CSE1005 – Software Engineering

Lab Slot: L41 + L42

DAILY EXPENSE TRACKER SYSTEM FOR EMPLOYEES

By Team 11:

19BCE7382 - KONDA VENKATESWARA REDDY

19BCD7246 - DEEPALA SRIHARSHITHA

19BCE7033 - ANDHAVARAM HARSHITA

19BCE7061 - DOLLU VAKULADEVI

19BCE7124 - PERNI BINDU MADHAVI

19BCE7585 - KOOKUTLA SATWIKA

19BCE7368 - HARSHITH RAJASEKHARAM

19BCE7557 - GADDAM PUNITHSAI REDDY

19BCE7659 - KANCHIRAJU HARI ANEESH SIDDHARTHA

19BCE7169 - R S V S R GANESH

Submitted To:

Dr. Subrata Tikadar

Assistant Professor Sr. Grade 1

SCOPE, VIT-AP University

Contents

1. ABSTRACT:

As the name itself suggests, this project is an attempt to manage employees' daily expenses in a more efficient and manageable way. The system attempts to free the user from the burden of manual calculation and keep track of the expenditure as much as possible. Rather than keeping a log of expenditures on smartphones and laptops, this system allows the user to keep track of the expenses and plans by preserving the past budget in mind.

With the help of this system, the user adds, deletes, and changes the currently entered bill entry efficiently for their reminder. The dashboard representation of the budget is the lucrative part of the system as it appeals to the user more and is easy to understand and incorporate for future planning.

2. INTRODUCTION:

2.1 – Goal

The objective behind this solution is to design a refined system that will allow employees to manage their expenses efficiently.

2.2 – Problem Statement

As of now, there is no such simple and easy-to-use solution, or we should say free of charge, which allows employees to track their daily expenditures. To do so an Employee has to keep a log in a diary or in a computer, also all the calculations need to be done which may sometimes result in errors leading to losses. As a result of the lack of a complete tracking system, daily entries of expenditures are constantly overwhelmed.

2.3 – Problem Solution

To fix the above-addressed problems, we are trying to design a system that would make the task of keeping the expenses in check an efficient and delightful job. This system will include a web application that will allow users to maintain an automated digital diary for their daily expenses.

2.4 – Target Audience

It is intended for employees who wish to know what their savings are at the end of each day, month, and year, and whether they are making progress or not.

The system can be used by those who are careless in handling the device e.g., who usually finds their device stolen, broken etc. they can recover their data from the server

2.5 – Scope

This program has a large market because it can be used by the employees who want to keep track of their costs and save for future investments, among other things. There are no specific range requirements, professions, or genders that will be targeted yet, it will be widely used. It can assist people in a variety of ways, including:

- Encourages to think about money by instilling discipline and organization.
- Crisis Management
- Budget Calculator
- Being aware of spending habits
- Financial awareness, Effective spending, tracking and reporting

2.6 – Limitations

- Users have to enter every record manually.
- The category divided may be a blunder or messy.
- The person who is handling the system must have some technical knowledge.

3. REQUIREMENT GATHERING METHODS:

Source of Data: User

3.1 – Interview

Interviews were carried out with some random persons and asked about their daily life expenses. While taking those samples, we found that they always broke off at the end of the month, which means they do not calculate those expenses daily. So, to control unnecessary spending habits, an expense tracker is a must. While using this tracker, they can control their expenses and save some of those.

3.2 – Questionnaire

A set of questionnaires was prepared to determine how people track their budgets and what features they would like to see in our web application. According to the report, most people do not plan for what they have earned and do not keep track of their expenditures. Many want to see their expenditures daily, weekly, monthly, and yearly in dashboards, categorical or graphical representations.

4. FEASIBILITY ANALYSIS:

Feasibility Study is a study to evaluate feasibility of a proposed project or system. It is one of the important four stages of the Software Project Management Process.

4.1 – Technical Feasibility

This assessment focuses on the technical resources available. It helps to determine whether the technical team is capable of converting the ideas into working systems. It also involves evaluation of the hardware, software and other technology requirements of the proposed system.

Software Type	Web Application
Language Used	Python 3.8
Framework	Flask
Database	MySQL 5.x
User Interface Design	HTML, CSS, JQUERY, JAVASCRIPT

4.2 – Operational Feasibility

This assessment has a simple UI. Any employee with the basic technical knowledge can use this Expense tracker. Expense tracker takes a few seconds to take you from the home screen to the front page. With a click data are entered.

4.3 – Economic Feasibility

Effort and time of every team member is the cost involved for this project. Also, the user does not need to pay a single penny to use this application. They can access this application through any device by connecting to the internet. Hence, DET for Employees is economically feasible.

4.4 – Legal Feasibility

Our system is legally feasible because this application provides confidentiality to user data so it is authentic for everyone to use it.

5. RISKS INVOLVED:

- Following possible risks are involved in the proposed system: Inherent Schedule Flaws as in software development given the intangible nature and uniqueness of software, is inherently difficult to estimate and schedule.

- Requirements Inflation i.e., the project progresses more and more features that were not identified at the beginning of the project emerge that threaten estimates and timelines.

- Specification Breakdown i.e., when coding and integration begin it becomes apparent that the specification is incomplete or contains conflicting requirements

6. FUNCTIONAL REQUIREMENTS:

Server Side:

Server	XAMPP
RAM	2 GB
Hard Disk	50 GB
Processor	2.2 GHz

User Side:

Web Browser	Mozilla, Google Chrome, Opera, Brave
Operating System	Windows 10, Linux or MAC OS

7. NON-FUNCTIONAL REQUIREMENTS:

7.1 – Reliability

Reliability in Employee Expense Management will be ensured by thorough testing at each level. Test scenarios will be established to reflect the necessary level of reliability required of Employee Expense Management.

7.2 – Security Requirements

a. Passwords shall be displayed as "*" in the web pages wherever required.
b. Proper authentication is required for users to access any of the web pages including the home page.
c. Every user of the system is assigned a unique login and password to access the application over the internet.

7.3 – Maintainability

Employee Expense Management is completely created using basic HTML, PHP language which adds ease of understanding. All the state transition diagrams for each activity reflects the ease of understanding the project.

7.4 – Portability

Employee Expense Management is light software which does not require heavy computational power to run. Basic computers with little updated software can run it. As it can run in any windows operating system which is used by most of the users also make it easy to use the service.

7.5 – Usability

The system should be very easy to use with minimal required training. Individuals of varying skill levels and technical competence will use the system.

7.6 – Availability

The system is available all the time, no time constraint.

8. SPECIFIC REQUIREMENTS:

User Login

The system can be used by many people of that company. But accessing files should only be done by some people. This ensures security and no breach of the database.

- **Purpose:** Gateway of the employee expense management portal
- **Input:** Users select their designation and choose the dialog box beside their designation.
- **Processing:** System checks the validity of their ID and password.
- **Output:** System opens the portal for correct users.

Expense Creation Page

- **Purpose:** To create the expense.
- **Input:** User enters the product name and provides the expense.
- **Processing:** System adds all the expenses entered by the user along with the date to a table.
- **Output:** System displays all the expenses in a table form.

Expense Editing Page

Employee Expense management portal allows Employees to add new expenses and prepare a log based on them.

- **Purpose:** Editing portal of the expense management system
- **Input:** Users add new expenses and the amount related to it.
- **Processing:** System enters input into tables of a particular format.
- **Output:** system displays the entered details in a table form

9. DESIGN AND METHODOLOGY (UML DIAGRAMS):

9.1 - System Architecture Design (Three-Tier Architecture Design):

- The Presentation layer requires skills such as HTML, CSS, and possibly JavaScript, plus UI design.
- The Business layer requires skills in a programming language so that business rules can be processed by a computer.
- The Data Access layer requires SQL skills in the form of Data Definition Language (DDL) and Data Manipulation Language (DML), plus database design.

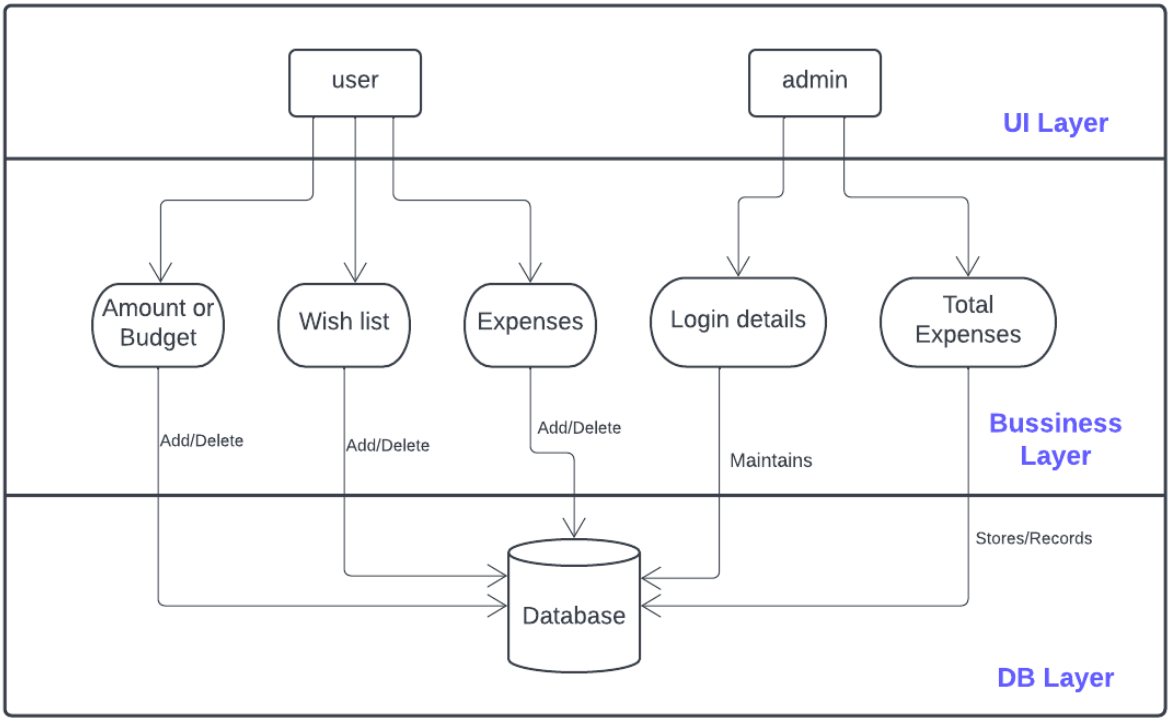


Figure 1: Three-Tier Architecture Design of Expense Tracker

Presentation Layer (UI Layer):

