**Contents**

[**Acknowledgement** 2](#_Toc5545972)

[**Abstract** 3](#_Toc5545973)

[**Chapter 1: Introduction** 4](#_Toc5545974)

[**1.1 Background of the project** 4](#_Toc5545975)

[**1.2 Problem Statement** 4](#_Toc5545976)

[**1.3 Features** 5](#_Toc5545977)

[**1.4 Aims** 6](#_Toc5545978)

[**1.5 Objectives** 6](#_Toc5545979)

[**Chapter 2: Analysis** 7](#_Toc5545980)

[**2.1** **Requirements** 7](#_Toc5545981)

[**2.2** **Use-Case Diagram** 8](#_Toc5545982)

[**2.3** **Architecture** 10](#_Toc5545983)

[**2.4** **Initial class diagram** 11](#_Toc5545984)

[**Chapter 3: Design** 12](#_Toc5545985)

[**3.1 Structural Design** 12](#_Toc5545986)

[**3.1.1 The final class diagram:** 12](#_Toc5545987)

[**3.1.2 ER Diagram** 13](#_Toc5545988)

[**3.2 Behavioural Design** 14](#_Toc5545989)

[**3.2.1 Activity diagram** 14](#_Toc5545990)

[**3.2.2 Sequence diagram** 17](#_Toc5545991)

[**Chapter 4 Database** 20](#_Toc5545992)

[**Chapter 5: Implementation** 22](#_Toc5545993)

[**5.1 Language** 22](#_Toc5545994)

[**5.2 Developing environment** 22](#_Toc5545995)

[**5.3 Deployment Strategy** 22](#_Toc5545996)

[**Chapter 6: Testing** 23](#_Toc5545997)

[**Chapter 7: Other project work** 23](#_Toc5545998)

[**7.1 Risk management** 23](#_Toc5545999)

[**7.2: Configuration management** 25](#_Toc5546000)

[**7.2 Future works** 26](#_Toc5546001)

[**Chapter 8: Conclusion** 26](#_Toc5546002)

[**Appendix A: Source code** 27](#_Toc5546003)

[27](#_Toc5546004)

[27](#_Toc5546005)

[**Appendix B: User Manual** 47](#_Toc5546006)

[**Reference** 55](#_Toc5546007)

# **Acknowledgement**

It was a great experience to be able to present Expenses Management System as part of Level 5 diploma(L5DC) degree. I had a great experience while doing my Computing Project (CP) which was based on Expenses Management System.

This project was created to help an individual or an organisation to manage their finances and help them save their expenses. Expenses Management system has different features which allows it to be user friendly and easy to use. To create this project, I have used PHP Laravel as it is effective to use.

I would like to thank MR Kiran Rana for proper guidance he had provided to finish this project; I would like to thank him for keeping us motivated and helping us manage our time properly in order to finish the project in time. I would also like to thank my friends who has given me immense guidance and support while creating the project

This project has given me a lot of knowledge about Laravel framework, php etc. It has given me skills such as managing time and resources to finish the project in time

# **Abstract**

Expenses Management system is a website which is created for user to input their financial records such as their expenses as well as their income. This website is created in order to lower the chances of human error such as user may forget where they have spent their money or they may not know how to manage their finance. Hence, expenses management system is created to make things easier for people to record their expenses and manage their money. The whole project was created for security of their finance. This project can also save user’s time as they do not have to remember each and every expense they have made and also, they do not have to write their expenses down in order to remember it. User can simply login to their account and easily get their information. This also helps the user to keep their financial information secret and secured.

User can add details such as their expenses amount and date (when they have spent) and where they have spent their expense. They can also edit and delete their information If not needed. By adding income user can plan their expenses such as how much money they have left to spend. User can create a report where they will get an exact calculation of their income and expenses and how much they have as a saving.

Even though there are different methodology, I have used waterfall model to create this project. Waterfall model is easy and effective to use as it can be used for smaller projects. It helps to maintain standard and manage time effectively to complete the project. I have used MVC pattern to manage the coding part of my project as it helps to test and debug easily. I have used MYSQL for my database and Laravel to create the project.

# **Chapter 1: Introduction**

Expenses management system is used by a business or an individual to audit their expenses. This system allows user to input their income and expenses. The objective of the project is to help user with their finance and encourage them to save.

This system tracks their spending which allows them to have an idea of their expenses. Expenses management system analyses overall expenses, identifies cost-saving opportunities and controls excessive spending.

## **1.1 Background of the project**

Expenses Management system allows user to register and login here they can add their income as well as their daily expenses. This allows them to control their expenses and think of their savings. Their income and expenses will be saved into their account so they can come back and edit their details. This system allows user to track their money and have control over their spending. Time has been managed properly for this project in order to finish it on time. This expense management system provides an integrated set of features to help you to manage your expenses and cash flow. It provides the ability to group your income/expenses into categories and lets you set a budget and track expenses in the category.

The tools I have used for this project are Laravel which will be used to create design and code for the project. I also used Project Libre to manage and list my schedule.

## **1.2 Problem Statement**

This project is focused on any individual or organization who wants to keep track of their income and expenses. Expenses management system reduces having to memorise each and every expense that a user has done. It keeps track of all the expenses and income hence user do not have to memorize it.

Before this system, user may have had to write their expenses down which involves the risk of losing the evidence. Hence, this system allows user to keep track without losing their data.

## **1.3 Features**

The features of the project are as follows:

* User can register on the page
* User can register using their email and password
* Login
* Username will be set
* User can login using their username and password
* Edit profile
* User can edit their profile such as username/ password
* Change password
* Dashboard for user
* Dashboard will help user to navigate through the page easily
* They can go from expenses page to income page
* Log out
* Add income
* User will need to add income to calculate their salary and expenses
* Edit income
* Add expenses
* User need to add their expenses by remembering their spending
* Their expenses will be saved.
* Edit and view expenses
* Add expense category such as “food” “fuel”
* View details of expense category
* Calculate their spending according to their income
* Their spending will be calculated by subtracting it from their income
* This will help user to track their income
* Report of the expense
* Report will be generated by calculating their expenses

## **1.4 Aims**

* To build a system where user can add, update and track their expenses
* To help user control their spending.
* To generate expenses report at the end.
* To calculate spending

## **1.5 Objectives**

* To perform better design
* To perform better code
* To work according to the requirements
* To perform different testing to make sure there are no flaws.
* To perform step by step process to finish the project
* To use Development methodology
* To use Design pattern
* To use structural architecture

# **Chapter 2: Analysis**

**Introduction**

Analysis is the process of breaking down the project into smaller parts for better understanding. This is the phase where all the requirements are gathered and problems are identified. For the first phase of Analysis, all the requirements are gathered which is also broken down into functional and non- functional requirements and MoScoW prioritisation. Furthermore, analysis phase is described with various diagrams such as use-case diagram and initial class diagram.

## **Requirements**

Functional requirement (F) are the main things that user expects from the software.

Non-function requirement (NF) are things that are not straight forward requirement of the system rather related to usability.

MoScoW prioritisation is a technique to help and understand and manage priorities.it helps to deliver the most immediate requirement in earlier stage. It simple simplifies the most, and the lowest importance during the project. It stands for:

**Must have, Should have, Could have, Won’t have**

|  |  |  |
| --- | --- | --- |
| Functional/non-functional | Requirements | MoSCoW Prioritization |
| F(R1) | Admin Login | Must have |
| F(R2) | User login | Must have |
| F(R3) | Add Expenses | Must have |
| F(R4) | Add income | Must have |
| NF(R5) | View Profile | Should have |
| NF(R6) | Update Profile | Should have |
| F(R7) | Add Expenses Category | Must have |
| F(R8) | Add Income Category | Must have |
| NF(R9) | Edit Expenses Category | Should have |
| NF(R10) | Edit Income Category | Should have |
| F(R11) | Delete Expense Category | Should have |
| F(R12) | Delete income category | Should have |
| NF(R13) | Change Password | Could have |
| NF(R14) | View List of Expenses | Should have |
| F(R15) | Edit Existing Expenses | Must have |
| F(R16) | Edit Existing Income | Must have |
| F(R17) | Log out | Must have |
| F(R18) | Calculate Saving | Must have |
| NF(R19) | Report of Expenses | Could have |

## **Use-Case Diagram**

Use case is a graphical interpretation of the interaction among the elements of a system. it is used to identify, clarify and organize system requirement. Use case diagram shows relationship between and among the actors and the use cases.

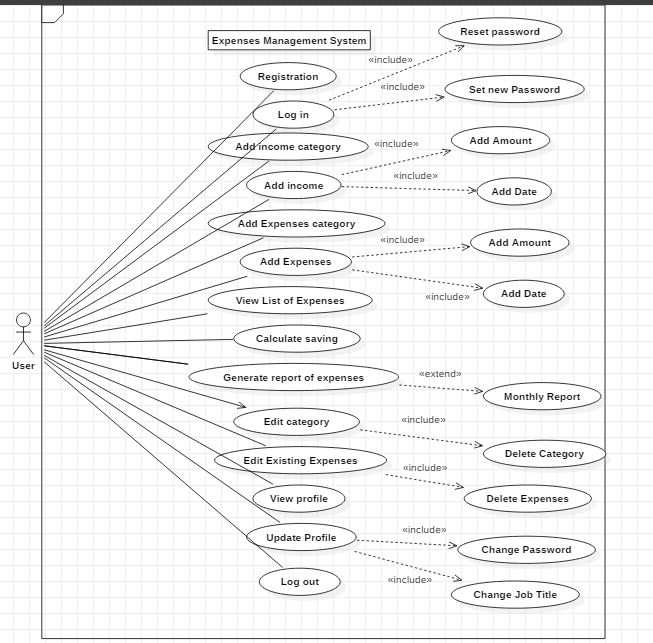


Figure 1:Use-Case diagram

**Justification**

I have used use case diagram so that it helps us design a system from user’s perspective. It is a useful technique for communicating system behaviour in the user's terms. Use case provides observable and valuable result to the actors of the system. it helps us to understand the requirement of the system and what the system is supposed to do. I have created a use case diagram for expenses management system where it shows different actions and steps taken by the user. It starts from registration and login.

This use case has one actor which is user and user can do all things when the user is logged in. First, the user starts by adding their username and password into the system. then, the user is given access to their account and hence, user starts to use the system by adding their income category, expense category, income and expenses. User can also edit their profile as well as update their expenses.

**Advantage**

* It helps to gather functional requirements of a system
* It is easily understandable
* It gathers additional behaviour that can improve system robustness.

**Disadvantage**

* Use case only summarises some of the relationships between use cases, actors and systems
* Use case does not show the order in which steps are performed
* Does not gather the non-functional requirements.

## **Architecture**

Architecture define the structure, behaviour, view and design of the system. In architecture, I have mainly used 3-tier architecture which is used as database, business login and presentation. I have used MVC design pattern for my project which can be designed as software architecture.

**Model: I**t represents an object. It also has logic to update controller if the data changes.

**View:** It represents the visualization of the data that model contains.

**Controller:** It acts on both model and view. It controls data flow into model object.

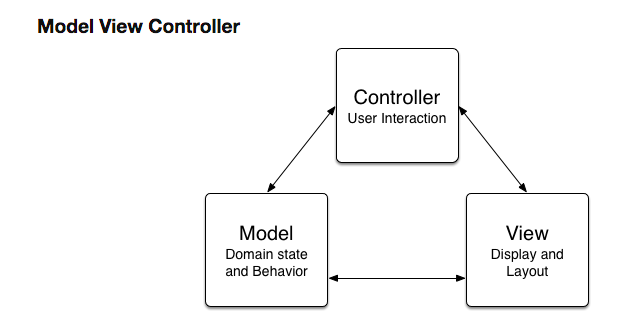


Figure 2: MVC diagram

Advantage of MVC pattern:

* Helps develop application faster
* Multiple developer can work at one project
* Modification does not affect the entire model because model part does not depend on the views part.
* MVC pattern returns data without applying any formatting so the same components can be used and called for use with any interface.

Disadvantage of MVC pattern:

* Complex
* Need multiple programmers
* Inefficiency of data access in view

## **Initial class diagram**

Class diagram is also a part of structural diagram which describes the structure of a system. it shows the system’s classes, attributes, operations and relationships. Class diagram defines the method and variables in an object. It is useful in all forms of object-oriented programming. Initial class diagram shows simple function and attributes where final class diagram shows details about the attributes and the source code.

**Initial Class Diagram:**

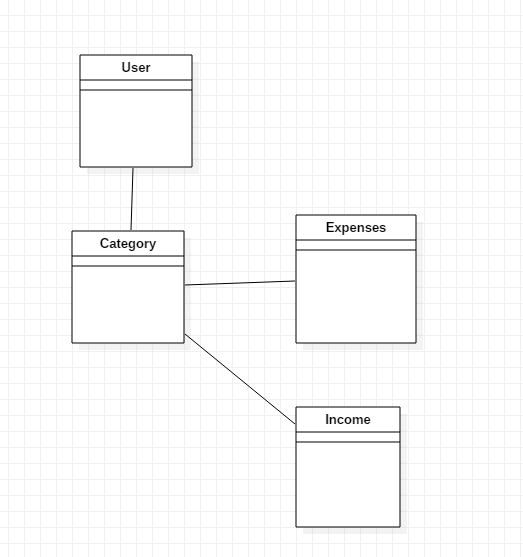


Figure 3:Initial class diagram

**Justification**

In the above class diagram, different classes are shown such as user, category, expense and income. User class is used to control all the actions on the system. User can add, edit and delete category, income and expense; they are all controlled the user. Income and expense have their own category hence they are dependent on category.

# **Chapter 3: Design**

Design phase is divided into parts such as Structural design, behavioural design etc.

## **3.1 Structural Design**

Structural design is static diagrams such as final class diagram, ER diagram. Structural design remains unchanged throughout the project once it has been created. They are the foundation of the project and it defines the infrastructure of the system.

