PERSONAL EXPENSE TRACKER APPLICATION

DOMAIN: CLOUD APPLICATION DEVELOPMENT

TEAM MEMBERS:

1. VARSHA S
2. MANJAARIKA K R
3. RAJASHREE S
4. SHUTHI M

INDEX

10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview

In simple words, personal finance entails all the financial decisions and activities that a Fiakes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.

Personal finance applications will ask users to add their expenses and based on their expense . Wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditure in graphical forms. They have an option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

Nowadays people are concerned about the regularity of their daily expenses. This is done mainly to keep a track of the users' daily expenses to have a control of users' monthly expenses. We have developed an android application named as "Expense Tracker Application" and this application is used to manage the user's daily expenses in a more coherent and manageable way [10]. This application will help us to reduce the manual calculations for their daily expenses and also keep track of the expenses. With the help of this application, the user can calculate his total expenses per day and these results will be stored for unique users. As with the traditional methods of budgeting, we need to maintain Excel sheets, Word Documents, notes, and files for the user's daily and monthly expenses. There is no such full-fledged solution to keep a track of our daily expenses easily. Keeping a log in diary is a very monotonous process and also may sometimes lead into problems due to the manual calculations. Looking at all the above given conditions, we are trying to satisfy the user requirements by building a mobile application which will help them reduce their burdens. "Expense Tracker Application" is an application where one can enter their daily expenses and end of the day, they know their expenses in charts

1.2 Purpose

The motivation to work in this project is actually our real-life experience. As a user We face many difficulties in our daily file. In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Daily Expense Tracker user can be tracking expenses day to day and making life tension free.

The idea of developing this project on a mobile platform for user convenience. Because whenever they make expenses immediately, they add in a mobile application. Some of the concerns of maintaining a personal expense is a BIG problem, in daily expenses many times we don’t know where the money goes. Some of the conventional methods used to tackle this problem in normal circumstances are like making use of sticky notes by common users. Proficient people deal with this kind of problems by using spreadsheets to record expenses and using a ledger to maintain the large amounts of data, especially by expert people. As this shows that it is various methods used by different people. This makes using this data contrary. There is still complication in areas like there is no assurance for data compatibility, there are chances of crucial inputs can be missed and the manual errors may sneak in. The Data recorders are not always handled, and it could be a hectic process to have an overall view of those expenses. We believe in a handy design and a handy mobile application which handles these troubles. Such that app is capable of recording the expenditure and giving broad view with easy to use the user interface and this application is intelligent enough to shows the history of expenses noted in the app

2 LITERATURE SURVEY

S. NO	TITLE	AUTHOR	YEAR	ABSTRACT	TECHNOLOGY
1.	EXPENSE MANAGER APPLICATION	Velmurugan A, Albert Mayan J, Niranjana P and Richard Francis	2020	This application is used to keep record of user personal expenses, his/her contribution in group expenditures, top investment options, view of the current stock market and grap the best ongoing offer in the market. It eliminate the sticky notes, spredsheets confussion and data handling in consistency problems.	Android studio, Kotlin, java, SQLite, Android OS, Figma designing tool.

2.	EXPENSE TRACKER	Nidhi Jitendra Jadhav, Rutuja Vijay Chakor , Trupti Mahesh Gunjal , Damayanti. D. Pawar	2022	This system takes the user's income and divides it into daily expense allowances. If you exceed that day's expense, it will be deducted from your income and replaced with a new daily expense allowance. If the amount is smaller, it will be saved. At the end of the month, the daily spending tracking system will provide a report that shows the income expenditure curve.	Mobile application, Using Database layer which holds all of the data and financial information, supported by User Interface.

3.	Daily Expense Tracker	Tamia Ruvimbo Masendu , Aanajey Mani Tripath	2022	Daily Expense Tracker is a gadget that being developed to help customers in budget planning. It offers end customers to file their earnings and costs within the finances that have been planned beforehand.	This application is a GUI (Graphics User Interface) based application.Tec hn ology used Java (Apache NetBeans IDE 13) and my MySQL Workbench.