The presentation phase is the user recognition and communication layer of the app, in which the end-user interacts with the app. Its main purpose is to display information and collect information from the user. The user can register or log in through the web browser. The user request is sent to the web browser. The HTTP protocol is used to take user to the next page.

Business Layer (Logic layer):

The logic tier is where all the thinking happens, and it knows what your application is allowed and what might happen, and it makes some decisions. This logic section is also the one that writes and reads data in the data section. The user is directed to the web page once he gets registered/logged in. Web application is built with HTML, CSS, JAVASCRIPT. The data of the user is sent to the Database through XAMPP Server and the Data-tier starts from here.

Data Base Layer:

The data tier, sometimes called database tier, data access tier, or back-end, is where the information processed by the application is stored and managed. In the data layer, Users' data is stored in the database using MYSQL, through the XAMPP server. The data in the database is again used by the logic layer. Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

9.2 - Data Flow Diagram (DFD):

A Data Flow Diagram (DFD) is a structured analysis and design tool that can be used for flowcharting. A DFD is a network that describes the flow of data and the processes that change or transform the data throughout a system. This network is constructed by using a set of symbols that do not imply any physical implementation. It has the purpose of clarifying system requirements and identifying major transformations. So, it is the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. DFD can be considered to an abstraction of the logic of an information-oriented or a process-oriented system flow-chart. For these reasons DFD's are often referred to as logical data flow diagrams.

LEVEL 0 DFD:

The Level-0 DFD, also called the context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

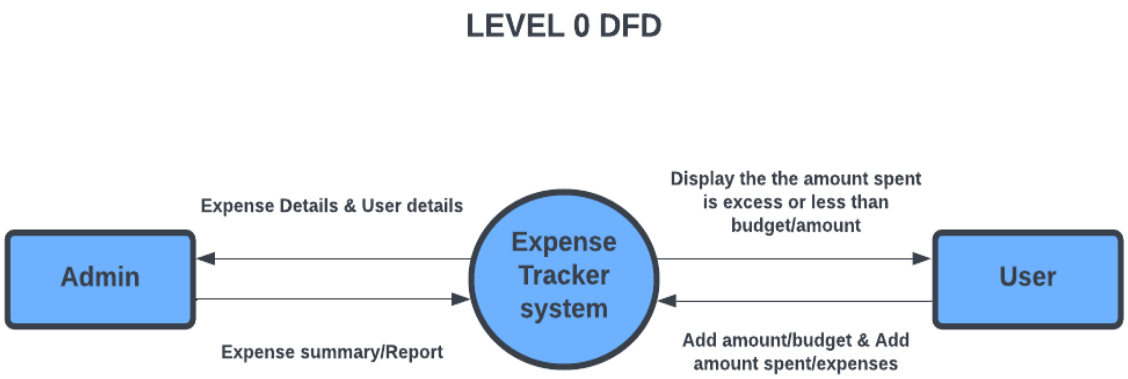


Figure 2.1: Data Flow Diagram Level-0 of Expense Tracker

LEVEL 1 DFD:

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. At this level, we highlight the main objectives of the system and break down the high-level process of 0-level DFD into subprocesses

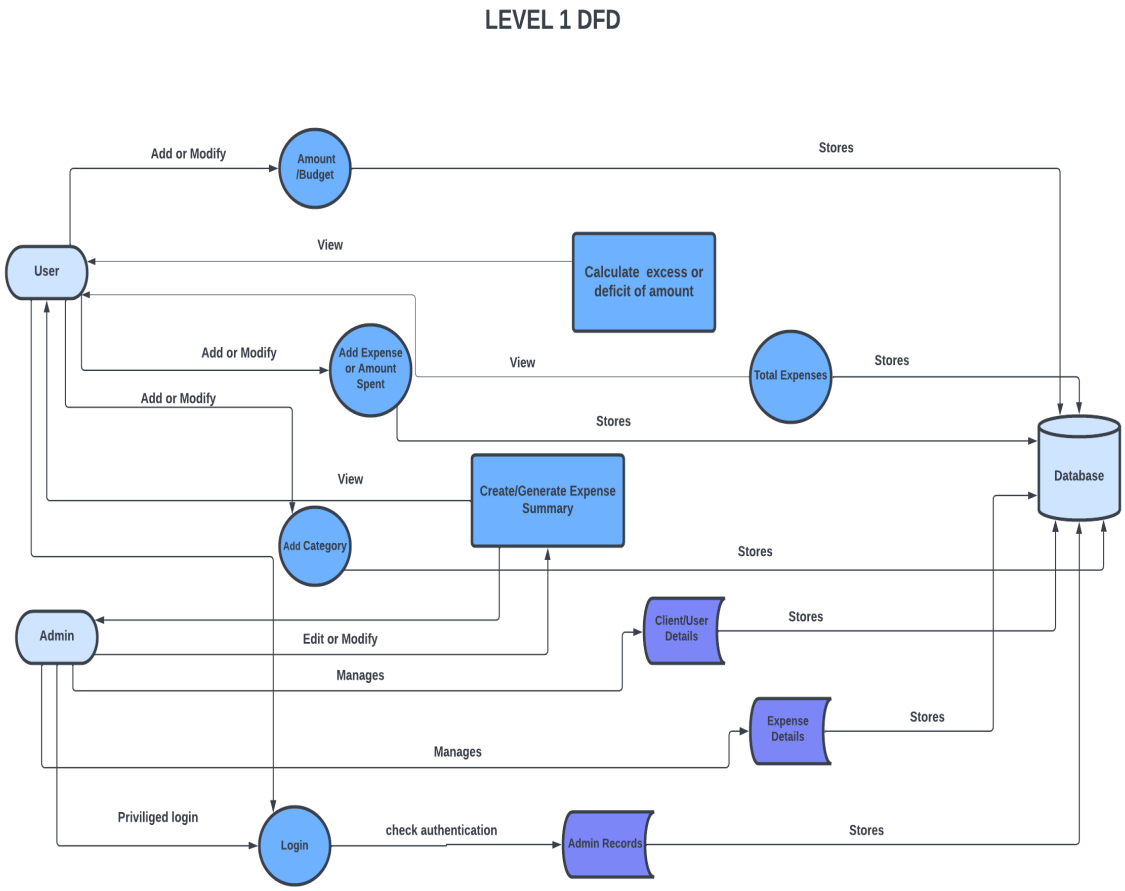


Figure 2.2: Data Flow Diagram Level-1 of Expense Tracker

9.3 - ENTITY RELATIONSHIP DIAGRAM (ERD):

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database.

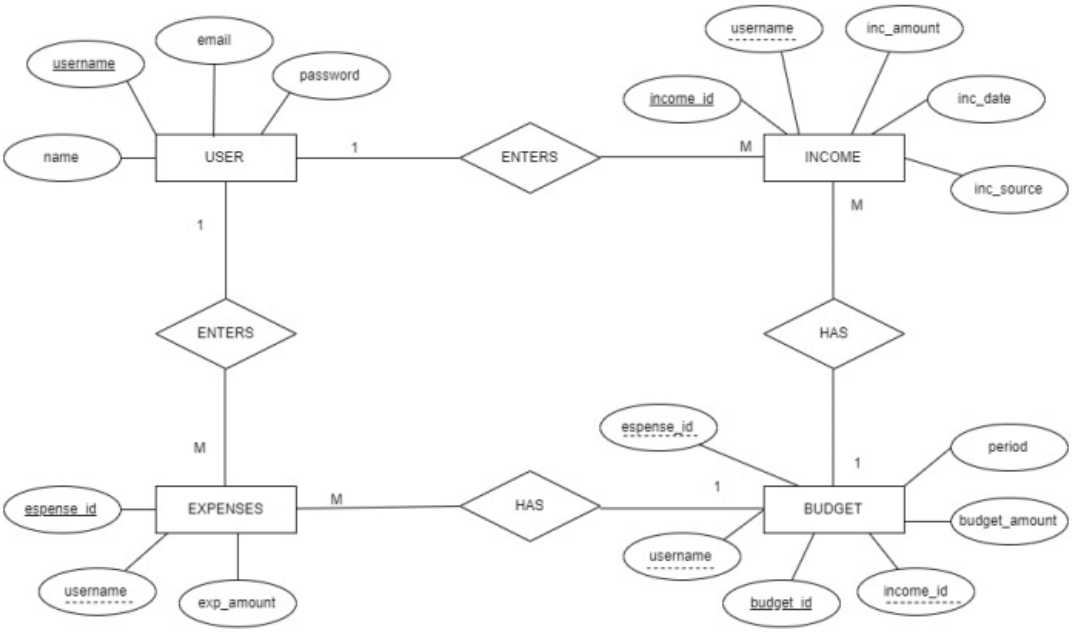


Figure 3: ER Diagram of Expense Tracker

9.4 - USE CASE DIAGRAM:

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

Once the user gets login, he/she can access the website. After entering the website, he/she can add the items list, amount or budget details and expense or amount spent on the item. Based on that he/she will get the price that exceeds or deficit the amount based on expense and the account holder/user will get a report on expenses/amount spent by the user.