### **3.1.1 The final class diagram:**

Figure 4:Final class diagram

**Justification**

I have created the final class diagram for expenses management system. It is created according to the MVC pattern where controller passes values to the model. It shows different classes and how they are related to each other. I have used class diagram to describe the structure of a system. Class diagram helps to understand the system and plan accordingly.

**Advantage:**

* Provides detail insight into the structure of the system
* They are simple and fast to read
* Reduces maintenance time.

**Disadvantage:**

* If it is complicated, it may be difficult to correlate with the code
* They do not have any dynamics.

### **3.1.2 ER Diagram**

ER diagram is a part of a structural diagram used to design database. It has different symbols and connectors that are important. It shows the relationship of different entities and how they are related to each other in a system. these entities have attributes that define its properties. ER diagram is used for creating database which shows different tables and content that has to be created.

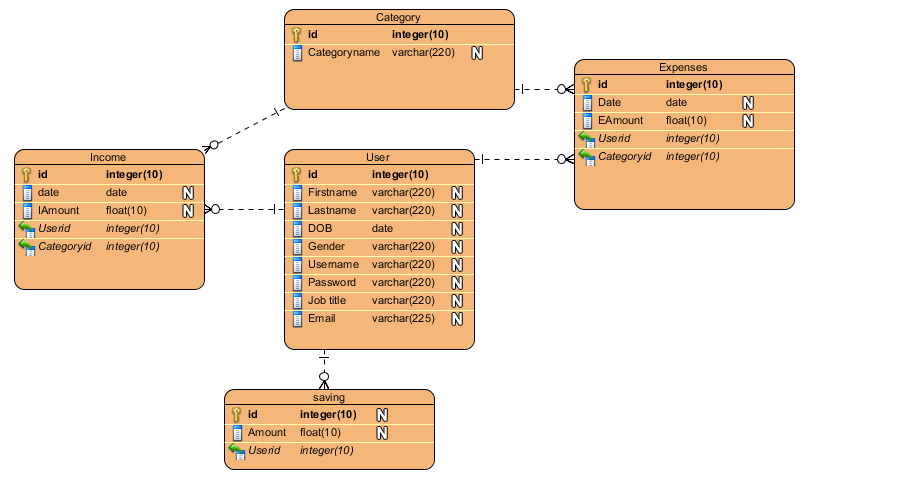


Figure 5:ER diagram

This diagram shows relationship between attributes for Expenses Management System. It shows relationship between tables such as user, expense, income, category. There different tables that will be appropriate for database and how they are related to each other. I have used Entity relationship diagram to provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization.

In this diagram, expense and income are related to category because category needs to be added before income and expense and they have one to many relationships because there is one category to many expenses.

**Advantage:**

* It is very simple
* It gives a better visual representation
* It can be easily converted to any data model.

**Disadvantage:**

* There can be loss of information or content
* It can be difficult to show data manipulation
* There is limited relationship representation.

# **3.2 Behavioural Design**

Behavioural design is dynamic design in the system. They change according to the requirement in the system. They may also change their structure according to the requirement of the user. I have used diagrams such as activity and sequence for this project.

## **3.2.1 Activity diagram**

Activity diagram describes the dynamic aspects of the system. it represents the flow of the system, and show one activity to another activity. Activity diagram is sometimes considered as a flow of diagram. It also describes the parallel, branched and concurrent flow of the system.

I have designed three different activity diagrams for expense, income and report system:

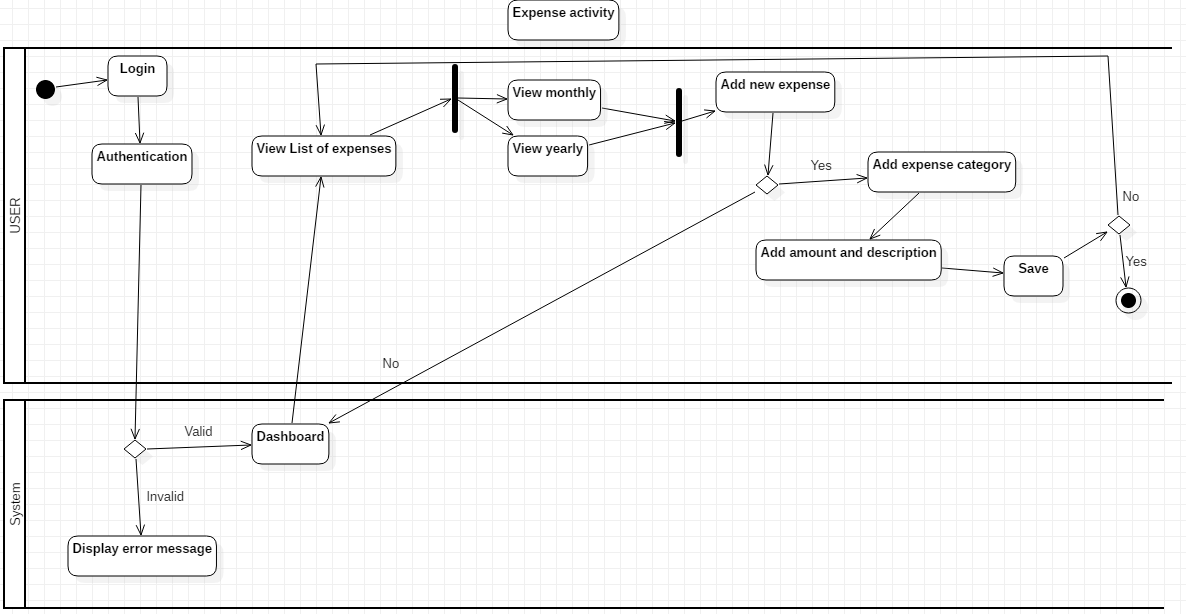
 **Expense Activity:**

Figure 6:Expense activity diagram

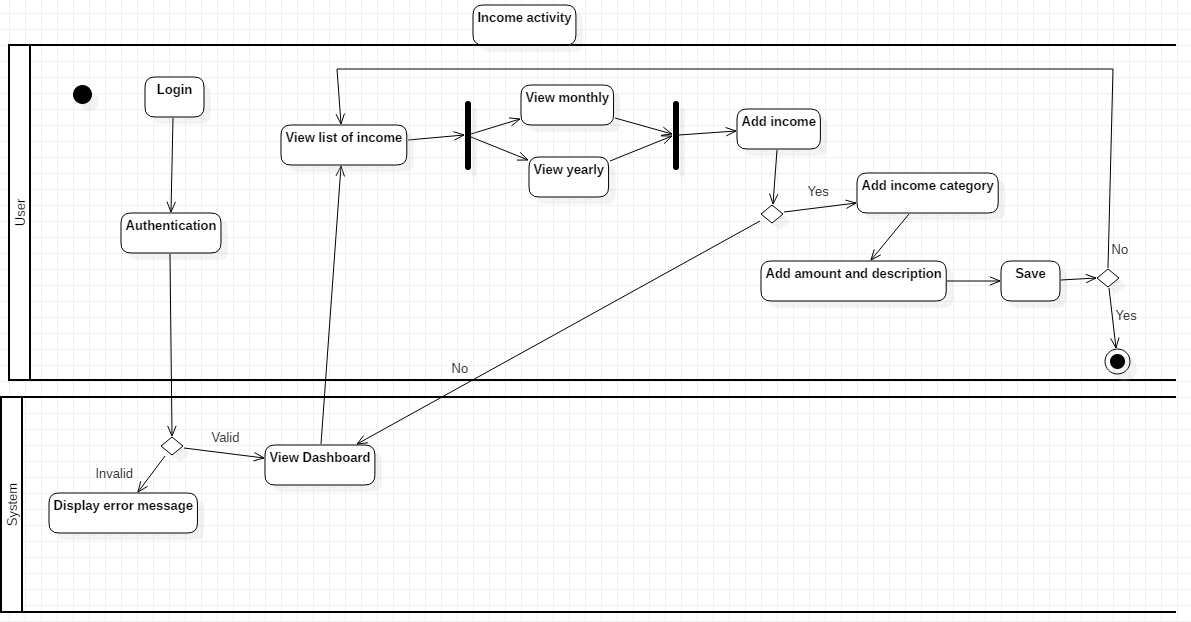
**Income Activity:**

Figure 7:Income activity diagram

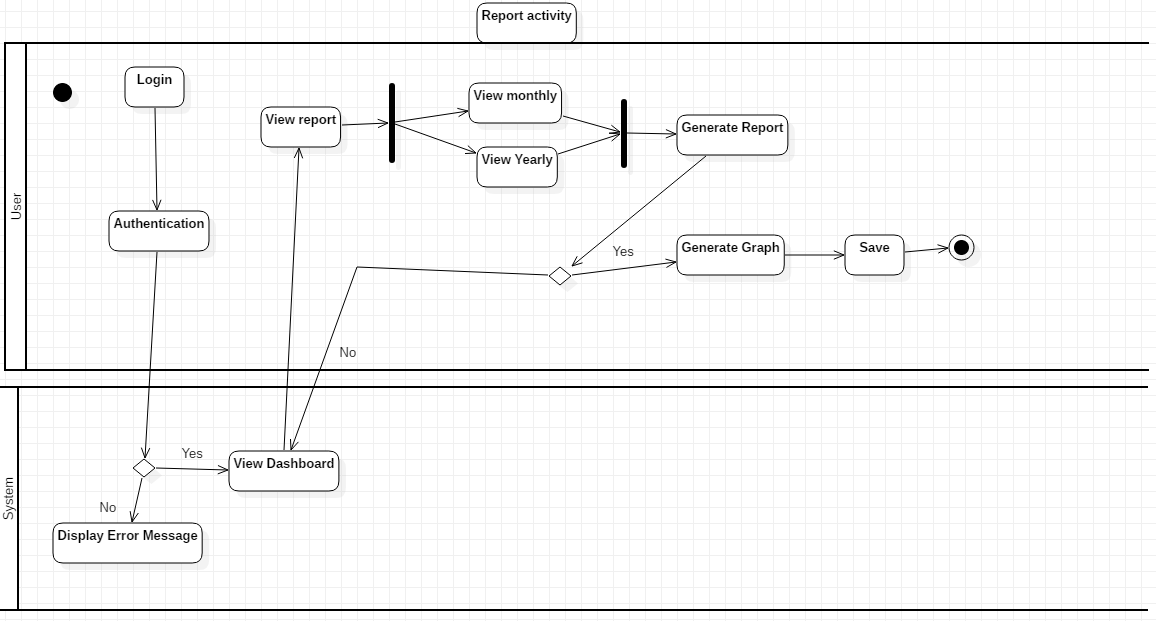
**Report Activity:**

Figure 8:Report activity diagram

I have used activity diagram to capture the dynamic behaviour of the system. It shows the flow of one activity to another activity. It helps to show clear understanding of the system. Activity diagrams are an essential part of the modelling process. I have used this diagram as they are also useful for analysing a use case by describing what actions need to take place.

In these activity diagrams I have showed different flow of expense, income and report. When the user logs and click on dashboard they can go to expense, where they are allowed to view their previous expense such as monthly or yearly. They can add their new expense and expense category and then save. They can do the same with income. To create report, they can print their expense and income monthly or yearly.

**Advantage**

* It is simple to understand
* It is user-friendly
* It displays multiple conditions and actors within a work flow.
* It describes the steps performed in a UML use case.

**Disadvantage**

* It does not give detail about how object behave.
* Does not show any message flow from one activity to another.

## **3.2.2 Sequence diagram**

Sequence diagram are also called event diagram. They are interaction diagrams that show how operations are carried out. Sequence diagram shows high level of interaction between the user and the system. In sequence diagram elements are shown as they interact over time which are organised according to object.

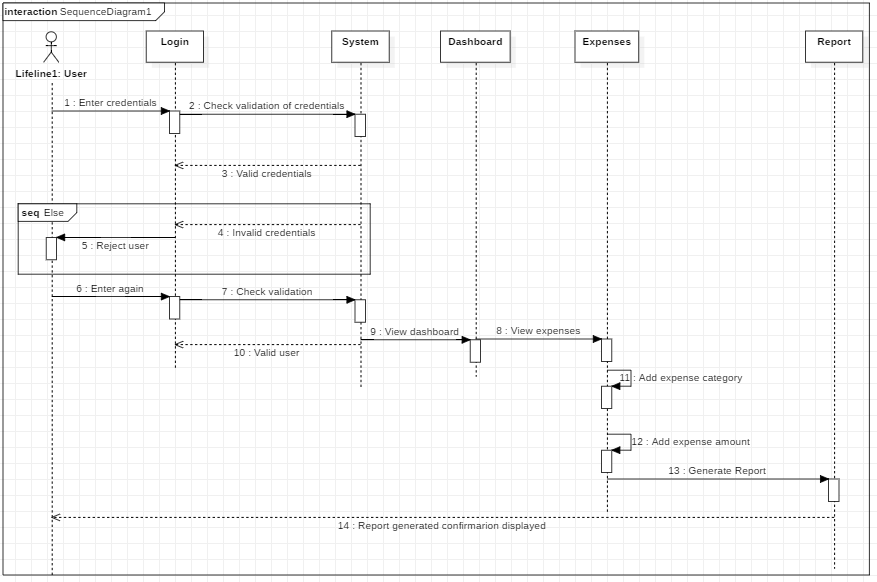
Sequence Diagram for Expenses:

Figure 9: Expense sequence diagram

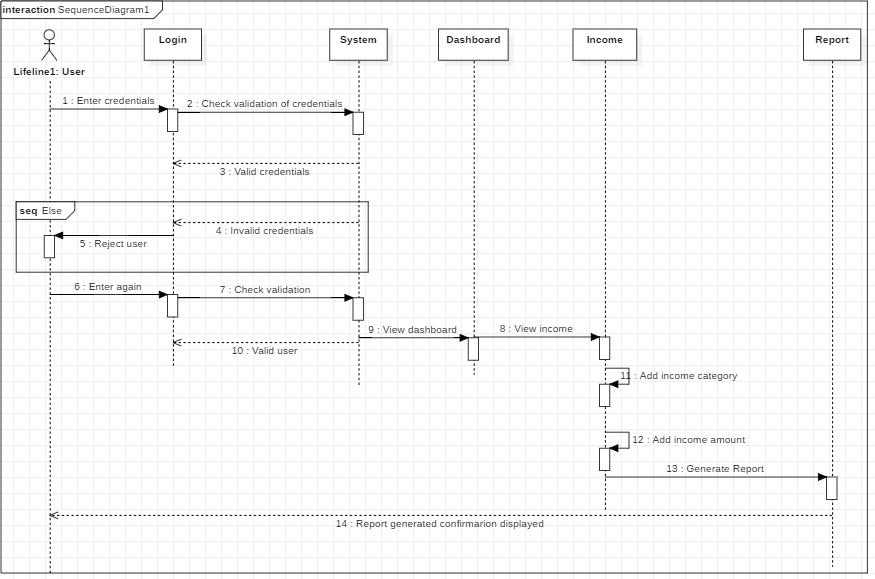
Sequence Diagram for income:

Figure 10:Sequence diagram for income

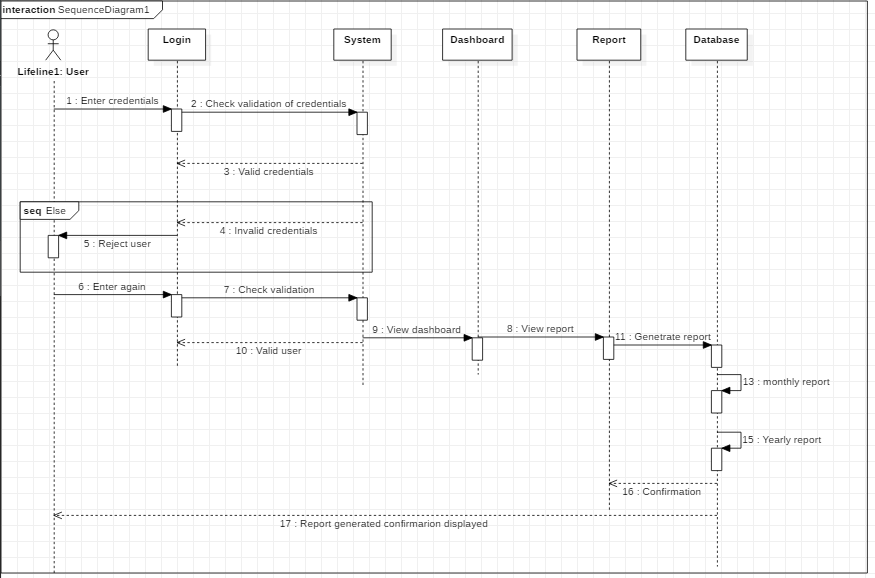


Figure 11:sequence for report

I have created a sequence diagram for Expenses Management. It shows the interaction between the user and the system. There are six lifelines which are: user, login, dashboard, income, expenses and report. User is an actor which interacts with object to show elements. This sequence diagram shows what messages are sent and when and is organised according to time. This diagram has different notations such as an actor, lifeline, activation, message, return message and self-messages. It shows different time period of how user interacts with the system i.e user logs in to the system which leads to dashboard page and then income and expenses are added to the system and report is generated according to the information.

According to my sequence diagram for expense, user logs into the account and the system checks the credentials and if it matches the user is taken to the dashboard ad if it does not, the system throws the user out and asks to try again. When the user is logged on, they are taken to the dashboard where they can add, edit and view their information.

**Advantage:**

* It is easier to read
* It shows how components of the system interacts with each other
* Understand detail functionality of the system.

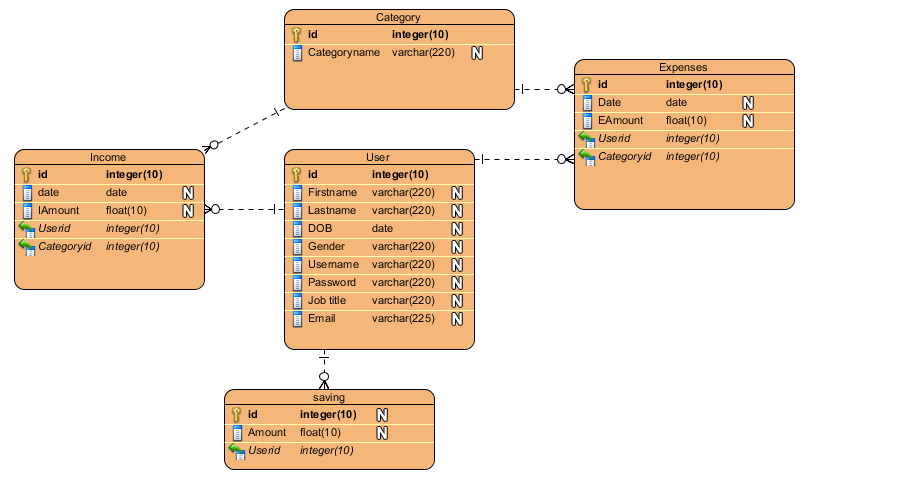
**Disadvantage:**

* You cannot represent every condition in sequence diagram
* It takes a lot of space to create the diagram
* It does not represent the inter relationship between the object very well.

# **Chapter 4 Database**

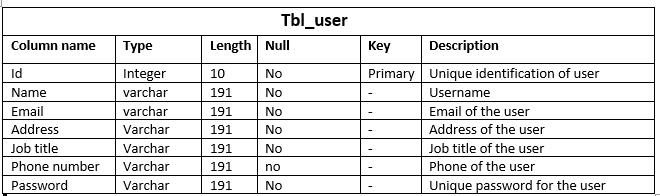
Database shows the relationship between tables and how data are exchanged in the system.

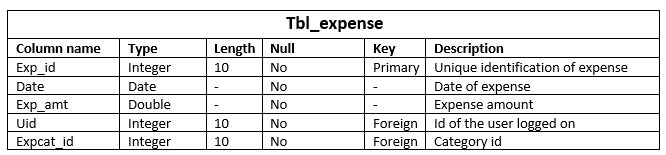
**ER diagram:**

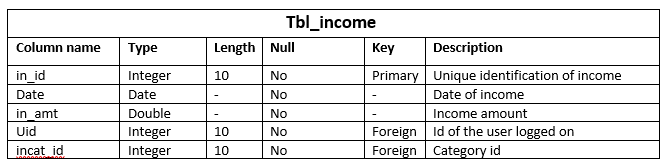
 They show the relationship between entities and the flow of the system.

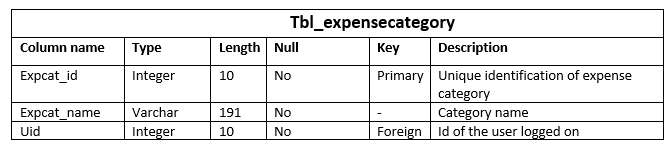
**Metadata**

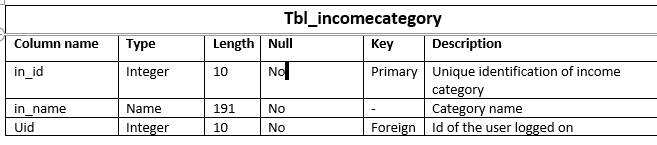
Meta data is the blueprint of the ER diagram shown above, they show the different tables created and their datatypes, length and keys with details.











# **Chapter 5: Implementation**

Implementation is the process of putting an idea or plan into real time working using code with different programming languages. Implementation is done according to the client’s requirement. It’s the process of executing plan into work.

## **5.1 Language**

Programming language is a set of rules that instructs computer to perform specific tasks. This language is converter into machine language for the computer to understand. No matter which language is used, it is converted into machine language.

According to my project, I have used PHP programming language and I have used Laravel as my PHP framework. I have used this language because it is easy and understandable. It also organises my work as it follows MVC design pattern. PHP is an open-source language and free to use. Database is also used with this language to store data hence, I have used MYSQL as my database. MYSQL is a relational database management system which has set of advanced features and it is easy to use.

## **5.2 Developing environment**

For this project, I have used Laravel framework to complete my project. I have used this because there are inbuilt set of features such as testing, validation etc which makes the work easier. It is an open-source and free to use. Laravel also follows MVC design pattern which organises the work effectively and makes the developer work quickly.

To design the system, I have used the help of CSS and JavaScript. They are used for user-interface and layouts of the system.

## **5.3 Deployment Strategy**

Deployment is the process of implementing the project with programming language into the user’ machine.

I have developed and designed this project in Windows. The project is web based hence, Morzilla is used to open the project

# **Chapter 6: Testing**

Testing is the process of finding whether there are bugs or error in the system. It evaluates the system with the intent to find whether it meets the requirement. Testing is done to minimize the project error. Testing is done to each function in the system to check the correctness and expected outputs were gained.

There are different kinds of testing for e.g. Black box, unit testing, white box texting etc which are documented via test case.

**White box testing:**

|  |  |
| --- | --- |
| **TEST CASE** | **T1** |
| Purpose of the case | Testing whether user can login |
| Test data | Email: [suvekshakhnl98@gmail.com](mailto:suvekshakhnl98@gmail.com)  Password: london |
| Class name | ExpmsTest |
| Function name | testLogin |
| Expected result | Login successfully |
| Actual result | User successfully logged in |
| Conclusion | Expected result matches with the actual result |

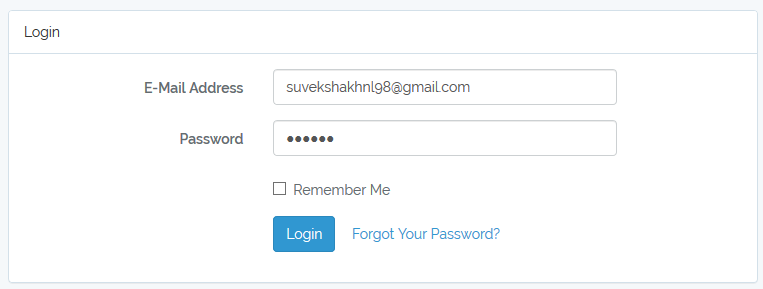
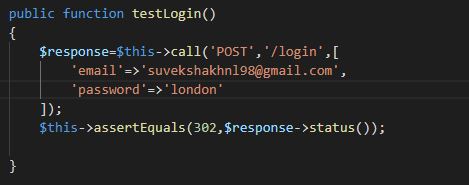
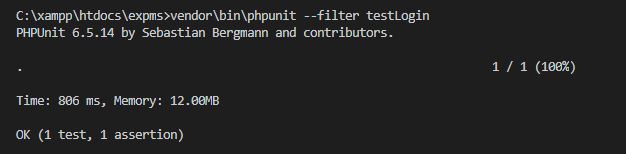


Figure 12:Login test





|  |  |
| --- | --- |
| **TEST CASE** | **T2** |
| Purpose of the case | Testing whether user can register to the system |
| Test data | Name: suveksha khanal  Email: [suvekshakhnl98@gmail.com](mailto:suvekshakhnl98@gmail.com)  Address: dharan  Jobtitle: student  Phno: 9840251859  Password: London  Confirm password: london |
| Class name | ExpmsTest |
| Function name | testRegister |
| Expected result | Registered successfully |
| Actual result | User successfully registered |
| Conclusion | Expected result matches with the actual result |

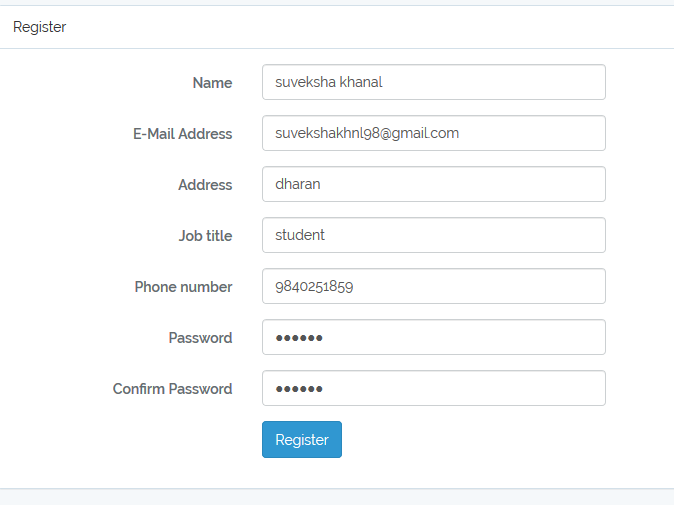
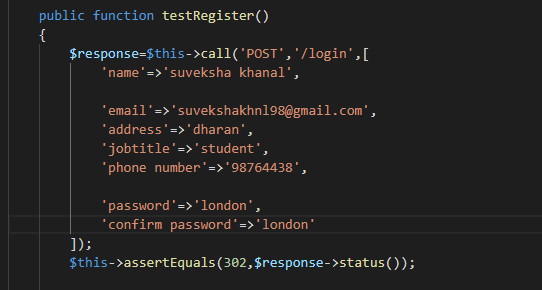
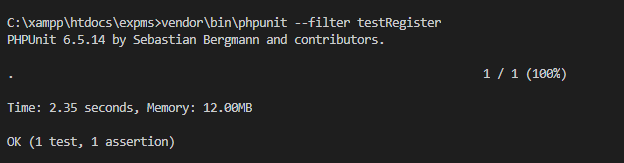


Figure 13:Registration test

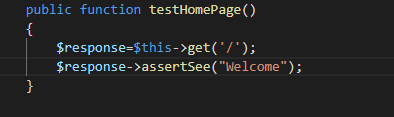


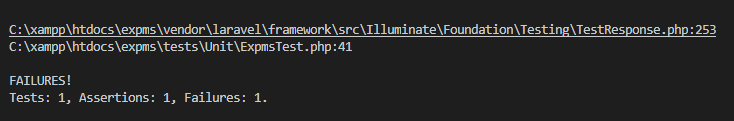


|  |  |
| --- | --- |
| **TEST CASE** | **T3** |
| Purpose of the case | Testing whether the home page displays “welcome”. |
| Test data | “welcome” |
| Class name | ExpmsTest |
| Function name | testHomePage |
| Expected result | Testing failure. |
| Actual result | User successfully registered |
| Conclusion | Expected result matches with the actual result |



