figure 29: testing of signup form (8) 57](#_Toc23514637)

[Figure 30: testing of add category 58](#_Toc23514638)

[Figure 31: testing of add category (2) 59](#_Toc23514639)

[Figure 32: testing of add category (3) 60](#_Toc23514640)

[Figure 33: testing of add transaction 61](#_Toc23514641)

[Figure 34: testing of add transaction (2) 62](#_Toc23514642)

[Figure 35: testing of add transaction (3) 63](#_Toc23514643)

[Figure 36: testing of add transaction (4) 64](#_Toc23514644)

[Figure 37: testing of add transaction (5) 65](#_Toc23514645)

[Figure 38: unit test code for login 66](#_Toc23514646)

[Figure 39: unit test result for login fail 66](#_Toc23514647)

[Figure 40: unit test code for login pass 67](#_Toc23514648)

[Figure 41: unit test result for login pass 67](#_Toc23514649)

[Figure 42: unit test code for validation fail 68](#_Toc23514650)

[Figure 43: unit test result for validation fail 68](#_Toc23514651)

[Figure 44: unit test code for validation pass 69](#_Toc23514652)

[Figure 45: unit test result for validation pass 69](#_Toc23514653)

[Figure 46: Configuration management 73](#_Toc23514654)

[Figure 47: Code for user\_model 88](#_Toc23514655)

[Figure 48: Code for transaction\_model 90](#_Toc23514656)

[Figure 49: Code for category\_model 91](#_Toc23514657)

[Figure 50: Code for user\_presenter 92](#_Toc23514658)

[Figure 51: Code for transaction\_presenter 94](#_Toc23514659)

[Figure 52: Code for category\_presenter 95](#_Toc23514660)

[Figure 53: Code for IUser\_view 96](#_Toc23514661)

[Figure 54: Login form 97](#_Toc23514662)

[Figure 55: Code for login 99](#_Toc23514663)

[Figure 56: UI for sign up 100](#_Toc23514664)

[Figure 57: Code for signup 103](#_Toc23514665)

[Figure 58: Code for abstract\_view 107](#_Toc23514666)

[Figure 59: UI for transaction 108](#_Toc23514667)

[Figure 60: Code for transaction 110](#_Toc23514668)

[Figure 61: UI for category 111](#_Toc23514669)

[Figure 62: Code for category 112](#_Toc23514670)

[Figure 63: UI for graph 113](#_Toc23514671)

[Figure 64: Code for graph 114](#_Toc23514672)

[Figure 65: UI for report 115](#_Toc23514673)

[Figure 66: Code for report 116](#_Toc23514674)

[Figure 67: Code for abstract\_transaction 120](#_Toc23514675)

[Figure 68: UI for add\_transaction 121](#_Toc23514676)

[Figure 69: Code for add\_transaction 122](#_Toc23514677)

[Figure 70: UI for edit\_transaction 123](#_Toc23514678)

[Figure 71: Code for abstract\_transaction 125](#_Toc23514679)

[Figure 72: UI for add\_category 126](#_Toc23514680)

[Figure 73: Code for add\_category 127](#_Toc23514681)

[Figure 74: UI for edit\_category 128](#_Toc23514682)

**List of tables**

[Table 1: Table of functional requirement 16](#_Toc23514683)

[Table 2: Table of non-functional requirement 17](#_Toc23514684)

[Table 3: MoSCoW Prioritization of functional requirement 18](#_Toc23514685)

[Table 4: MoSCoW Prioritization non-functional requirement 18](#_Toc23514686)

[Table 5: summary of use case diagram 21](#_Toc23514687)

[Table 6: notation used in flowchart 27](#_Toc23514688)

[Table 7: Notation used in activity diagram 29](#_Toc23514689)

[Table 8: Notation used in activity diagram 31](#_Toc23514690)

[Table 9: Data dictionary of user table 34](#_Toc23514691)

[Table 10: Data dictionary of category table 34](#_Toc23514692)

[Table 11: Data dictionary of transaction table 35](#_Toc23514693)

[Table 12: risk management 72](#_Toc23514694)

# **Chapter 1 Introduction**

## Introduction to your system

The term "budget" is a powerful word since, it has feature of both tedious and challenging. In this era of expensive market, it is really essential to keep track of your spending of money.

It is the initial stage towards understanding how an individual is managing their money, and to taking control of your budget. As one of the most crucial tools in building a successful financial future, it helps to achieve the most satisfaction and utility from limited finance. Tracking user’s expenses can help them stop feeling like you have little to show for your hard work.

Expense tracker- This app will help you track all the expenses bared by a user and moreover, help to manage user’s personal finance.

## Background to the system

Normally, people manage their budget by simply memorizing it or writing it on a notebook. No financial situation is same to each person; foundation can also not be same. It differs from person to person. It may seem like a simple work, but increased transaction led towards more complexity.

### Problem statement

These type of traditional practices usually led to miscalculation and caused several problems. This made people monotonous due to repetitive tasks of recording transaction.

## Overview of the proposed system Chapter

The app is basically designed to replace the traditional methods of tracking expenses and making this task easier and effective. It is related not only with where the money goes but also with where money comes from. Users can plan their budget accordingly. This tracking application is rich with numerous powerful tools like, Income/Expense, Bills, Accounts, Reports etc.

## Justification of the project

The rise of technology has made great impact on our lives. Nowadays people want tasks to be quick and easy. Similarly, to solve the task of tracking expenses more effortlessly I have come up with this idea of an expense tracker. Individuals can use this app to manage their costs according to their budget.

## Aims and objectives of the project

It aims to track expenses to create financial awareness. Few aims of the projects are as follows:

1. Finance awareness

The main reason to track expenses is to build financial awareness. People can organize their budget as their needs leading to maximum satisfaction from their limited income.

1. Stabilizing the budget

The expense should not exceed beyond the expected budget. For this reason, the user can make expenses as per their assumption.

1. Identify spending issues

Ignoring small expenses lean towards high impact on the budget. By tracking these types of expenditures user are made alert about their current situation.

Main objectives of this project are:

1. Remind people to stick to their budget
2. Provide information of their current budget
3. Alert user about their expenses
4. Provide details regarding the nature of income
5. Obtain maximum utility through properly mobilized limited budget

# **Chapter 2: Analysis**

## Introduction

The term ‘Analysis’ means examining something in detailed. Similarly, in software development the process of examining the system requirement, data and document.

### Analysis methodology, diagrams associated with methodology

Analysis methodology is an organized way of carrying out analysis. There are different methodologies that is suited for solving different types of issues. Among these methodologies I have chosen OOM (Object Oriented Methodology).

OOM is a methodology which does not concentrate solely on processes or data but rather views the system as group of objects which works together to solve a task. Objects are the representation of items in the real world. The structure is divided into different steps which consists of tasks which is further divided into sub-tasks. To solve a problem different model are created like dynamic model, object model and functional model.

The object model represents the static structure of the system. The dynamic model represents the dynamic structure that is flow of control and events. The functional model represents the internal processes.

OOM is fit for this project because of the re-usability of the analysis, design, objects and code. It is easy to understand. It is also more flexible to change as it is easier to make update in response to requirement change. The system can also be developed rapidly.

The diagram related to OOM like event-trace diagram, activity diagram, sequence diagram and class diagram will be drawn in the Design phase.

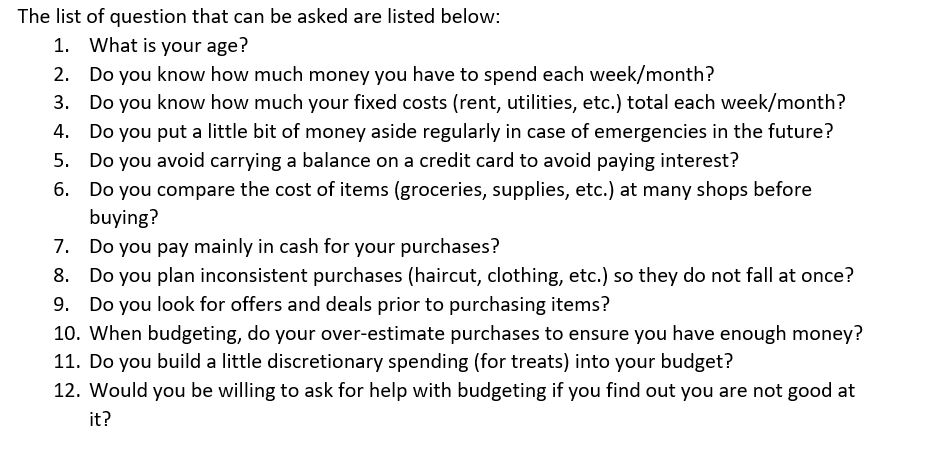
## Information gathering technique

Information gathering is the process of collecting information from different stakeholders that will be used to produce requirement definition (SRS). However, this task is not an easy one as just asking stakeholders will not always solve the problem as we may not get what they are trying to explain or we may get something else than they are trying to explain. There are various methods to gather information like interview, questionnaires, survey, observation, etc.

For this project I have chosen questionnaires and interview.

* **Questionnaires**

Questionnaires is a method in which a group of questions are presented to a group of people. The questions are generally of three types: fixed-alternatives, scale and open ended. This method can be used to explore qualitative data. It provides large amount of data for relatively low cost.



* **Interview**

Interview is a one on one conversation between the interviewer and interviewee where questions are asked and answered. Interview will be a good fit to know what the problems are and how they hope them to be solved. However, this is not an easy task.

## Feasibility study

Feasibility study is an analysis that takes all project’s relevant factors into account to determine the success of the project. These factors include technical, economic, scheduling, legal, social, etc. It is important as it helps to find if the project is feasible or not. It identifies the pros and cons of the project to find whether the project is worth the investment of money and time.

* + 1. **Technical Feasibility**

Technical feasibility study is done to find whether is technical feasible like whether the computer can handle the processes or a new one is needed. For the current project the

* + 1. **Economic Feasibility**

In this study the economical part is taken into account and tested whether the project benefit overshadow the cost. As this project is done for educational purpose there is actually no budget. The basic requirements like a PC with specification are fulfilled as I already have a laptop with required specification so, budget will not be an issue.

* + 1. **Scheduling Feasibility**

Time is a major factor for the success of the project. Scheduling and strictly following the time schedule must be done to fulfil the requirements on time. We have about 3 month time to complete this project and I have divided the time into suitable periods to fulfil a specific task. If all the tasks are completed in the specified time then this project will be feasible in the given time.

* + 1. **Legal Feasibility**

Legal issues should not be taken for granted. It may be one of the major cause for the failure of the project. It can lead to a law suit if any laws are broken. There is no illegal issues in this project. It follow all the laws of the country so, it will not have legal issues.

* + 1. **Social Feasibility**

In a society there are many people with various cultures and traditions. Social feasibility study is done to find whether the current project can be properly implemented to the society. This project is a simple one and does not target any stereotype of people. So, this project is socially feasible.

## SRS (System Requirement Specification)

SRS is a part of documentation that describes the features and behavior requirements of the system. For this documentation, it contains both functional and non-functional requirements.

### Functional requirement

These requirements are the product’s functionalities like features, capabilities, usability and operations. The list of features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Functional Requirements | Description | Rational | Dependency |
| F1 | Add Expense | User can add expense | To add expense to keep record | N/A |
| F2 | Add Income | User can add income | To add income to keep record | N/A |
| F3 | Update Expense | User can update expense | To edit the input expense | F1 |
| F4 | Update Income | User can update income | To edit the input income | F2 |
| F5 | Delete Expense | User can delete falsely entered expense | To delete the incorrect expenses | F1 |
| F6 | Delete Income | User can delete falsely entered expense | To delete the incorrect income | F2 |
| F7 | Generate graph | To produce a graph from the given data | It helps to visualize the data in graphs like bar or pie chart | F1,F2 |
| F8 | Generate report | To produce a report from the given data | It helps to generate valuable report | F1,F2 |
| F9 | Add Expense categories | To specifies the type of expense | It helps to differentiate expenses | N/A |
| F10 | Add income categories | To specifies the type income | It helps to differentiate income | N/A |
| F11 | Provide notification | Gives notification for remainder or any other purposes | Can be as a remainder or give status of the system | F1,F2 |
| F12 | Automate income | Automates repetitive tasks | Can help to ignore tedious work | F1,F2 |
| F13 | Automate expense | Automates repetitive tasks | Can help to ignore tedious work | F1,F2 |
| F14 | Export graph | Exports graph to other extensions | It helps to export valuable graph | F1,F2,F7 |
| F15 | Export report | Exports report to other extensions | It helps to report valuable graph | F1,F2,F8 |

Table 1: Table of functional requirement

### Non-functional requirement

Rather than specific behaviors, these requirement specifies criteria to judge the operation of a system, are non-functional requirement.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Description | Dependency |
| N1 | Availability | The system should be accessible when it is needed. As this is an offline app it will be available even when there is no internet connection. | N/A |
| N2 | Integrity | The data should not change without intended to. Change in data may cause huge problems. | N/A |
| N3 | Documentation | This project will have proper documentation of the phases of development. It will also have an user manual. | N/A |
| N4 | Durability | The system should not have to undergo maintenance phase frequently. | N/A |
| N5 | Efficiency | The app should produce correct results. The output should not be inappropriate. | F1,F2,F7,F8 |
| N6 | User-friendly | The system should have a friendly user interface so that even user who have never used the system can easily use the system. A user manual will also be prepared to help the new user. | N/A |
| N7 | Security | Data should be always secure. It should not fall into the wrong hand. Data breach cause huge problems | N/A |
| N8 | Scalable | The project should be easy to add features or other updates. In the future there should not be problems to expand the system | N/A |
| N9 | Maintainability | The system should be easy to maintain. It should have easy fixes to bug and minor problems. | N/A |
| N10 | Reliability | The output generated should reliable. The expected results should be obtained without any failure in the system. | N/A |

Table 2: Table of non-functional requirement

### Prioritization

For the prioritization of requirements, I have done MoSCoW prioritization in which the requirements are categorized into Should have,Must have, Could have and Won’t have.

The MoSCoW prioritization for functional requirements are:

|  |  |  |
| --- | --- | --- |
| ID | Functional Requirements | MoSCoW |
| F1 | Add Expense | M |
| F2 | Add Income | M |
| F3 | Update Expense | M |
| F4 | Update Income | M |
| F5 | Delete Expense | M |
| F6 | Delete Income | M |
| F7 | Generate graph | S |
| F8 | Generate report | S |
| F9 | Add Expense categories | M |
| F10 | Add income categories | M |
| F11 | Provide notification | C |
| F12 | Automate income | C |
| F13 | Automate expense | C |
| F14 | Export graph | S |
| F15 | Export report | S |

Table 3: MoSCoW Prioritization of functional requirement

The MoSCoW prioritization for non-functional requirements are:

|  |  |  |
| --- | --- | --- |
| ID | Title | MoSCoW |
| N1 | Availability | M |
| N2 | Integrity | M |
| N3 | Documentation | M |
| N4 | Durability | S |
| N5 | Efficiency | M |
| N6 | User-friendly | M |
| N7 | Security | S |
| N8 | Scalable | C |
| N9 | Maintainability | C |
| N10 | Reliability | M |

Table 4: MoSCoW Prioritization non-functional requirement

Where,

M: Must have

S: Should have

C: Could have

### Requirement specification

**Software Requirement:**

Programming Language: C#

Operating system: Windows (7 or higher)

Database: SQL Server

Platform: Windows

**Hardware Requirement:**

RAM: Minimum 3 GB, recommended 8 GB

Memory: 2 GB, 4 GB recommended

Processor: 2.16 gigahertz Intel Core i3 **processor** (dual core)

Display: 1280\*800 minimum screen resolution

## Use case

Use case diagram represents the interaction amongst the system and entities. It identifies and organizes the requirements and shows the expected behavior rather than exact method it happens. The main aim of using a case modeling is to help in designing a system from end user's perspective. An effective technique for communicating system behavior specifying all externally visible system behavior of user's terms.

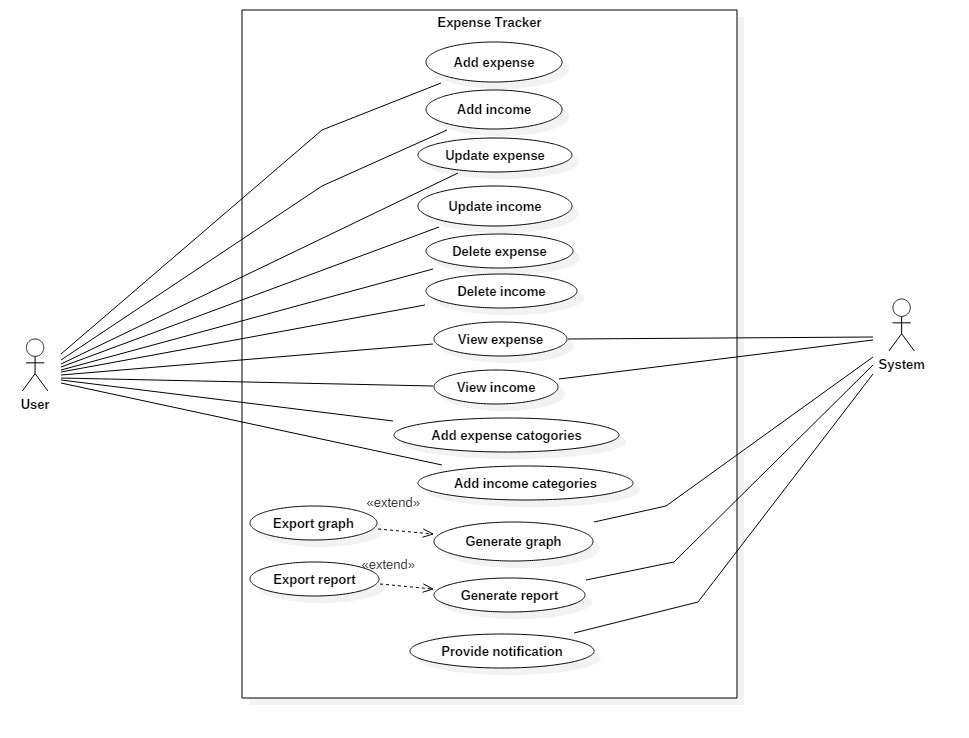


Figure 1: Use case diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Use Case title** | **Summary** | **Actor** |
| C1 | Add expense | The user can add expenses | User |
| C2 | Add income | The user can add income | User |
| C3 | Update expense | User can update the entered expense | User |
| C4 | Update income | User can update the entered expense | User |
| C5 | Delete expense | User can delete expense | User |
| C6 | Delete income | User can delete income | User |
| C7 | View expense | The user can view the expense added and the system can use this data too. | User, System |
| C8 | View income | The user can view the income added and the system can use this data too. | User, System |
| C9 | Add expense categories | User can add new types of expense to categorize expense | User |
| C10 | Add income categories | User can add new types of income to categorize income | User |
| C11 | Generate graph | The system generate graph from the expenses or/and income. This graph can also be exported. | System |
| C12 | Generate report | The system generate report from the expenses or/and income. This report can also be exported. | System |
| C15 | Provide notification | The system can provides notification to remind the user. | System |

Table 5: summary of use case diagram

## System architecture

For the system architecture I have chosen 3 tier architecture because it divides the system into three parts. The user uses and sees only the presentation layer and the background work is done in application layer and data layer. Our system will be most suited in this architecture as the work will be divided and there will be no confusion on which tasks to do where. Similarly, the 3 layer can work in the following way:

**Presentation layer:** This is the only layer that the user can see so, this should be made user friendly. This is the layer where input is taken from the user.

**Application layer:** In this layer the overall tasks of the system is carried out like calculations and operations. It is the layer between the presentation layer and the data layer so, the communication of these layers are completed through this layer.

**Data layer:** It is the lowest layer of this system. The basic task of this layer is to handle data like store, retrieve and update. It is the layer where database is stored.

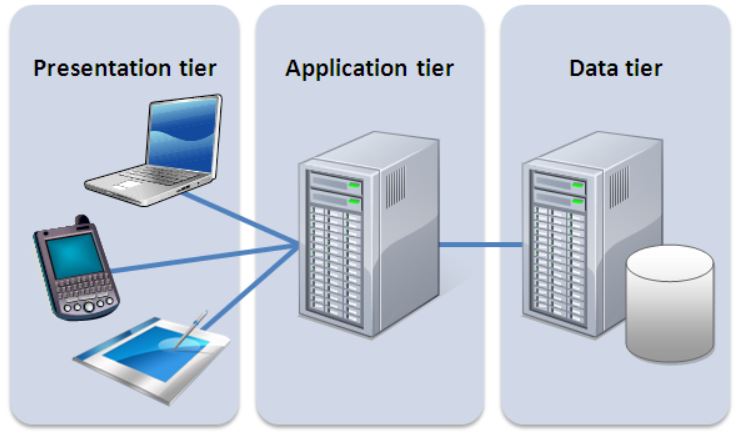


Figure 2: 3 tier architecture

### 

### Natural Language Analysis (NLA)

NLA is a method for filtering out noun for classes, verbs for methods and adjectives for attributes. This process of filtration is carried out by observation and discussion from a problem/ scenario. Helps us with identification of the candidate and actual resources, attributes and properties.

**Scenario**

The term "budget" is a powerful word since, it has feature of both tedious and challenging. In this era of expensive market, it is really essential to keep track of your spending of money.

It is the initial stage towards understanding how an individual is managing their money, and to taking control of your budget. As one of the most crucial tools in building a successful financial future, it helps to achieve the most satisfaction and utility from limited finance. Tracking user’s expenses can help them stop feeling like you have little to show for your hard work.

Tracking your expenses can help you stop feeling like you have little to show for your hard work.

Your task is to make an application which can help the users to track their transaction. The app should be able to store the income and outgoings of the user so that the user can view these records. The user should also be able to plan their budget in the future. These are the basic requirements for the app:

* Store and view the expenses
* Store and view the incomes
* Plan the budget for future

There should be categories of income (Example: salary, wages, tips, bonuses, commissions) and expenses (Example: food, transportation, entertainment). User should also be able to create their type of income or expenses. Then the user can visualize the categories in:

* Graphs. Graphical representation can help in better understanding the data. It helps to compare the expenses made with other expense or total income.
* Report. When transactions are classified into different categories or arranged in a specific order then it will be easy to view and understand.

It would be tedious to add the same expenses like food or transportation daily. To make the app interesting the app could be having automatic feature or semi- automation. For example, if a user gets a fixed salary every month than this could be done automatically.

The candidate class from the above scenario are:

|  |  |  |  |
| --- | --- | --- | --- |
| Budget | people | money | tools |
| expenses | User | Transaction | Income |
| Outgoing | Records | Future | App |
| Categories | Salary | Wages | Tips |
| Bonuses | Commission | Food | Transportation |
| Entertainment | Graph | Report | Automatic |

Actual class can be identified from the candidate classes listed above.

|  |  |  |
| --- | --- | --- |
| S.N. | Actual class | Description |
| 1 | Expenses | It can hold the expenses and categories of expenses |
| 2 | Income | It can hold the income and categories of income |
| 3 | Transaction | It could be the super class to the expenses and income |
| 4 | Records | It can be used to hold the records |
| 5 | Graph | It can be used to generate graph |
| 6 | Report | It can be used to generate report |
| 7 | Automatic | It will be used to do the automatic work |

### Initial class diagram

Looking at the above actual class diagrams the initial class diagram could look something like this:

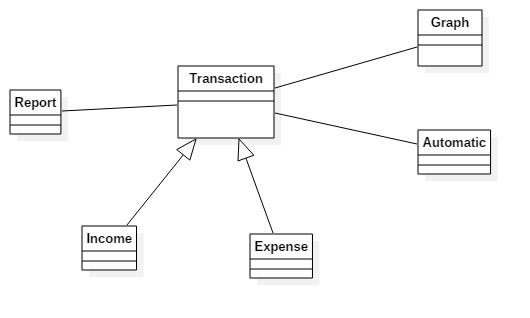


Figure 3: Initial class diagram

# **Chapter 3: Design**

## Introduction

Design phase is the stage where the system architecture is designed based on the user requirements. The logical system design produced during the analysis phase is converted into physical design in design phase. Different tools and techniques are used to describe the design of the system.

## Structural Modeling

Structural model provides the structural perspective of the data that is processed by the system.

### Class diagram

Class diagram is a static structural diagram which shows the structure of classes of the system, its methods, attributes and relationship among objects.

**Justification**

The reasons for using class diagrams are as follows:

1. Shows static structure of the class and objects
2. Because I am using object-oriented methodology
3. Gives the overview of the system and helps to maintain time

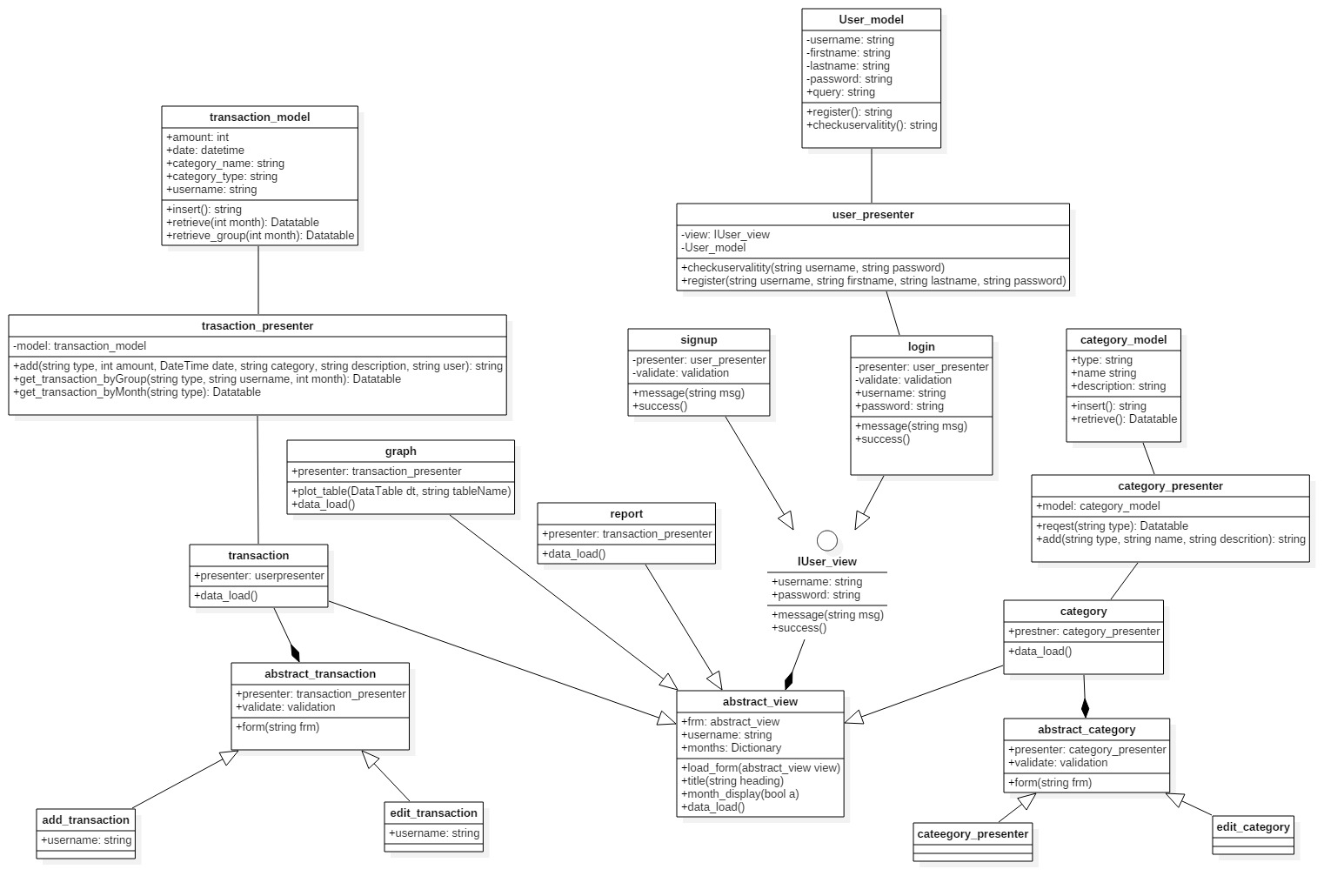


Figure 4: Class Diagram

### Flowchart diagram

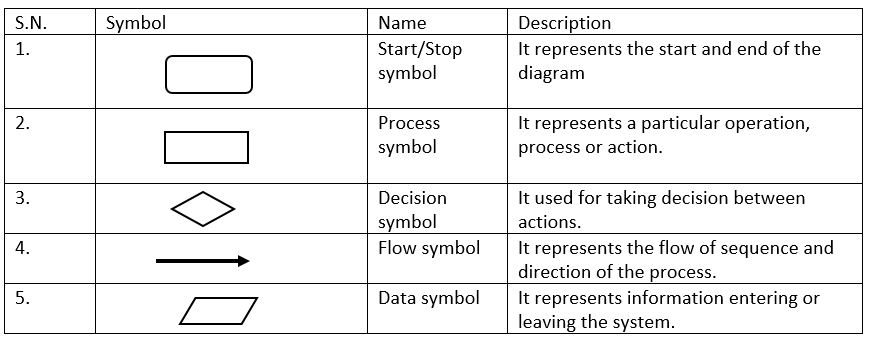
Flowchart is a representation of steps in graphical form. It shows the sequential flow of steps to represent the algorithm, workflow or processes.