Admin has the access to check user profiles and Expenses details. Admin can also edit the report/summary generated. Users can add and delete their budget, expense and the category list anytime.

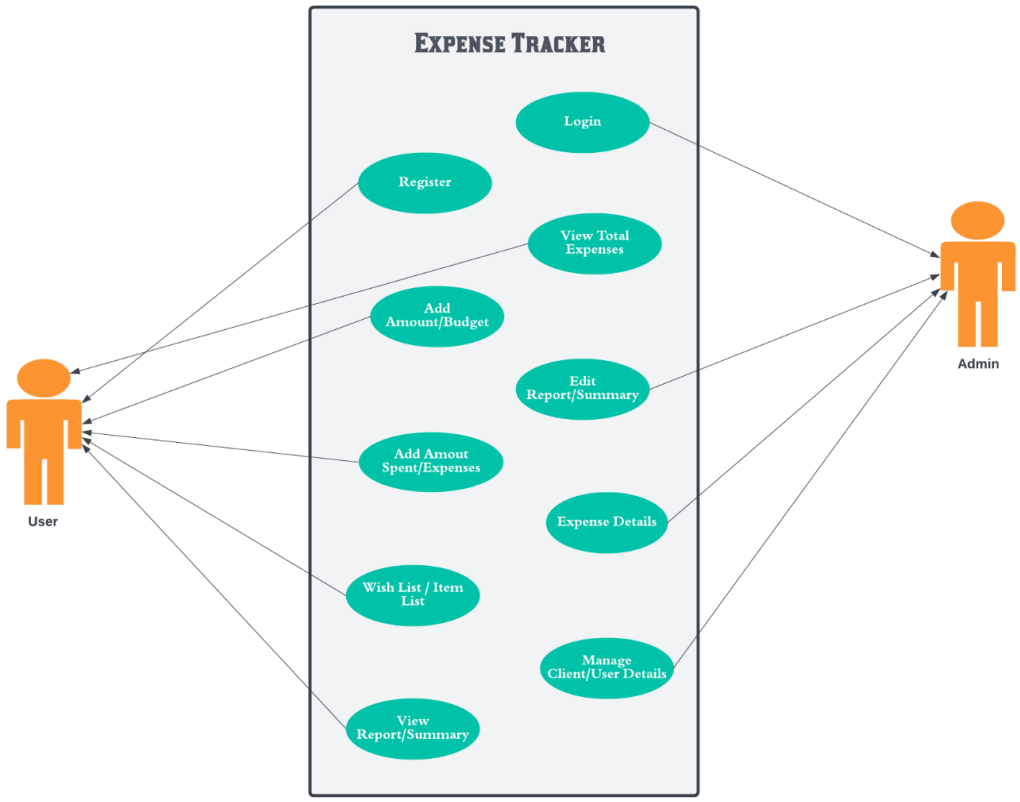


Figure 4: Use Case Diagram of Expense Tracker

9.5 - ACTIVITY DIAGRAM:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Activity Diagram is similar to use case diagram. In Use Case Diagram it shows the activities done by the user and admin whereas in case of activity diagram it shows the sequence of activity that is done one after another. Once the user gets login, he/she can access the website. After entering the website, he/she can add the items list, amount or budget details and expense or amount spent on the item. Based on that he/she will get the price that exceeds or deficit the amount based on expense and the account holder/user will get a report on expenses/amount spent by the user. This is the sequence of the activities done in Expense Tracker.

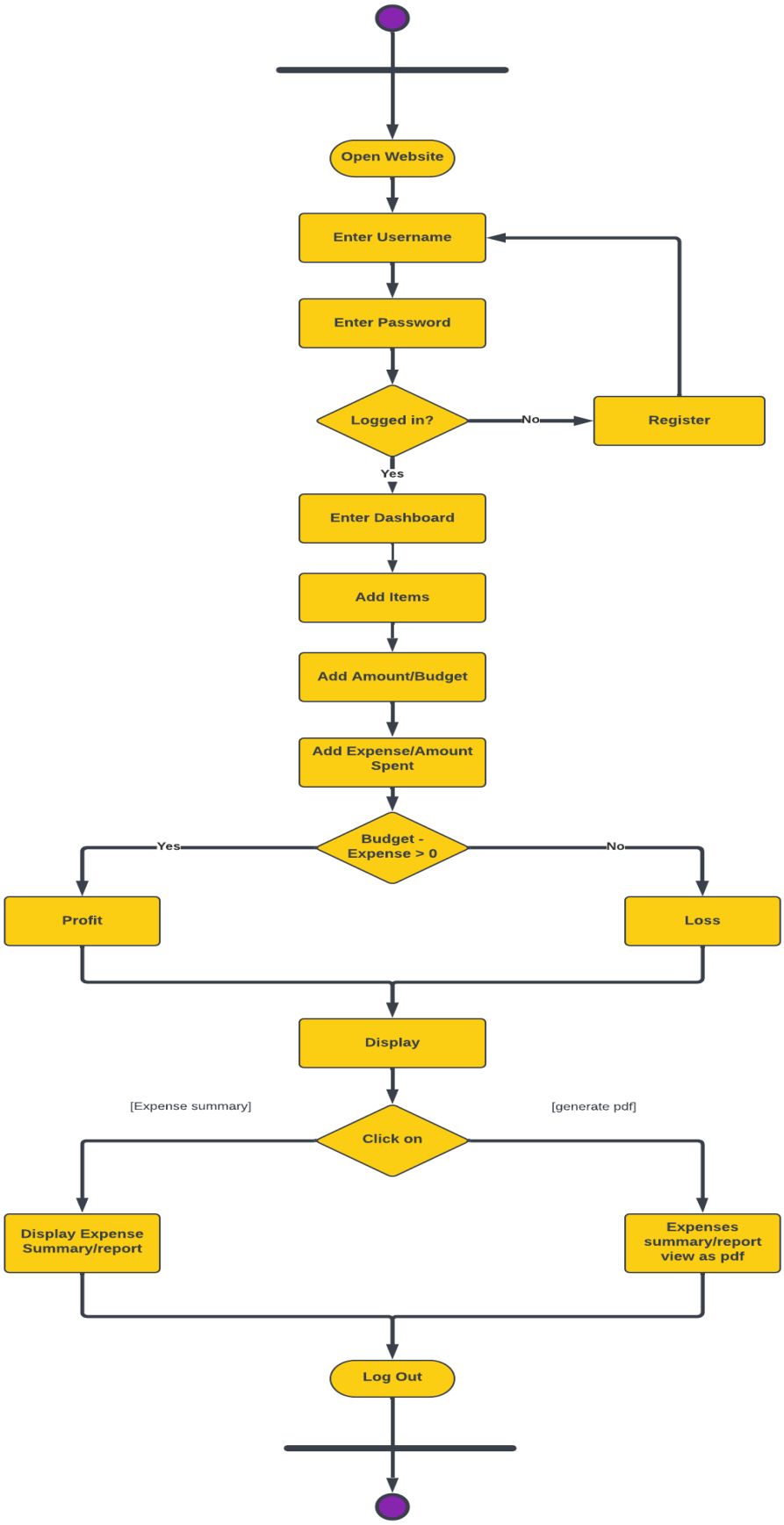


Figure 5: Activity Diagram of Expense Tracker

9.6 - SEQUENCE DIAGRAM:

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you to both documents and validate your logic, and are used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system.