Figure 14:Home page test





|  |  |
| --- | --- |
| **TEST CASE** | **T4** |
| Purpose of the case | Testing to see if expense can be added. |
| Test data | Expense category: food  Date: 20/01/2019  Amount: 2000 |
| Class name | ExpmsTest |
| Function name | testAddExp |
| Expected result | Expense added successfully |
| Actual result | Expense added successfully |
| Conclusion | Expected result matches with the actual result |

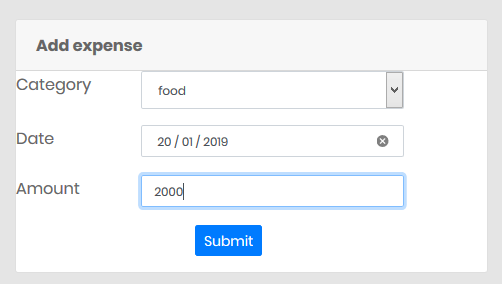
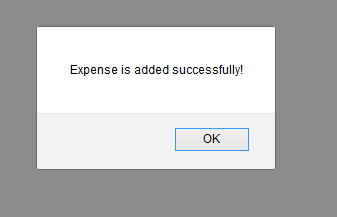
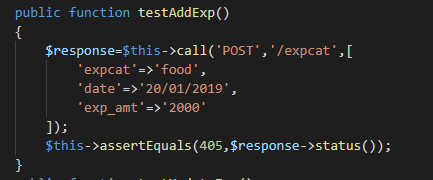
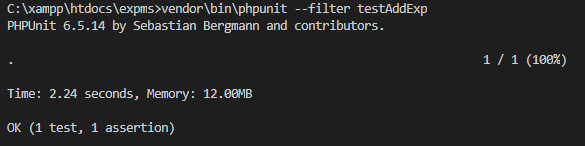


Figure 15:Add expense test

**output:**





|  |  |
| --- | --- |
| **TEST CASE** | **T5** |
| Purpose of the case | Testing to see if expense can be updated |
| Test data | Expense category: food  Date: 20/01/2019  Amount: 50000 |
| Class name | ExpmsTest |
| Function name | testUpdateExp |
| Expected result | Expense should be updated |
| Actual result | Expense updated successfully |
| Conclusion | Expected result matches with the actual result |

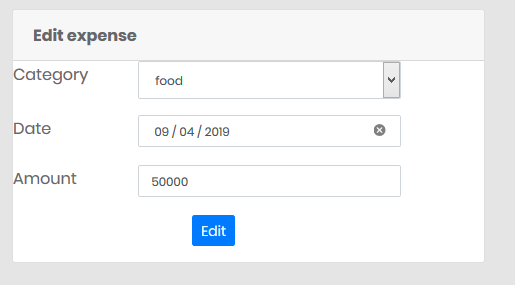
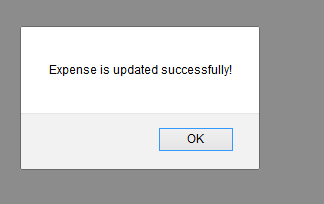
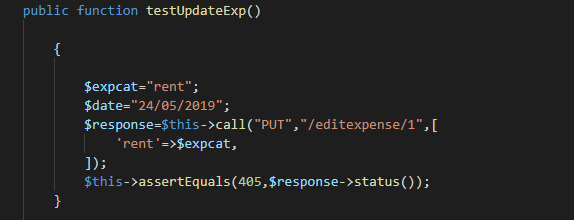
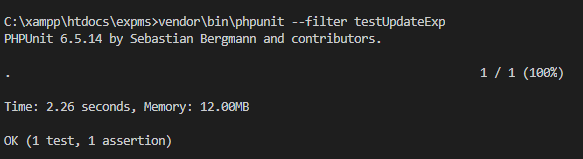


Figure 16:edit expense test

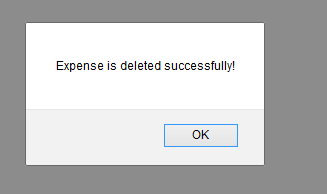
**Output:**

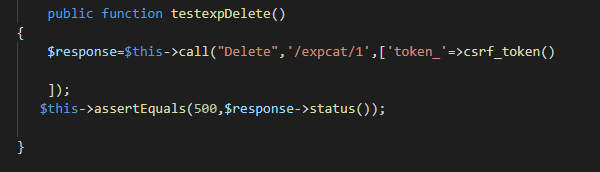


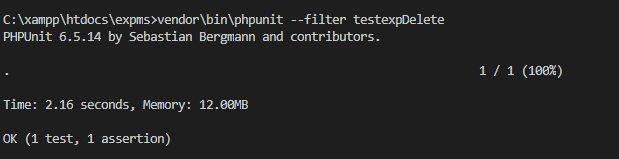


|  |  |
| --- | --- |
| **TEST CASE** | **T6** |
| Purpose of the case | Testing to see if expense can be deleted |
| Test data | Expense category: food  Date: 20/01/2019  Amount: 5000 |
| Class name | ExpmsTest |
| Function name | testexpDelete |
| Expected result | Expense should be deleted |
| Actual result | Expense deleted successfully |
| Conclusion | Expected result matches with the actual result |

**Output:**







|  |  |
| --- | --- |
| **TEST CASE** | **T7** |
| Purpose of the case | Testing to see if income can be added. |
| Test data | Income category: salary  Date: 08/04/2019  Amount: 300000 |
| Class name | ExpmsTest |
| Function name | testAddinc |
| Expected result | Income should be added |
| Actual result | Income added successfully |
| Conclusion | Expected result matches with the actual result |

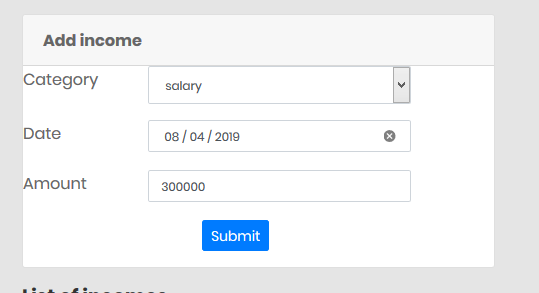
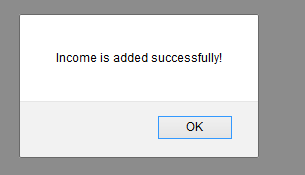
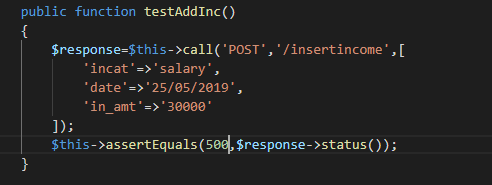
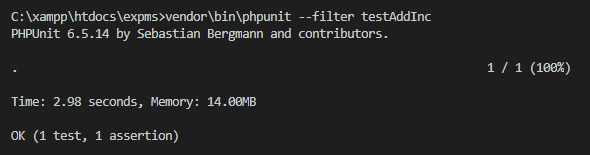


Figure 17:Add income test

**Output:**







|  |  |
| --- | --- |
| **TEST CASE** | **T8** |
| Purpose of the case | Testing to see if income can be updated |
| Test data | Expense category: salary  Date: 09/04/2019  Amount: 400000 |
| Class name | ExpmsTest |
| Function name | testUpdateInc |
| Expected result | Income should be updated |
| Actual result | Income upated successfully |
| Conclusion | Expected result matches with the actual result |

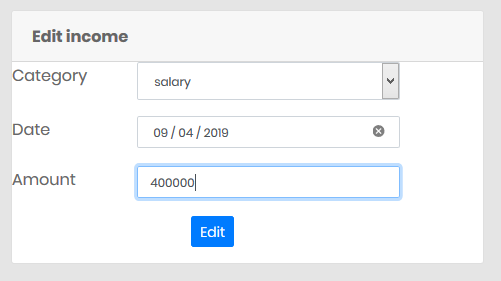
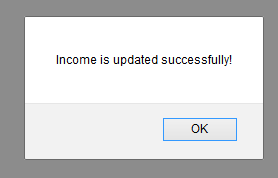
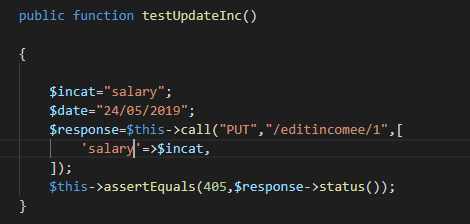
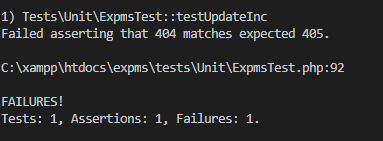


Figure 18:Edit income test

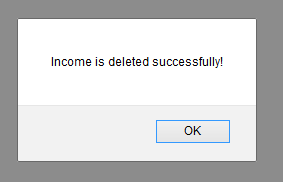
**Output:**

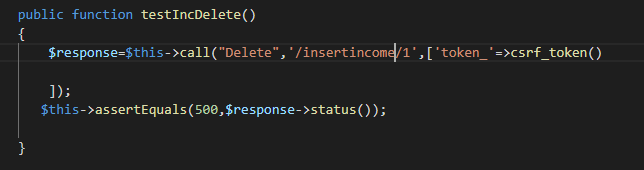


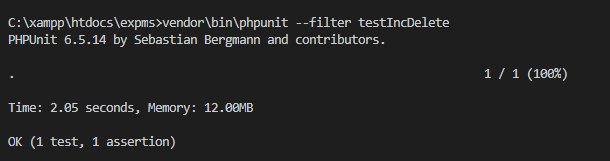




|  |  |
| --- | --- |
| **TEST CASE** | **T9** |
| Purpose of the case | Testing to see if income can be deleted . |
| Test data | Expense category: salary  Date: 09/04/2019  Amount: 400000 |
| Class name | ExpmsTest |
| Function name | testIncDelete |
| Expected result | Income should be deleted |
| Actual result | Income deleted successfully |
| Conclusion | Expected result matches with the actual result |

**Output:**





|  |  |
| --- | --- |
| **TEST CASE** | **T10** |
| Purpose of the case | Testing to see if income category can be added |
| Test data | Income category name: Pocket money |
| Class name | ExpmsTest |
| Function name | testAddInccat |
| Expected result | Income category should be added |
| Actual result | Income category added successfully |
| Conclusion | Expected result matches with the actual result |

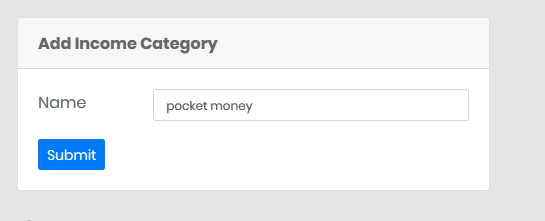
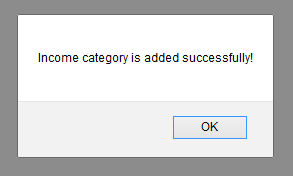
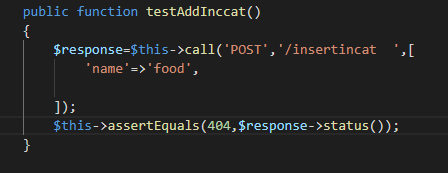
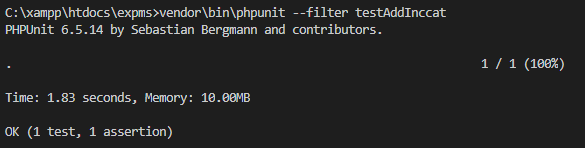


Figure 19:Income category test

**Output:**







# **Chapter 7: Other project work**

## **7.1 Risk management**

Risk management is the process of identifying, accessing and controlling threats or risks of a system. It is easier to identify risk before they occur so it can be solved faster when they actually occur. Risk management also accounts human factors including potential errors. Risk management helps minimize risks and time before they happen. Extra money and time are saved through risk management.

Impact= likelihood\* consequence

Risk Likelihood values are shown as follows

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

Risk Consequence values are shown below

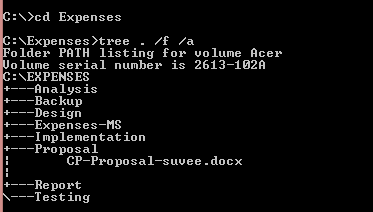
|  |  |
| --- | --- |
| Consequence | Value |
| Very low | 1 |
| Low | 2 |
| Medium | 3 |
| High | 4 |
| Very High | 5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Risks | Likelihood | Consequences | Impact | Solution |
| 1 | Lack of resources | 2 | 3 | 6 | Resources needed for the project must be available. |
| 2 | OS failure | 2 | 4 | 8 | Back up of file should be kept. |
| 3 | Electricity cut | 1 | 5 | 5 | UPS must be set up and backup should be done. |
| 4 | Mistakes of human | 3 | 3 | 9 | Testing should be done identify mistakes and solution. |
| 5 | Virus | 2 | 4 | 8 | Antivirus should be installed. |
| 6 | Human illness | 3 | 2 | 6 | Time should be managed properly. Extra hour should be done at the end. |
| 7 | Hard disk failure | 1 | 4 | 4 | Cloud back up should be done. |

## **7.2: Configuration management**

Configuration management is a system engineering process for maintaining consistency of a product's performance, functional, and physical attributes. It’s the process of saving and managing the directory of the file. The files of the project are saved in appropriate folders as it will be easy to find hence, saves time. Configuration management is easy to maintain reliability, performance and efficiency

Following is my folders in GitHub and tree structure:



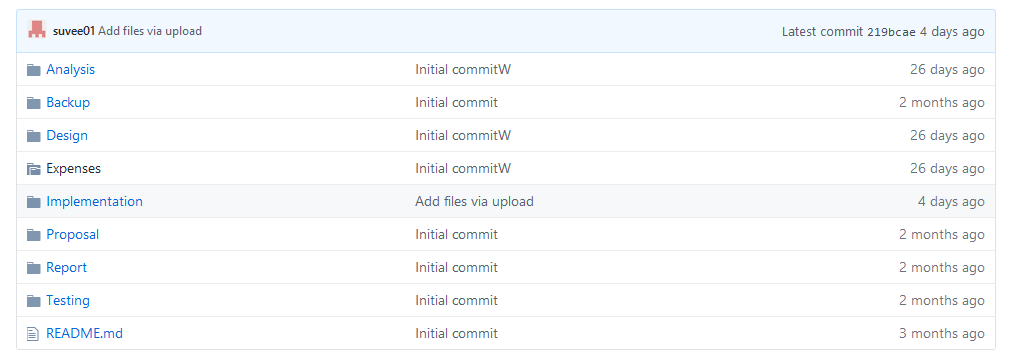


Figure 20:GitHub folder structure

The above directory is my GitHub folders. All the files are saved in appropriate folders. I have used six main folders such as analysis, design, implementation etc. This project is divided into parts hence, it is easy to maintain efficiency.