**Justification**

The reason for using flowchart diagram are as follows:

1. Flowchart represents the overall logic of the system to all involved which improves communication.
2. The flow of process is shown in a systematic way.
3. All the process can be identified clearly

Notation used

 Table 6: notation used in flowchart

Diagram

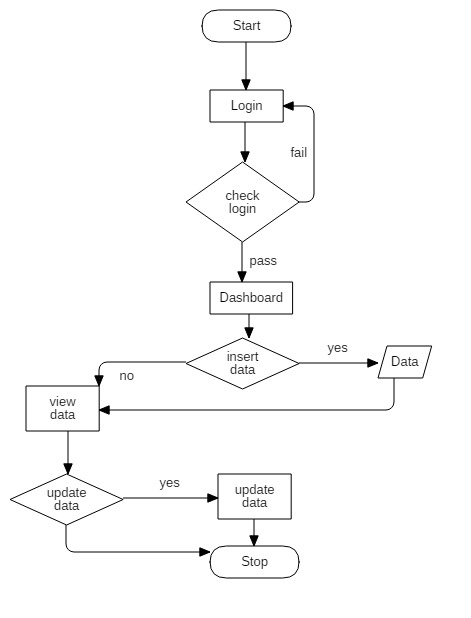


Figure 5: Flow chart diagram

The given diagram shows that user can get into the Dashboard after successful and valid login details. The user enters its data and start its transaction and can view, update their details.

## Behavioral Modeling

Behavioral model is a type of system model in which dynamic behavior of the system are shown how it responds to the system.

### Activity Diagram

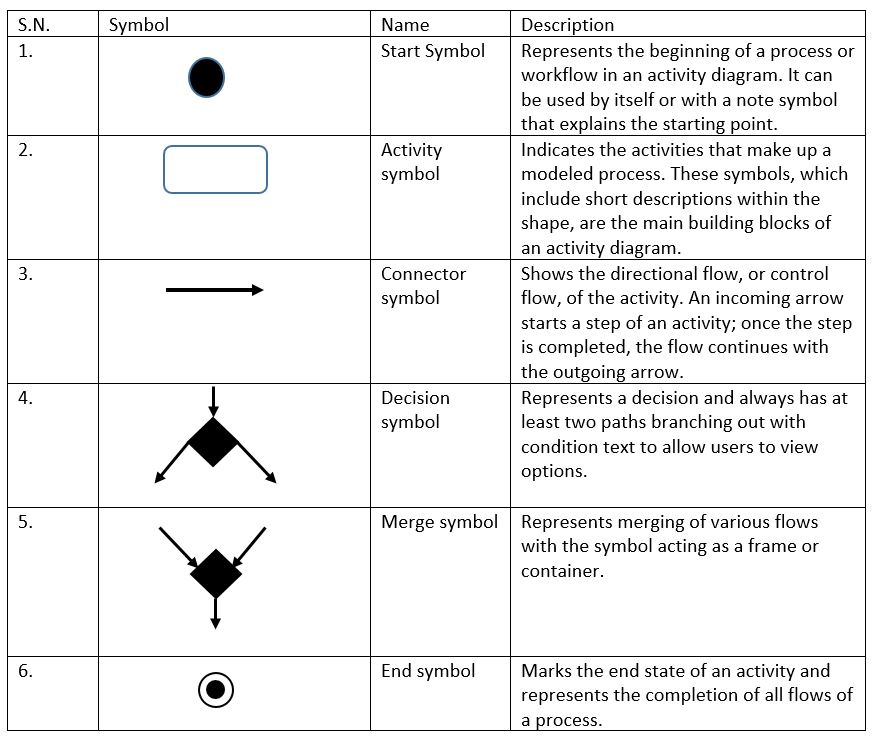
Activity diagram is a dynamic behavioral diagram which shows the activity involved in data processing or a process.

**Justification**

The reasons for using activity diagrams are as follows:

1. It displays the flow of activity between the user and the system
2. It interprets the parallel, concurrent and branched flow of the system
3. It can be used for analyzing a use case
4. It describes the sequence of an activity to another

Notation used

Table 7: Notation used in activity diagram

Diagram

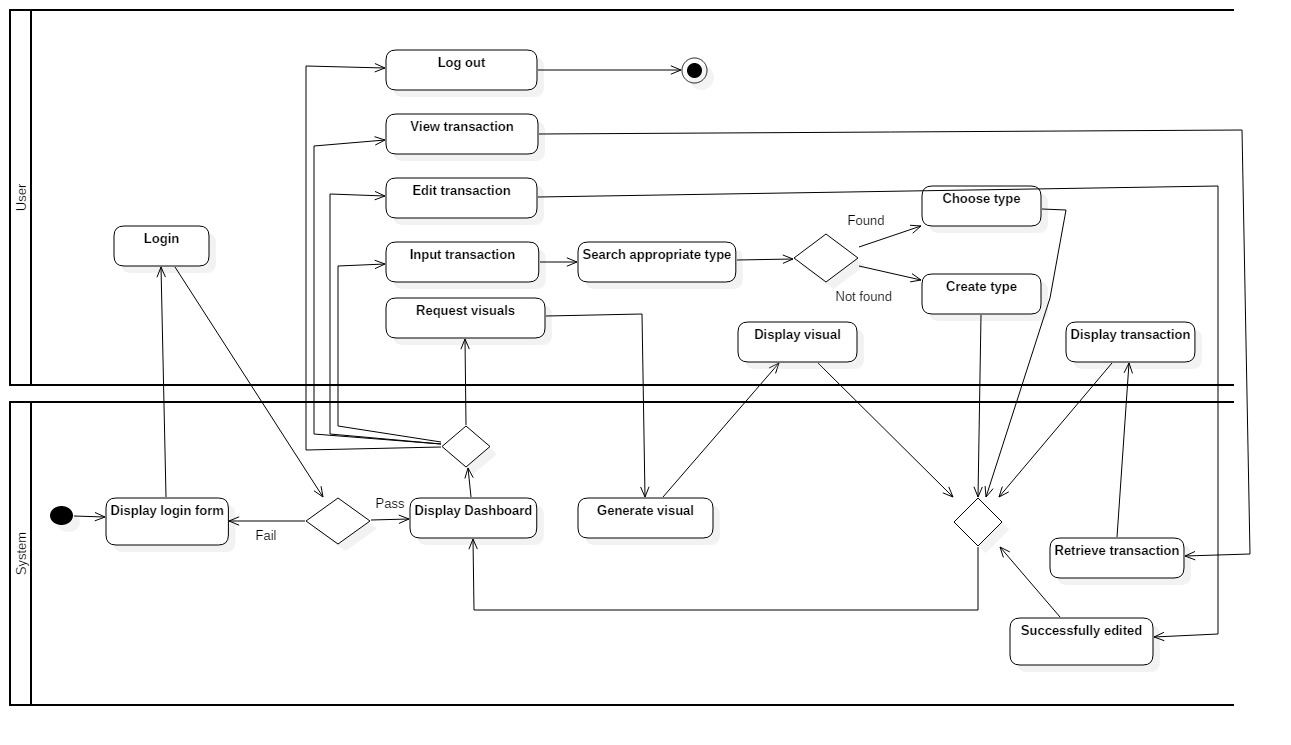


Figure 6: Activity diagram

The above diagram, at first the system displays the login form so that the user can login. If the user enters the right credentials, then the system will display else back to the login form. From the dashboard the user can either input transaction, view transaction, edit transaction, request visuals or logout. If the user wants to enter the transaction, then he\she must search for appropriate type. If found the user can choose it else, he\she can also create type. After this the system take the user back to the dashboard. If the user requests visuals, then the system first generate the visuals based on the data it has stored then display it to the user. After this again the system takes the user back to the dashboard. The user can also logout to terminate the system.

### Sequence Diagram

Sequence diagram is a dynamic behavioral diagram. It represents the interaction between actors, system and system components.

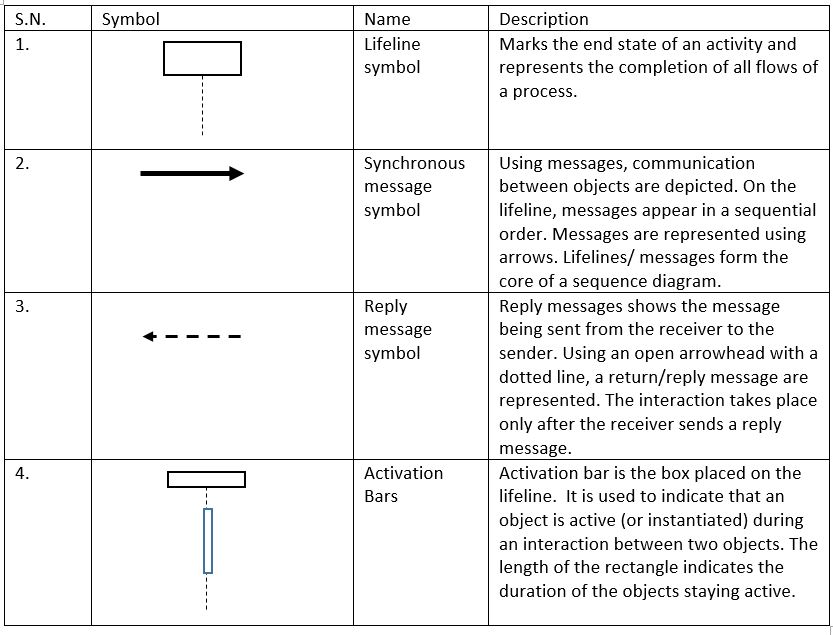
**Justification**

The reasons for using activity diagrams are as follows:

1. It models high-level interaction between the active objects in a system
2. Allows reverse engineering
3. Shows how objects and components interact with each other to complete a process

Notation used

Table 8: Notation used in activity diagram



**Diagram**

**Login sequence**



Figure 7: sequence diagram of login

The above diagram shows the login sequential flow done by user to the system. The user inserts the email and password and the login details data is check for validation in the database. If the input data is correct then the user dashboard is displayed but if the input data is invalid, then user need to enter the email and password to login.

**User sequence**



Figure 8: sequence diagram of system

The above diagram shows sequence of work done by the user. The user can add income or expense. They can also choose a category to the income or expense. Similarly, the user can also request graph or reports. The system fetches the expenses and income entered by the user then it generates a graph or report based on the data provided. User can add, update and delete transaction and store data in database. User has access to view transaction and also has authority to view visual by sending message to database after that database reply by providing transaction details.

## Database Modeling

### ER modelling

Entity Relationship (ER) modeling is graphical representation of database design. Entity is a real world object or thing that can be distinguished from the environment. The relation of these entity are represented in the ER diagram.

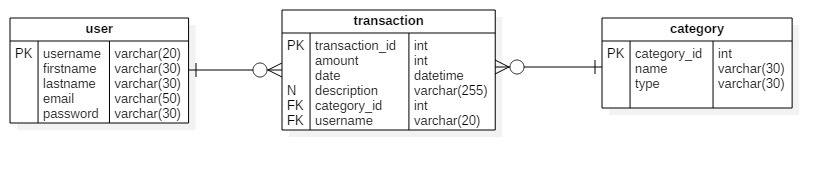
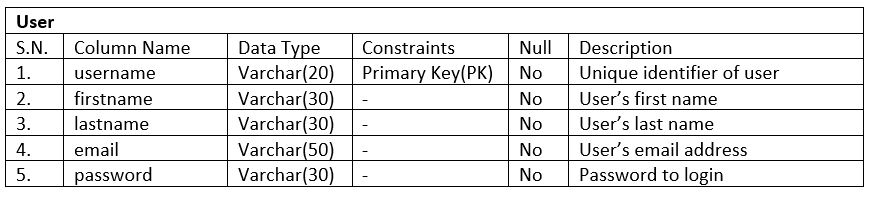
The ER diagram of this project is presented below

Figure 9: ER Diagram

### Data dictionary

Data dictionary is a file or a group of files in which metadata as well as data ownership, data relationship and other data are stored. The database user does not interact with data dictionary. It is invisible to the user but it is an important part of the relational database. Only the database administrator handles data dictionary.

The metadata of this project are listed below:

 Table 9: Data dictionary of user table

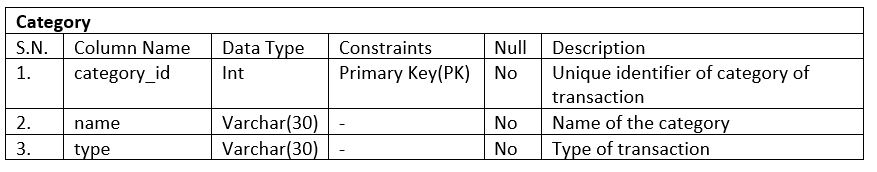


Table 10: Data dictionary of category table

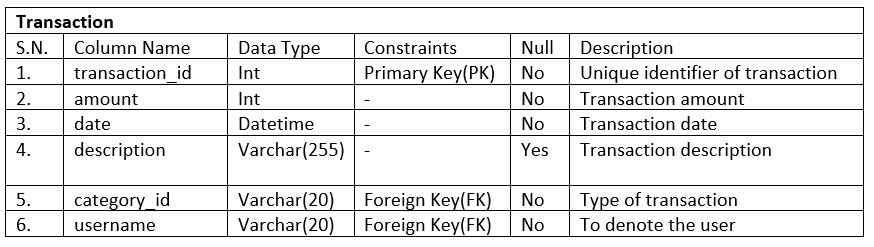


Table 11: Data dictionary of transaction table

## UI Modeling

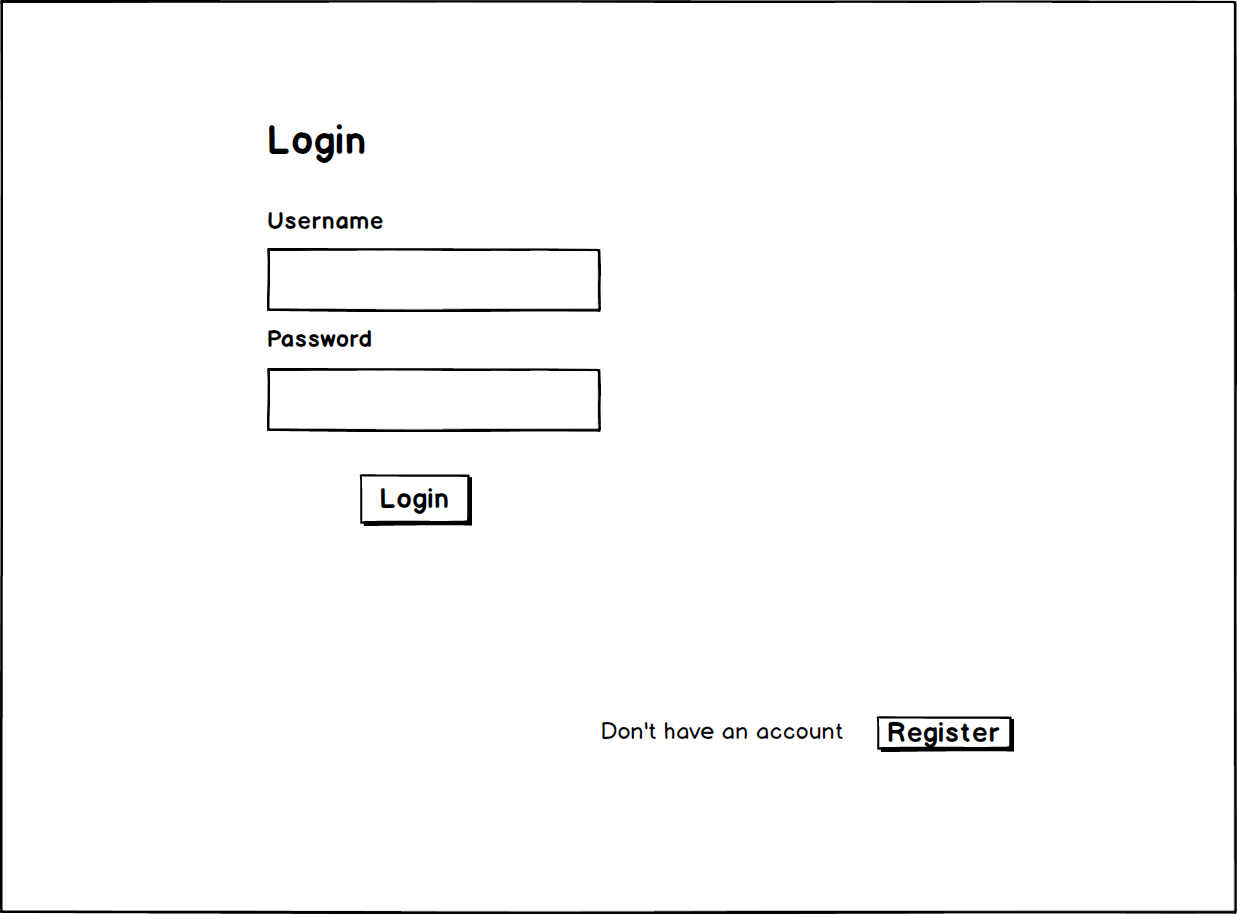


Figure 10: UI for login

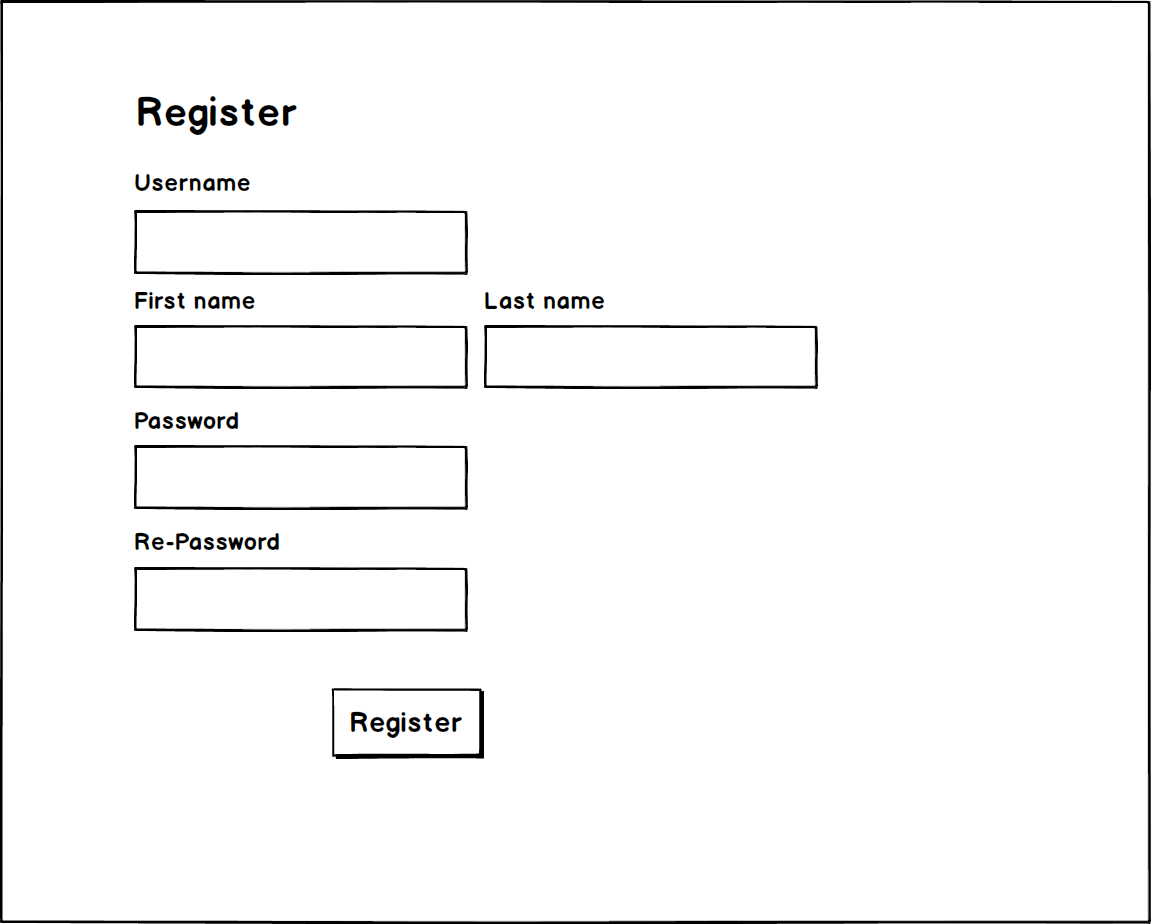


Figure 11: UI for register

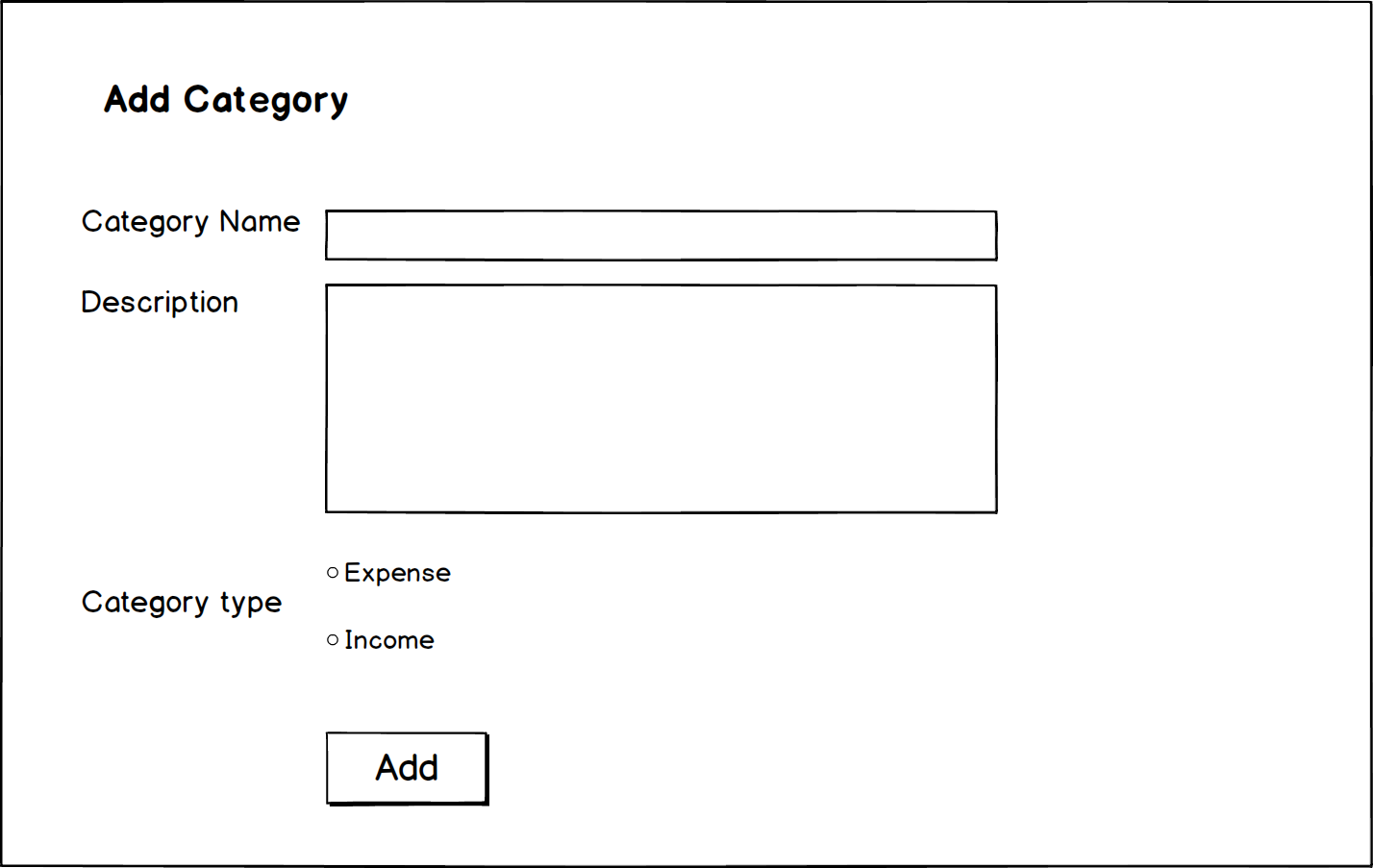


Figure 12: UI for add category

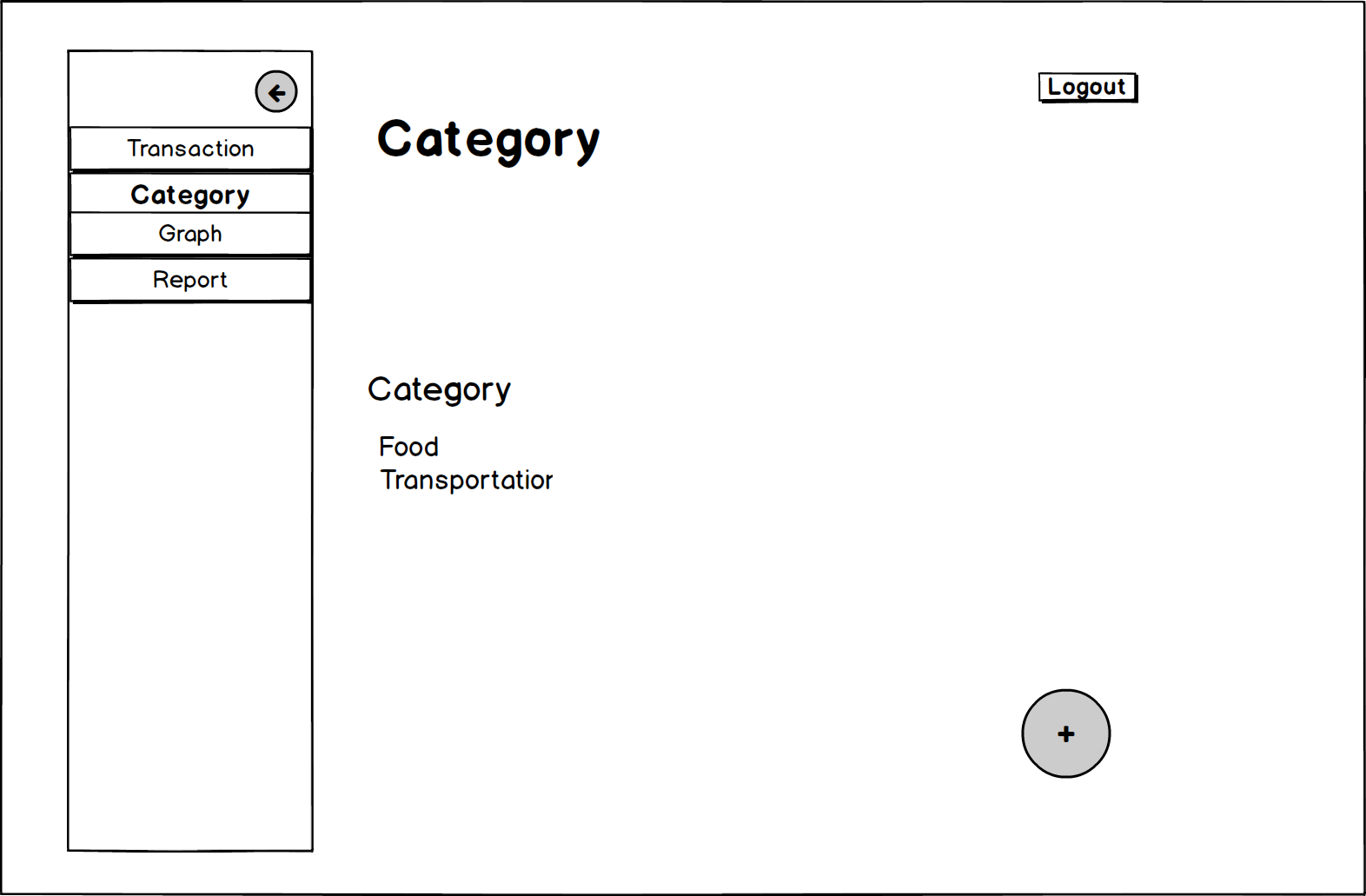


Figure 13: UI for category

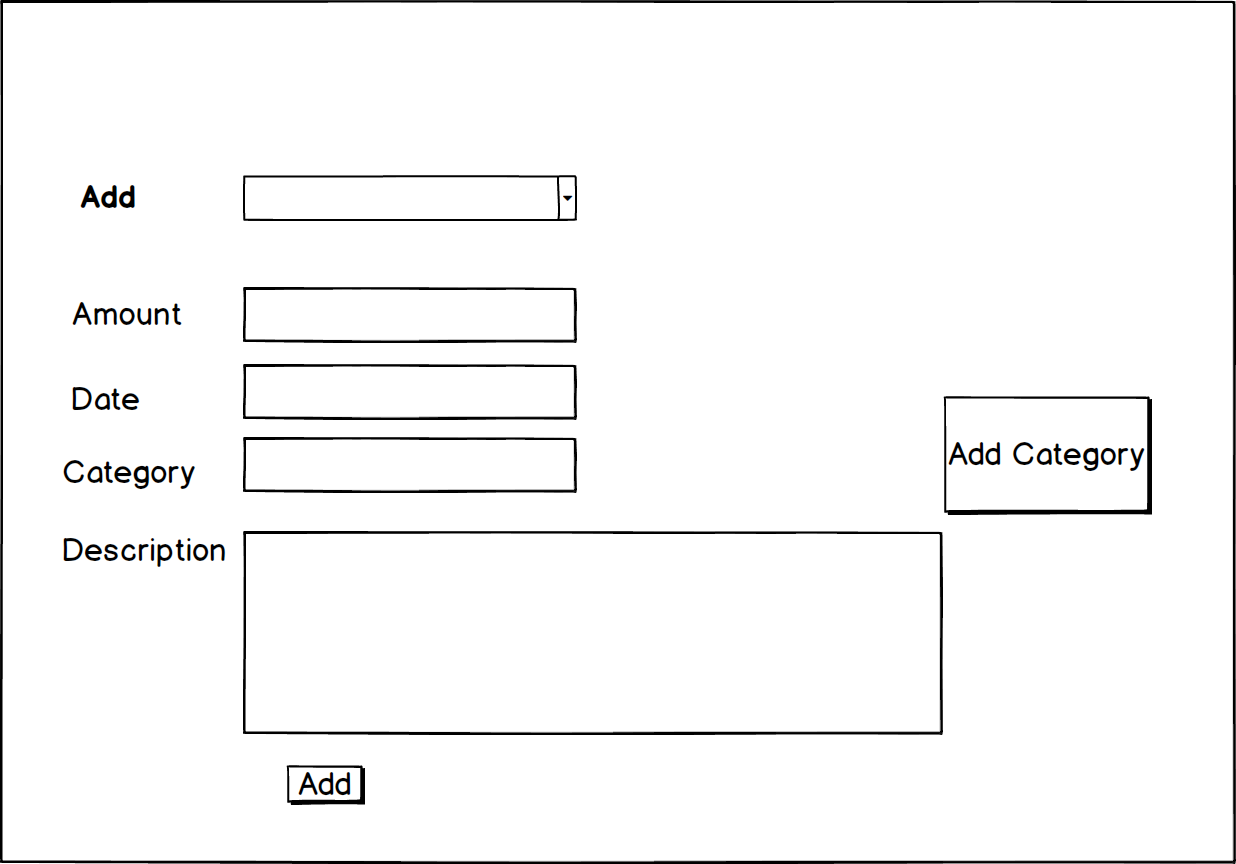


Figure 14: UI for add transaction

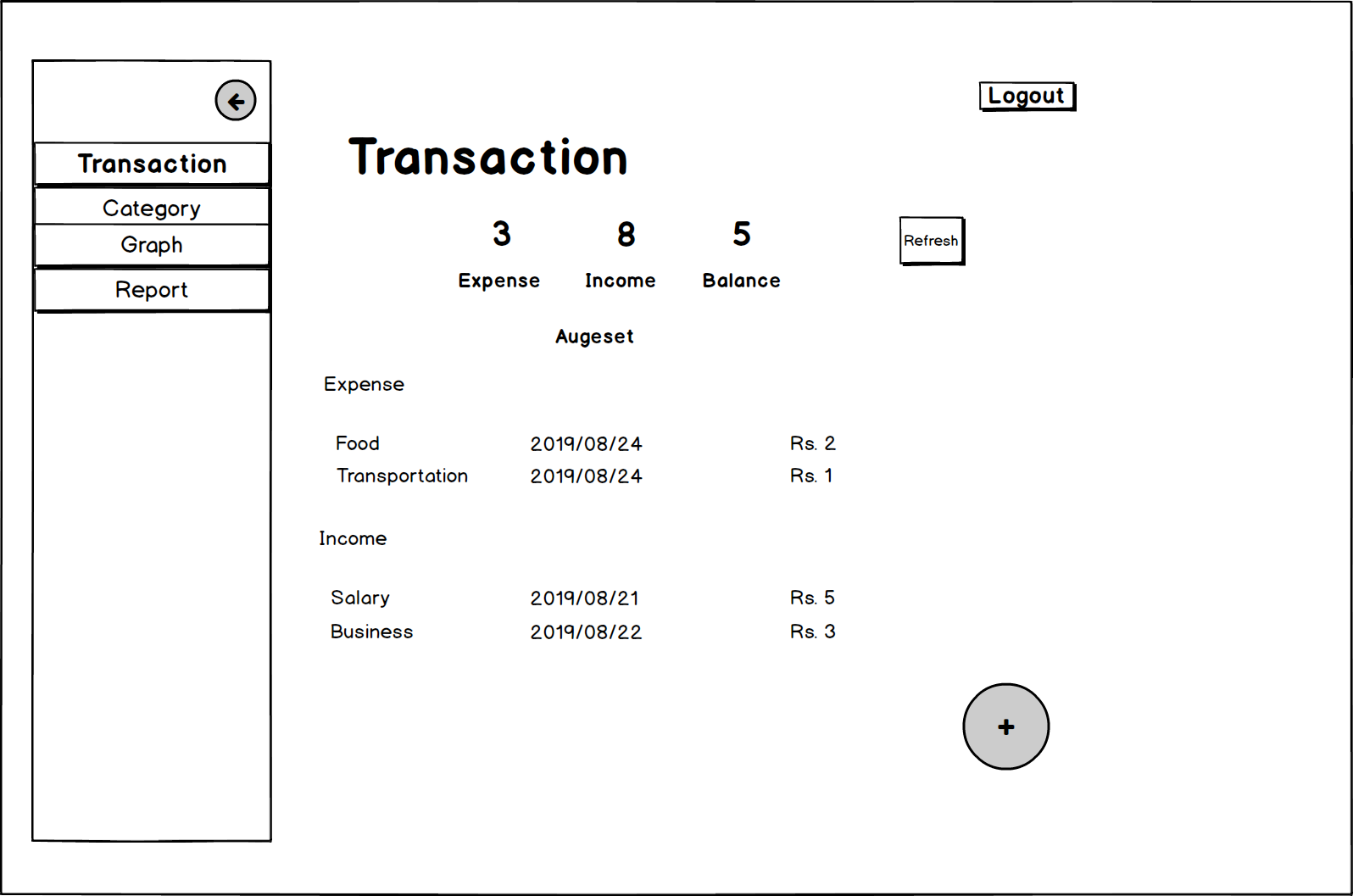


Figure 15: UI for transaction

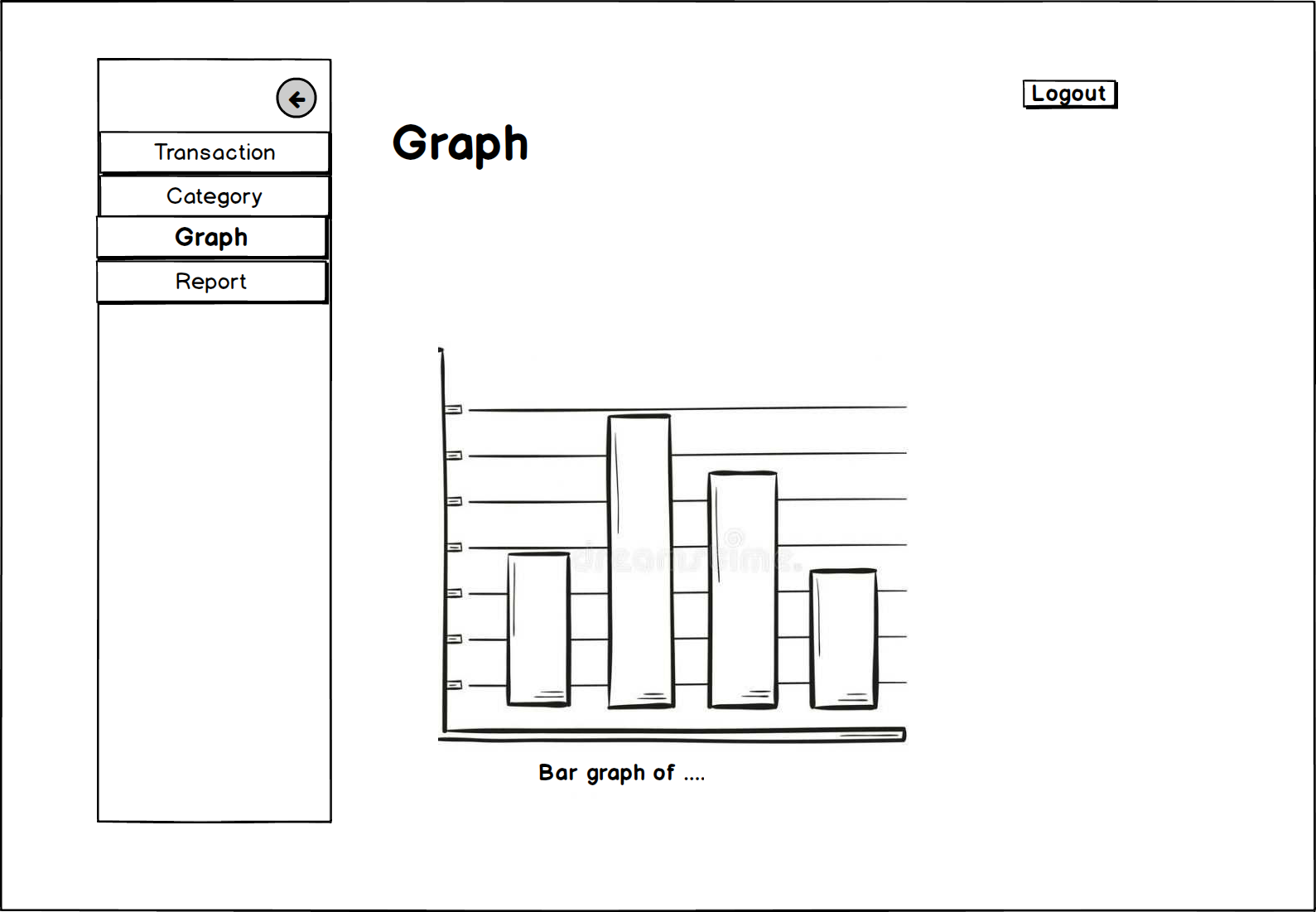


Figure 16: UI for graph

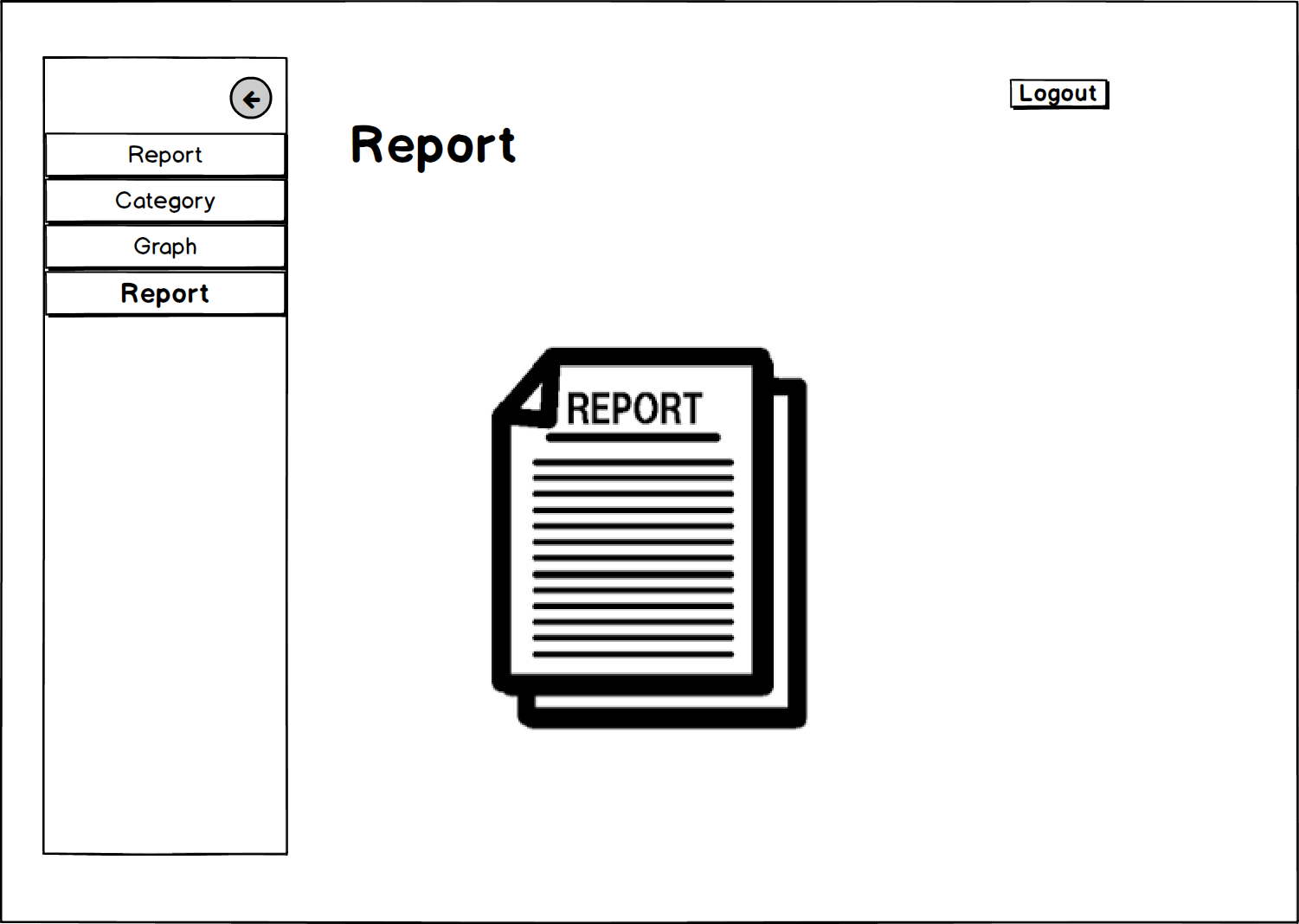


Figure 17: UI for report

# **Chapter 4: Implementation**

The code of implementation is placed in the appendix.

# **Chapter 5: Testing**

**Testing**:

Testing is execution of a system that identifies errors, gaps or missing requirements in contrast to actual requirements of that system.

It is defined as a process that involves checking whether the actual results match the expected results. This process intend to ensure that the system is error free.

We have numerous types of testing to make a system bug free. Among them I have choose **Unit testing** and **Black box testing** with test planning.

Basically, test plan is a document, describes about the scope, resources, objectives, approach, schedule, etc. of the proposed test activities.

**Unit** **Testing**:

Unit testing is concerned with breaking of program into pieces, and applying each piece to a series of tests. It is done to check whether there are any issues made by the developer.

The individual modules are identified, analyzed and fixed the defects. Advantages of Unit Testing are:

* Help reduce bugs and defects by changing the existing functionality.
* Help reduce expenses of testing since bugs are found at the early phase.
* Help improve design and refactoring of code.
* Facilities changes and simplifies integration.

**Black box testing:**

Black-box testing is concerned with examining the functionality of an application based on the specifications. It can be applied to all levels of software testing.

Advantages of Black box testing:

* Identifies contradictions and functional specifications.
* Test designed can be made immediately after completion of specifications.
* Tests are unbiased since, the designer and tester work independently
* Tester does not require knowledge of specific programming languages to test.

Test Name: Login for user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 1 | **Username**:<blank>  **Password**: <blank> | Show error message | Please enter username | Pass |

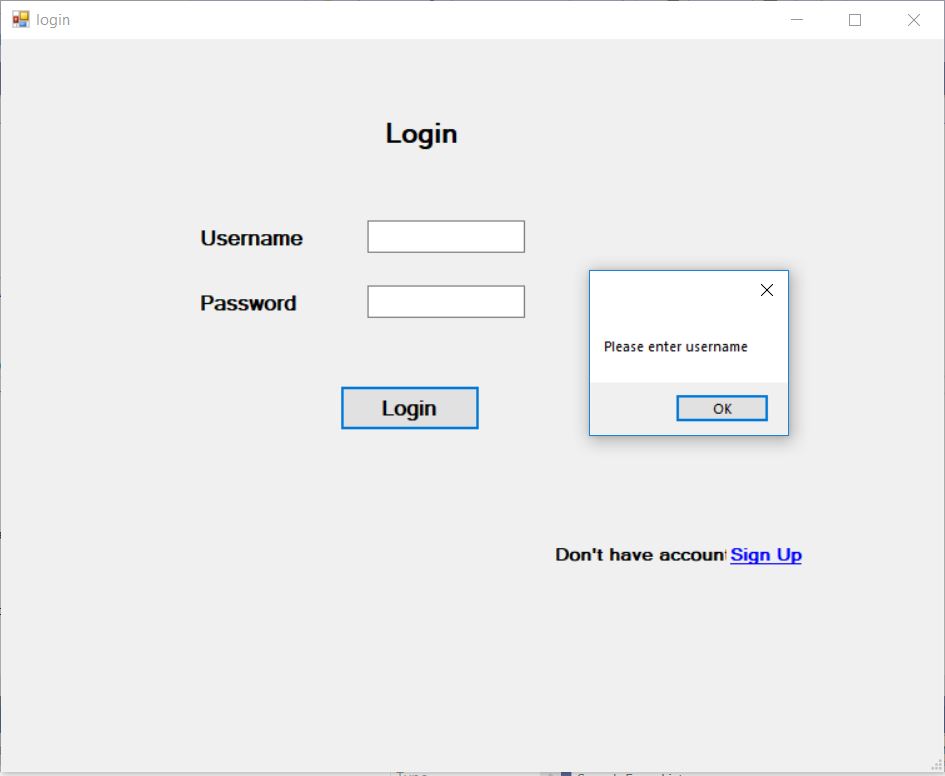


Figure 18: testing of login form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 2 | **Username**:abc  **Password**: <blank> | Show error message | Please enter password | Pass |

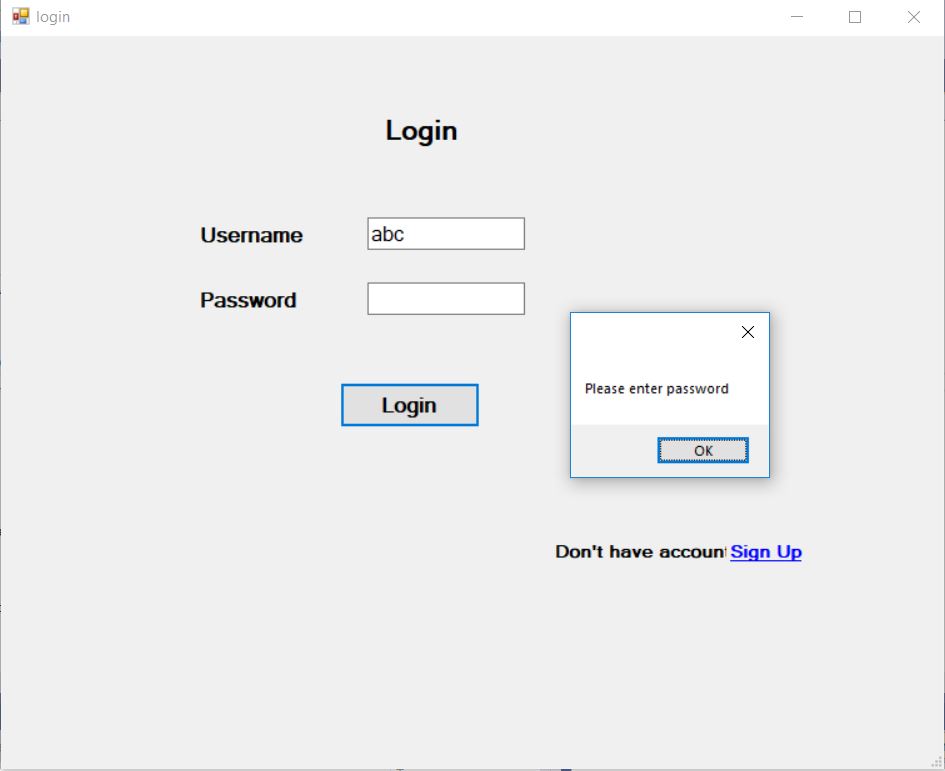


Figure 19: testing of login form (2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 3 | **Username**: abc  **Password**: abc | Show error message | Username and Password not matched! | Pass |

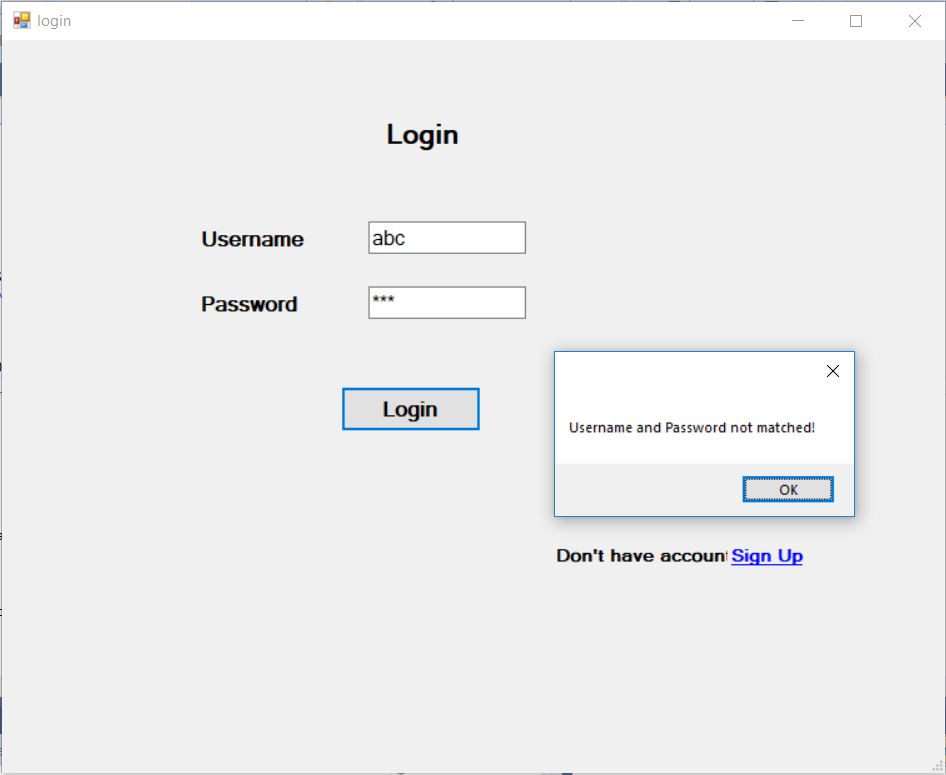


Figure 20: testing of login form (3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 4 | **Username**: admin  **Password**: admin | Show error message | Login successful! | Pass |

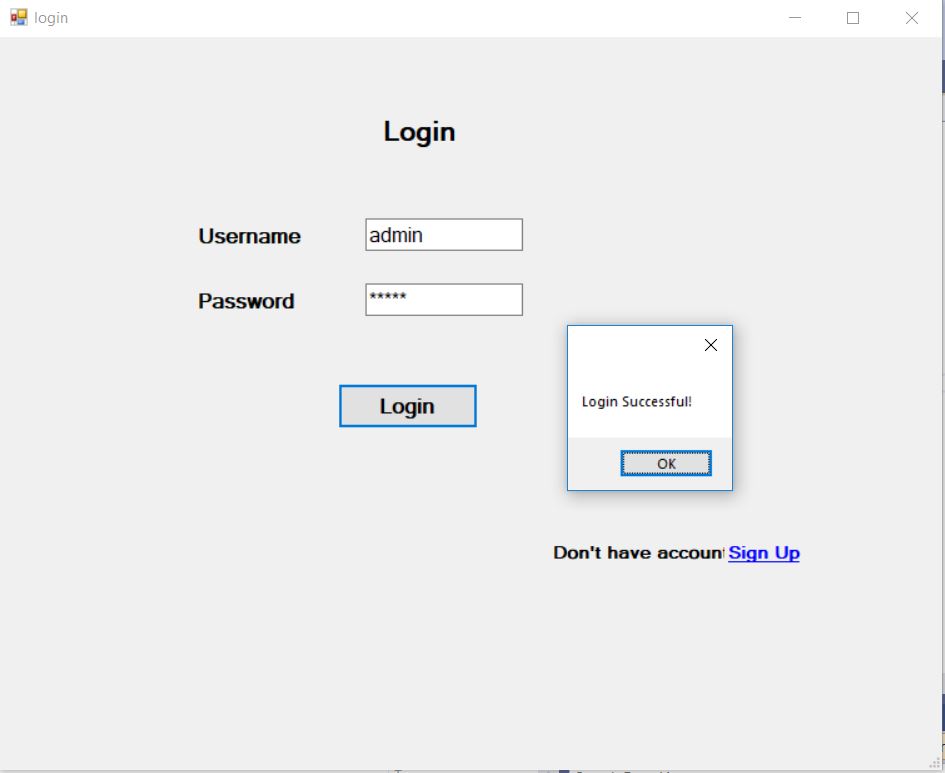


Figure 21: testing of login form (4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 5 | **Username**: <blank>  **First Name:** <blank>  **Last name:** <blank>  **Password**: <blank>  **Re-enter Password:** <blank> | Show error message | Please enter username | Pass |