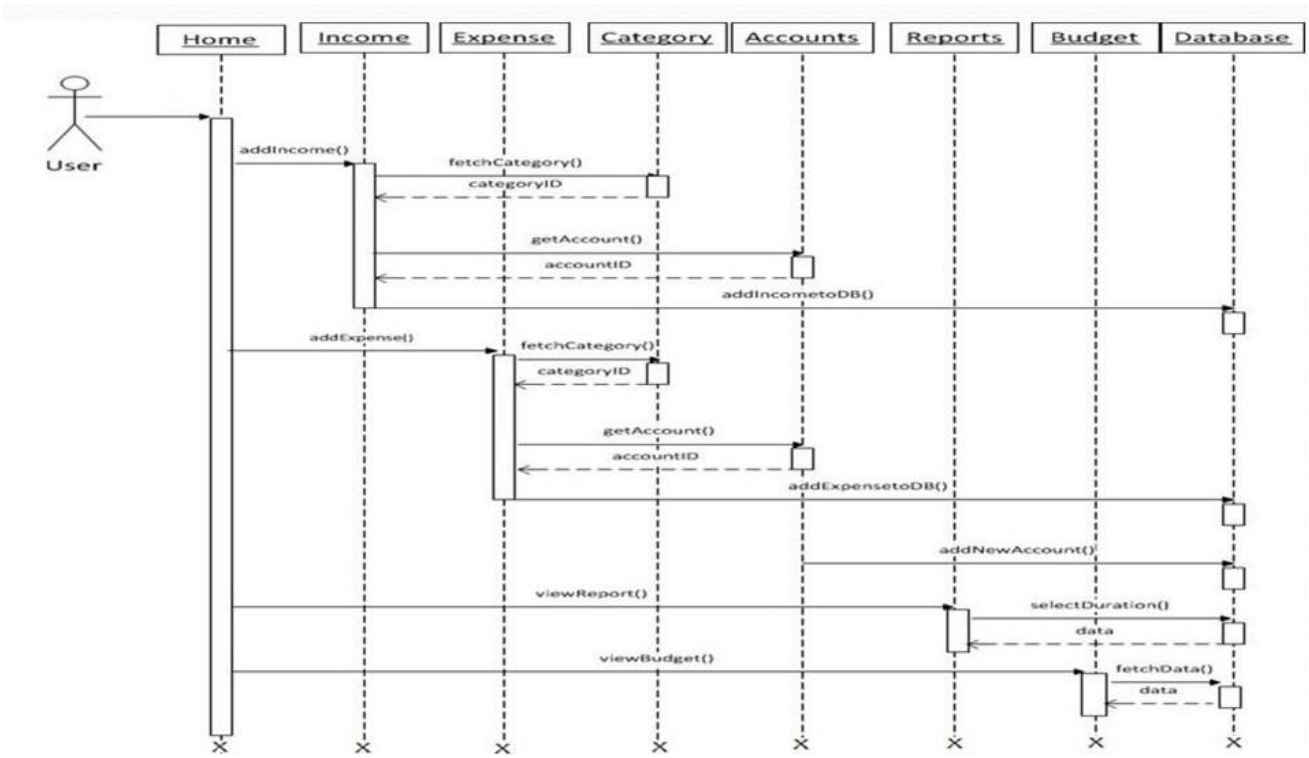


Figure 6: Sequence Diagram of Expense Tracker

9.7 - CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

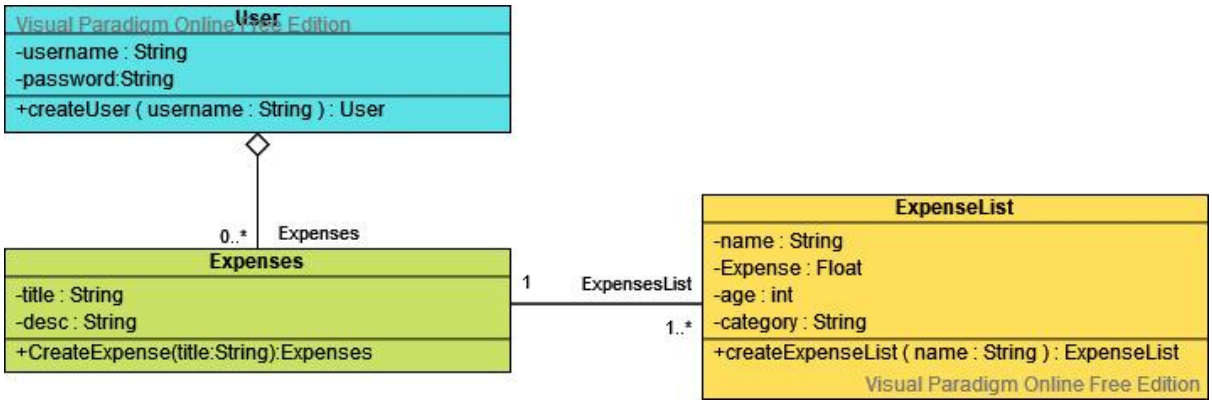


Figure 7: Class Diagram of Expense Tracker

10. TEST CASES:

UTC-1 (Register):

User Test Case	Register
Related Requirements	User ID, Email, Password
Test Case Goal	To add as a new user

Pre-Condition	The user has a unique ID, Email and Password
Post-Condition	• The user has successfully registered as a new user • And now he can use web application
Expected Result	Registered Successfully and now he can login
Status	Pass or Fail

UTC-2 (Login):

User Test Case	Login
Related Requirements	Login id, password
Test Case Goal	To login in the system and add bill or assess

Pre-Condition	The user has a valid id and password
Post-Condition	• The user successfully logged in the system and can access all the functionality of the system • The user denied of the access
Expected Result	Login was successful and now can view the dashboard screen
Status	Pass or Fail

UTC-3 (Add Expense):

User Test Case	Add Expense
Related Requirements	User account logged in
Test Case Goal	To add a bill

Pre-Condition	The user is logged in
Post-Condition	• New bill is added successfully • There might be a redundant entry
Expected Result	The new bill added successfully
Status	Pass or Fail

UTC-4 (Budget History):

User Test Case	Budget History
Related Requirements	User account logged in
Test Case Goal	To view the history sheet

Pre-Condition	The user is logged in
Post-Condition	• The user gets the most recent updated log of the budget
Expected Result	User has the full access to the recent log
Status	Pass or Fail

UTC-5 (View Expenses):

User Test Case	Login View Expenses
Related Requirements	User account logged in
Test Case Goal	The user is logged in

Pre-Condition	The user has a valid id and password
Post-Condition	• The user can view the detailed expenditure record
Expected Result	User can view the history sheet of the expenses
Status	Pass or Fail

UTC-6 (Delete Bill):

User Test Case	Delete Bill
Related Requirements	User account logged in
Test Case Goal	To Delete a bill

Pre-Condition	The user is logged in
Post-Condition	• Existing Bill Deleted Successfully
Expected Result	The log in Database is updated
Status	Pass or Fail

UTC-7 (Expense Summary):

User Test Case	Expense Summary
Related Requirements	User account logged in
Test Case Goal	To track the report

Pre-Condition	The user is logged in
Post-Condition	• Complete Expenditure is Displayed
Expected Result	View Expenditure History
Status	Pass or Fail

11. CODES AND OUTPUTS:

Login.html

```
<html>
    <head>
      <meta charset="UTF-8">
      <title> Login </title>
      <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
    </head>
    <body></br></br></br></br></br>
      <div align="center">
      <div align="center" class="border">
            <div class="header">
                  <h1 class="word">Login</h1>
            </div></br></br></br>
            <h2 class="word">
                  <form action="{{ url_for('login') }}" method="post">
                  <div class="msg">{{ msg }}</div>
                        <input id="username" name="username" type="text"
placeholder="Enter Your Username" class="textbox"/></br></br>
                        <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br></br>
                        <input type="submit" class="btn" value="Sign
In"></br></br>
                  </form>
            </h2>
            <p class="bottom">Don't have an account? <a class="bottom"
href="{{url_for('register')}}"> Sign Up here</a></p>
      </div>
      </div>
    </body>
</html>
```

Register.html

```
<html>
    <head>
      <meta charset="UTF-8">
      <title> Register </title>
      <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
    </head>
    <body></br></br></br></br></br>
      <div align="center">
      <div align="center" class="border">
            <div class="header">
                  <h1 class="word">Register</h1>
            </div></br></br></br>
            <h2 class="word">
                  <form action="{{ url_for('register') }}" method="post">
                  <div class="msg">{{ msg }}</div>
                        <input id="username" name="username" type="text"
placeholder="Enter Your Username" class="textbox"/></br></br>
                        <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br>
                        <input id="email" name="email" type="text"
placeholder="Enter Your Email ID" class="textbox"/></br></br>
                        <input type="submit" class="btn" value="Sign Up"></br>
                  </form>
            </h2>
            <p class="bottom">Already have an account? <a class="bottom"
href="{{url_for('login')}}"> Sign In here</a></p>
      </div>
      </div>
    </body>
</html>
```

Home.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <title>Expense Tracker</title>

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>

  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></s
cript>

  <script src="https://cdn.jsdelivr.net/gh/emn178/chartjs-plugin-
labels/src/chartjs-plugin-labels.js"></script>

  <script
src='https://cdnjs.cloudflare.com/ajax/libs/Chart.js/1.0.2/Chart.min.js'></sc
ript>

  <script src='static/Chart.min.js'></script>

  <style>

      .row.content {height: 1500px}


      .sidenav {

      background-color: #f1f1f1;

      height: 100%;

      }


      footer {

      background-color: #555;

      color: white;

      padding: 15px;

      }


      @media screen and (max-width: 767px) {
```

```
      .sidenav {
      height: auto;
      padding: 15px;
      }
      .row.content {height: auto;}
      }

      .hamburger {
      width: 35px;
      height: 3px;
      background-color: black;
      margin: 6px 0;
      margin-right: 0px;
      margin-left: auto;
      }
      #Chart_disp{
      display: flex;
      }
      #Titles{
      display: inline-flex;
      }

  </style>
</head>

<body>

<div class="container-fluid">
  <div class="row content">
      <div class="col-sm-3 sidenav">
      <h4>Expense Tracker</h4>
      <ul class="nav nav-pills nav-stacked">
      <li class="active"><a href="{{url_for('home')}}">Dashboard</a></li>
      <li><a href="{{url_for('about')}}">Create/Edit Expenses</a></li>
      <li><a href="{{url_for('list')}}">Result</a></li>
```