## **7.2 Future works**

Due to limited time, the project does not have all the features that can be implemented. So, for the future work, there are some features that can be used to make the project bigger and better. All of these future work features fall under won’t have prioritisation as they have not been implemented yet. Future works are as follows:

* Mobile Accessibility
* Receipt Imaging- uploading image of the receipt directly into the system
* Credit Card Integration- integrate credit card account right into the system.
* Graphs showing the trends over time.
* Website can be connected by social media
* Better UI design

# **Chapter 8: Conclusion**

At the end, the project has been completed by overcoming different obstacles. Expenses Management system is a website which helps people to manage their money by including the time and date and amount they spent and what they spent on. They can add the source from where they got their finance such as salary. They can get report on their monthly spending and income. User can also get the chance to know their saving money. This project is trying to make people from having to try to remember their financial details so, instead they can just upload it on the website and their time is saved.

I went through different phases to complete this project such as analysis, design, implementation and testing phase. First of all, I have used waterfall methodology for SDLC. To begin, requirements are gathered which are further divided into functional and non-functional. MoSCoW prioritisation is also used for requirement gathering process. Different diagrams such as use case diagram and initial class diagram were done for the analysis phase. Use case diagram shows the steps required for the project. For design phase, more diagrams were done such as ER diagram, activity diagram, sequence diagrams according to the requirement. For the coding phase, PHP programming language is used with Laravel framework. PHP is used because it is easy and effective to use. Coding is made easier by using MVC design pattern. The code is written in Visual Studio Code and diagrams are done In Visual paradigm and Star UML. For the documentation part, Microsoft Word is used.

Lastly, I have completed the task given according to the requirement. I have faced man y problems but also found solution. Many problems were overcome but at the end the project is finished. However, the project still lacks some features which is mentioned in future works.