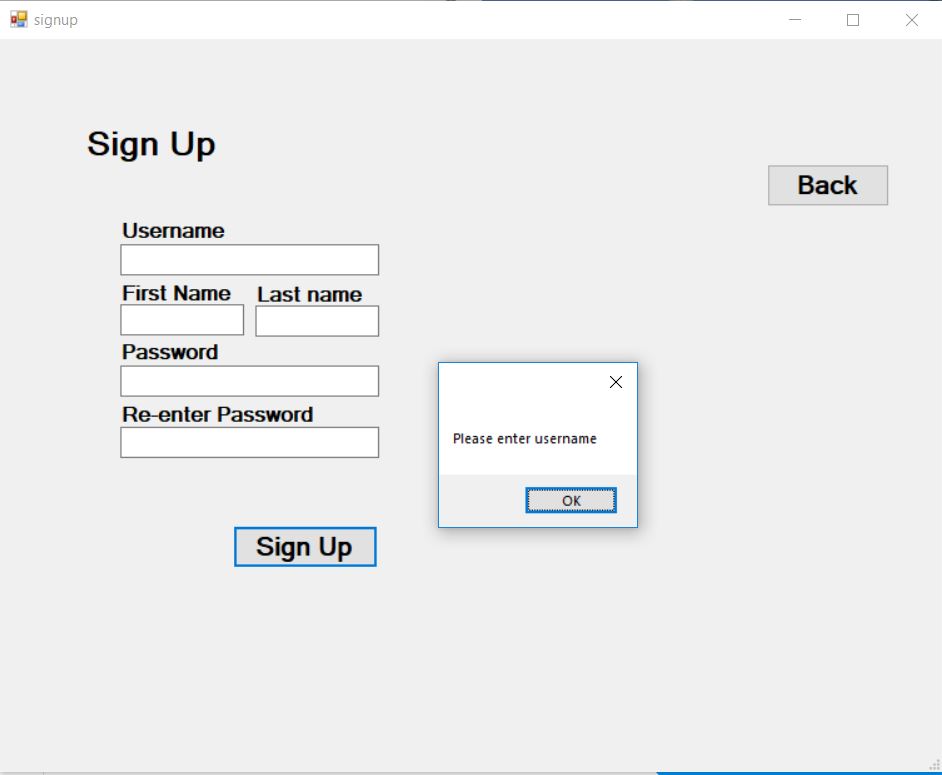


Figure 22: testing of signup form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 6 | **Username**: admin  **First Name:** <blank>  **Last name:** <blank>  **Password**: <blank>  **Re-enter Password:** <blank> | Show error message | Please enter Firstname | Pass |

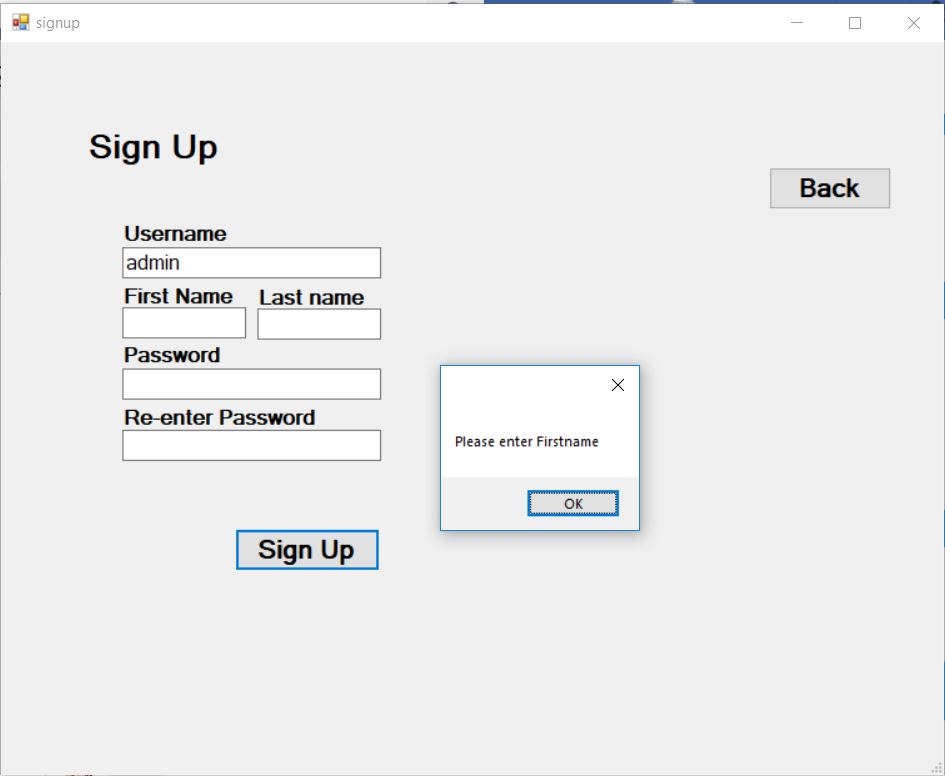


Figure 23: testing of signup form (2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 7 | **Username**: admin  **First Name:** admin  **Last name:**  **Password**: <blank>  **Re-enter Password:** <blank> | Show error message | Please enter Lastname | Pass |

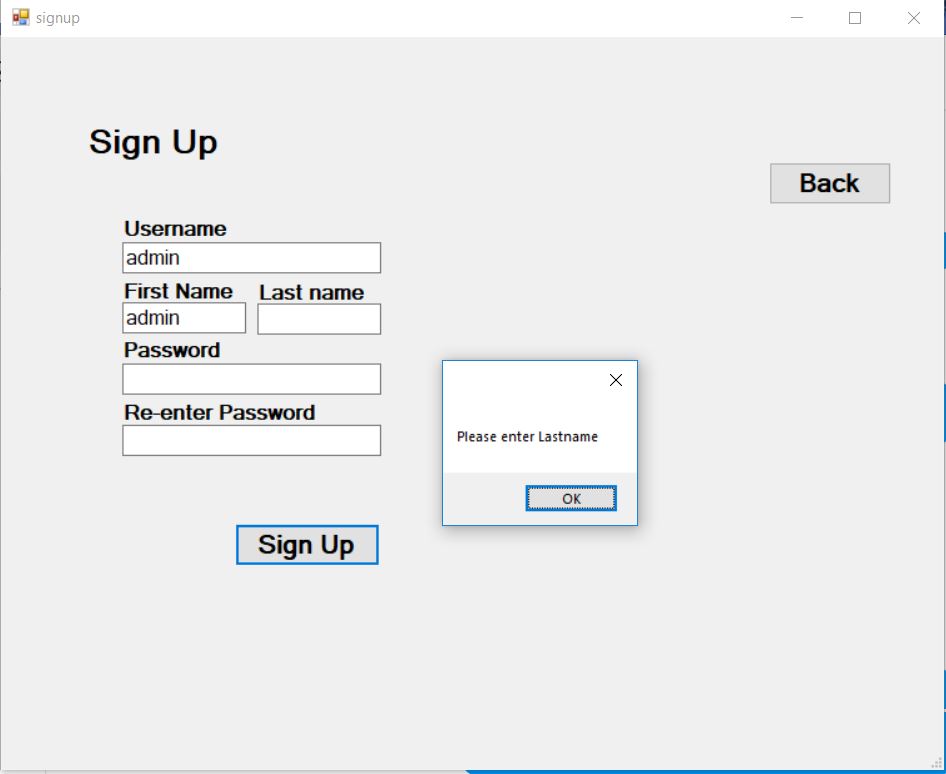


Figure 24: testing of signup form (3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 8 | **Username**: admin  **First Name:** admin  **Last name:** admin  **Password**: <blank>  **Re-enter Password:** <blank> | Show error message | Please enter password | Pass |

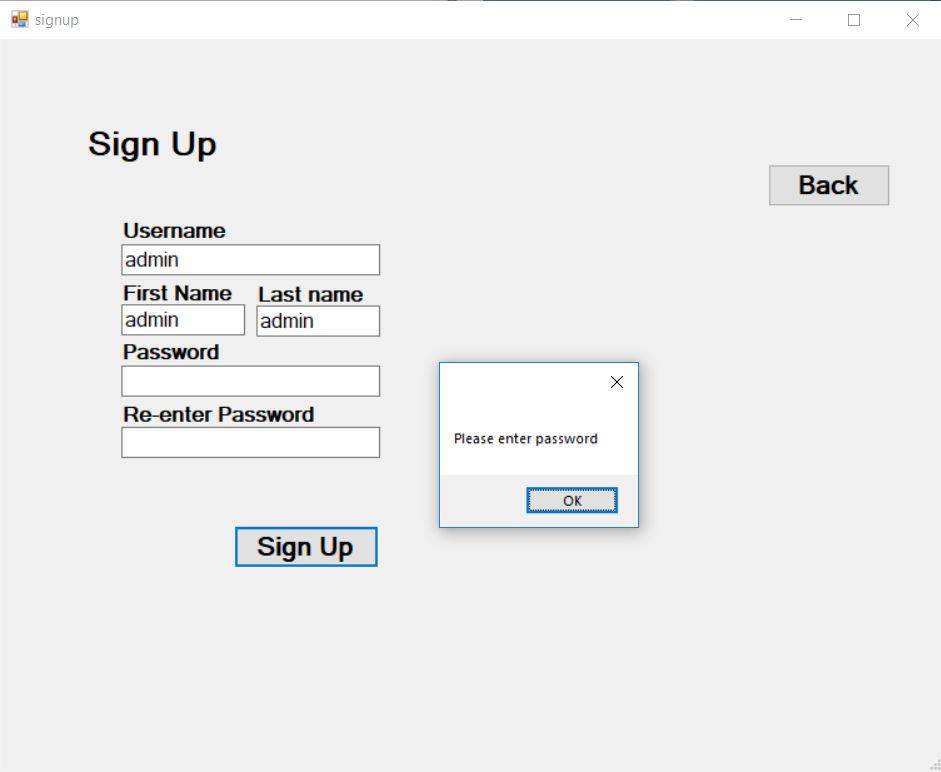


Figure 25: testing of signup form (4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 9 | **Username**: admin  **First Name:** admin  **Last name:** admin  **Password**: admin  **Re-enter Password:** <blank> | Show error message | Please enter re-password | Pass |

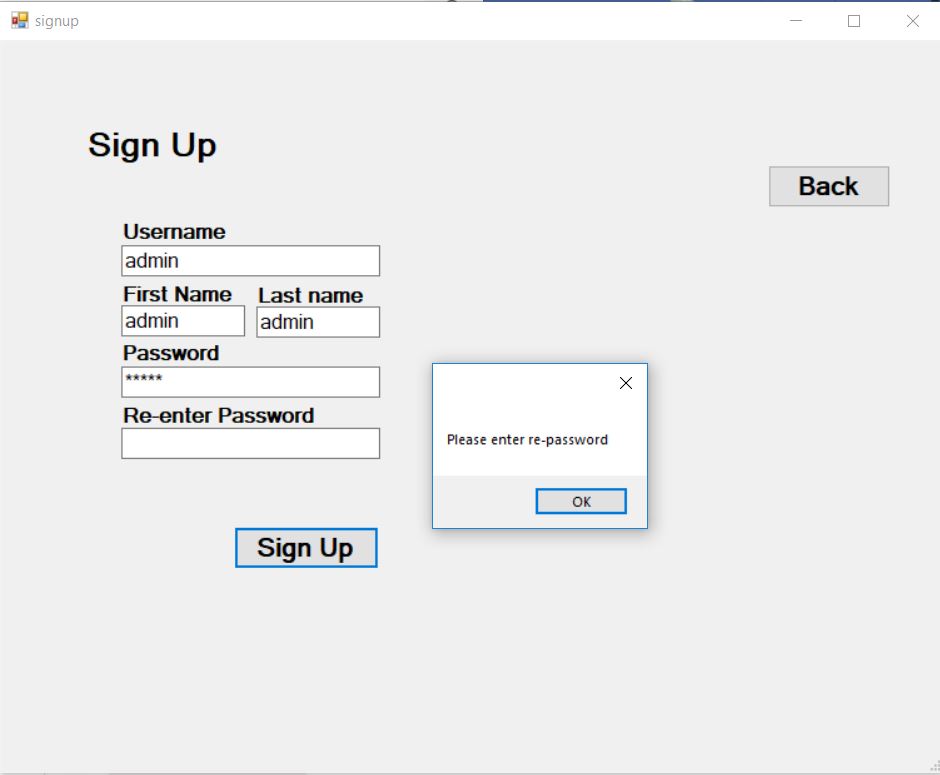


Figure 26: testing of signup form (5)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 10 | **Username**: admin  **First Name:** admin  **Last name:** admin  **Password**: admin  **Re-enter Password:** a | Show error message | Password and re-password does not match | Pass |

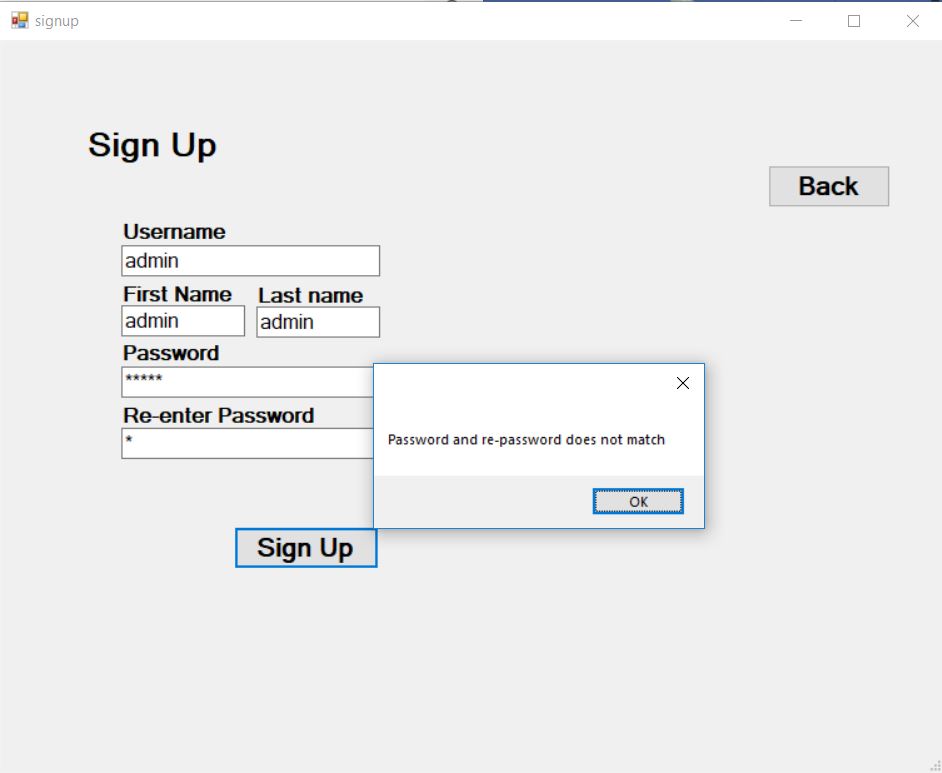


Figure 27: testing of signup form (6)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 11 | **Username**: admin  **First Name:** admin  **Last name:** admin  **Password**: admin  **Re-enter Password:** admin | Show error message | This username is already taken | Pass |

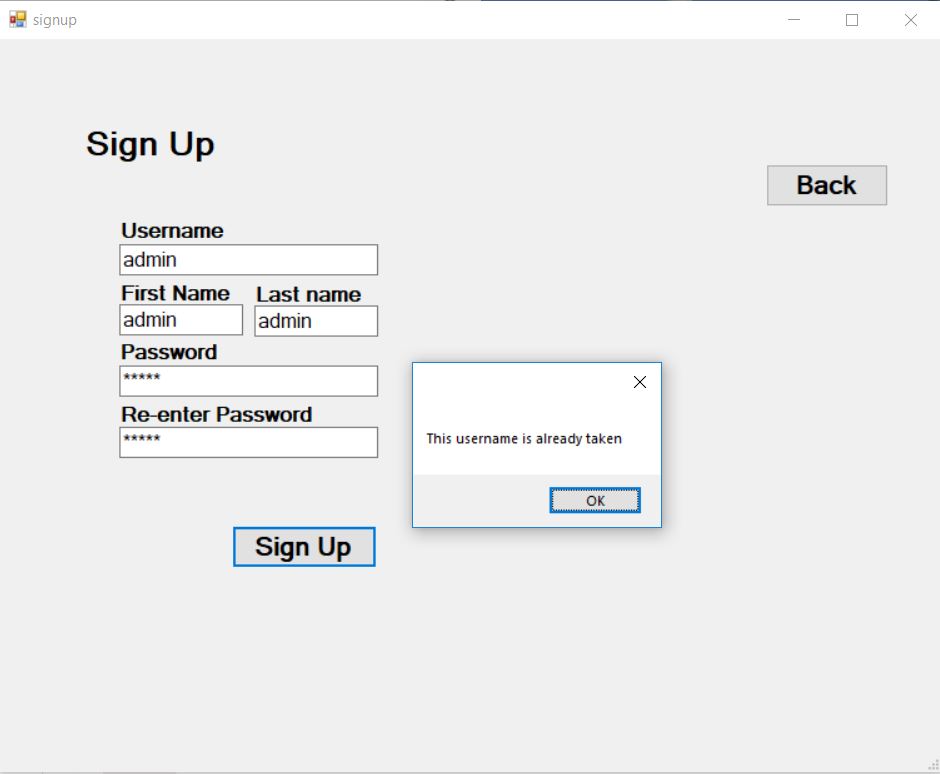


Figure 28: testing of signup form (7)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 12 | **Username**: admin  **First Name:** admin  **Last name:** admin  **Password**: admin  **Re-enter Password:** admin | Show error message | Registration successful!! | Pass |

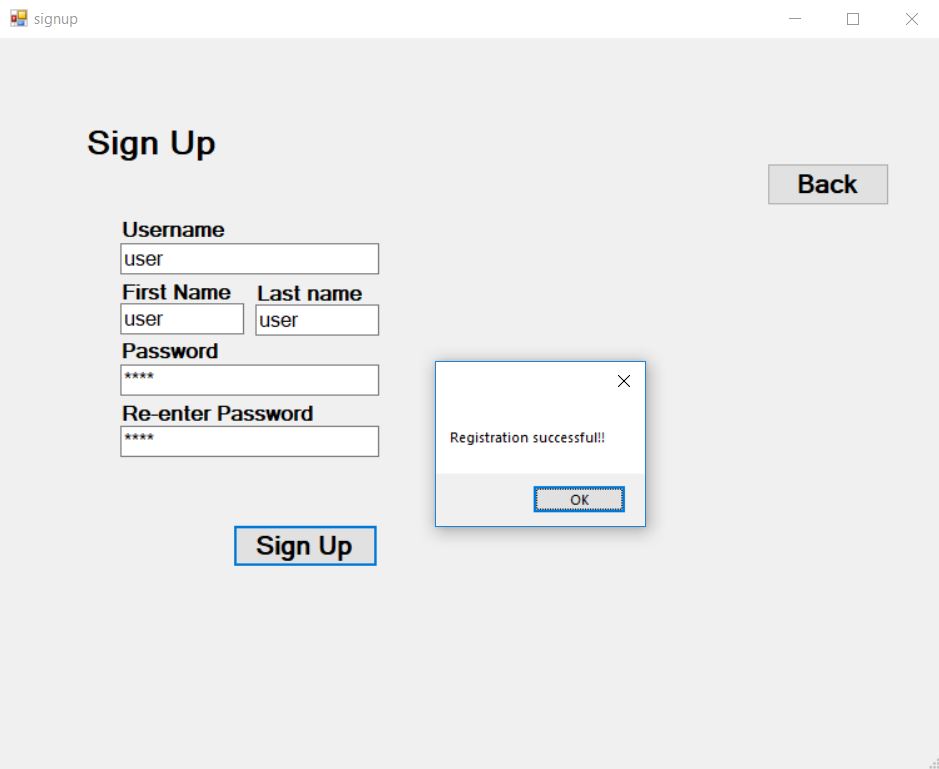


Figure 29: testing of signup form (8)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 13 | **Category Name:** <blank>  **Description:** <blank>  **Category Type:** Income | Show error message | Please enter Name | Pass |

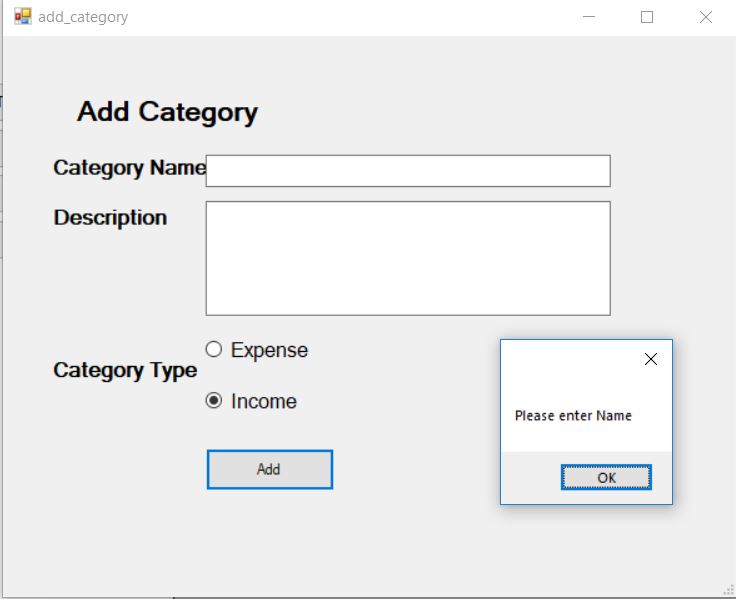


Figure 30: testing of add category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 14 | **Category Name:** salary  **Description:** <blank>  **Category Type:** Income | Show error message | Value added successfully | Pass |

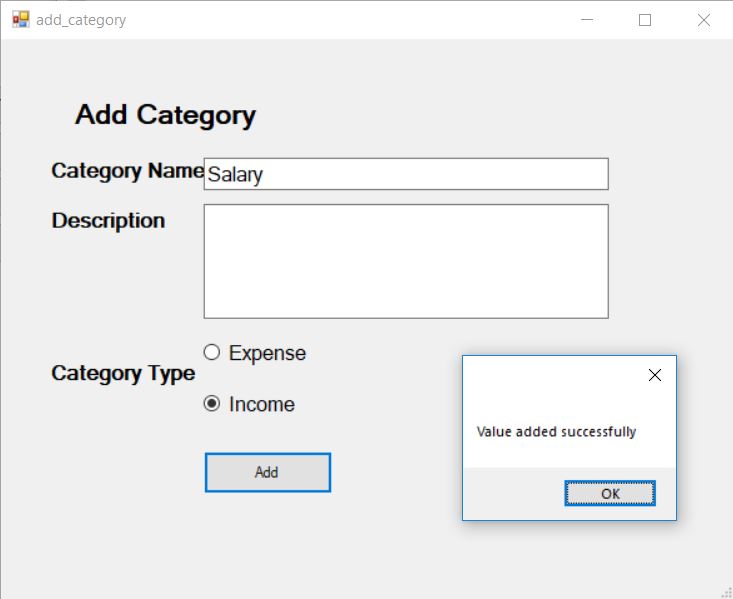


Figure 31: testing of add category (2)

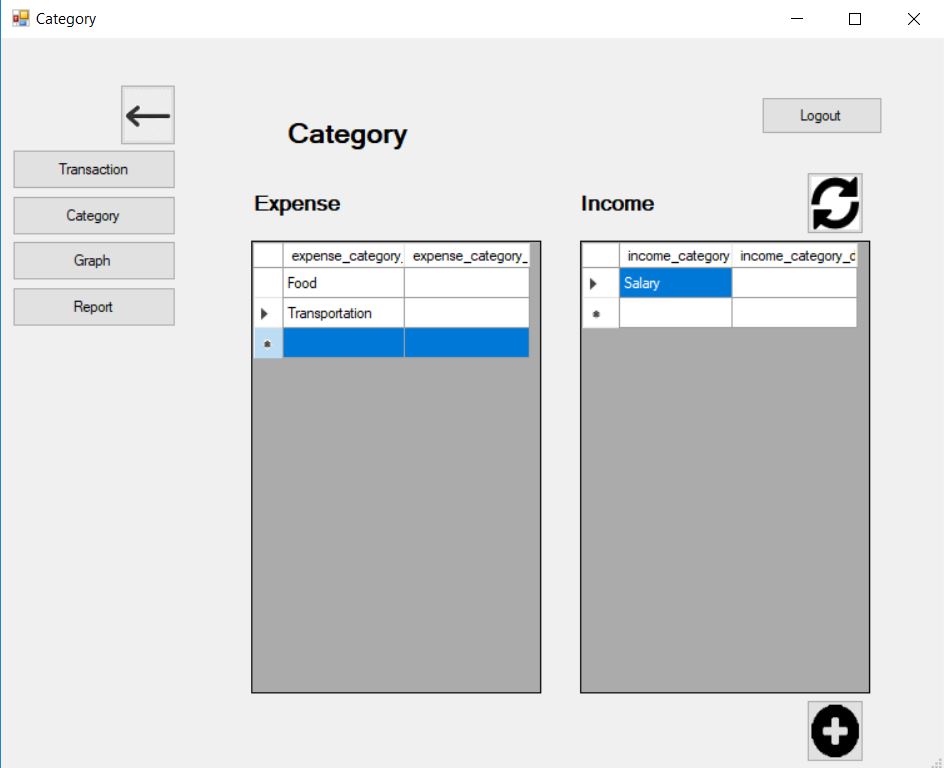


Figure 32: testing of add category (3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 15 | **Add:** Income  **Amount:** 5  **Date:** 9/30/2019  **Category:** salary  **Description:** <blank> | Show error message | Value added successfully | Pass |

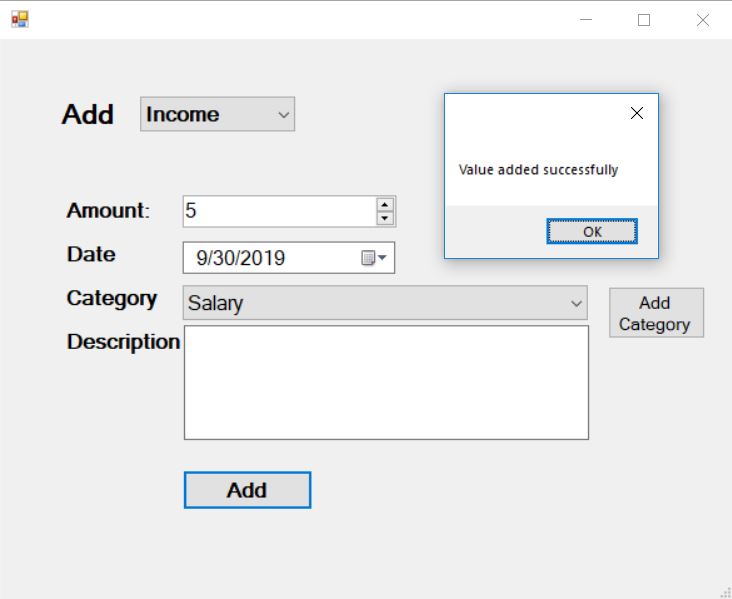


Figure 33: testing of add transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 16 | **Add:** <blank>  **Amount:** 0  **Date:** 9/30/2019  **Category:** <blank>  **Description:** <blank> | Show error message | Please enter Type | Pass |