```
      <li><a href="{{url_for('logout')}}">Logout</a></button></li>
      </ul><br>
      </div>

      <div class="col-sm-9">
      <h1><small>Dashboard
      </small></h1>
      <h3>Welcome <b>{{session.username}}..!</b></h3>
      <hr>
      <p></p>
      <br><br>

      </div>

      <div id="Titles">
                                       
                 <h1>Expenses amount pie chart</h1>
                                   
                          
                                    
                <h1>Budget Overview</h1>
      </div>
      <br>
      <br>
      <div id="Chart_disp">
      <canvas id="chart" style="width:100%;max-width:600px"></canvas>
      <canvas id="chartOne" style="width:100%;max-width:600px"></canvas>
  </div>
  </div>
</div>




<script>

  var chartData = [
```

```
 {% for data in dataitems %}
     {
     value : {{data.val}},
     label : '{{data.lab}}',
     labelColor: 'white',
     labelFontSize: '16',
     labelAlign: 'center',
     color : '{{data.color}}'
     },
     {% endfor %}
 ];

var options = {
     title: {
     display : true,
     text : 'Pie Chart'
     },
     segmentShowStroke : true,
     animateScale : true,
     tooltips: {enabled: true},
     hover: {
     animationDuration: 0},
     showTooltips: true,
     onAnimateComplete: function() {
     this.showTooltips(this.chartData[0].points, true);
     },
     interaction: {
     mode: 'nearest',
     axis: 'x',
     intersect: false
     }
 }


 var chartDataOne = [
```

```
 {% for data in dataitems %}
    {
    value : {{data.bud}},
    label : '{{data.lab}}',
    labelColor: 'white',
    labelFontSize: '16',
    labelAlign: 'center',
    color : '{{data.color}}'
    },
    {% endfor %}
 ];

var options = {
    title: {
    display : true,
    text : 'Pie Chart'
    },
    segmentShowStroke : true,
    animateScale : true,
    tooltips: {enabled: true},
    hover: {
    animationDuration: 0},
    showTooltips: true,
    onAnimateComplete: function() {
    this.showTooltips(this.chartData[0].points, true);
    },
    interaction: {
    mode: 'nearest',
    axis: 'x',
    intersect: false
    }
 }

  Chart.defaults.global.animationSteps = 50;
  Chart.defaults.global.tooltipYPadding = 16;
```

```
   Chart.defaults.global.tooltipCornerRadius = 0;
   Chart.defaults.global.tooltipTitleFontStyle = "normal";
   Chart.defaults.global.tooltipFillColor = "rgba(0,0,0,0.8)";
   Chart.defaults.global.animationEasing = "easeOutBounce";
   Chart.defaults.global.responsive = false;
   Chart.defaults.global.scaleLineColor = "black";
   Chart.defaults.global.scaleFontSize = 16;

  var ctx = document.getElementById("chart").getContext("2d");

  var PieChartDemo = new Chart(ctx).Pie(chartData, options);



  var ctxOne = document.getElementById("chartOne").getContext("2d");

  var PieChartDemoOne = new Chart(ctxOne).Pie(chartDataOne, options);



a.click();

</script>

<footer class="container-fluid">
     <p>Footer Text</p>
  </footer>
</body>
</html>
```

AddExpenses.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Expense Tracker</title>
```

```
  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>

  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></s
cript>

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></s
cript>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>

  <style>

      .row.content {height: 1500px}


      .sidenav {

      background-color: #f1f1f1;

      height: 100%;

      }


      footer {

      background-color: #555;

      color: white;

      padding: 15px;

      }


      @media screen and (max-width: 767px) {

      .sidenav {

      height: auto;

      padding: 15px;

      }

      .row.content {height: auto;}
```

```
      }

      .hamburger {
      width: 35px;
      height: 3px;
      background-color: black;
      margin: 6px 0;
      margin-right: 0px;
      margin-left: auto;
      }
  </style>

</head>
<body>

<div class="container-fluid">
  <div class="row content">
      <div class="col-sm-3 sidenav">
      <h4>Expense Tracker</h4>
      <ul class="nav nav-pills nav-stacked">
      <li><a href="{{url_for('home')}}">Dashboard</a></li>
      <li class="active"><a href="{{url_for('about')}}">Create/Edit
Expenses</a></li>
      <li><a href="{{url_for('list')}}">Result</a></li>
      <li><a href="{{url_for('logout')}}">Logout</a></button></li>
      </ul>
      </div>

      <div class="col-sm-9">
      <h1><small>Create/Edit Expenses
      </small>
      </h1>
      <hr>
      <h2>Expenses List</h2>
      <br>
```

```
      <form action="{{ url_for('about') }}" method="post">

      <input name="name" type="text" placeholder="Enter Expense Name"
class="textbox" id="name"/></br></br>

                        <input name="expenses_amount" type="text"
placeholder="Enter Expense Amount" class="textbox"
id="expenses_amount"/></br></br>

           <input name="expenses_budget" type="text" placeholder="Enter
Budget" class="textbox" id="expenses_budget"/></br></br>

      <input type="submit" class="btn" value="Add"></br>

      </form>

      <br>

      <br>

      </div>

   </div>

      </div>

  </div>

</div>

</body>

</html>
```

Result.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <title>Expense Tracker</title>

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>

  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></s
cript>
```

```
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></s
cript>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>

  <style>

      .row.content {height: 1500px}


      .sidenav {

      background-color: #f1f1f1;

      height: 100%;

      }


      footer {

      background-color: #555;

      color: white;

      padding: 15px;

      }


      @media screen and (max-width: 767px) {

      .sidenav {

      height: auto;

      padding: 15px;

      }

      .row.content {height: auto;}

      }


      .hamburger {

      width: 35px;

      height: 3px;

      background-color: black;

      margin: 6px 0;

      margin-right: 0px;
```

```
      margin-left: auto;
      }

      td {
      width: 150px;
      text-align: center;
      border: 1px solid black;
      padding: 5px;
      }
  </style>
</head>
<body>

<div class="container-fluid">
  <div class="row content">
      <div class="col-sm-3 sidenav">
      <h4>Expense Tracker</h4>
      <ul class="nav nav-pills nav-stacked">
      <li><a href="{{url_for('home')}}">Dashboard</a></li>
      <li><a href="{{url_for('about')}}">Create/Edit Expenses</a></li>
      <li class="active"><a href="{{url_for('list')}}">Result</a></li>
      <li><a href="{{url_for('logout')}}">Logout</a></button></li>
      </ul>
      </div>

      <div class="col-sm-9">
      <h1><small>Result
      </small>
      </h1>
      <hr>
      <br>
      <h2><u>Expenses List for {{session.username}}:</u></h2>
      <br>
      <table>
      <thead>
```

```
            <tr>
            <center><th>Name</th></center>
            <center><th>Expenses</th></center>
            <center><th>Budget</th></center>
            </tr>
            </thead>
            <tbody>
            {% for r in row %}
                  <tr>
                        <td>{{r[0]}}</td>
                        <td>{{r[1]}}</td>
                        <td>{{r[2]}}</td>
                  </tr>
            {% endfor %}
            </tbody>
      </table>
      <br><br>
      <h3>Total Budget:  {{value}}</h3><br>
      <h3>Total Expenses Amount:  {{value1}}</h3><br>
      <h3>Total Saving Amount: {{saving}}</h3>
      </div>
   </div>
      </div>
  </div>
</div>
</body>
</html>
```

Main.py

```
from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import mysql.connector
import matplotlib.pyplot as plt
```

```
import mpld3
import numpy as np
#import os
import re

import random
def getColor():
    color = "%06x" % random.randint(0, 0xFFFFFF)
    return "#"+color

from sqlalchemy import null
app = Flask(__name__)
app.secret_key = 'your secret key'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'harshith.4123'
app.config['MYSQL_DB'] = 'geeklogin'

mysql = MySQL(app)

@app.route('/home',methods=['GET','POST'])
def home():
    current_user = str(session['id'])
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT expenses_name,expenses_amount,expenses_budget FROM
expenses_list WHERE user_id=%s',[current_user])
    rows=cursor.fetchall()
    dataitems = []
    fld={}
    # colors = [
      # "#F7464A", "#46BFBD", "#FDB45C", "#FEDCBA",
      # "#ABCDEF", "#DDDDDD", "#ABCABC", "#4169E1",
      # "#C71585", "#FF4500", "#FEDCBA", "#46BFBD"]
    for i in rows:
       fld['lab'] = i[0]
```

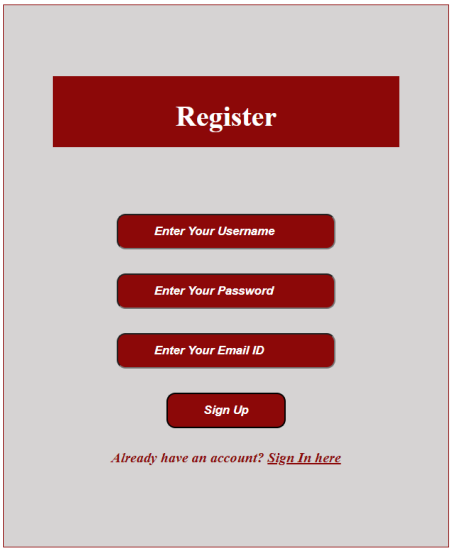
```
        fld['val'] = i[1]
        fld['bud'] = i[2]
        fld['color'] = getColor()
        dataitems.append(fld.copy())

    return render_template("Home.html",dataitems=dataitems)