# **Appendix A: Source code**

# 

Figure 21project folders

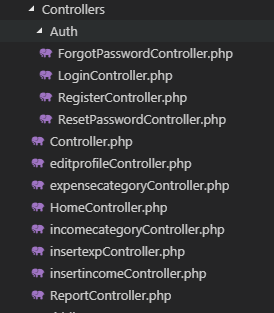


Figure 22:Controllers

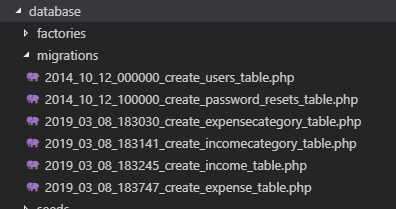


Figure 23:Database

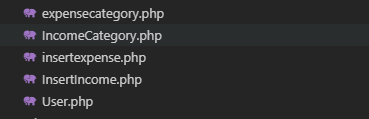


Figure 24:Model

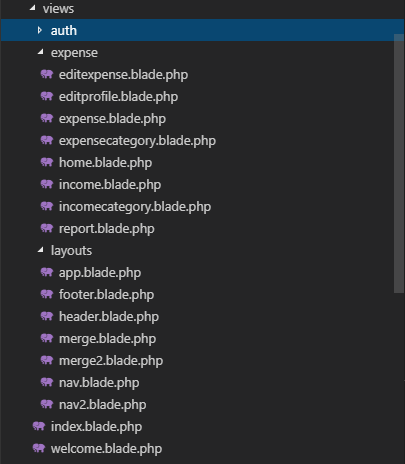


Figure 25:View

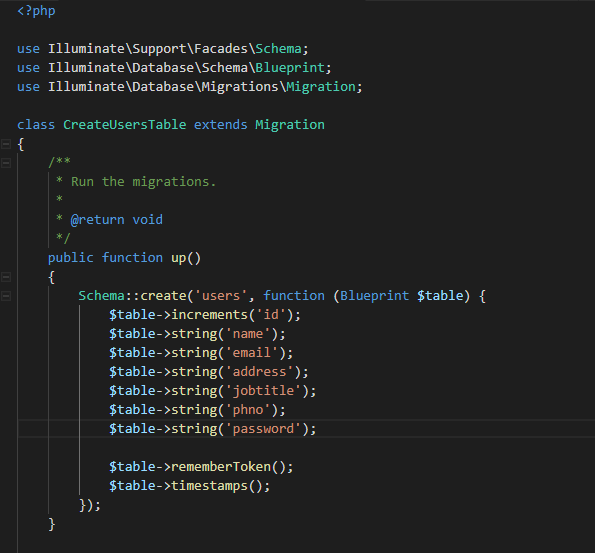


Figure 26:User table



Figure 27:Expense category table

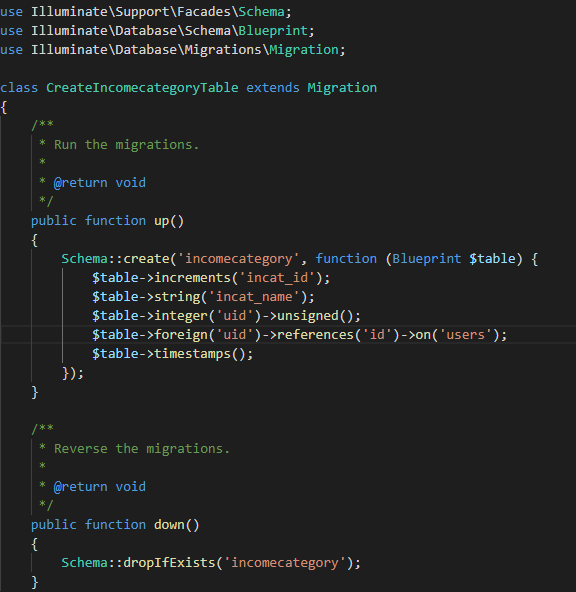


Figure 28:Income category table

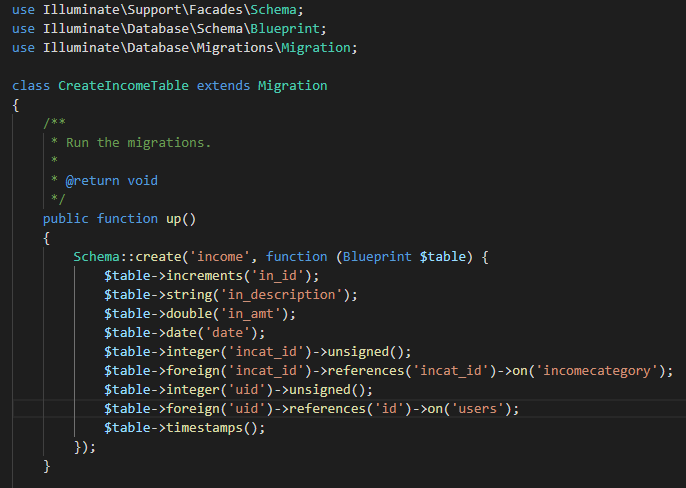


Figure 29:Income table



Figure 30:Expense table

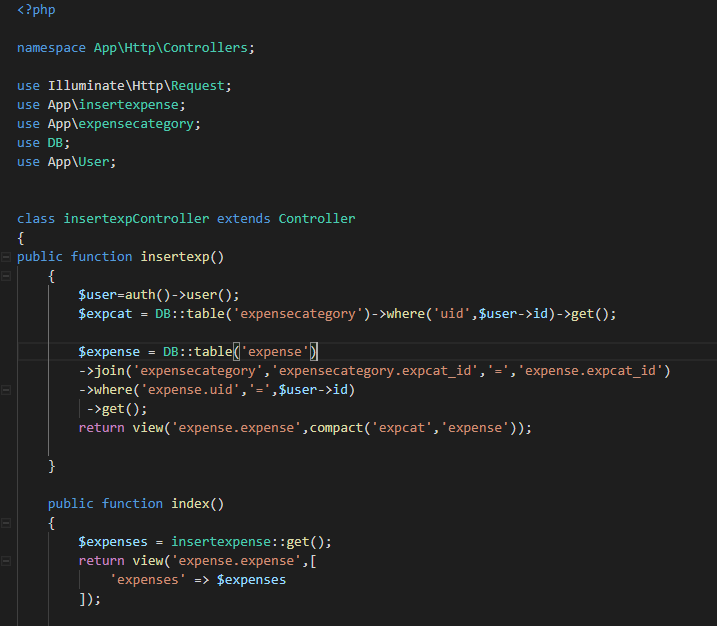


Figure 31:Expense controller

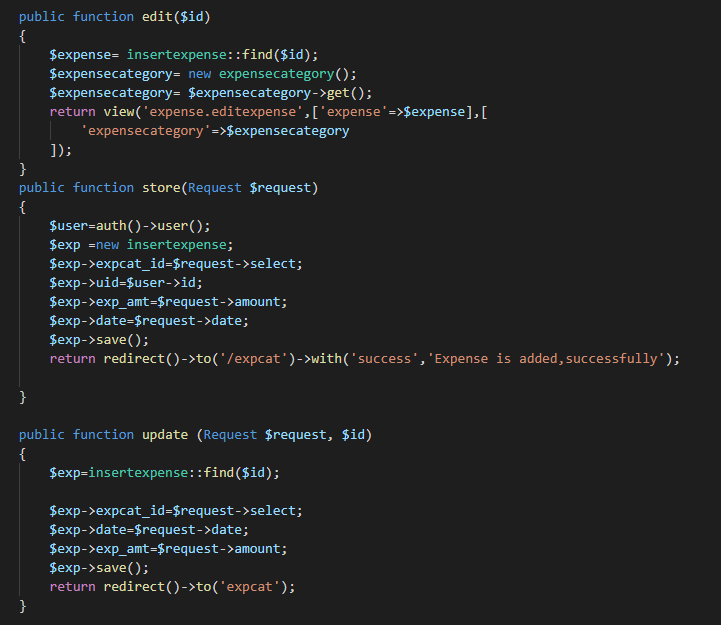


Figure 32:Add and update function

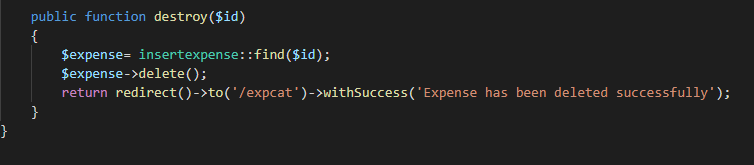


Figure 33:Delete function

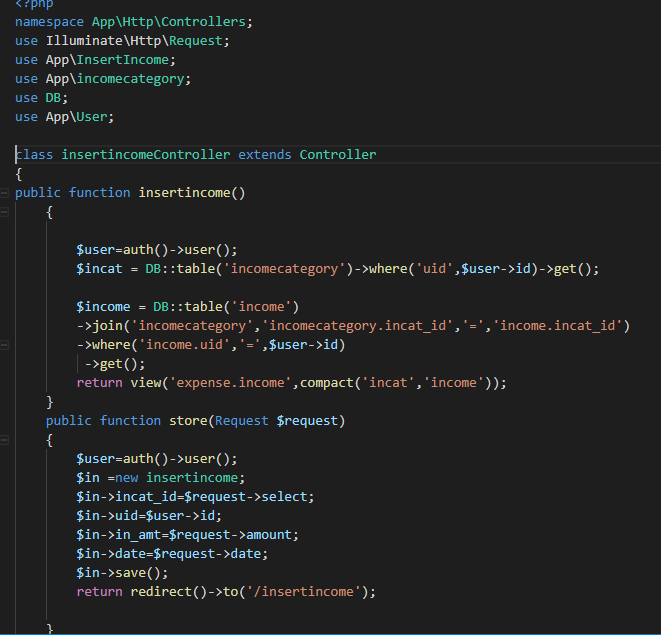


Figure 34:Income controller



Figure 35:Add, edit and delete function

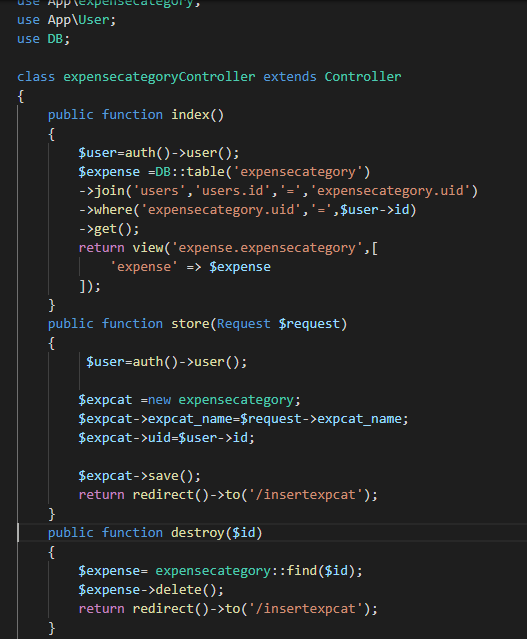


Figure 36:Expense category controller

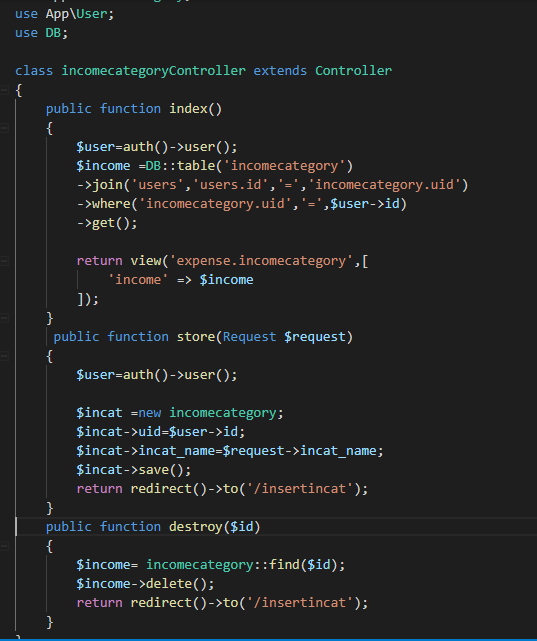


Figure 37:Income category controller

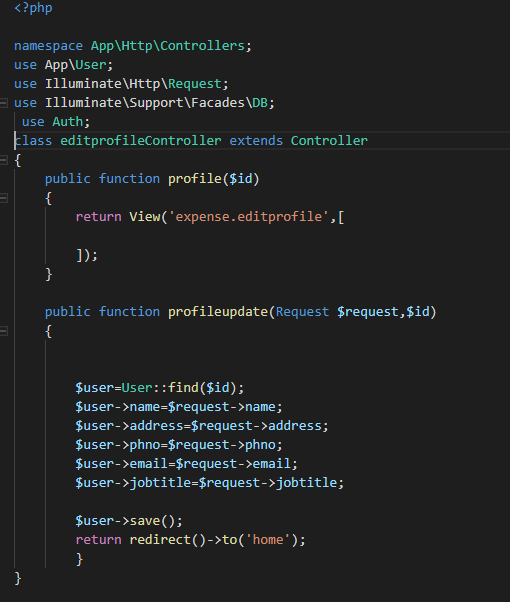


Figure 38:Edit profile controller

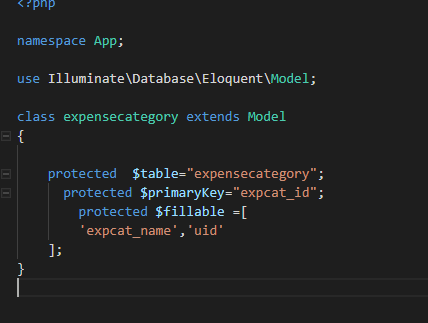


Figure 39:Expense category model

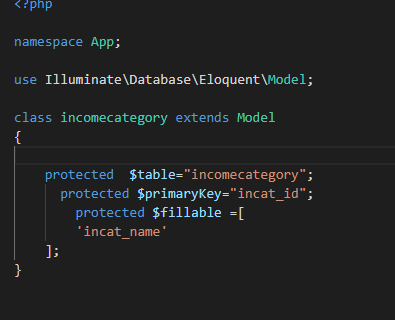


Figure 40:Income category model

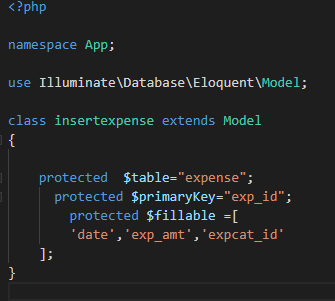


Figure 41:Expense model

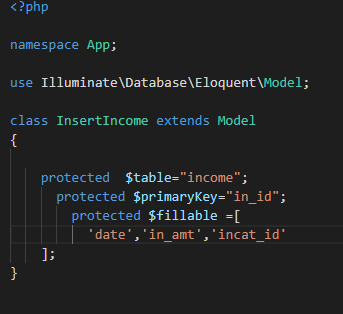


Figure 42:Income model

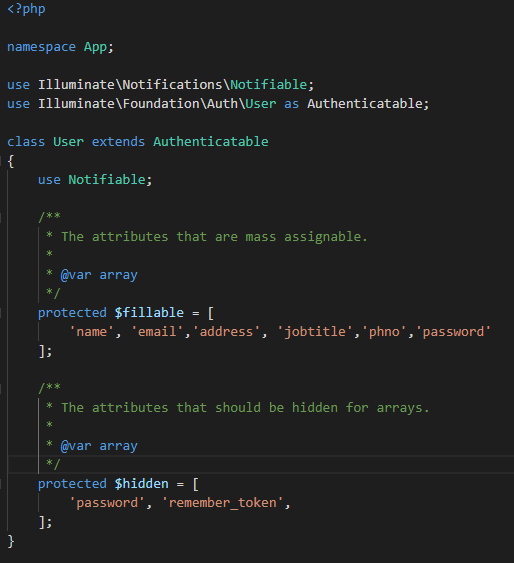


Figure 43:User model

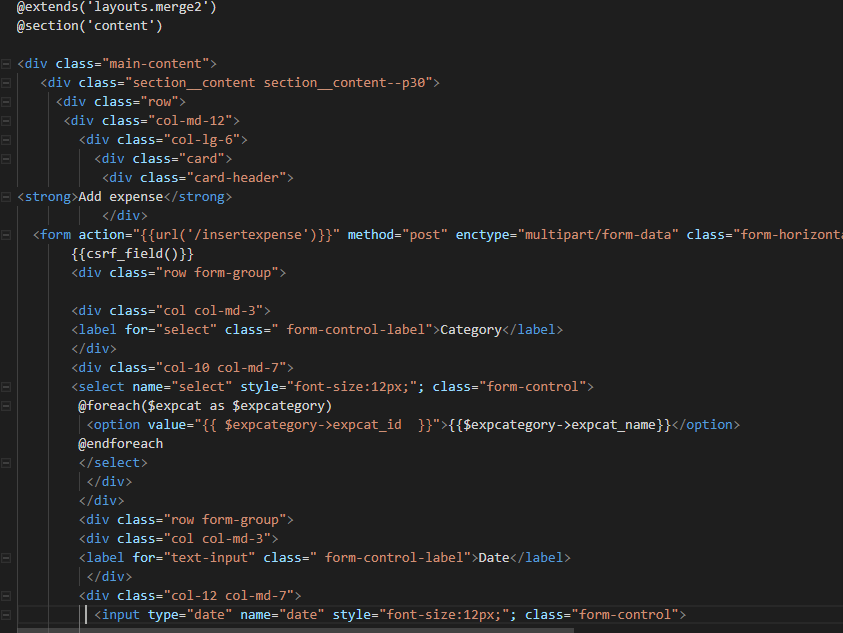
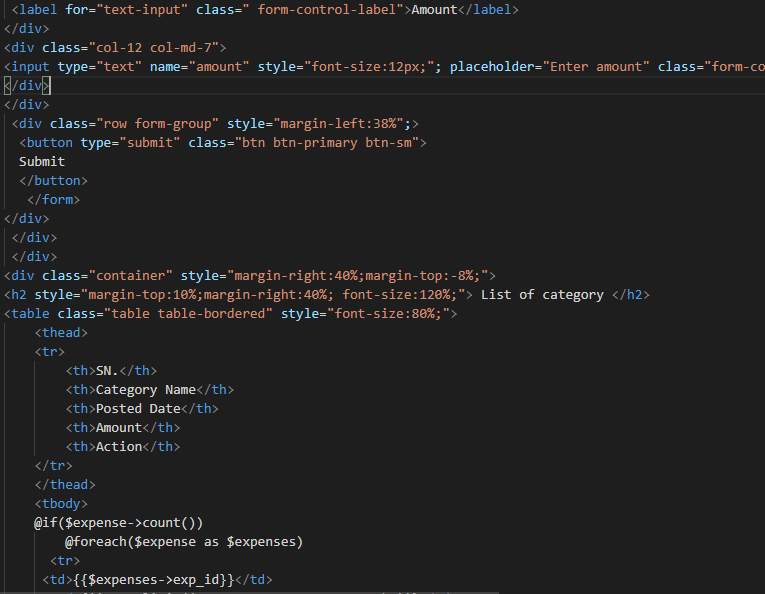
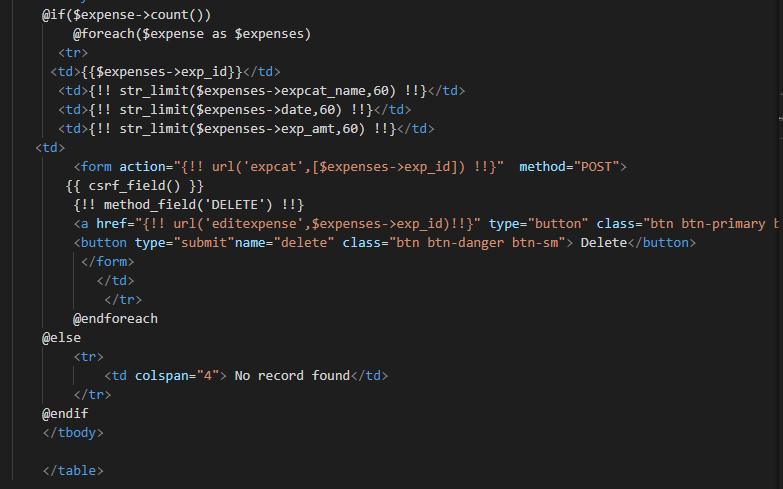


Figure 44:Expense View





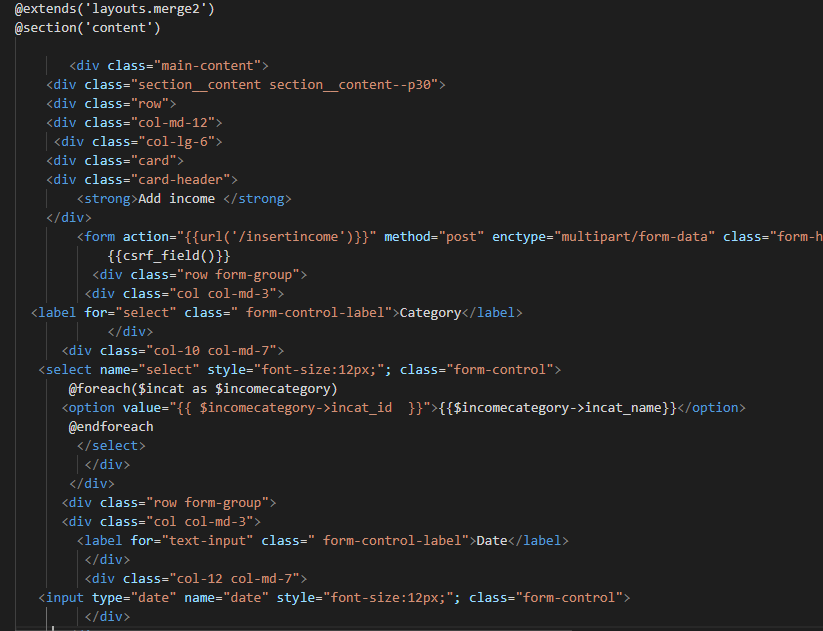
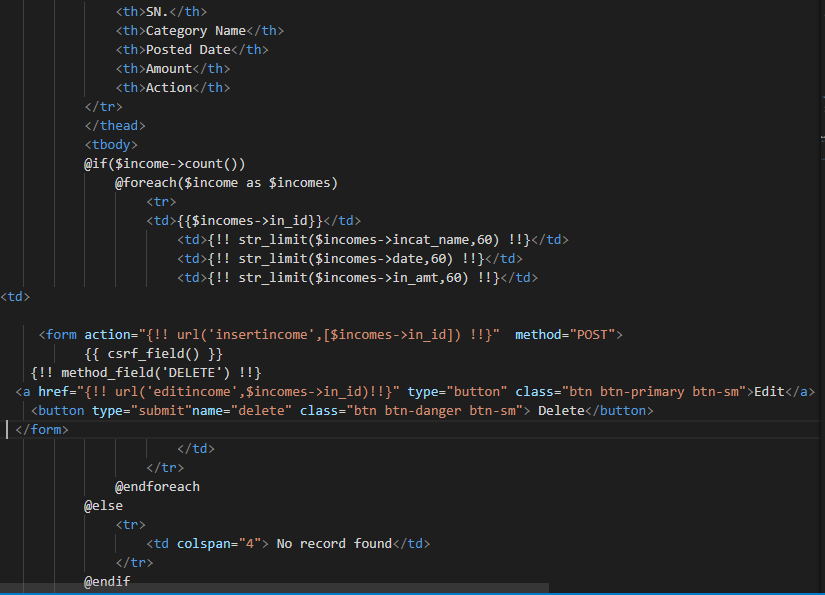