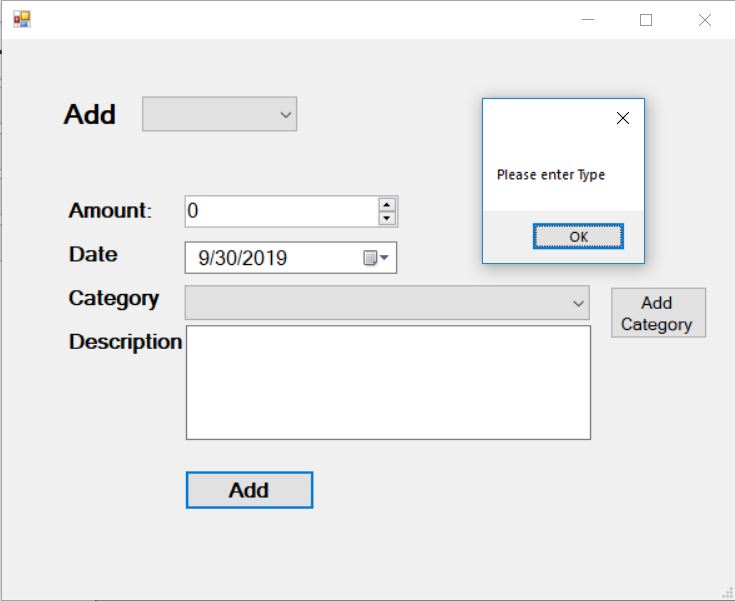


Figure 34: testing of add transaction (2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Test data** | **Expected result** | **Actual result** | **Result(Pass/Fail)** |
| 17 | **Add:** Income  **Amount:** 0  **Date:** 9/30/2019  **Category:** <blank>  **Description:** <blank> | Show error message | Please enter Category | Pass |

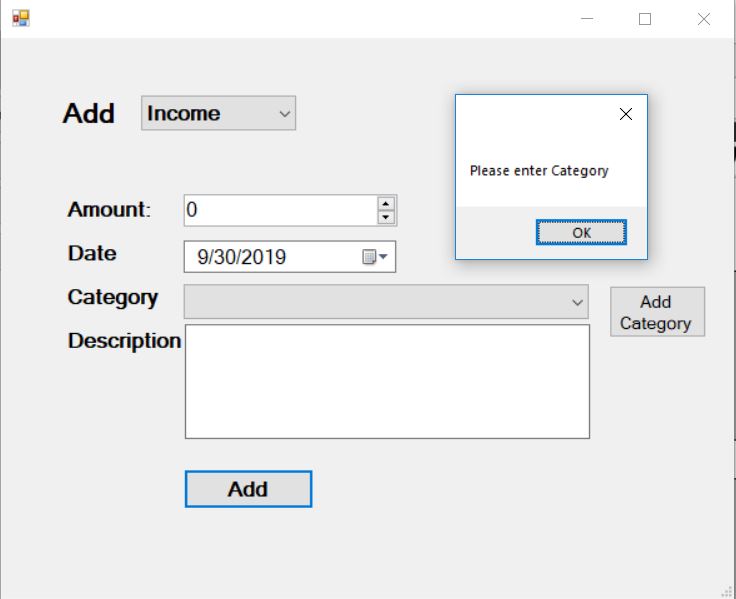


Figure 35: testing of add transaction (3)

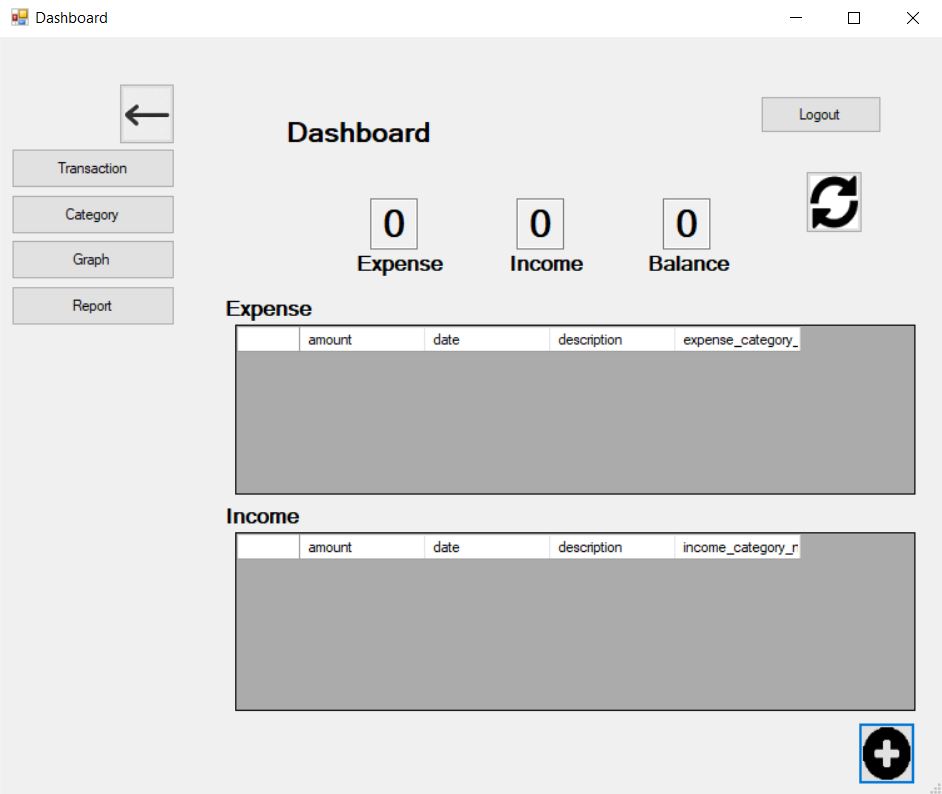


Figure 36: testing of add transaction (4)

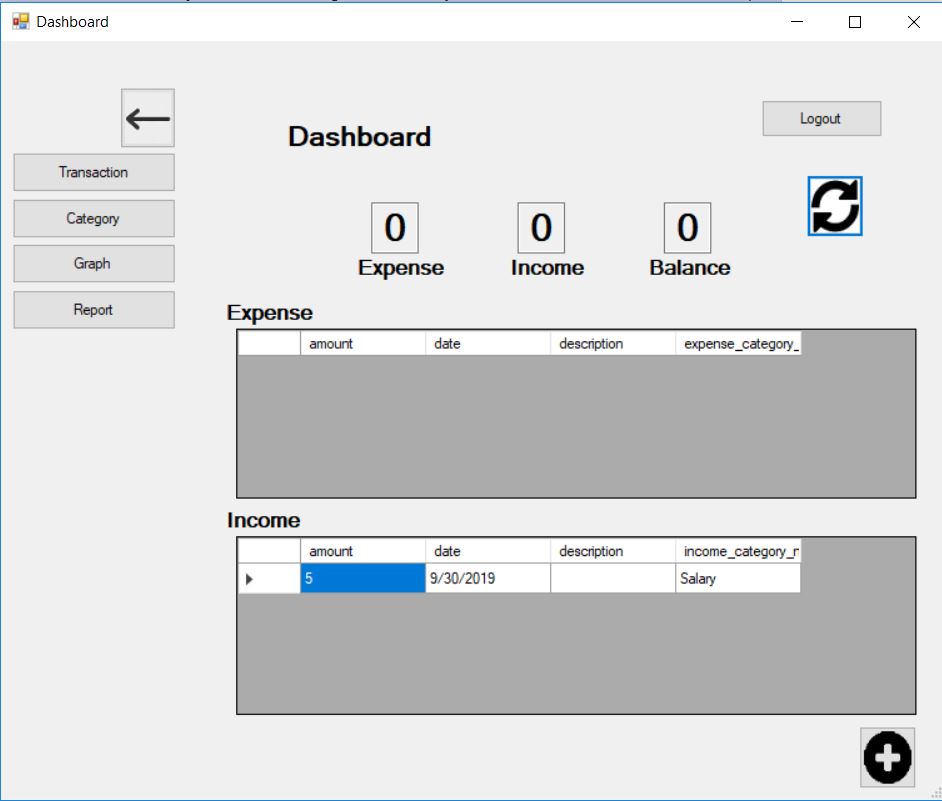


Figure 37: testing of add transaction (5)

## Unit Testing

1. Test for login fail



Figure 38: unit test code for login

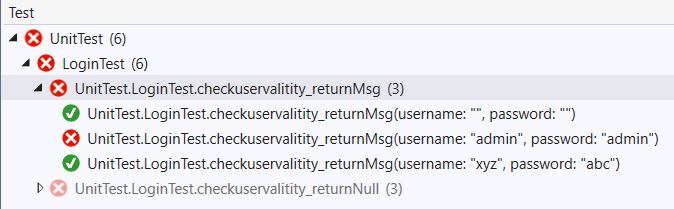


Figure 39: unit test result for login fail

1. Test for login pass



Figure 40: unit test code for login pass



Figure 41: unit test result for login pass

1. Test for validation fail



Figure 42: unit test code for validation fail

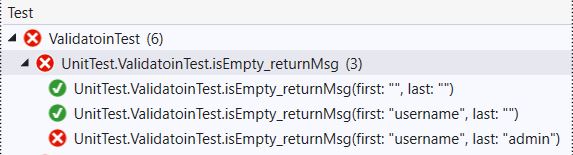


Figure 43: unit test result for validation fail

1. Test for validation pass

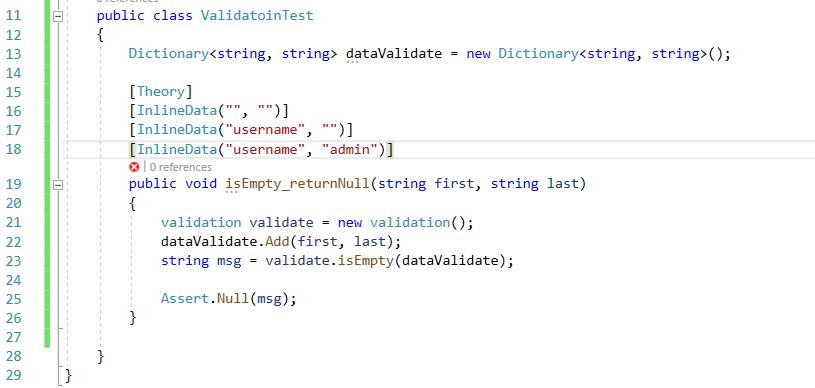


Figure 44: unit test code for validation pass

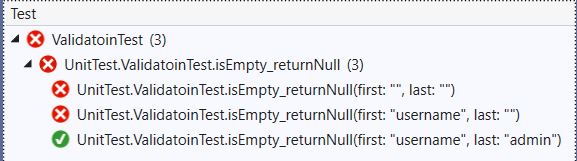


Figure 45: unit test result for validation pass

# **Chapter 6: Other project issues**

## Limitation

All systems have limitations. Similarly there are also few limitations to this project. The first limitation would be that the data are not fully secure as anyone with the access to the database can view the data as the data is not encrypted. This system also does not have the feature for data backup so, once a data is lost it may be hard to recover.

## Future work

The future work to my project would be making it more advanced and secure. More features can be added to this system to make it more useful in our day to day life. Some of the futures can be:

1. Link to bank account

By linking to the personal bank account to the system user can automatically get data about their current balance. They can also help keep record of the expense that they use through their bank account or ATM cards. This can reduce the tedious work to track each and every expenses and incomes.

1. Data Backup

Backup can be very useful. Even if the data is lost it can be recovered in the future. This helps in securing the data and make the data available when necessary. It can also be useful when hardware needs to be replaced. For example if a phone is damaged then the backup file can be recovered in the new phone.

1. Bug fixing

There may be few boxes in the system. Though the system was tested well there may still be bugs that I could not identify.

1. Commercial use

The current system is only for personal use but after few advancements it can be used for organizations as well to track their transactions.

## Risk Management

Process of identifying, assessing and controlling of threats that can harm an organization's capital and earnings are called Risk Management. These threats or risks hampers organization’s financial uncertainty, legal liabilities, strategic management errors, accidents, etc.

Security from threats and data-related risks has become a main priority for digitized companies. A risk management plan includes a complete process for identifying and controlling threats. These plans ensure security for digital assets like data, customer's personally identifiable information and intellectual property.

Risk management plans follows the same steps to overall risk management process:

1. **Risk identification**

The company identifies and defines Potential risks are identified and defined as per their impact on company’s process or project.

1. **Risk analysis**

Once specific types of risk are identified, the company then determines their occurrence and its consequences. In this stage risks are further understood.

1. **Risk assessment and evaluation**

Risk are evaluated for determining likelihood of occurrence combined with its overall consequence. The decisions of whether or not the risk is acceptable and whether the company is willing to take it on based on its risk appetite.

1. **Risk mitigation**

Using specific risk controls, companies develop a plan to lighten them. These plans include risk prevention tactics, risk mitigation processes and contingency plans, etc.

1. **Risk monitoring**

To continuously monitor and track new/ existing risks, part of the mitigation plan are carried out.

**Likelihood table:**

|  |  |
| --- | --- |
| **Likelihood** | **Value** |
| Low | 1 |
| Medium | 2 |
| High | 3 |

**Consequences table:**

|  |  |
| --- | --- |
| **Consequence** | **Value** |
| Very low | 1 |
| Low | 2 |
| Medium | 3 |
| High | 4 |
| Very high | 5 |

**Risk Management table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Risk** | **Likelihood** | **Consequence** | **Impact** | **Action** |
| 1 | Scope not well defined | 2 | 4 | 8 | Analysis of requirement must be done properly in the beginning of the project. |
| 2 | Virus thread | 2 | 3 | 6 | Daily scan and installation of antivirus. |
| 3 | Hard disk crash | 1 | 5 | 5 | Frequent data backup. |
| 4 | Shortage of resources | 2 | 5 | 10 | The resources required for the project should be identified and gathered. |
| 5 | Lack of management | 3 | 3 | 9 | The project should be well managed and have backup plans |
| 6 | Failure to follow methodology | 1 | 5 | 5 | Methodology should be choose considering all the aspects. |

Table 12: risk management

During the development of the project I faced few problems but with help of the risk plan I could recover from the risks and continue with the project.

## Configuration management

Configuration management is shared figure of tools, activities, processes and methods. Workers of this project can use to manage items during the project life cycle.

For this I have used GitHub to manages files and folder that are related to the project. And at the time of necessity files can be accesses easily.

The link to access the project is:

<https://github.com/aman0010/CP-Project>

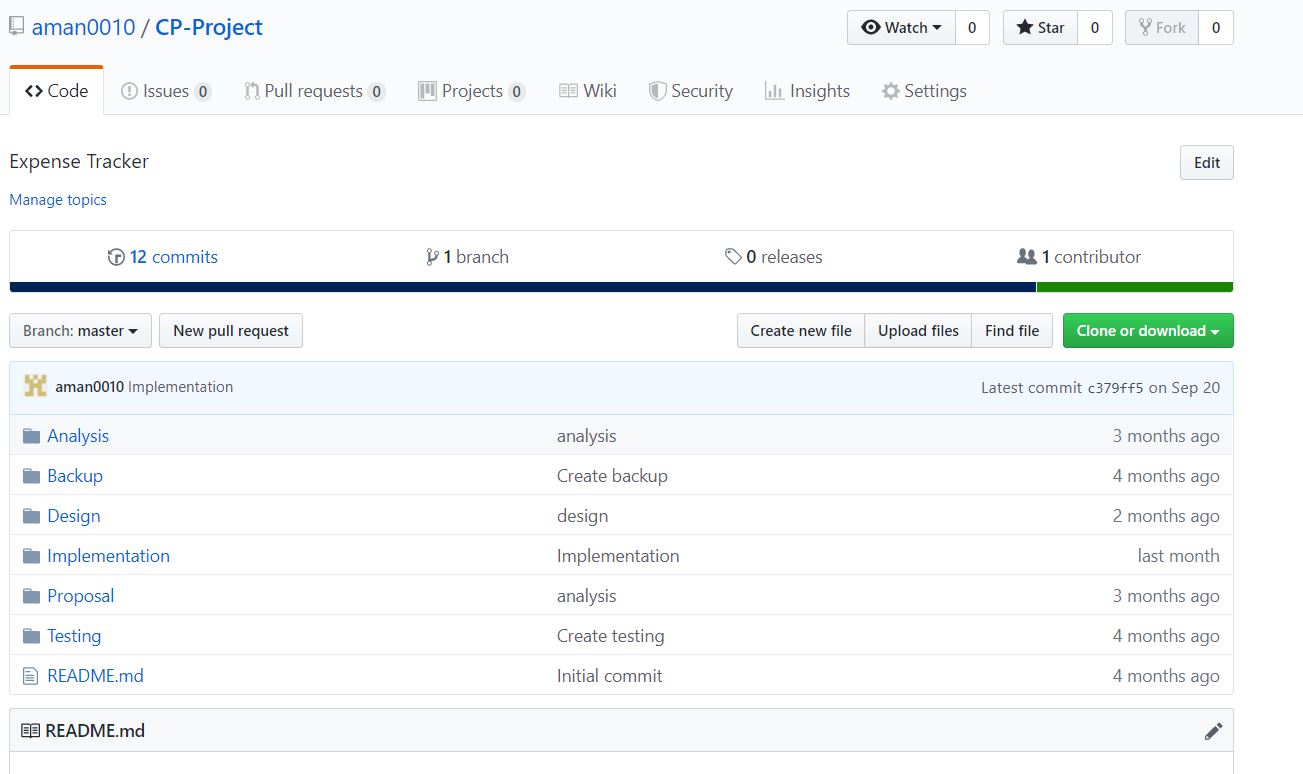


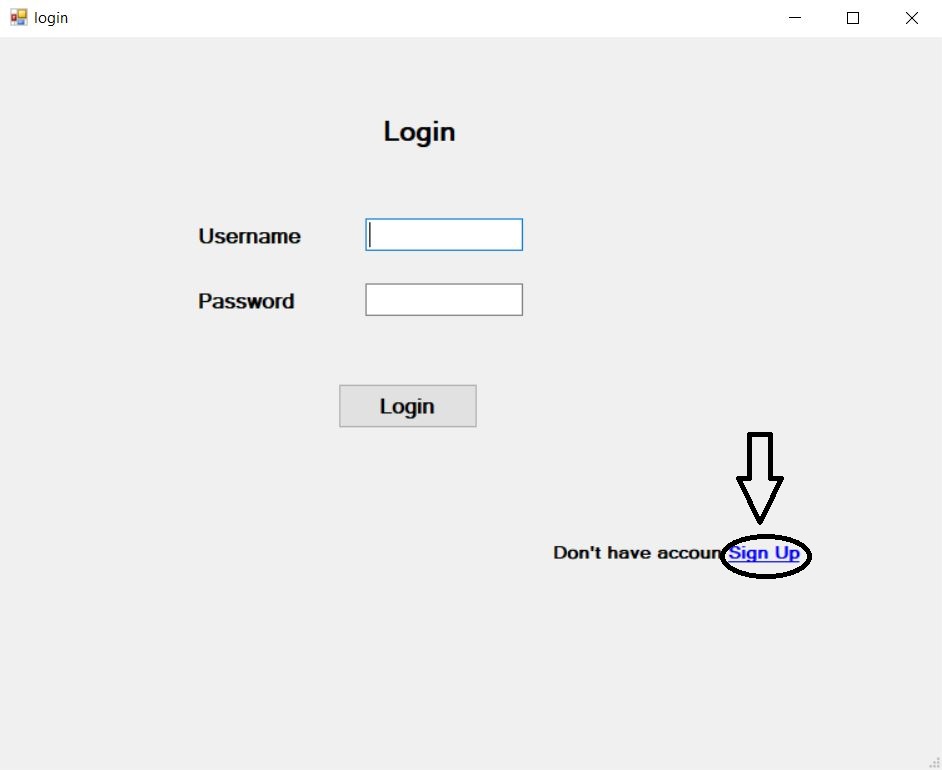
Figure 46: Configuration management

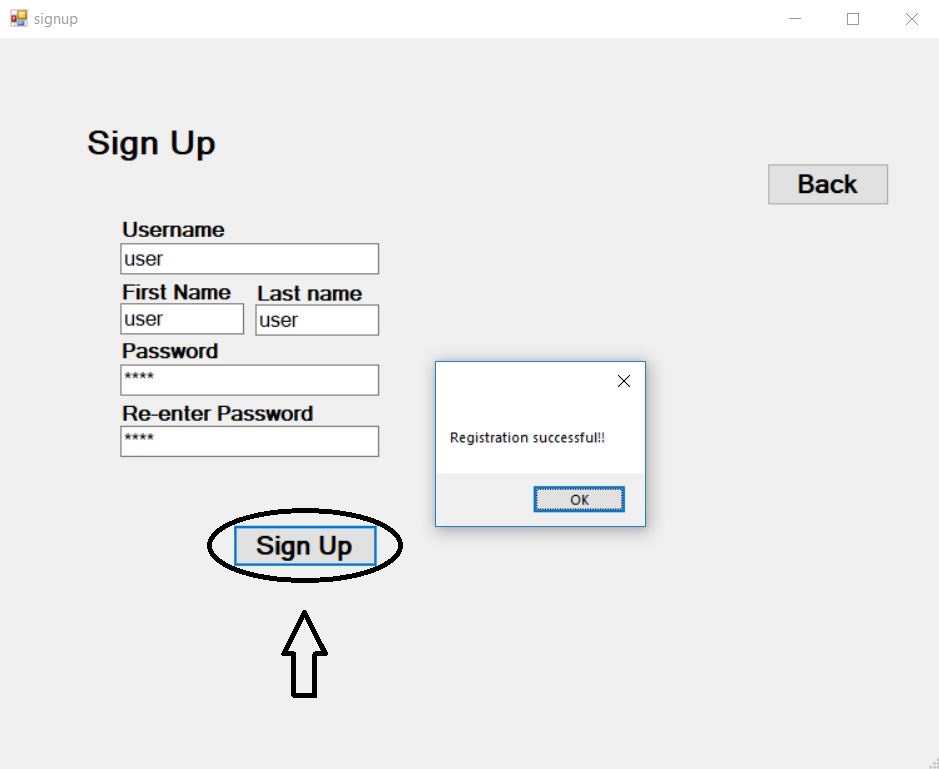
## Other project issues

While developing this project I faced few issues. Not totally following the design pattern was one of them I could not exactly make the best use of design pattern maybe due to my lack of knowledge. I did refer to the internet for help. It was not exactly as I expected but learned a lot. Managing the database was also a problem as I lost all the database file and had to create a new replica of the previous one. I also had problems when started coding. I didn’t know where to start. There were few mistakes at the first that I had to correct later. Overall there were few issues that held me back but solving them help me understand more.

## User manual

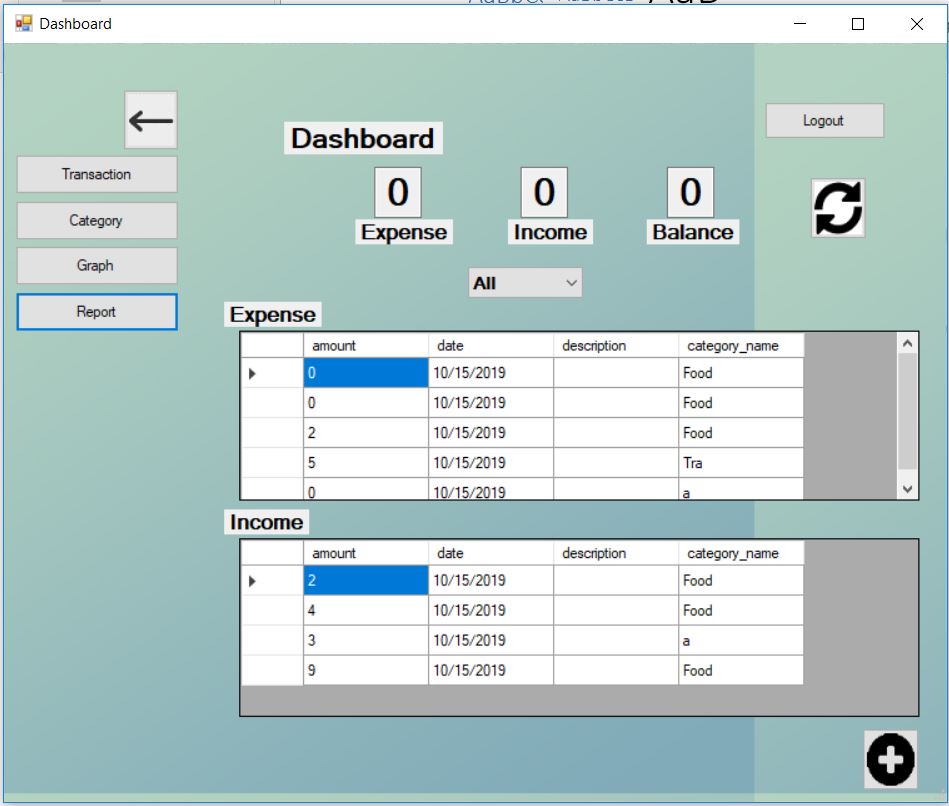
1. Getting Started (login and register)

At first you need to login to use the expense tracker. If you are not registered then you can register by clicking on the Sign Up text.

Then you should insert the details then click the Sign Up button

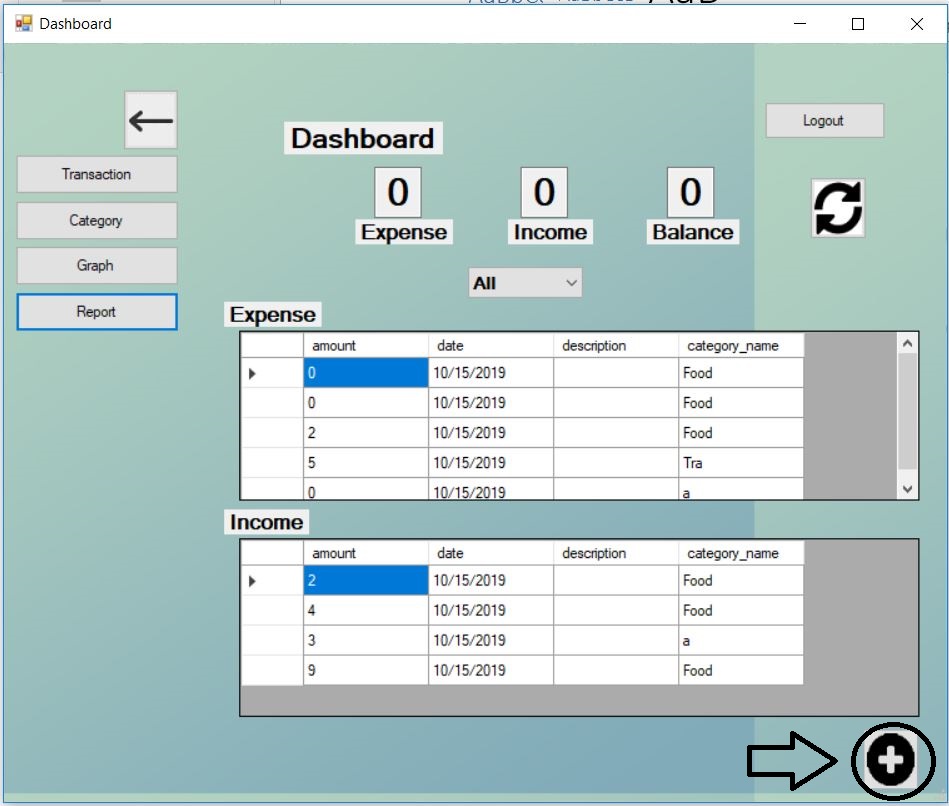
1. Navigating

After you login then dashboard is loaded then you can access other pages through the buttons at the left side.