@app.route('/result')
def list():
    cur = mysql.connection.cursor()
    current_user = str(session['id'])
    cur.execute('SELECT expenses_name,expenses_amount,expenses_budget FROM
expenses_list WHERE user_id=%s',[current_user])
    row = (cur.fetchall())
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT SUM(expenses_budget) FROM expenses_list WHERE
user_id=%s',[current_user])
    total_budget = (cursor.fetchall()[0][0])
    value = str(total_budget)
    cursor1 = mysql.connection.cursor()
    cursor1.execute('SELECT SUM(expenses_amount) FROM expenses_list WHERE
user_id=%s',[current_user])
    total_expenses = (cursor1.fetchall()[0][0])
    value1 = str(total_expenses)
    data1 = int(value)
    data2 = int(value1)
    data = data1-data2
    saving = str(data)
    return
render_template("result.html",value=value,row=row,value1=value1,saving=saving
)


@app.route('/about',methods =['GET','POST'])
def about():
    if request.method == 'POST' and 'name' in request.form and
'expenses_amount' in request.form and 'expenses_budget' in request.form:
        current_user = session['id']
```

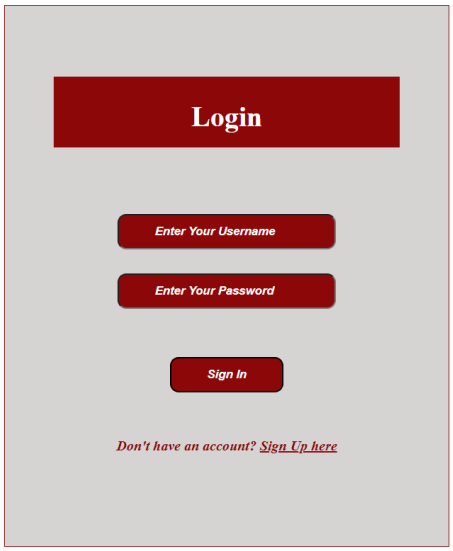
```
        expenses_name = request.form['name']
        expenses_amount = request.form['expenses_amount']
        expenses_budget = request.form['expenses_budget']
        cur = mysql.connection.cursor()
        cur.execute('INSERT INTO
expenses_list(expenses_name,user_id,expenses_amount,expenses_budget) VALUES
(%s,%s,%s,%s)',(expenses_name,current_user,expenses_amount,expenses_budget))


        mysql.connection.commit()
    return render_template('AddExpenses.HTML')


@app.route('/')
@app.route('/login', methods =['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password'
in request.form:
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM accounts WHERE username = % s AND
password = % s', (username, password, ))
        account = cursor.fetchone()
        if account:
              for row in cursor:
                    session['loggedin'] = True
                    session['id'] = row[0]
                    session['username'] = row[1]
              msg = 'Logged in successfully !'
              return render_template('home.html', msg = msg)
        else:
              msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)


@app.route('/logout')
def logout():
```

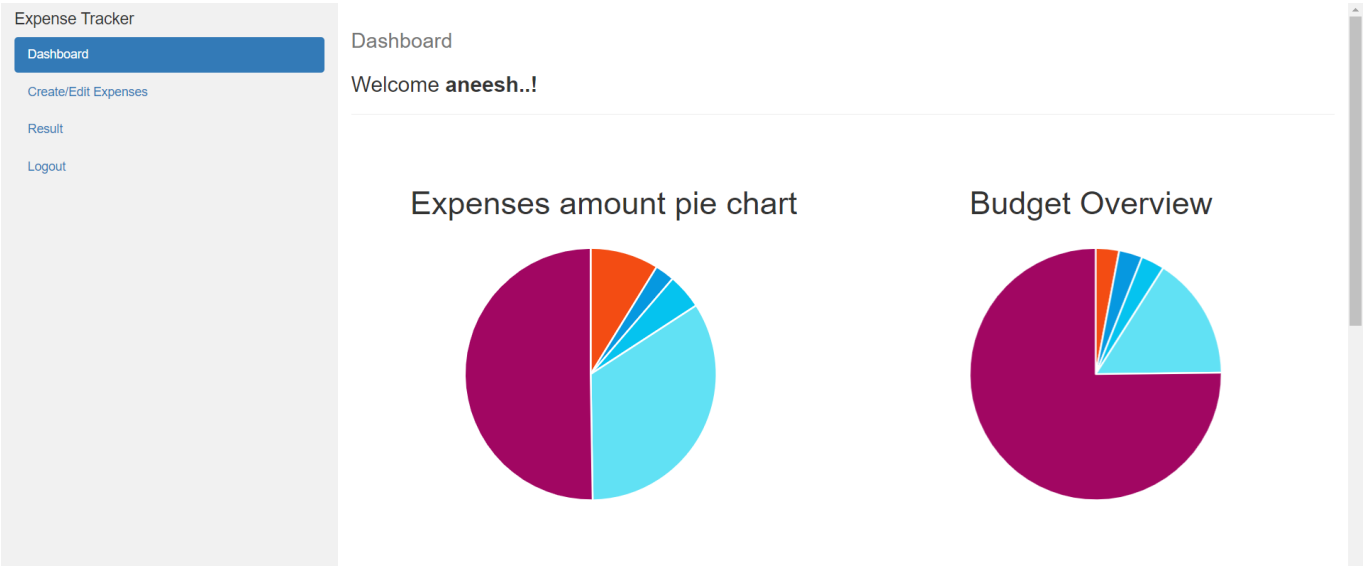
```
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))


@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password'
in request.form and 'email' in request.form :
      username = request.form['username']
      password = request.form['password']
      email = request.form['email']
      cursor = mysql.connection.cursor()
      cursor.execute('SELECT * FROM accounts WHERE username = % s',
(username, ))
      account = cursor.fetchone()
      if account:
            msg = 'Account already exists !'
      elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
      elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'Username must contain only characters and numbers !'
      elif not username or not password or not email:
            msg = 'Please fill out the form !'
      else:
            cursor.execute('INSERT INTO accounts VALUES (NULL, % s, % s, %
s)', (username, password, email, ))
            mysql.connection.commit()
            msg = 'You have successfully registered !'
    elif request.method == 'POST':
      msg = 'Please fill out the form !'
    if msg == 'You have successfully registered !':
      return render_template('Home.HTML')
    else:
      return render_template('register.html')
```