Figure 45:Income view



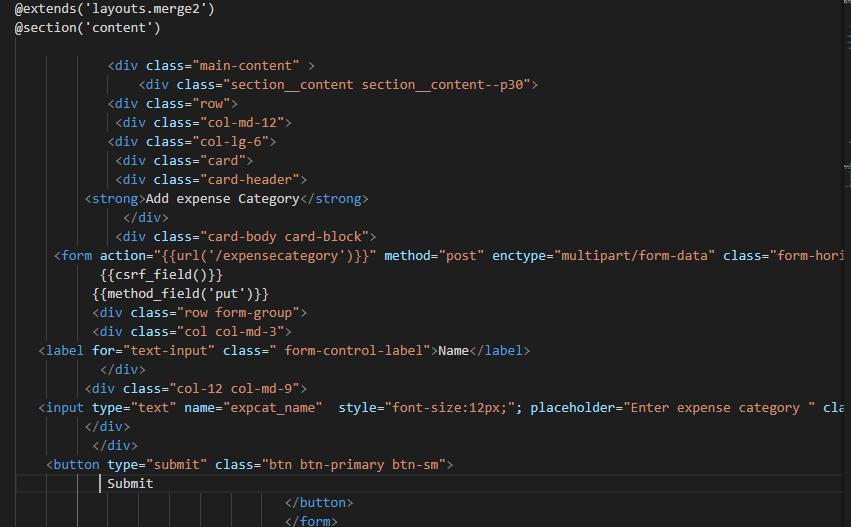


Figure 46:Expense category view

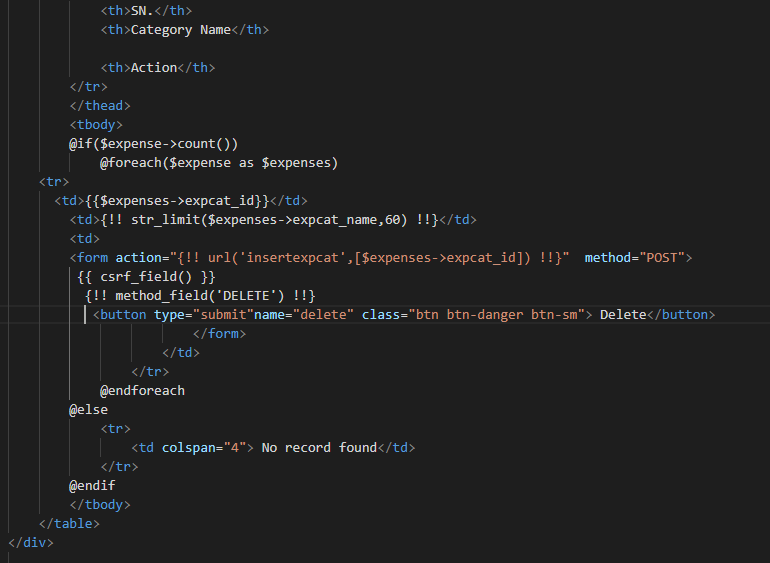
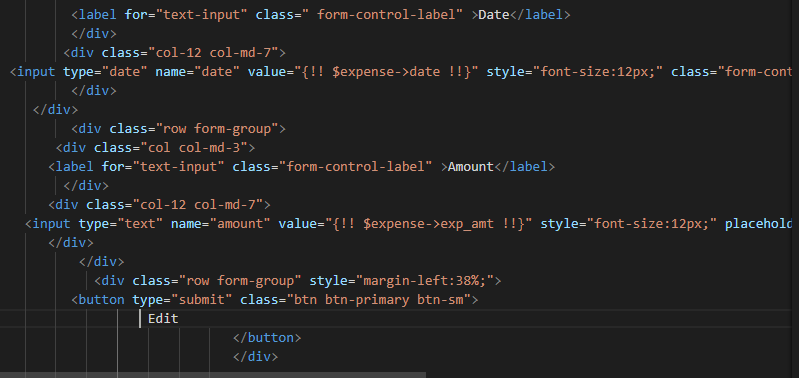




Figure 47:Edit expense View



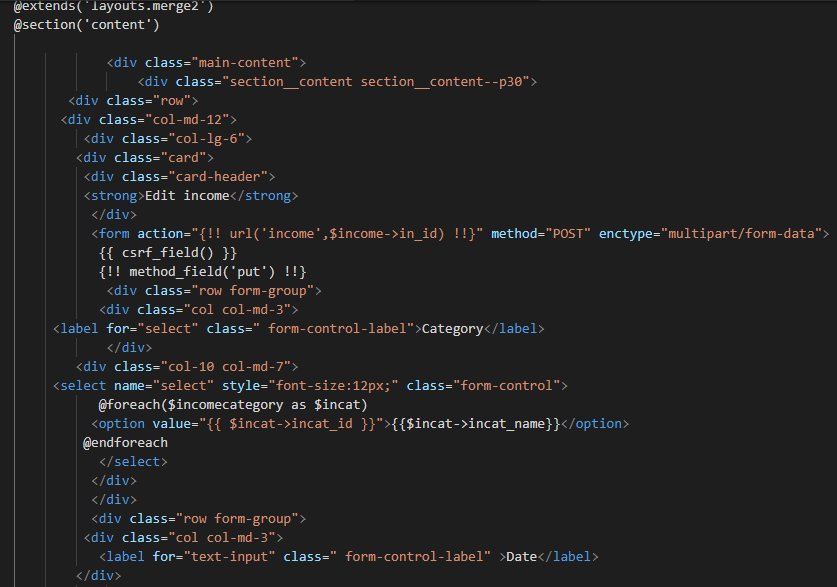
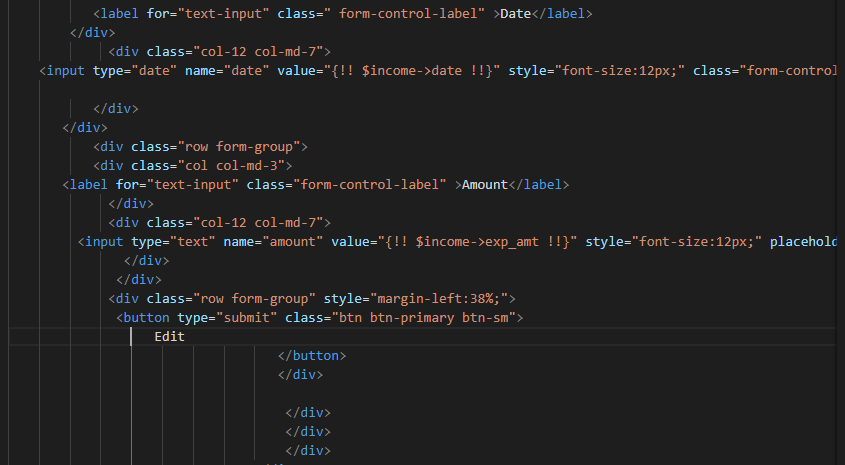


Figure 48:Edit income View



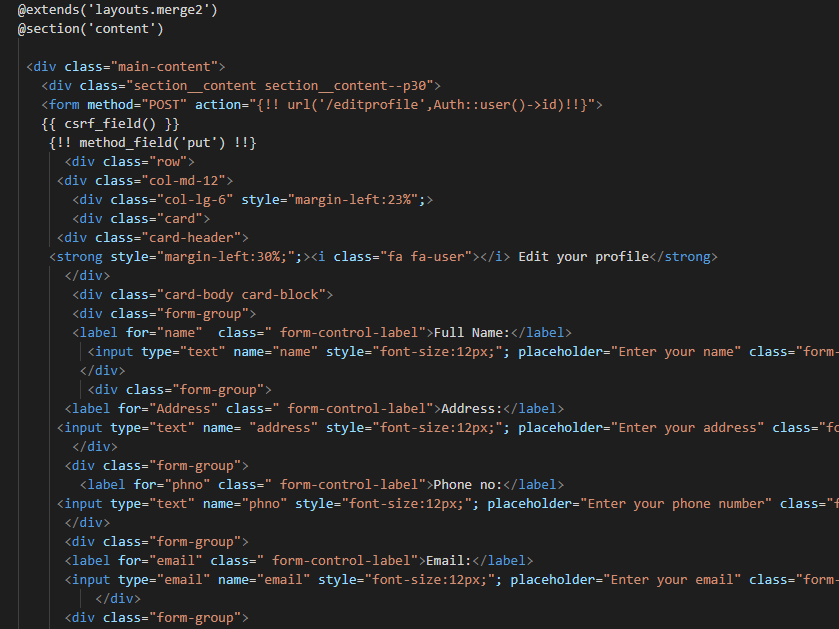


Figure 49:Edit profile view

#



Figure 50:Route





# **Appendix B: User Manual**

Expenses management is a system where user can log in and add their finance record. It helps them manage their money and time. Unregistered user cannot access this system as they have to register to use this system. Once they are logged in, only then they will be able to use the features.



Figure 51:Home page

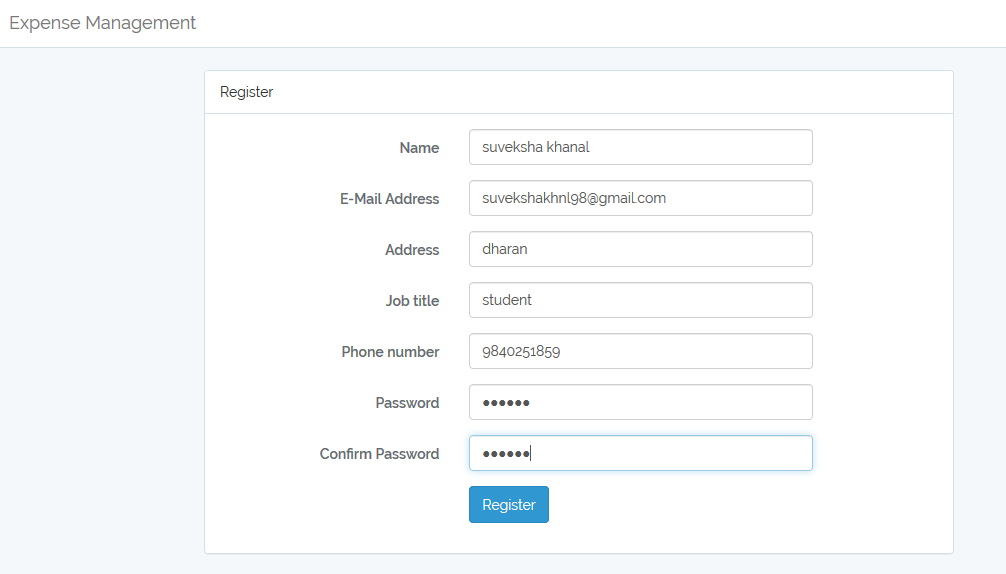


Figure 52:Registration

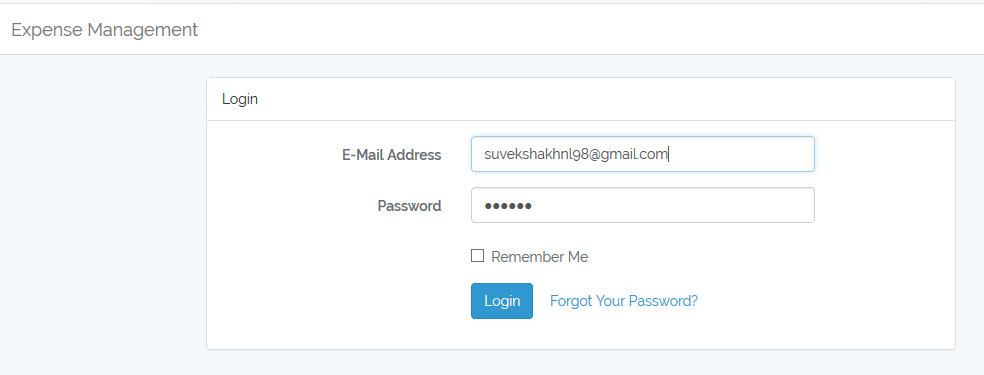


Figure 53:Login

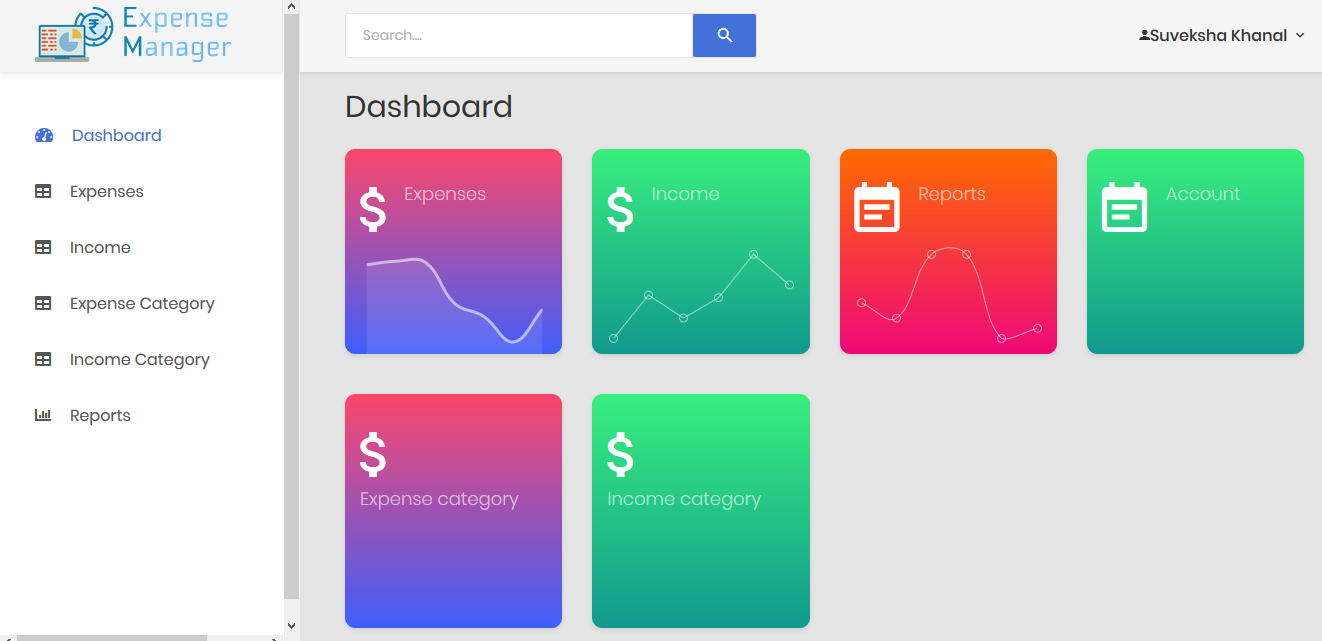
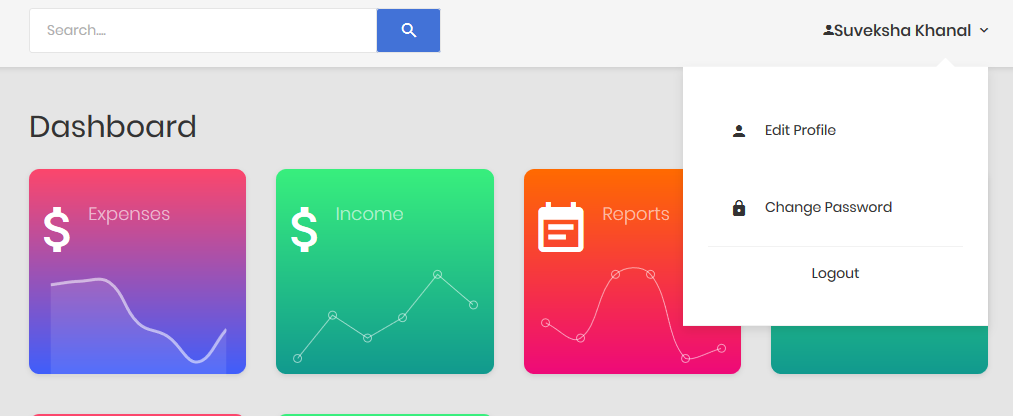