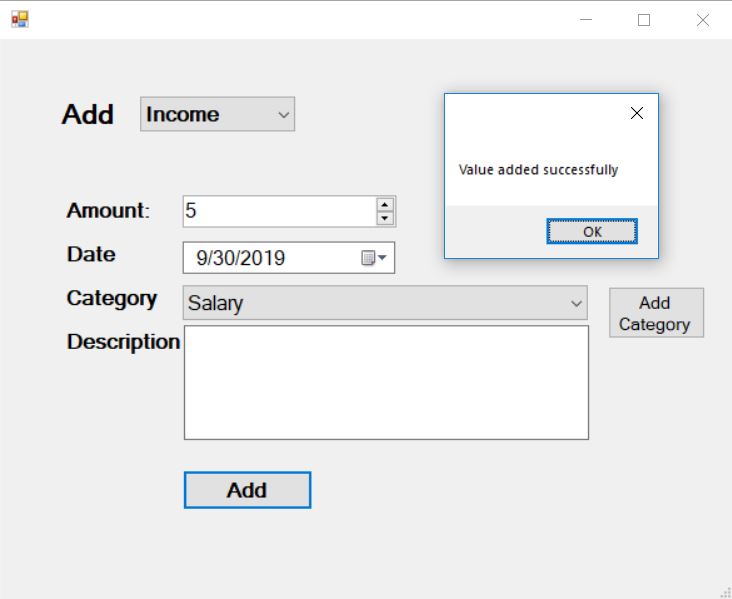


1. Adding transaction

* Click on the add button on the bottom right

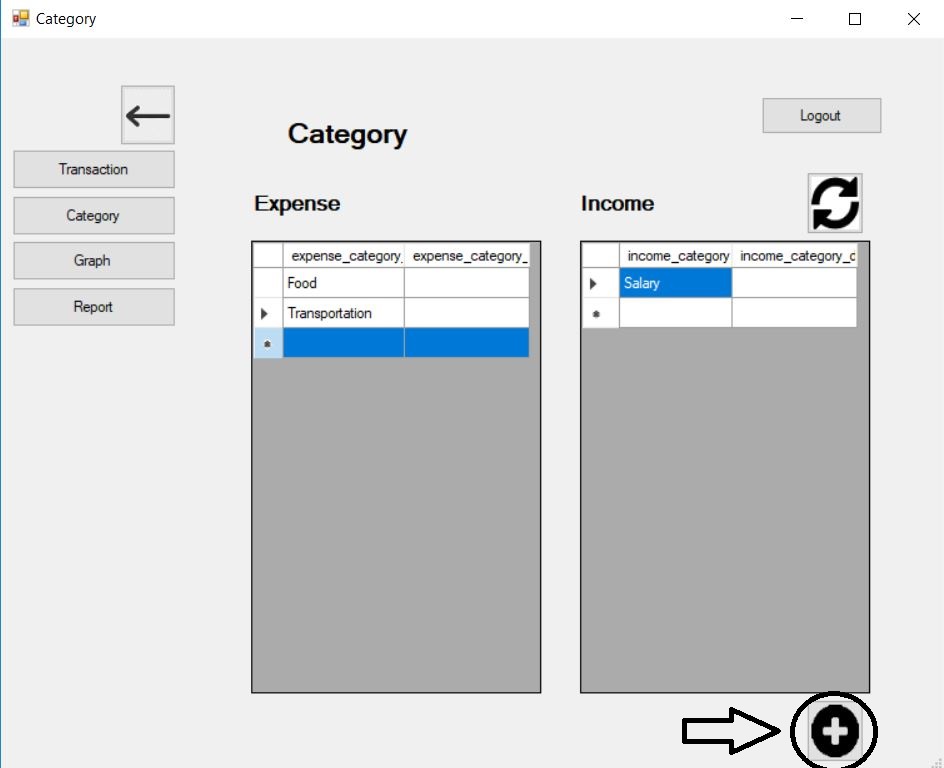


* Add all the details the press the Add button

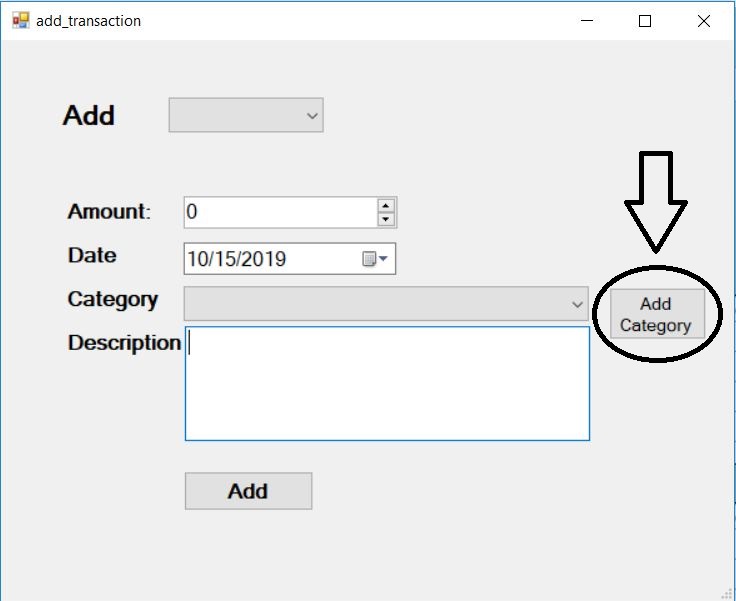
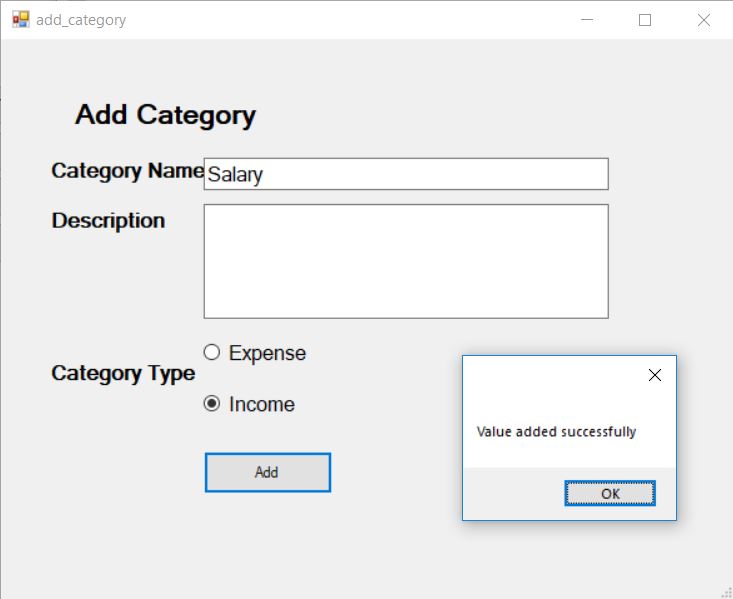


1. To add category

* You can either press go to the category form and press the add button

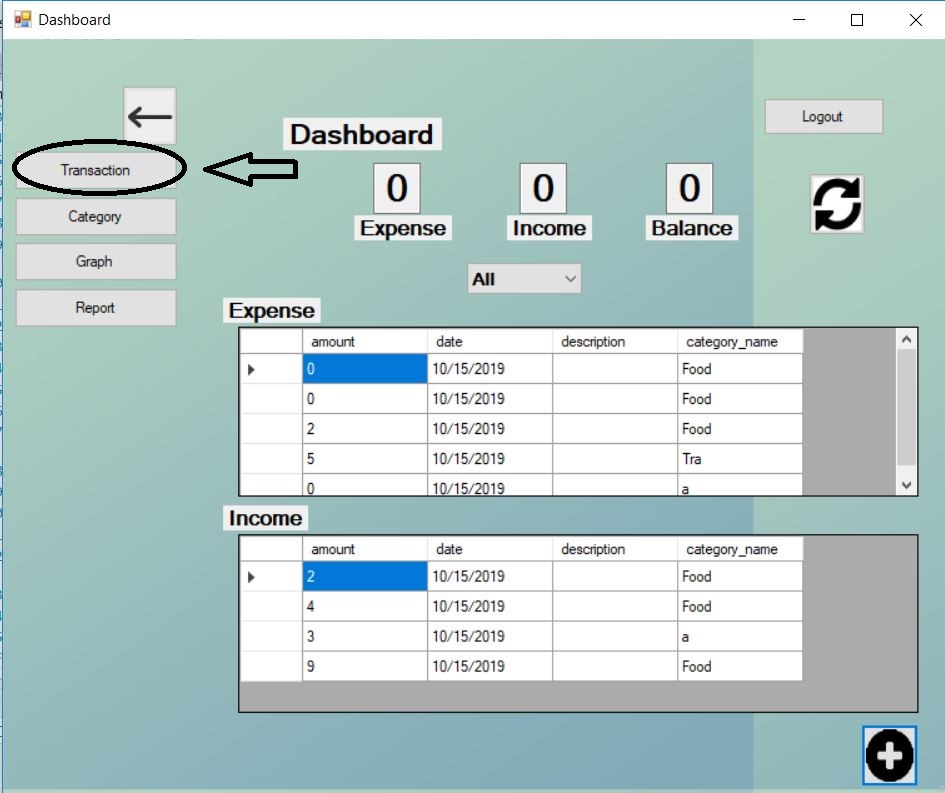


OR

* In the add transaction page you can press the Add Category button
* Then fill the form and press Add button

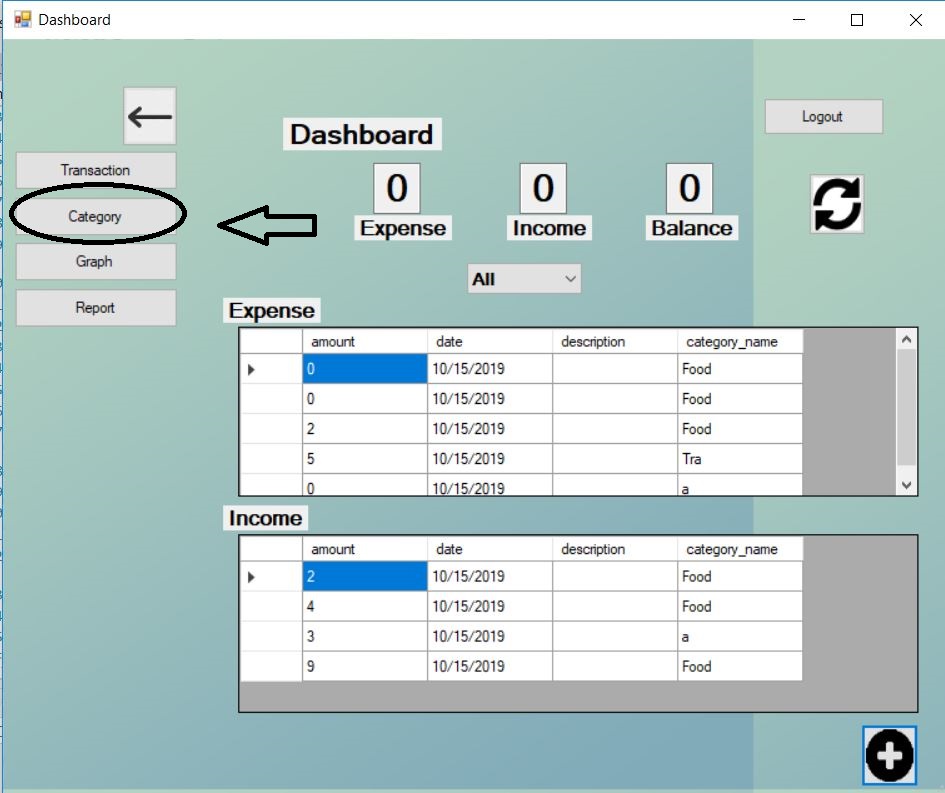
1. To view the transactions

* Press the Transaction Button on the left hand side. The data does not load you can refresh by clicking the refresh icon on the right hand side.



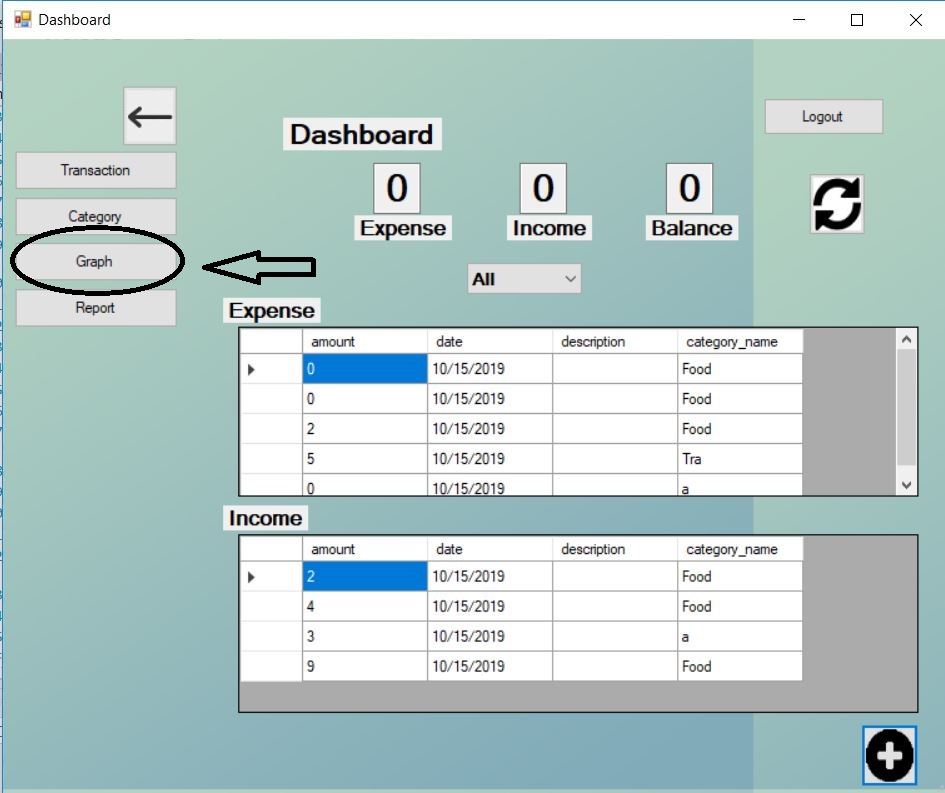
1. To view Category

* Press the Category Button on the left hand side. The data does not load you can refresh by clicking the refresh icon on the right hand side.



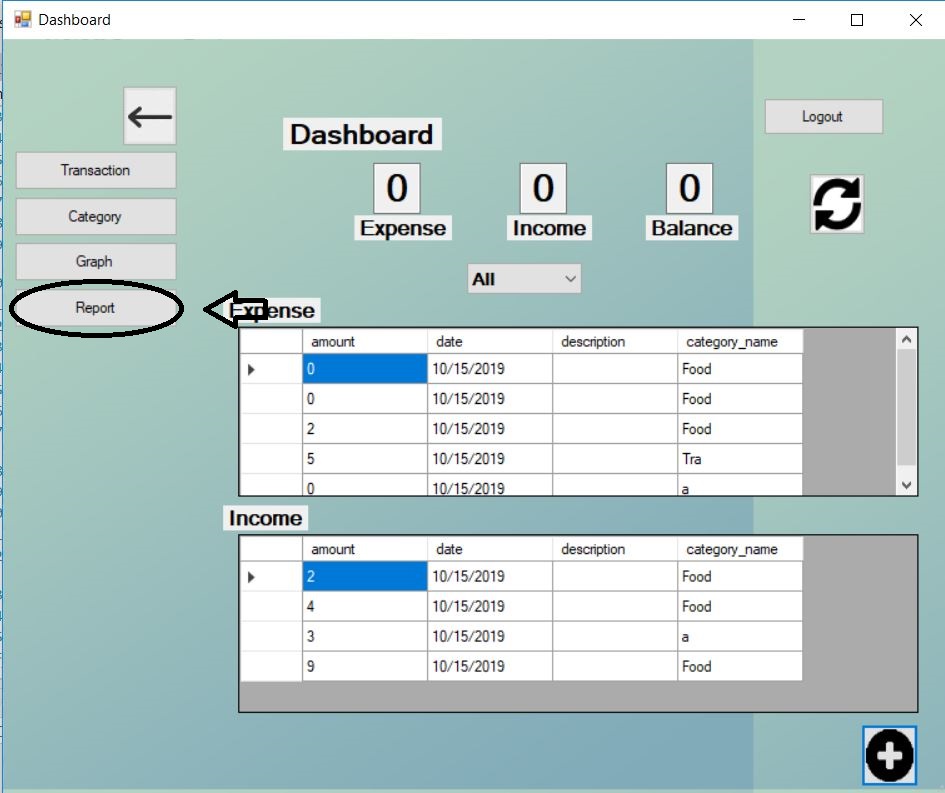
1. To view Graph

* Press the Graph Button on the left hand side. The data does not load you can refresh by clicking the refresh icon on the right hand side.



1. To View report

* Press the Report Button on the left hand side. The data does not load you can refresh by clicking the refresh icon on the right hand side.



# **Conclusion**

My attempt towards building an Expense Tracker has finally successful. I have tried my level best to come up with new ideas for this application.

Initially, learnt C# for developing this window based application. I have used SQL Server for writing SQL. Having no experience with C#, this project gave me a lot of knowledge and made me familiar with the bugs. Despite being new to this programming language, I have done my level best to come up with good features. With a view to attract users towards using this app, I have come up with features like graph representation and month wise view of income

# **Appendix**

**Model**

1. User\_model

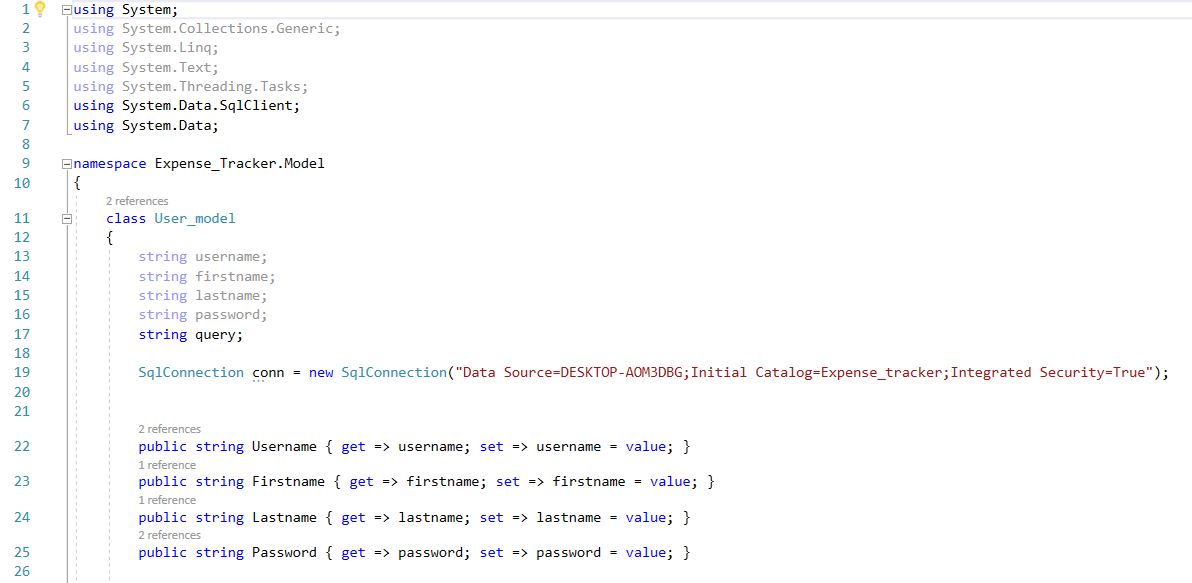


Figure 47: Code for user\_model

1. transaction\_model

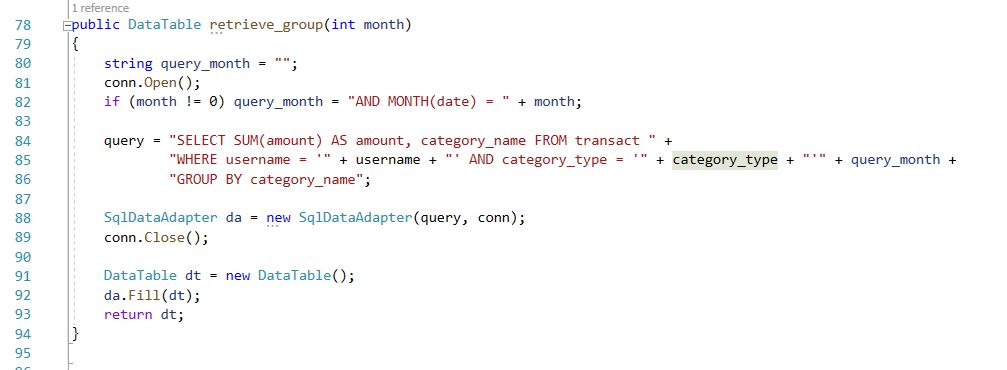
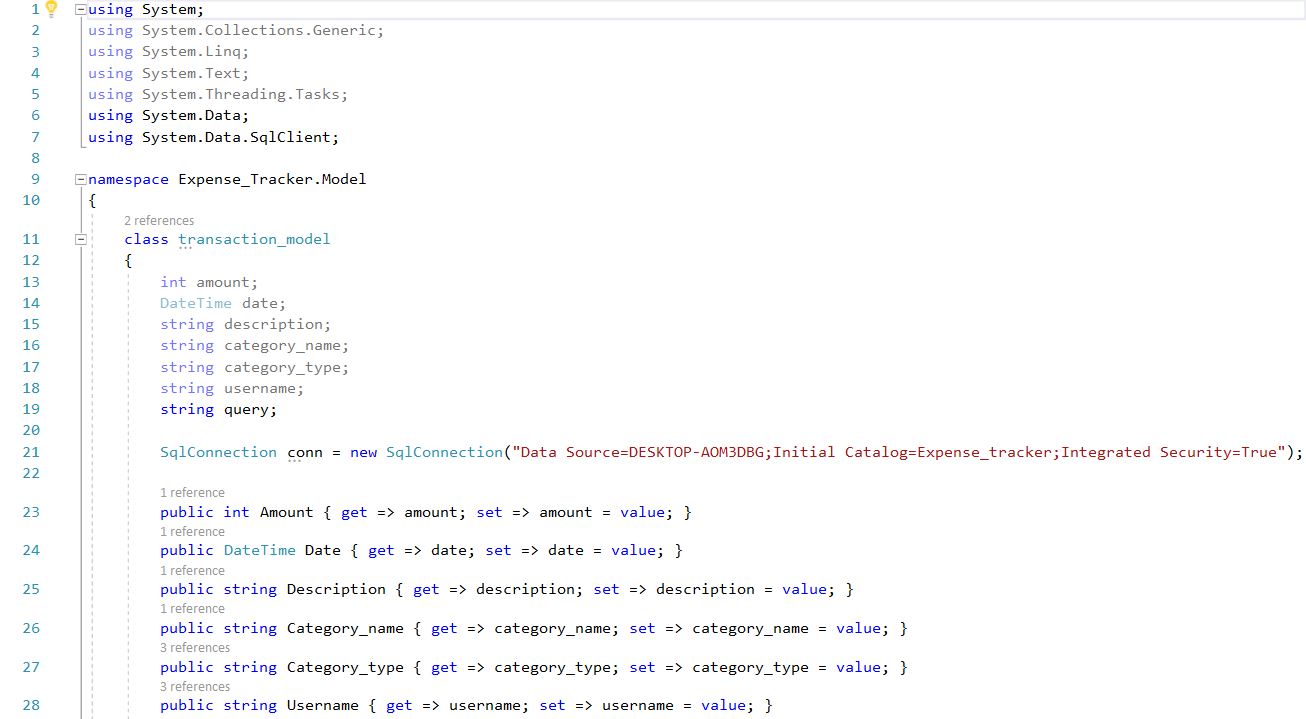


Figure 48: Code for transaction\_model

1. Category\_model

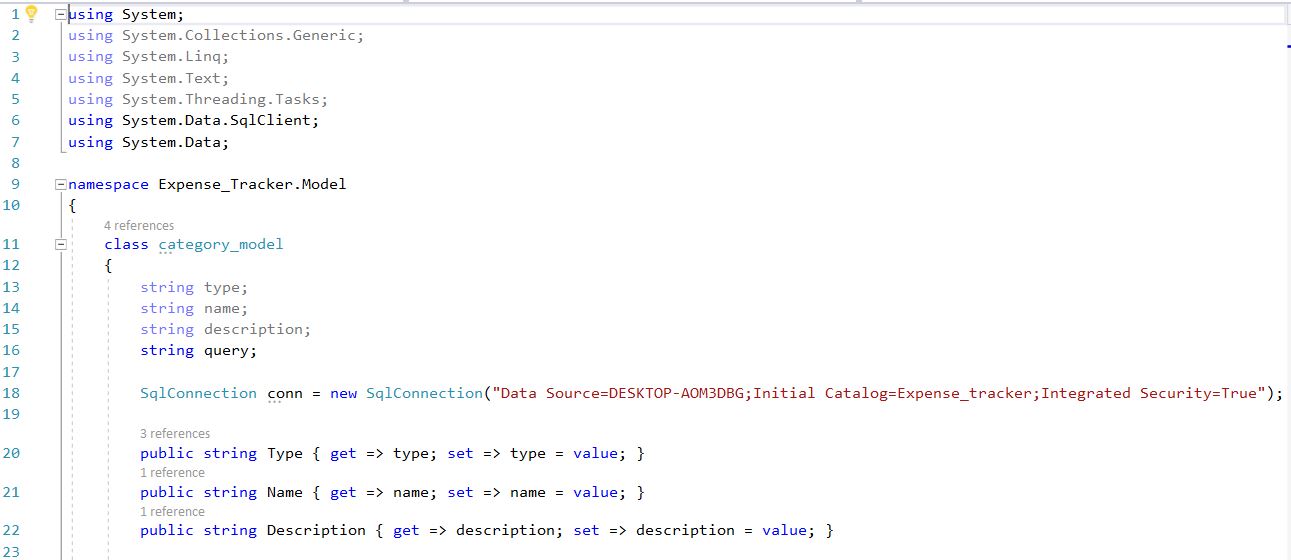


Figure 49: Code for category\_model

**Presenter**

1. User\_presenter

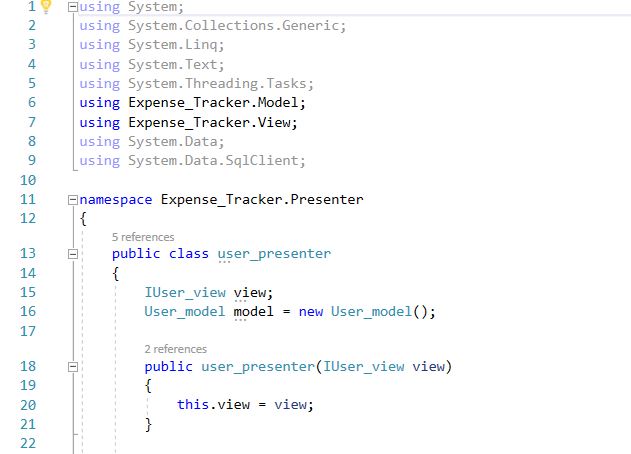


Figure 50: Code for user\_presenter

1. transaction\_presenter

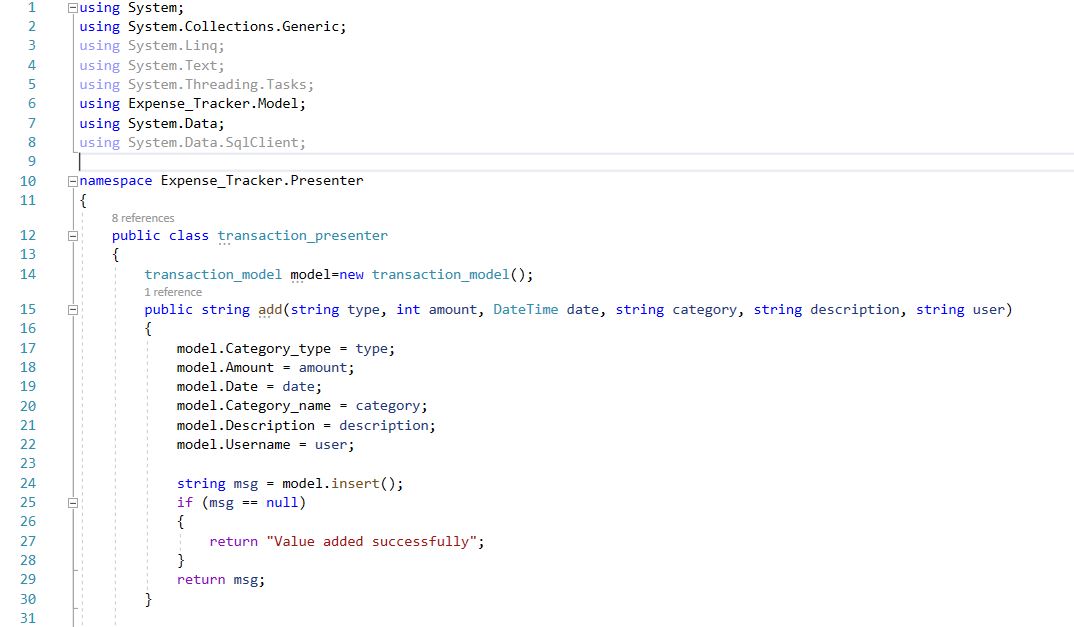
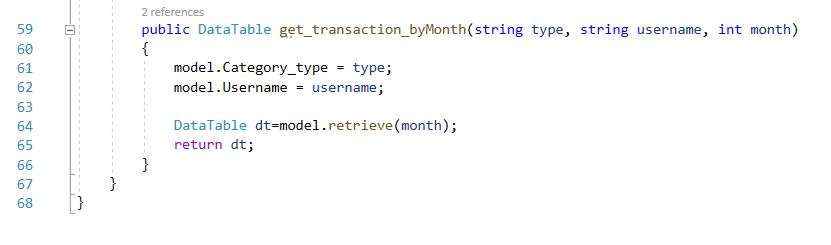
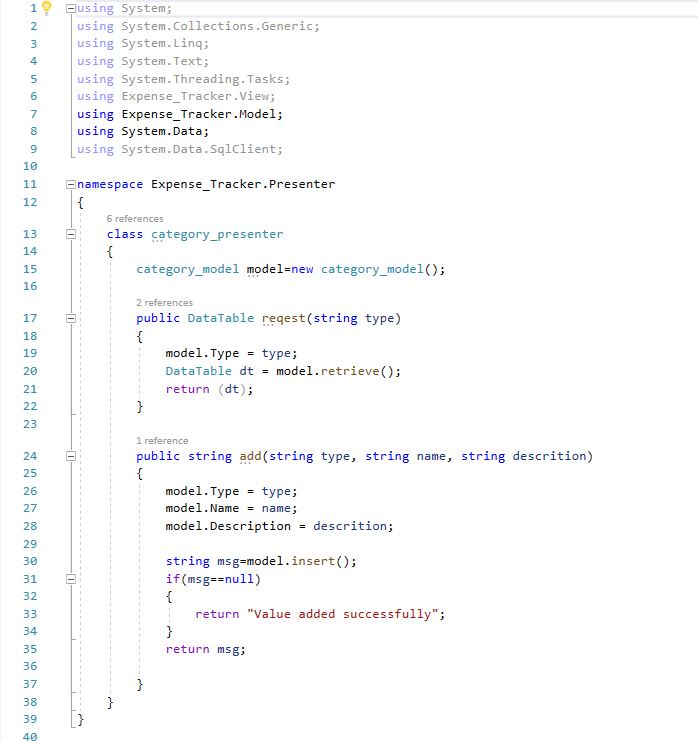