```
if __name__=='__main__':
      app.run(debug=True)
```

Output Snapshots:







Expense Tracker

Dashboard

Create/Edit Expenses

Result

Logout

Create/Edit Expenses

Expenses List

Enter Expense Name

Enter Expense Amount

Enter Budget

Add

Expense Tracker

Dashboard

Create/Edit Expenses

Result

Logout

Result

Expenses List for aneesh:

Name	Expenses	Budget
drink	35	20
pens	10	20
pencils	18	20
Biscuits	135	105
Mouse Pad	200	500

Total Budget: 665

Total Expenses Amount: 398

Total Saving Amount: 267

12. SOFTWARE TESTING REPORT:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as-yet-discovered error.

UTC-1 (Register User):

Functionality Unit Testing			**Module Name: Register User**
Project Title	Daily Expense Tracker System for Employees (Web Application)		
Test Case Name	**Register Users**	**Testing Date**	06/05/2022
Test Case ID	**UTC1**	**Test Type**	Black Box
Conducted By	Team 11	**Duration**	4 Hours
Precondition	User must have internet access to register on the web Application.		
Description	To perform user registration by providing the required details.		
Test Case Priority/Severity	**High** as the user will be allowed to perform all the task under DET. So, severity is high.		

MODULE EXECUTION

Module	**Steps**	**Result Expected**	**Actual Result**	**Result (Pass/Fail)**
1.1	Click on registration Button	System should open user registration page.	Requested Screen Opened	**Pass**

1.2	Enter the following information: Username: “abc” Password: “123” Email Id:” abc@gmail.com” And click Sign Up	User Should be registered and must show “User Registered”	New User added to the system successfully.	**Pass**
1.3	Leave any of the field blank Enter the following information: User Name: Password: “123” Email Id: abc@gmail.com And Click Sign Up	System should display “Please Enter your username”.	System displayed html tags instead of showing expected result<html> … </html>	**Fail**
1.4	Enter same registration details Enter the following information: User Name: “abc” Password: “123” Email: “abc@gmail.com” And Click Sign Up	System should display “User with similar email id is already registered”.	System displayed “New user is added” and redirect to login page	**Fail**

Result: Two error found in the module.

Measures Taken:

(Module 1.3 Error Rectification) Developer reviewed the code of Main.py file in DET web app file and provided front end validation file for that. If any field left blank then display appropriate error.

Measures Taken:

(Module 1.4 Error Rectification) Developer reviewed the code of Main.py file and check the error by identifying the add user code in database. If user with same email id is in database, then display message “User already registered”.

Review Test Case for failed Obligations

Module	Steps	Result Expected	Actual Result	Result (Pass/Fail)
1.3	Either of any required filed like password, Email etc. left blank	Error message should appear	**Identical to the expected Result**	**Pass**

1.4	Username or email already existed entered	Error message "User Already Exist"	**Identical to the expected Result**	**Pass**
Conclusion: Module worked perfectly without any error				

UTC-2 (Login User):

Functionality Unit Testing			**Module Name: Login User**
Project Title	Daily Expense Tracker System for Employees (Web Application)		
Test Case Name	**Login User**	**Testing Date**	06/05/2022
Test Case ID	**UTC2**	**Test Type**	Black Box
Conducted By	Team 11	**Duration**	4 Hours
Precondition	User must have internet access to login to the web Application.		
Description	To perform user login by providing the required details.		
Test Case Priority/Severity	**High** as the user will be allowed to perform all the task under DET. So, severity is high.		

MODULE EXECUTION

Module	**Steps**	**Result Expected**	**Actual Result**	**Result (Pass/Fail)**
2.1	Click on Login Button	System should open user Login page.	Requested Screen Opened	**Pass**

2.2	Enter the following information: Username: “abc” Password: “123”	User Should be registered And should be able to login with correct credentials.	User logged into the system successfully.	**Pass**
2.3	Enter the following information: Username:”abc” Password: “1234”	System should display “Username or password is invalid”.	User logged into the system successfully	**Fail**

Result: One error found in the module.

Measures Taken:

(Module 1.3 Error Rectification) Developer reviewed the code of Main.py file and check the error by identifying the username and password in the database. If user with password is in database, then display message “logged in” and if password or username is wrong it should display error.

Review Test Case for failed Obligations

Module	Steps	Result Expected	Actual Result	Result (Pass/Fail)