Figure 54:Dashboard



User can edit their profile

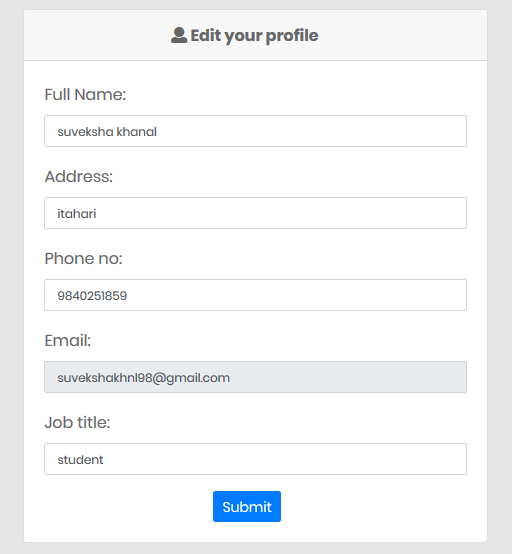


Figure 55:Edit profile

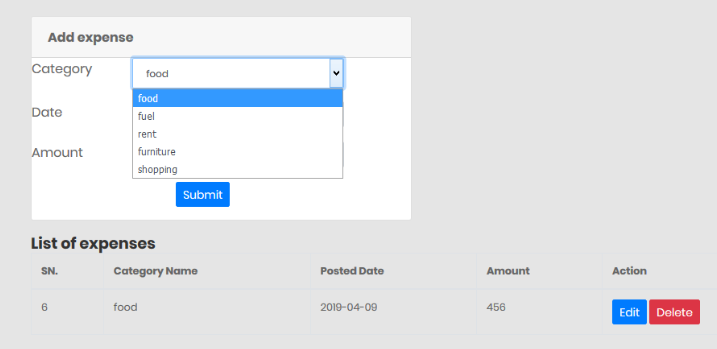
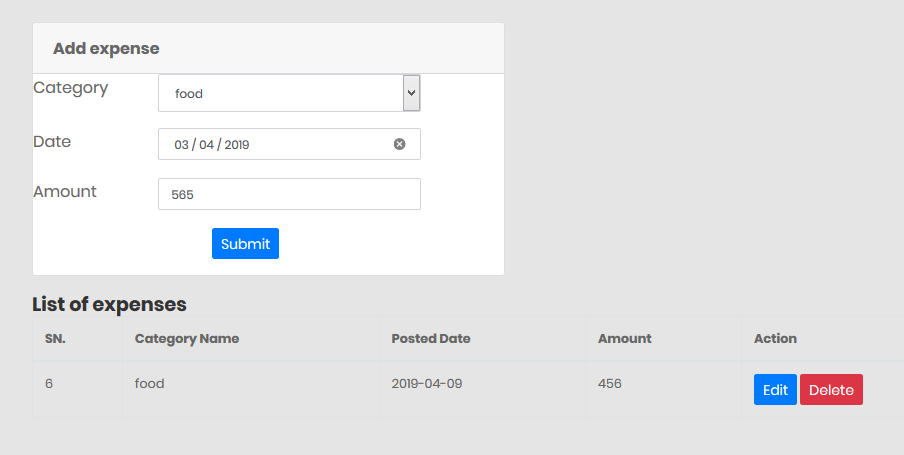
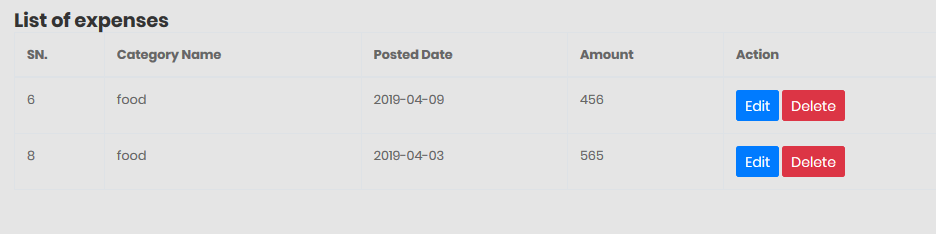


Figure 56:Expense page

Expenses is added by choosing expense category.





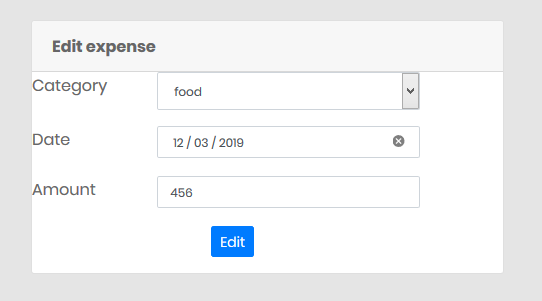
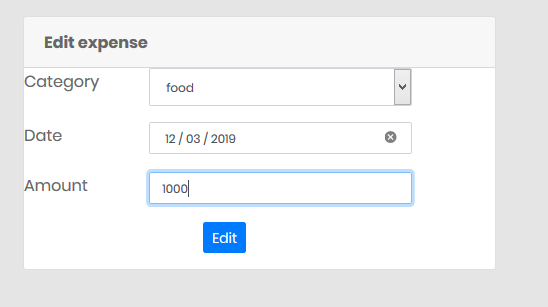
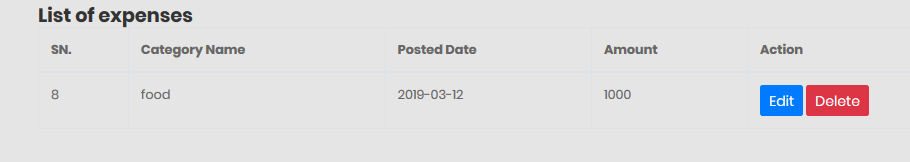


Figure 57:Edit expense page

Added expense can be edited by clicking on edit button.



Expense can also be deleted by clicking on delete button.



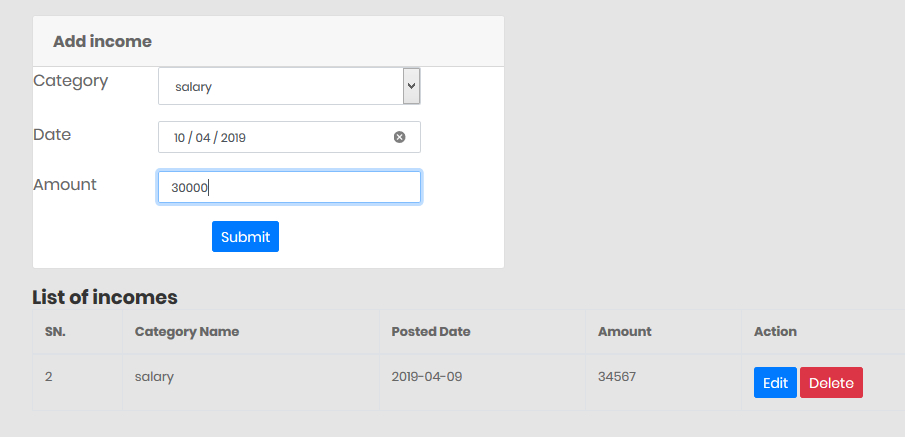
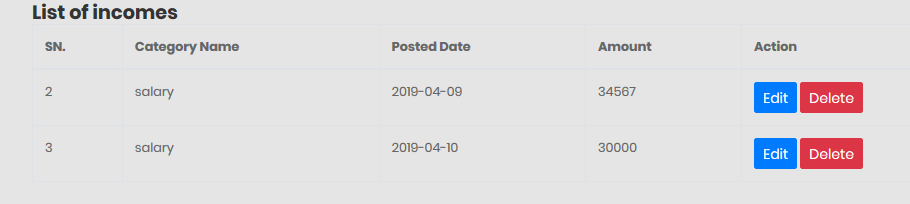


Figure 58:Income page

Income can be added by choosing income category

Income can be edited by choosing edit button

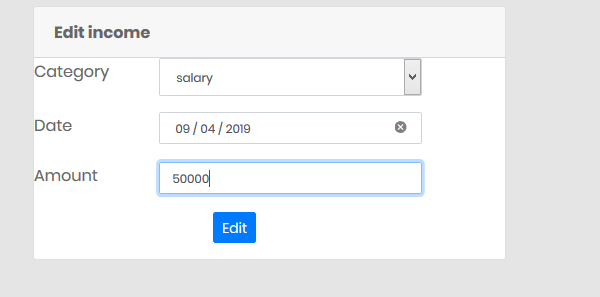
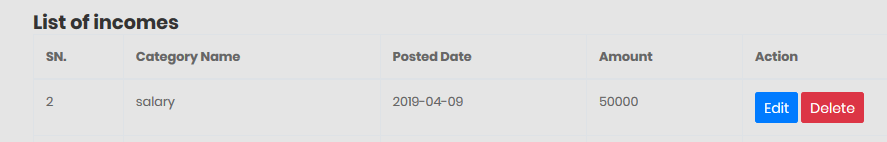


Figure 59:Edit income page

Income can be deleted by clicking on delete button.



User can add expense category as well as delete them with delete button..

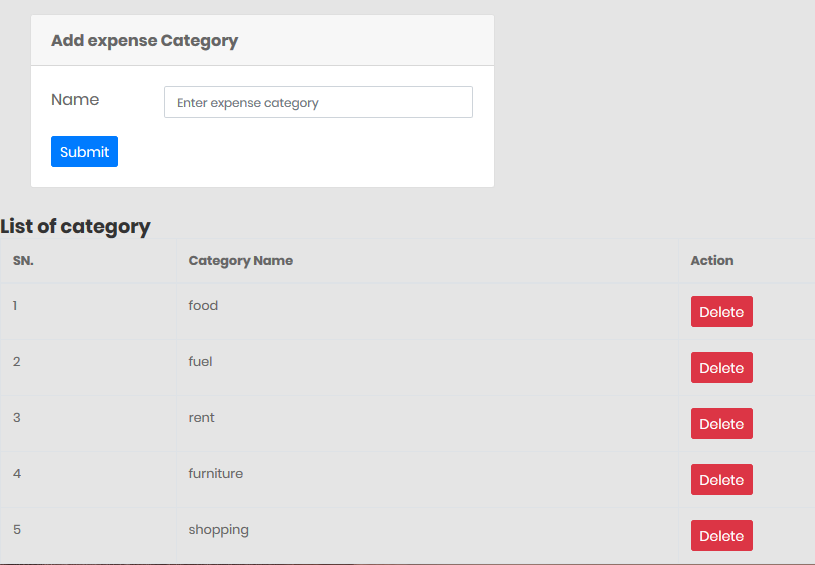
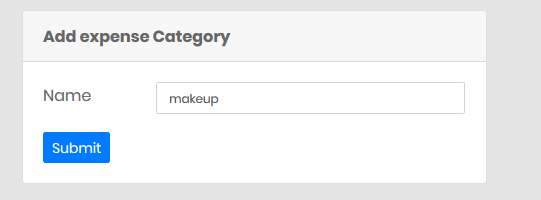
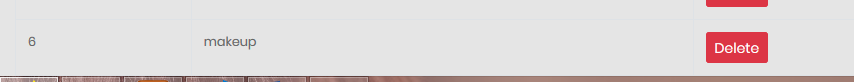


Figure 60:Expense category page





User can also add income category as well as delete them.

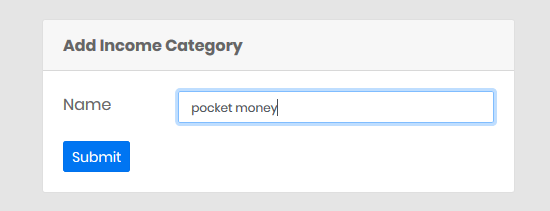
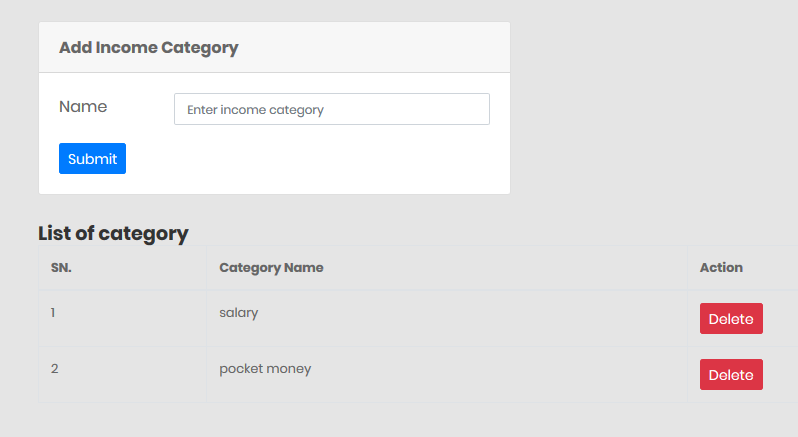


Figure 61:Income category page



Deleted category



# **Reference**

<https://www.g2.com/categories/expense-management>

<https://accounting-software.financesonline.com/c/expenses-management-software>

<https://kissflow.com/finance/expense-management/expense-management-tool-features/>

<https://www.kashflow.com/blog/what-is-expense-management/>

<https://www.webprojectbuilder.com/item/expense-management/live-demo/1>

<https://www.toolsqa.com/software-testing/waterfall-model/>

<https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>