Figure 51: Code for transaction\_presenter



1. category\_presenter

Figure 52: Code for category\_presenter



**View**

IUser\_view

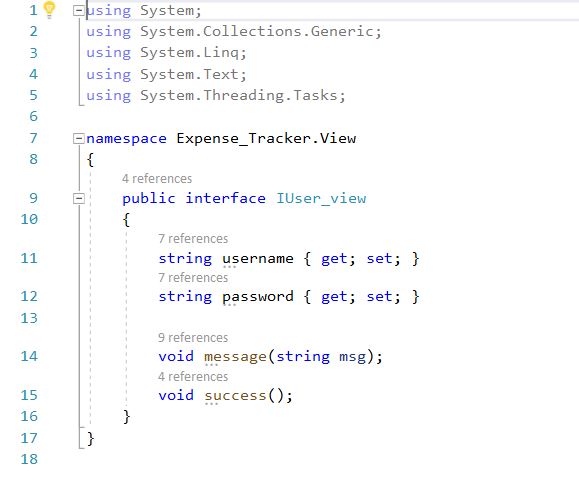


Figure 53: Code for IUser\_view

1. Login form

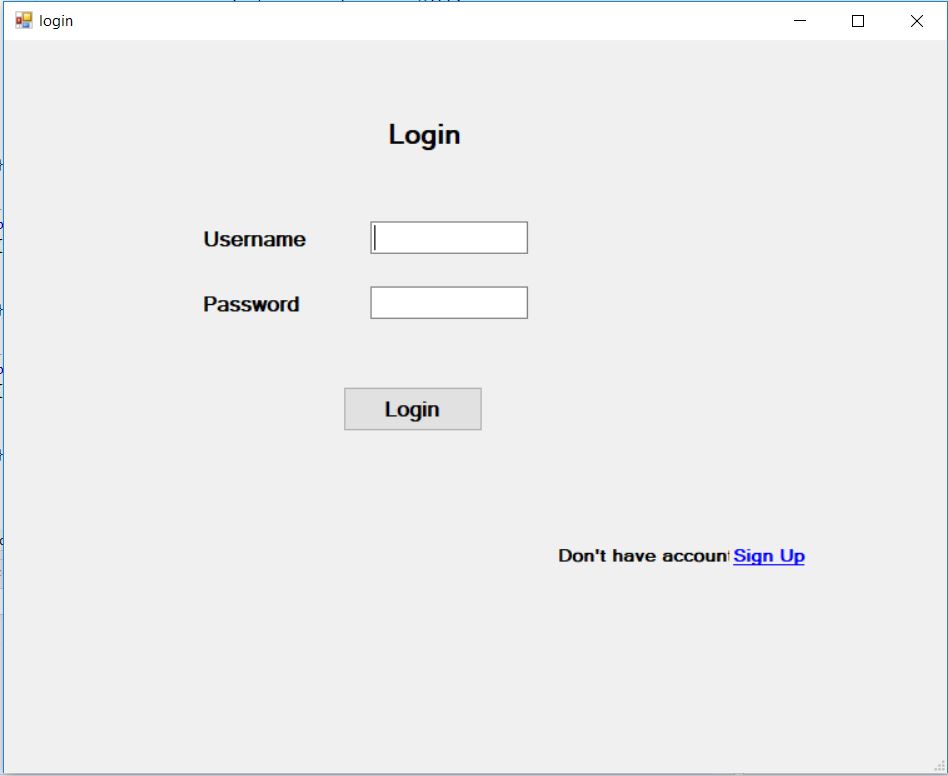


Figure 54: Login form

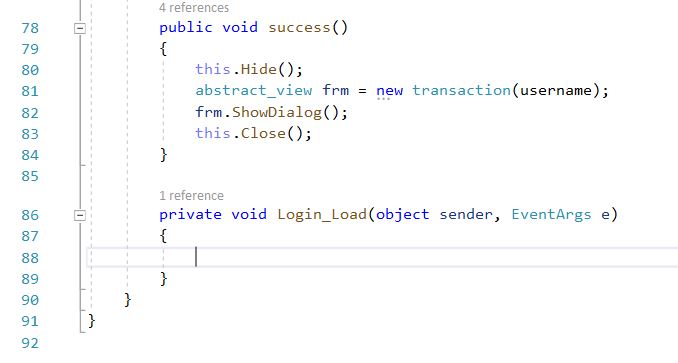
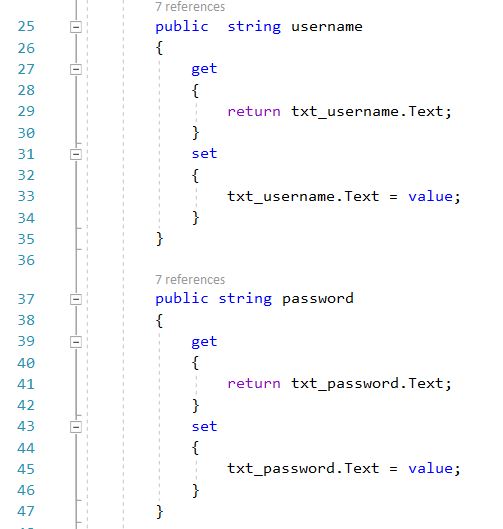
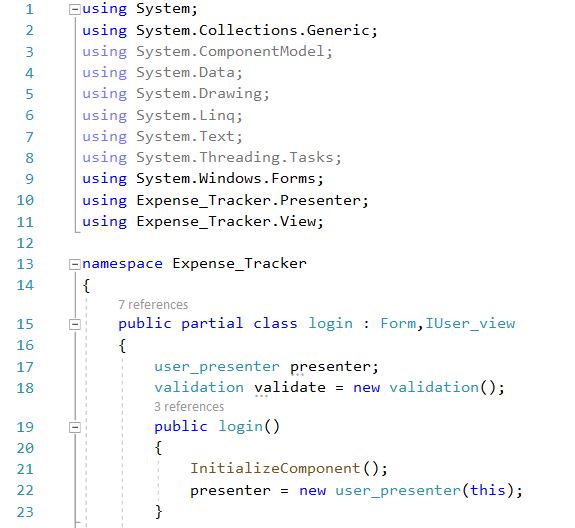


Figure 55: Code for login

1. Sign up

Figure 56: UI for sign up

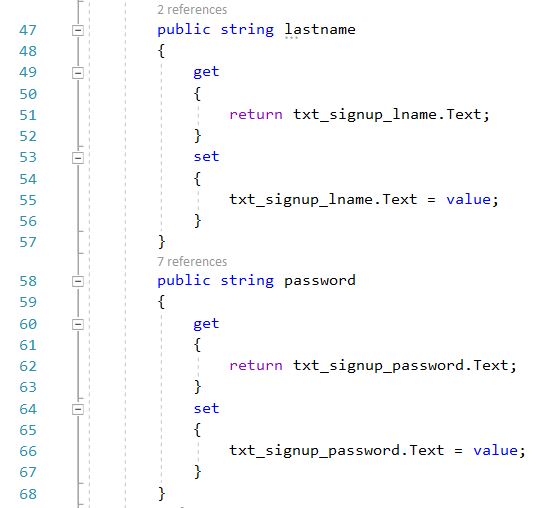
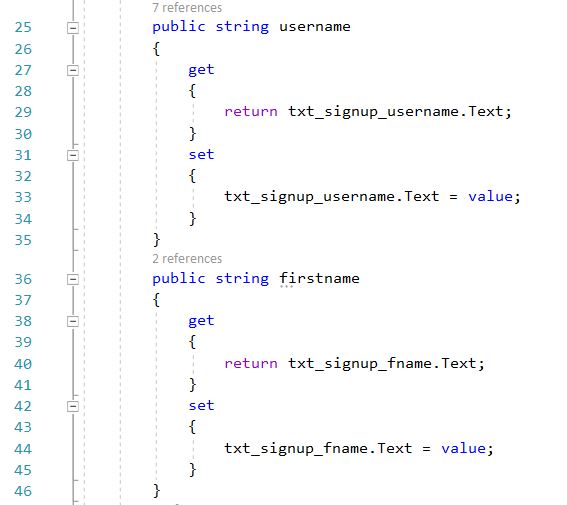
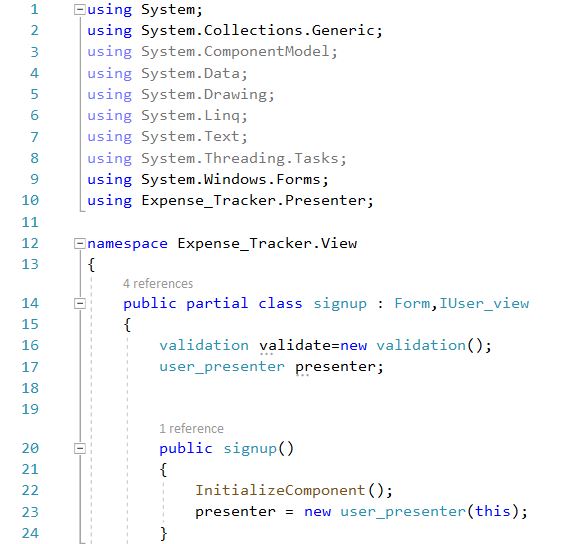
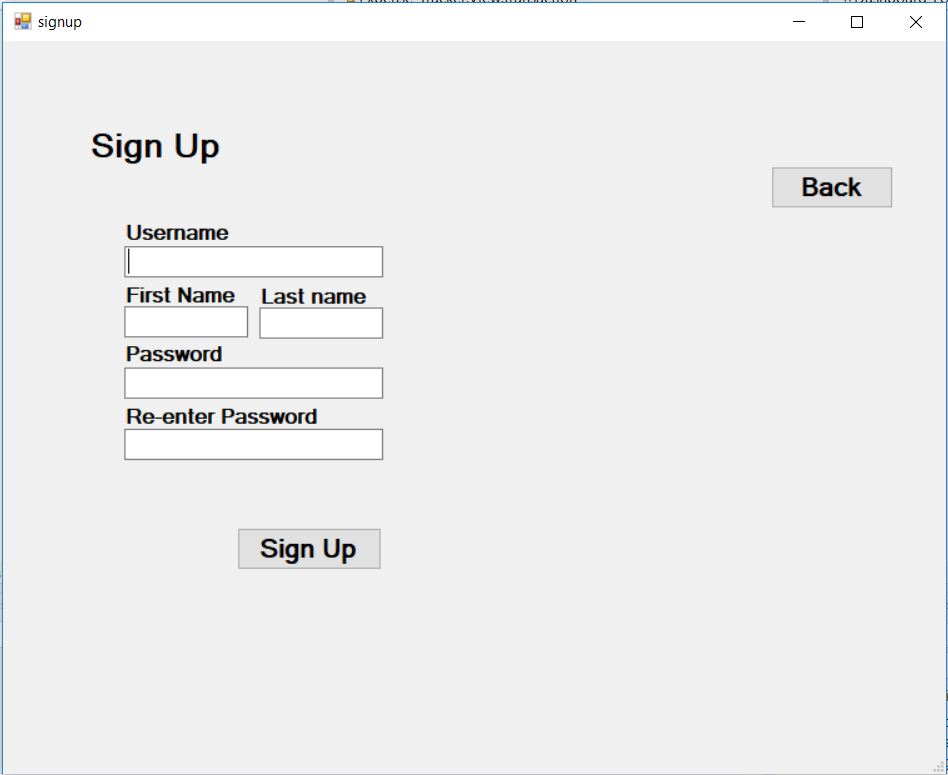


Figure 57: Code for signup

1. Abstract\_view

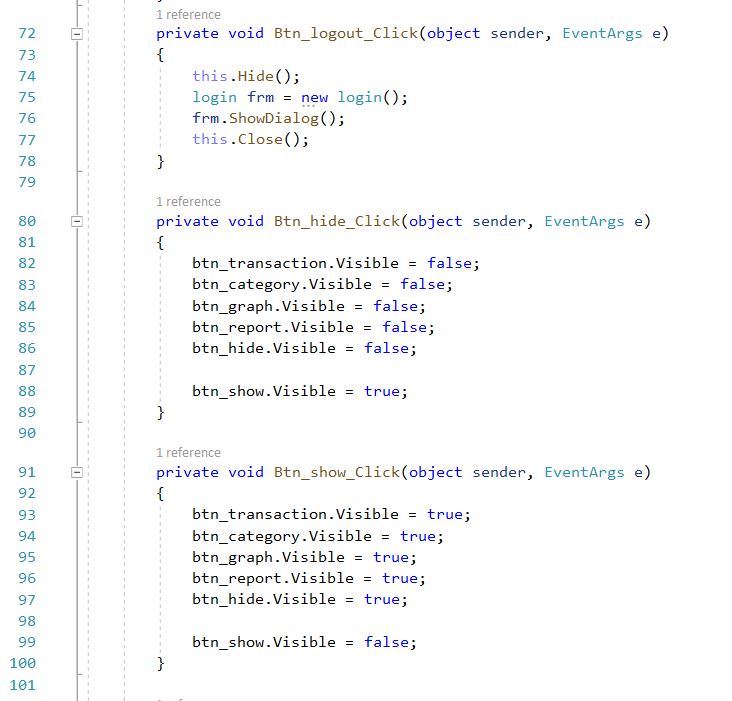
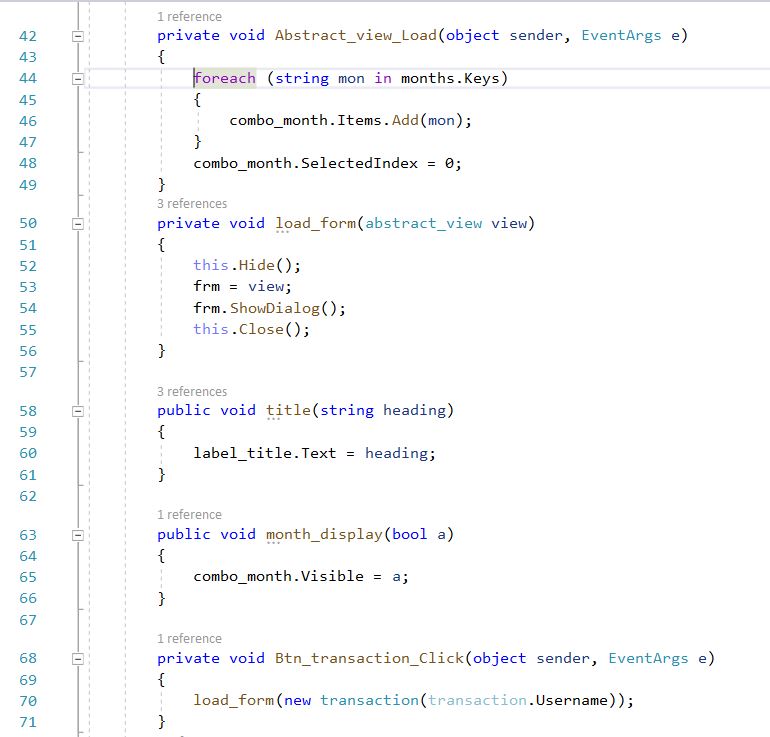
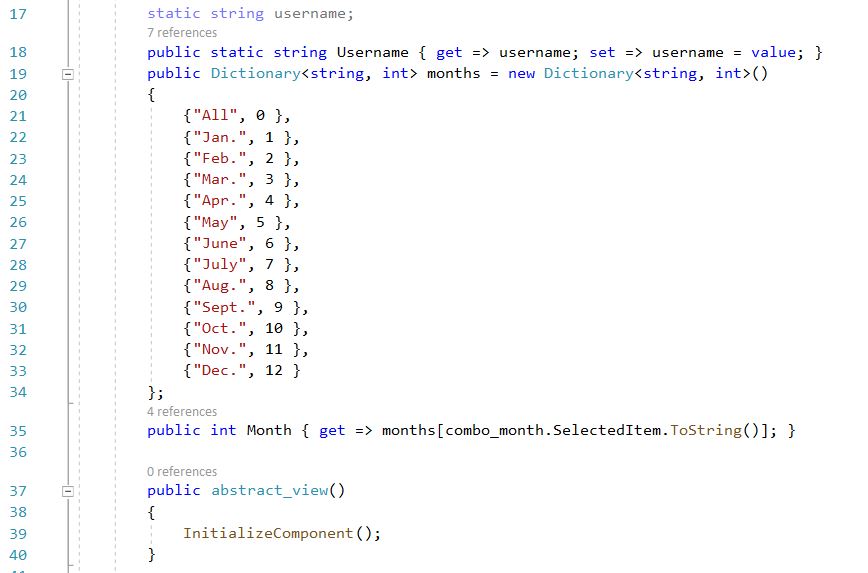
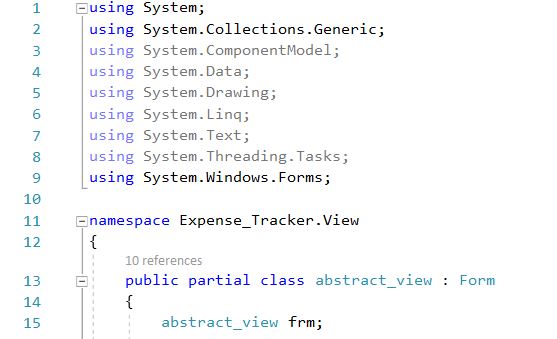


Figure 58: Code for abstract\_view

1. Transaction

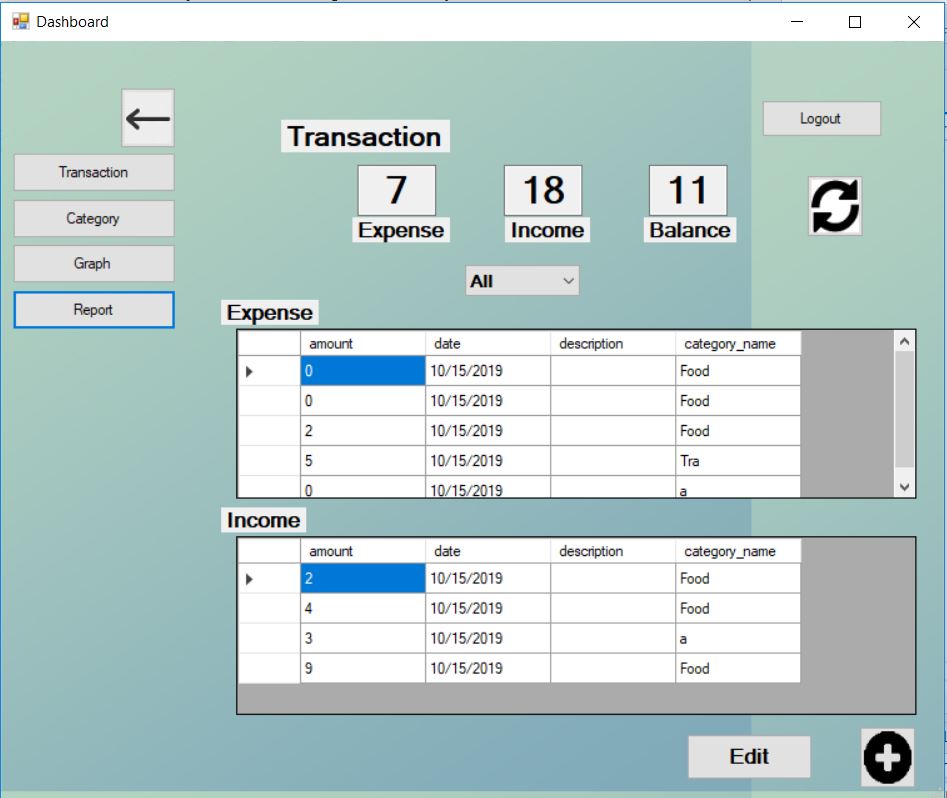


Figure 59: UI for transaction

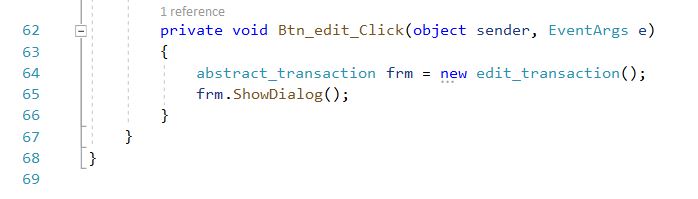
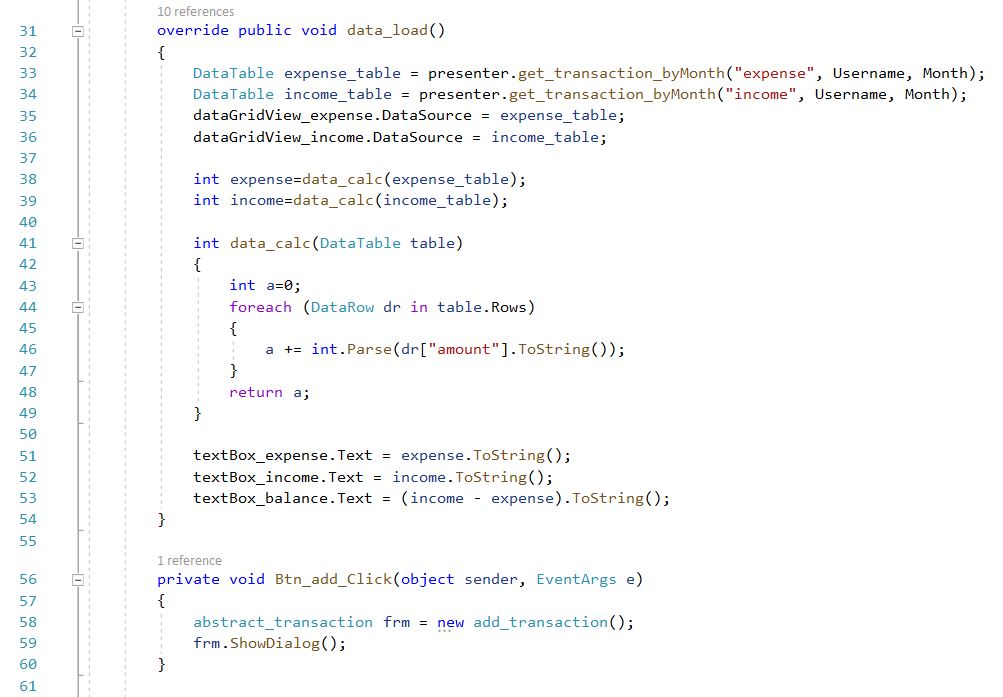
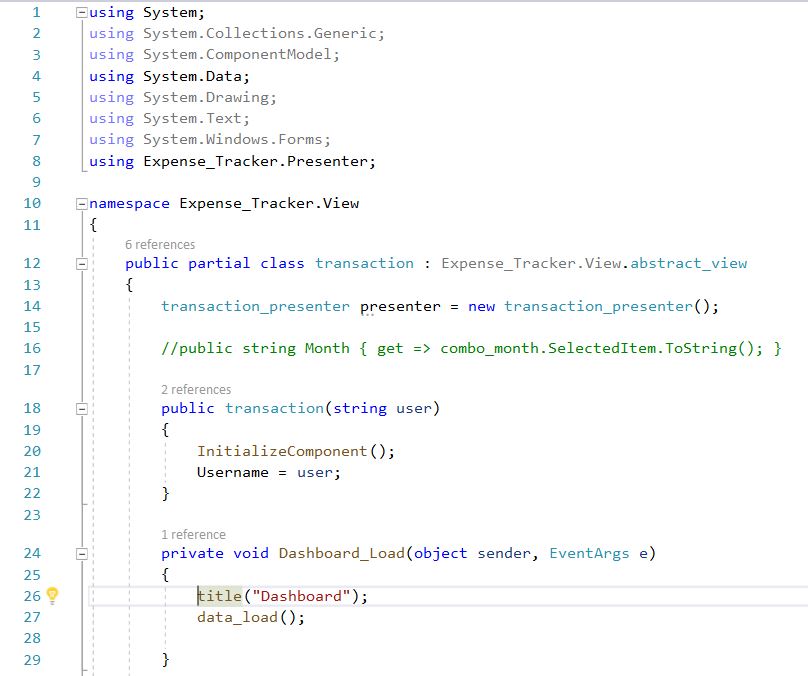