2.3	Either of any of the username or password is entered wrongly.	System should display “Username or password is invalid	**Identical to the expected Result**	**Pass**

Conclusion: Module worked perfectly without any error

UTC-3 (Add Expense):

Functionality Unit Testing	**Module Name: Add Expenses**
Project Title	Daily Expense Tracker System for Employees (Web Application)

Test Case Name	**Add Expenses**	**Testing Date**	08/04/2022
Test Case ID	**UTC3**	**Test Type**	Black Box
Conducted By	Team 11	**Duration**	6 Hours
Precondition	User must be logged in the application in order to add expense.		
Description	To add daily expenses in Web Application.		
Test Case Priority/Severity	**High:** as this feature is the core feature of the application, without this feature this system will not be successful.		

MODULE EXECUTION

Module	Steps	Result Expected	Actual Result	Result (Pass/Fail)
3.1	Click on Create/Add Expenses in the menu	System should open Create/Edit Expenses page.	Requested Screen Opened	**Pass**
3.2	Enter the following information: Expense Name: “Glasses” Expense Amount: “700” Budget: “1000” then Press Add	Expense should be saved successfully.	Insert Success	**Pass**
3.3	Enter the following information again: Expense Name: “Glasses” Expense Amount: “700” Budget: “1000” And Press Add	Expense should be saved successfully	Insert Success	**Pass**
3.4	Left all fields blank and then press Add	System should display “Please enter the details”.	Insert Success	**Fail**

Result: one error found in the module.

Measures Taken:

(Module 3.4 Error Rectification) Developer reviewed the code of Main.py in DET web app file and found there was no front end validation provided. So the developer provided front end validation also.

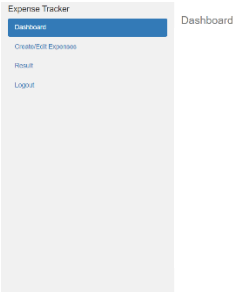
Review Test Case for failed Obligations				
Module	**Steps**	**Result Expected**	**Actual Result**	**Result (Pass/Fail)**
3.4	Either of any required filed like Expense Name, Expense Amount or Budget left blank	Error message should appear	**Identical to the expected Result**	**Pass**
Conclusion: Module worked perfectly without any error				

13. CHANGES MADE IN VERSION2:

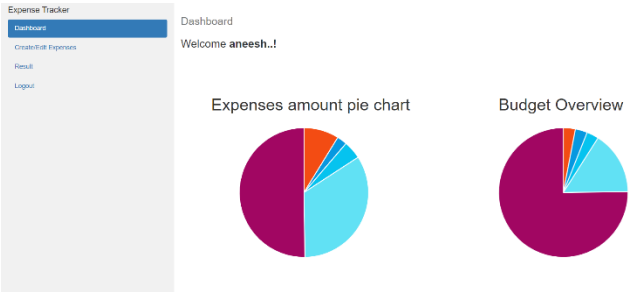
Adding *Pie Chart visualization* and *Welcome message* to the Dashboard module:

The dashboard module has not been implemented in the first version. The dashboard page is completely blank. The Pie chart visualization of expenses amount, Budget overview and "Welcome User" message was added later in version 2. Pie Chart visualizations are obtained by taking log data of Expenses and Budget from Result module and Welcome message of user is obtained by taking user data from the database.

version-1:



version-2:



14. CONCLUSION:

We are able to bring our vision to life, with a centralised log pertaining to all daily expenses. The user requires fewer or no manual calculations, with an effective and intuitive user

15. FUTURE SCOPE:

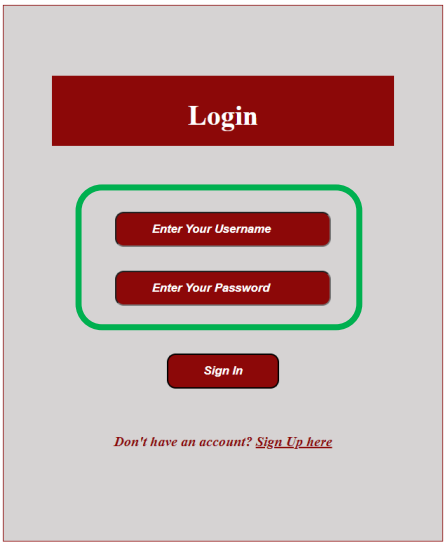
The following are some future development plans

- Group: Apart from keeping a personal log, we are planning to extend this system to incorporate a shared expense group.
- Payment gateway: We are planning to include a service so as to make the direct cash payment within the application itself.
- Representing the interface multilingual
- In the future emails and messages will be given to the user according to the expenses spent by him.

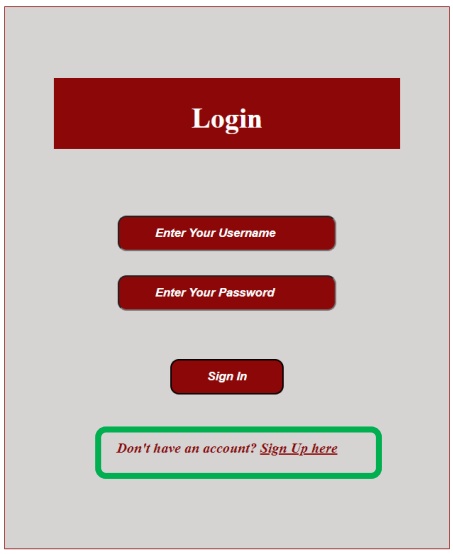
16. USER MANUAL:

Module: Create/Edit Expenses

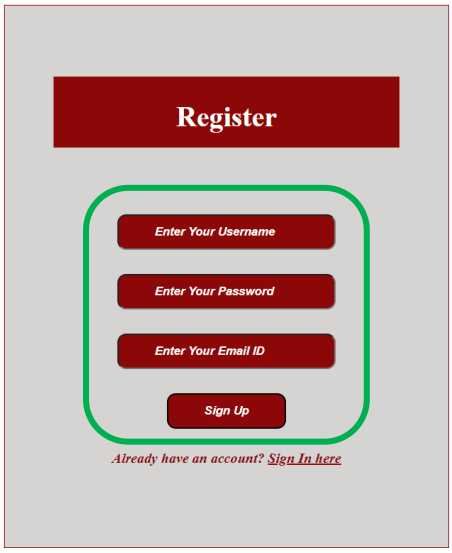
1) Open the webpage, login page will appear at first. If you are existing user, Sign in with your username and password.



2) If you are new user, register for a new account by clicking “Sign Up here”.

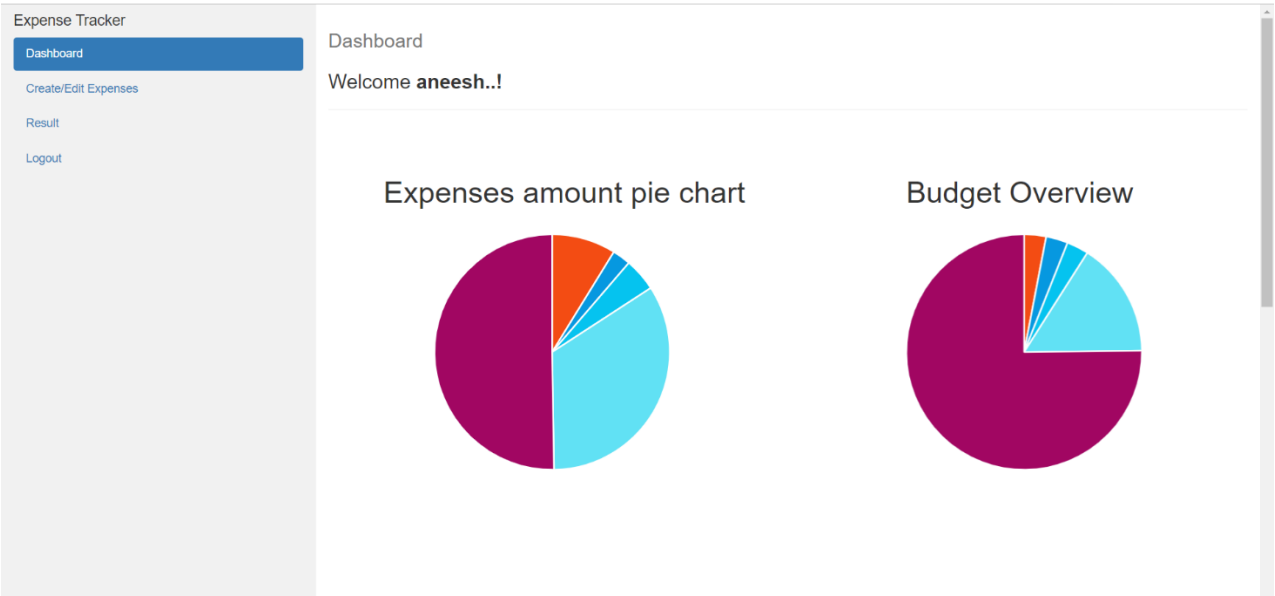


3) A new register page will appear for the registration. Enter your Username, Password and Email ID and then click Sign Up for account creation.

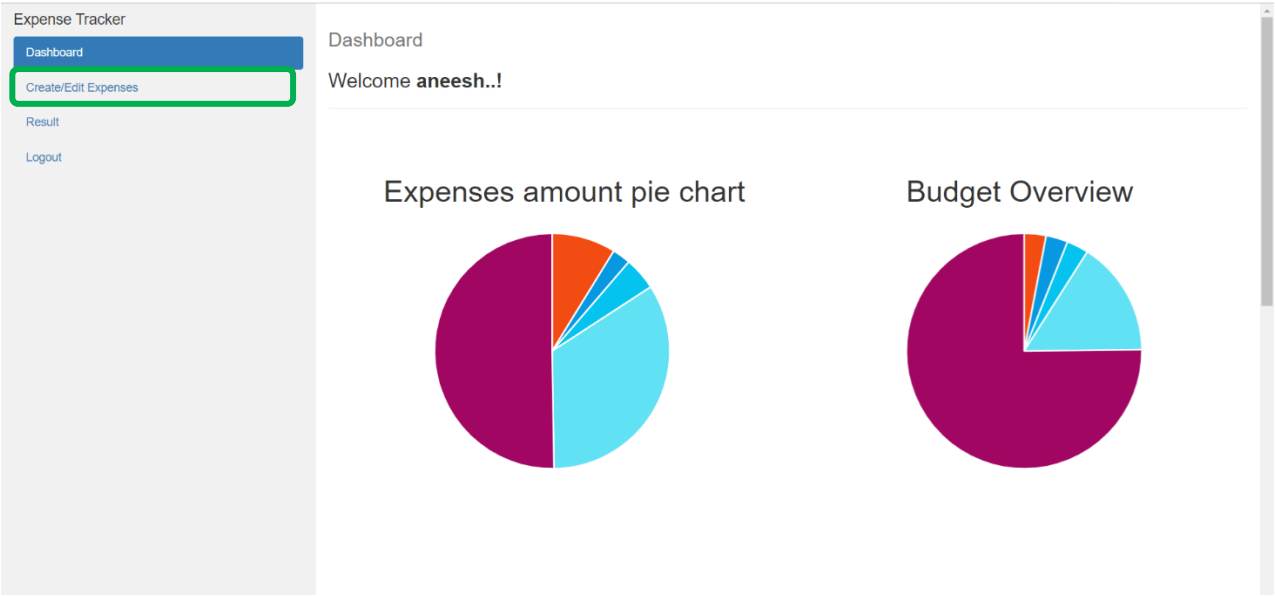


4) After creating the account, enter your user credentials in Sign In page as shown in first step.

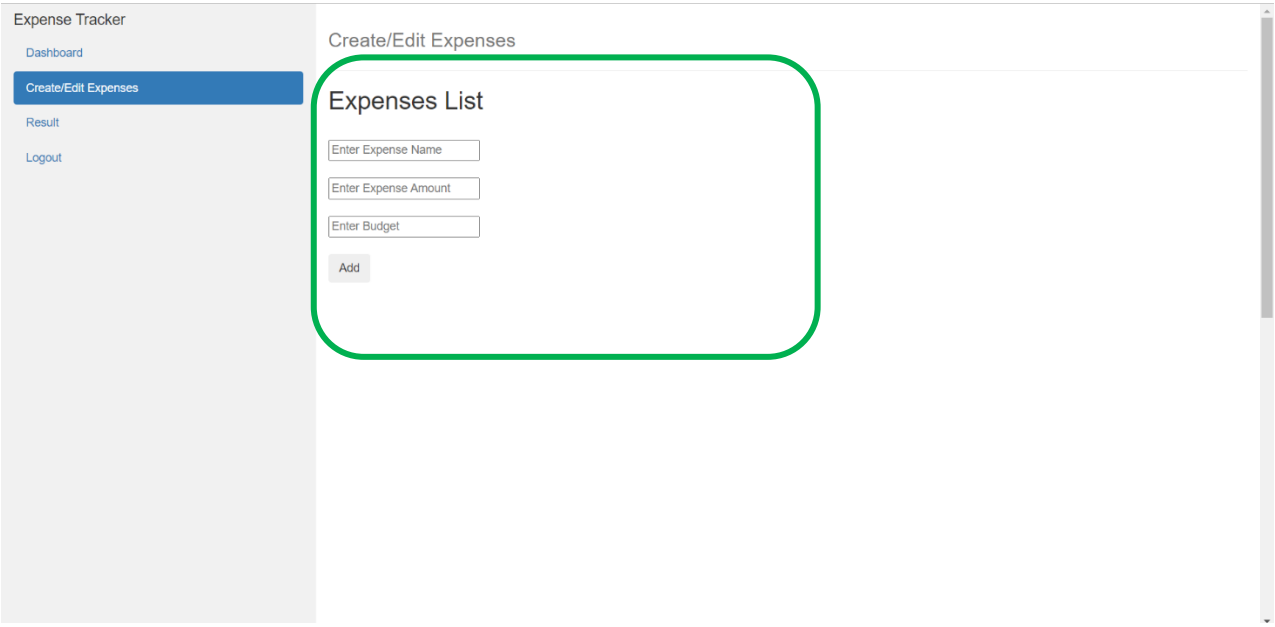
5) Upon logging in, a dashboard page (pie chart visualization of your expenses and budget overview) will appear. If you are a new user, you notice an empty page as you not added the expenses.



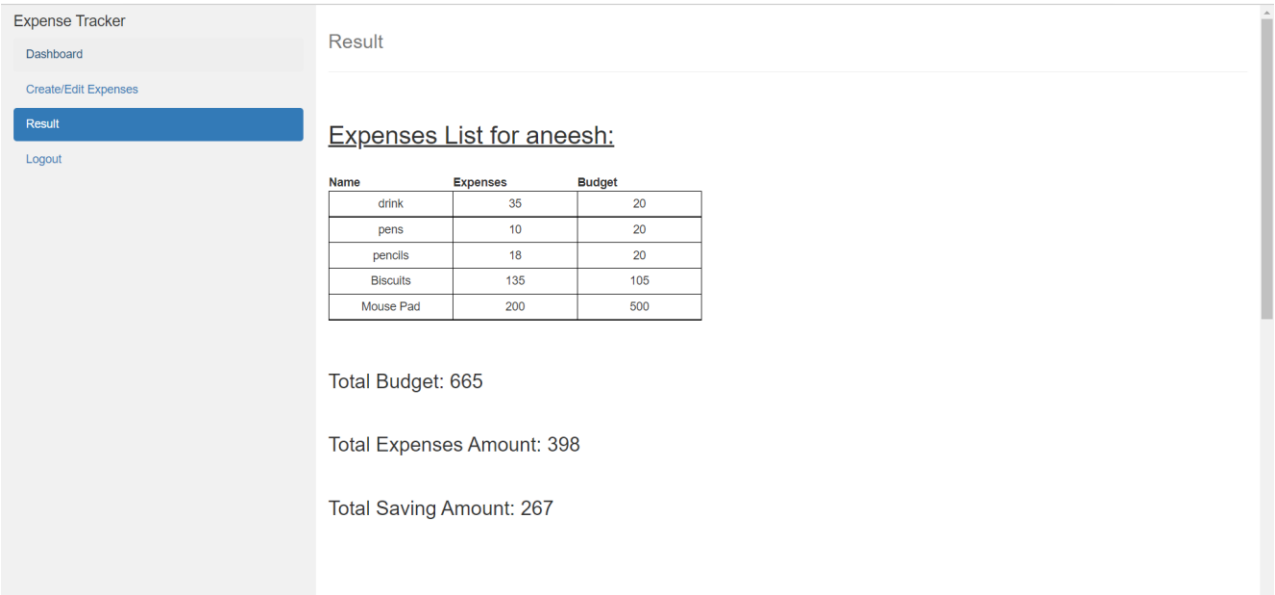
6) To add your daily expenses, click “Create/Edit Expenses” option in the sidebar menu.



7) Then add your expenses by specifying the expense name, expense amount and Budget



8) Then click on result to view your daily expenses.



17. REFERENCES:

- https://nevonprojects.com/daily-expense-tracker-system/
- https://www.opensourceforu.com/2016/06/future-expense-management-software/
- https://www.outlookindia.com/outlookmoney/technology/5-reasons-to-us e-an-expense-tracker-app-3559