Figure 60: Code for transaction

1. Category

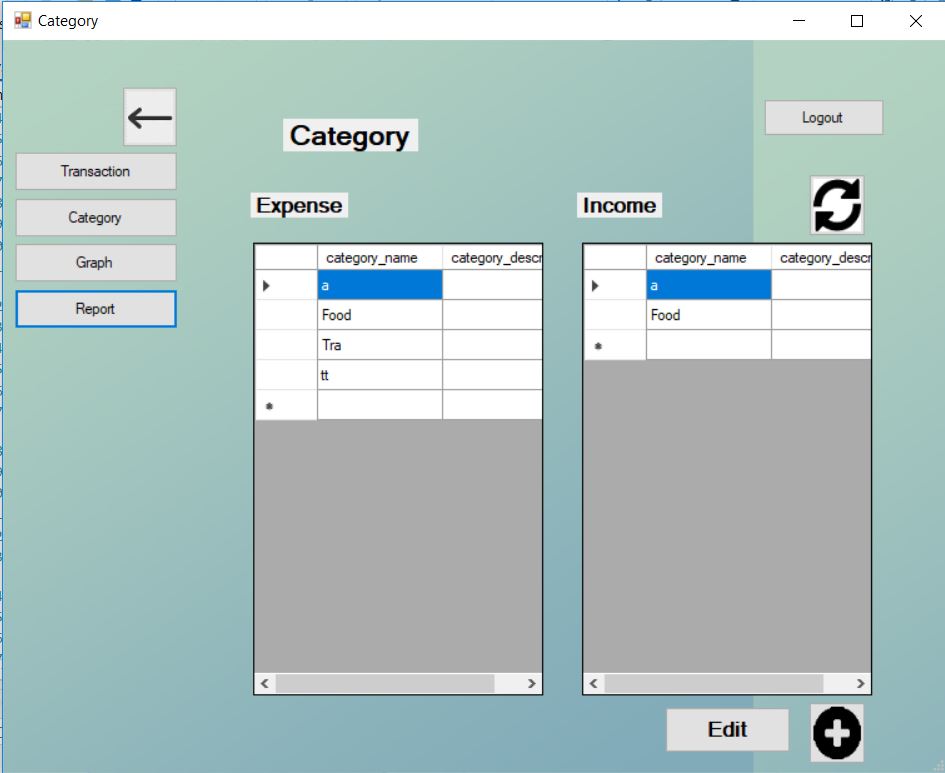


Figure 61: UI for category

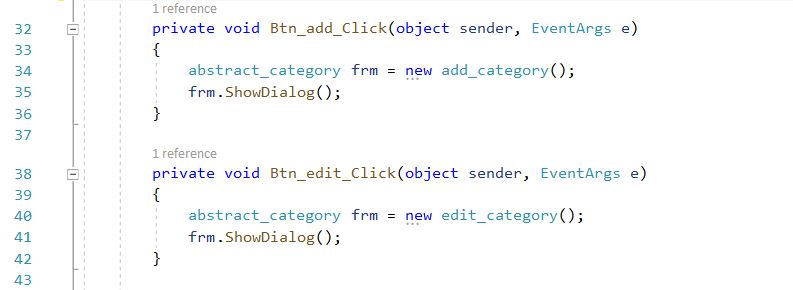


Figure 62: Code for category

1. Graph

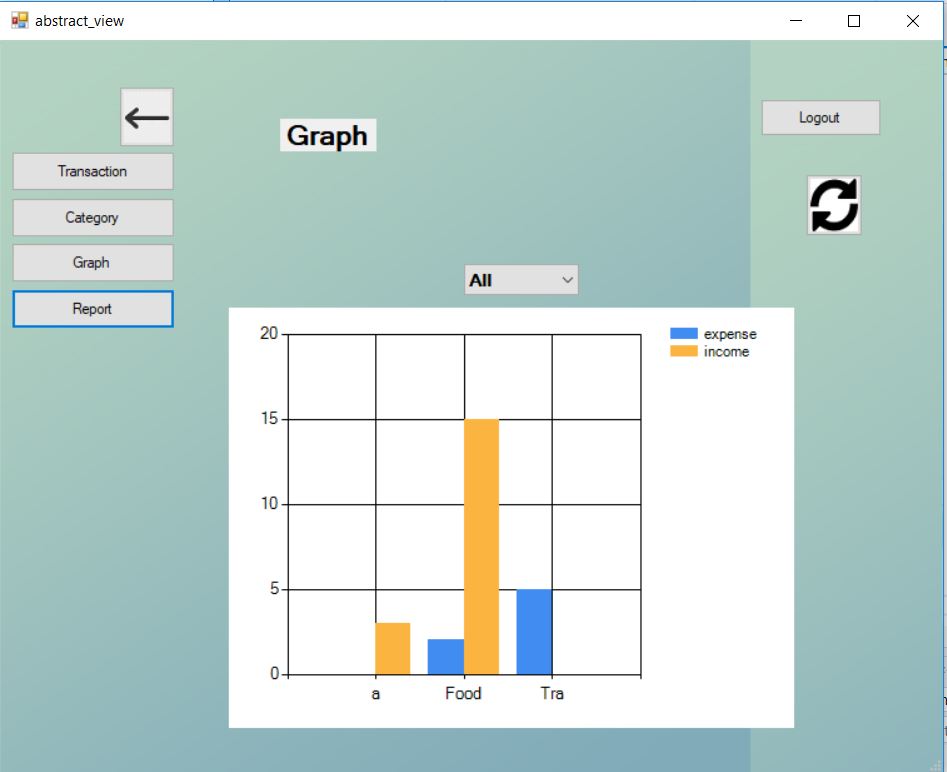


Figure 63: UI for graph

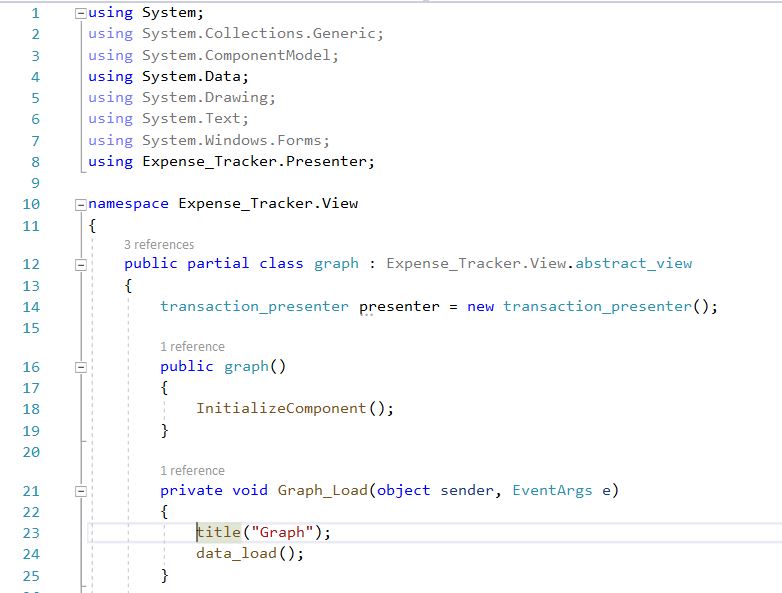


Figure 64: Code for graph

1. Report

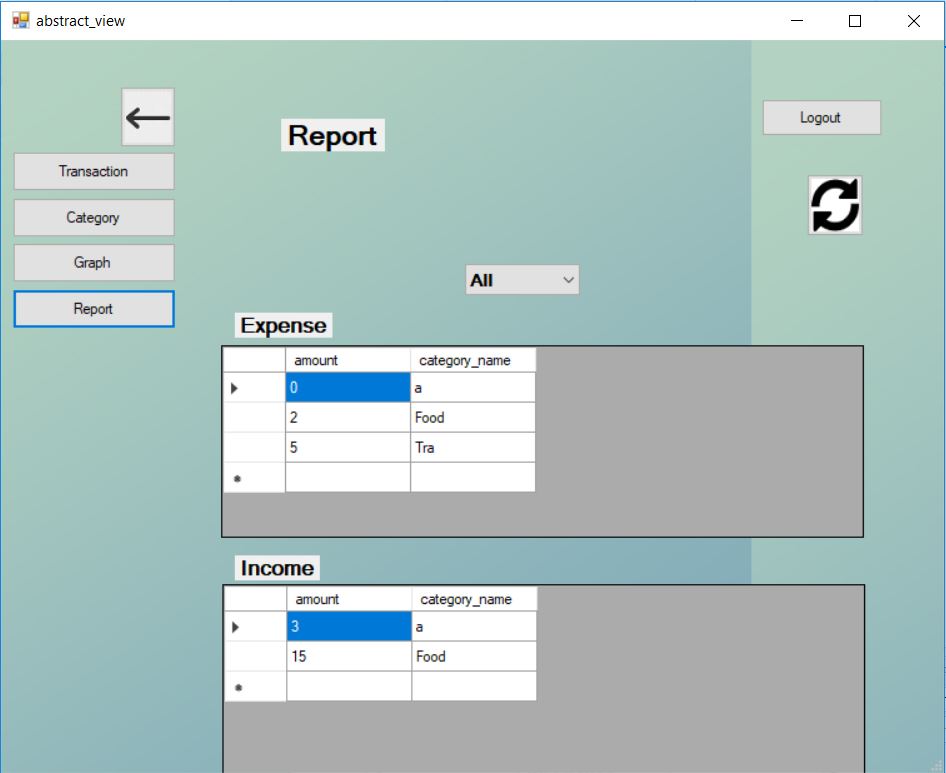


Figure 65: UI for report

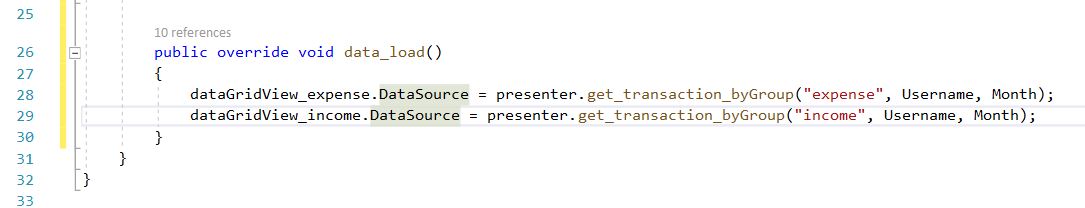


Figure 66: Code for report

1. Abstract\_transaction

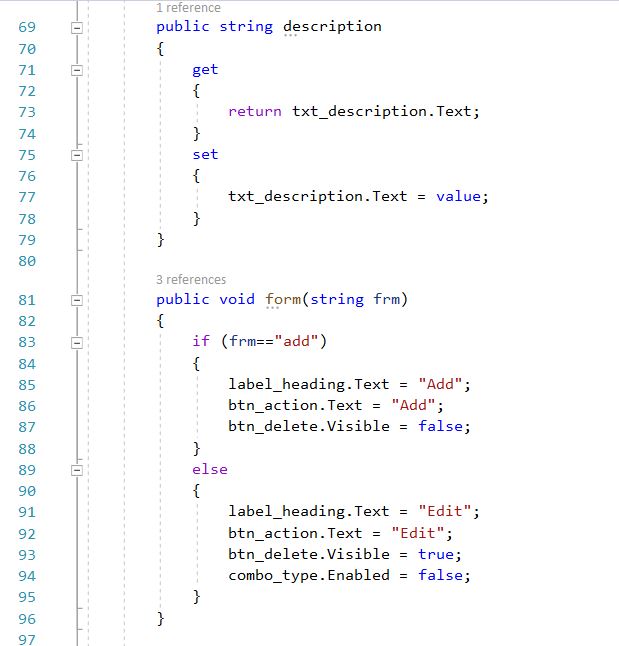
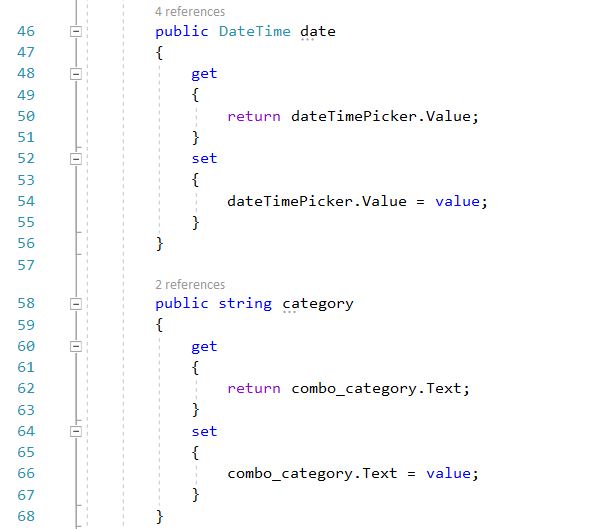
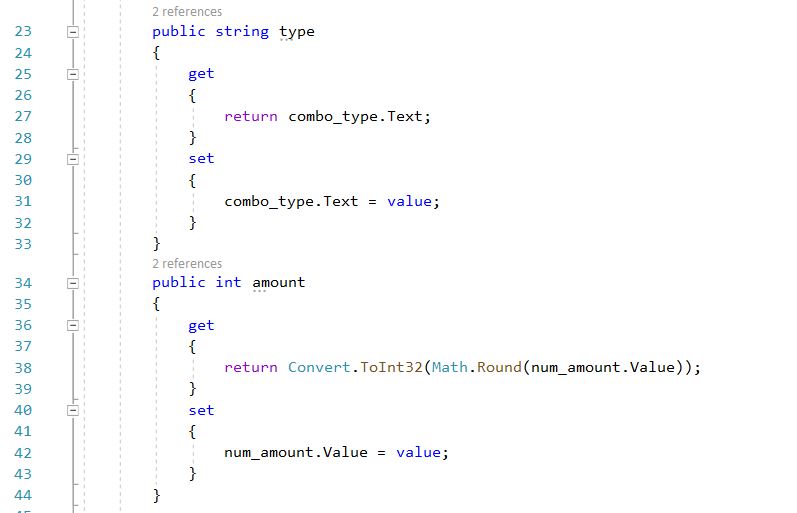


Figure 67: Code for abstract\_transaction

1. Add\_transaction

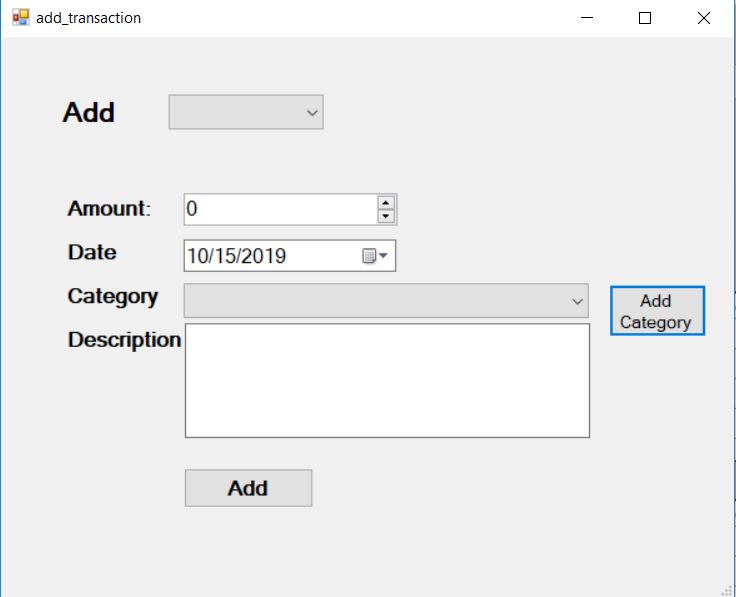


Figure 68: UI for add\_transaction

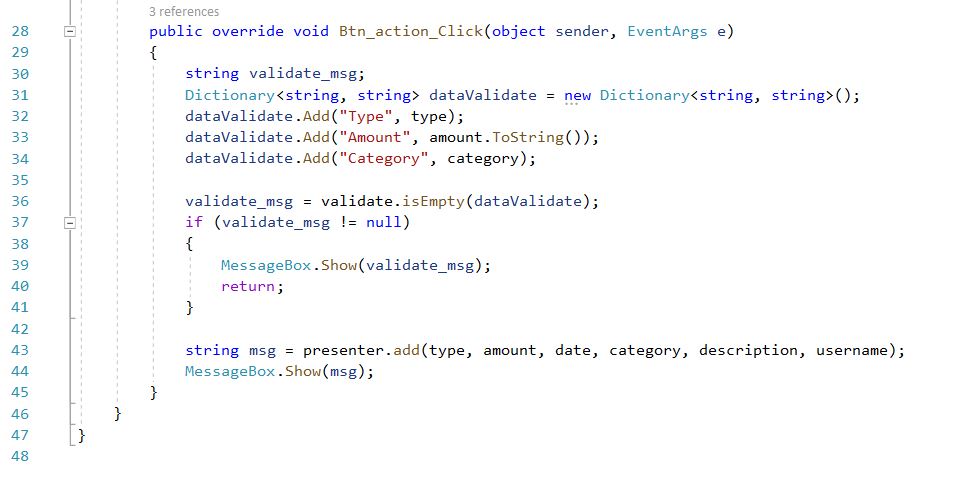
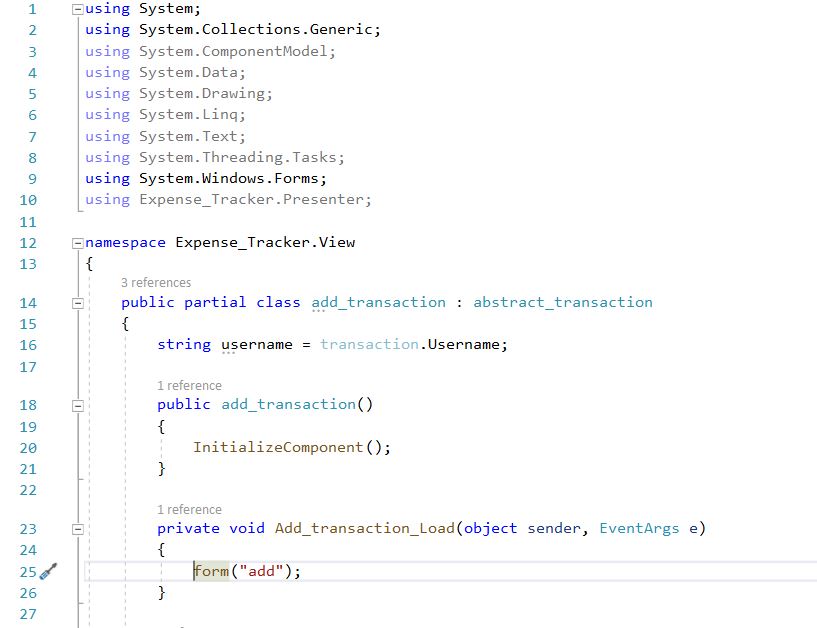


Figure 69: Code for add\_transaction

1. Edit\_transaction

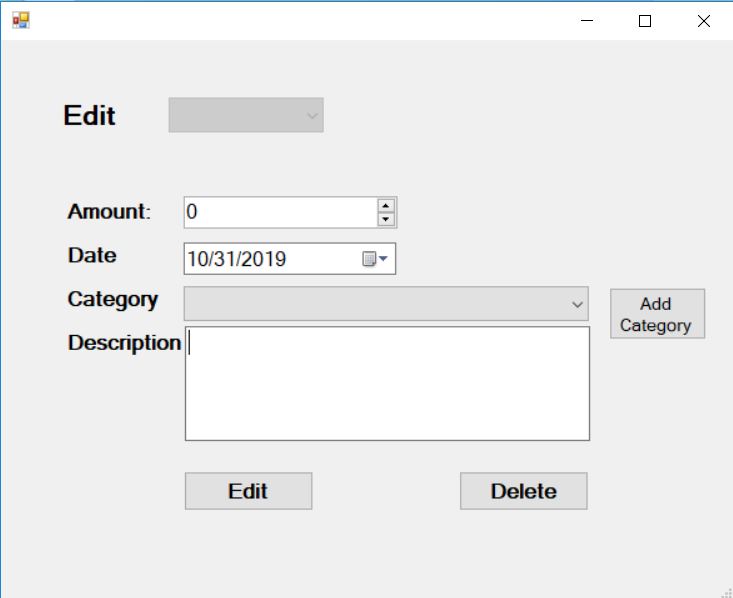


Figure 70: UI for edit\_transaction

1. Abstract\_category

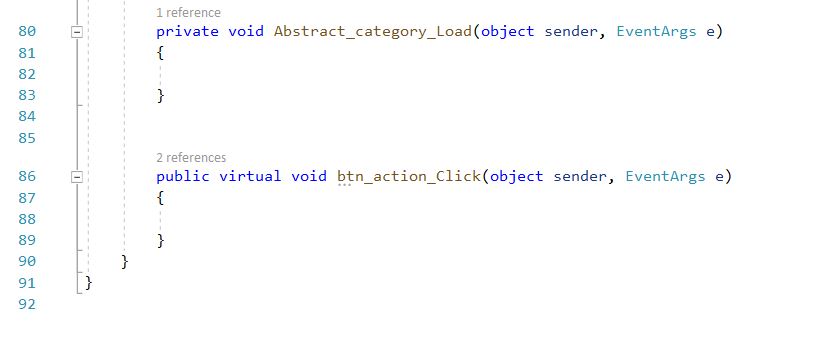
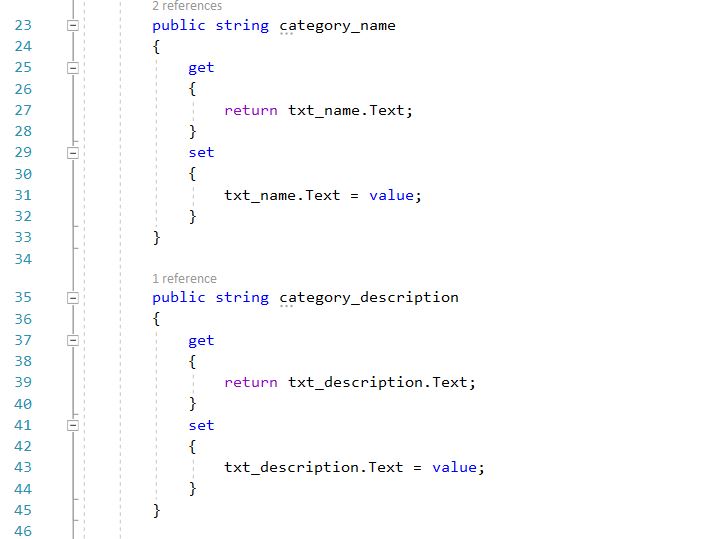
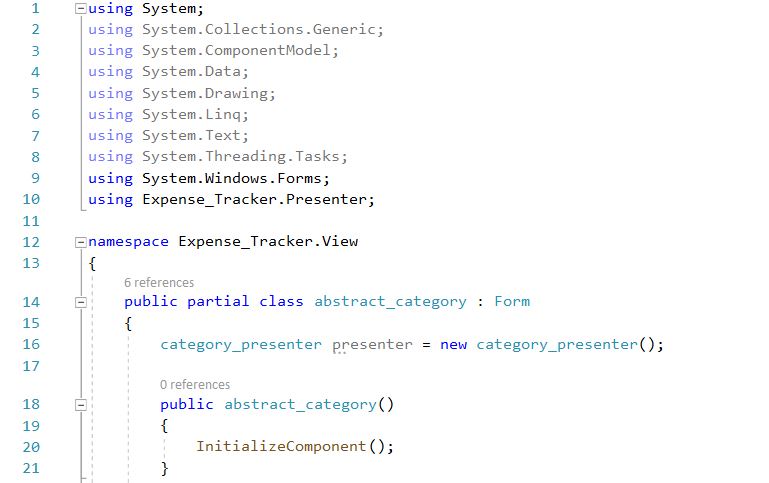


Figure 71: Code for abstract\_transaction

1. Add\_category

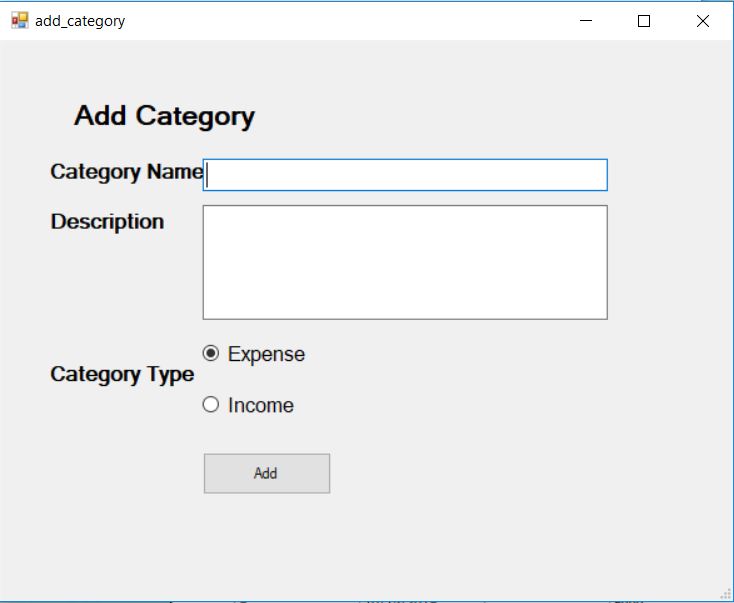


Figure 72: UI for add\_category

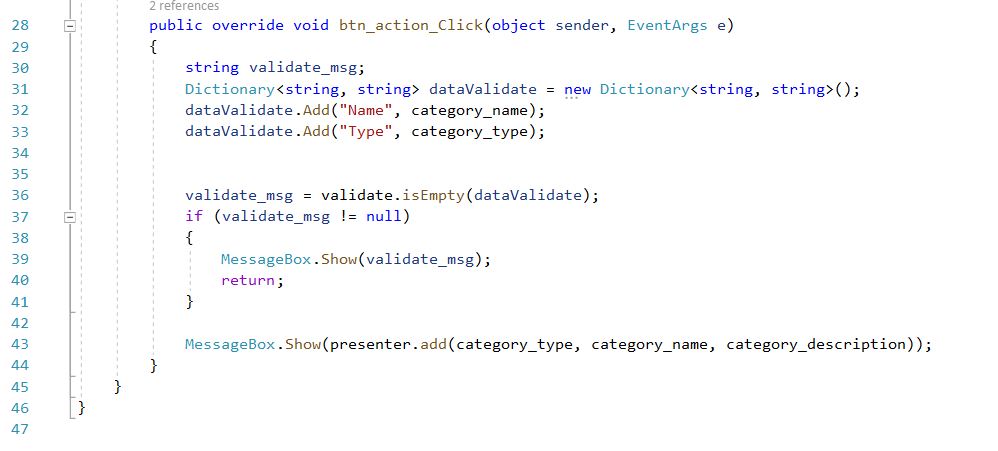
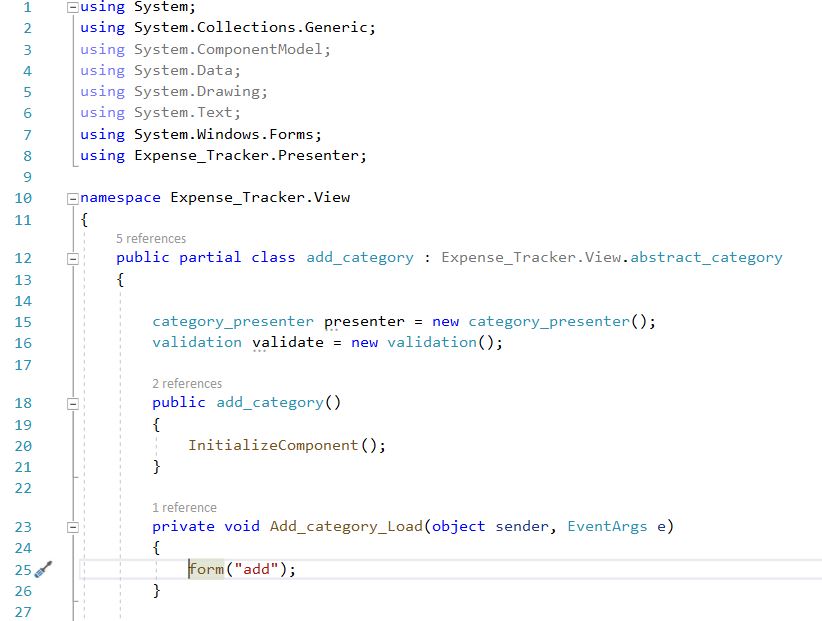


Figure 73: Code for add\_category

1. Edit\_category

Figure 74: UI for edit\_category