4.	Expense Tracker Application	Velmurugan. R , Mrs. P. Usha	2021	This application allows the user to maintain a computerized diary. which will keep a track of Expenses of a user on a day to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. It will generate report at the	Java, Xml, MySQL

				end of month to show Expense via a graphical representation.	

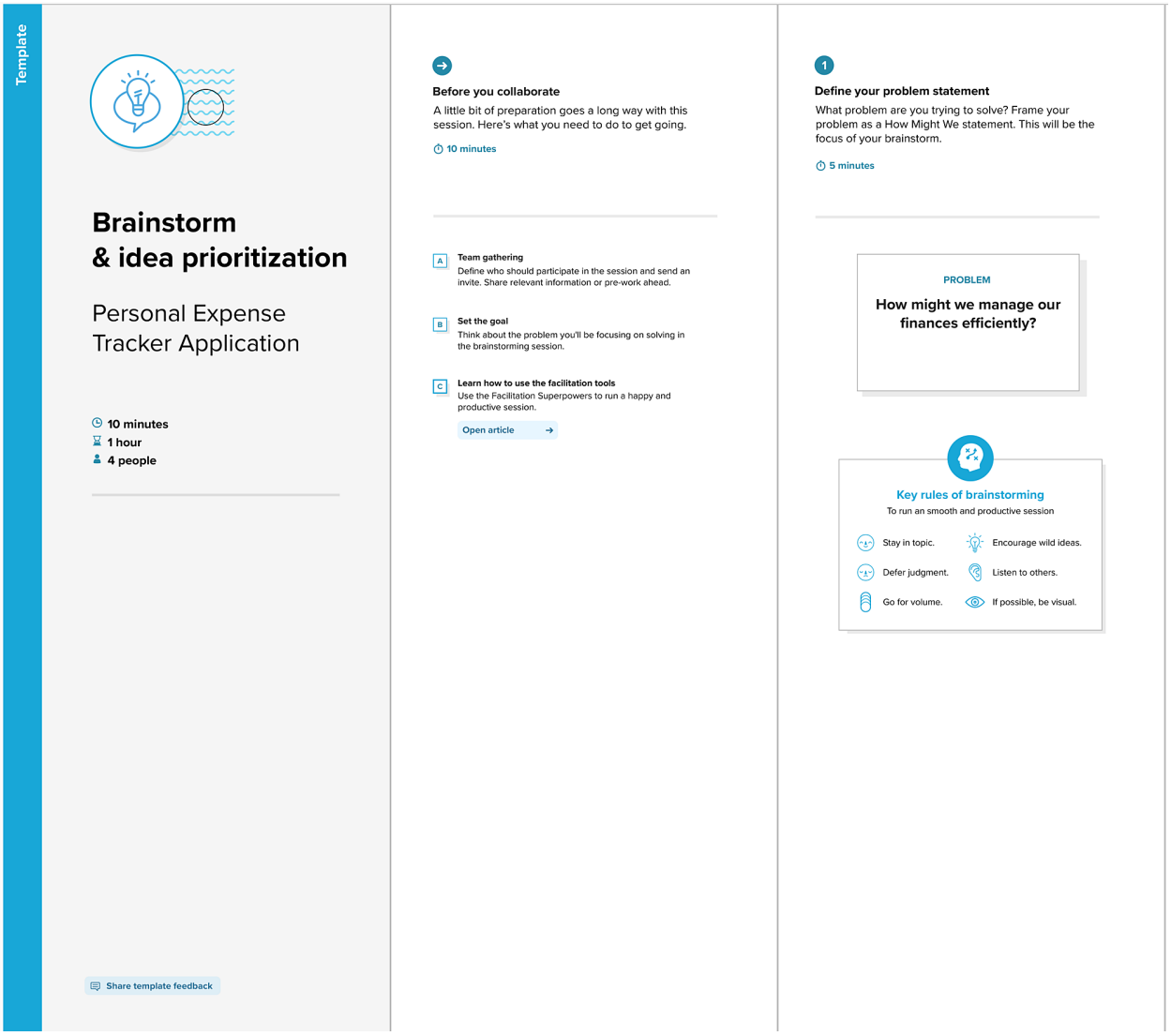
5.	EXPENDITURE MANAGEMENT SYSTEM	Dr. V. Geetha, G. Nikhitha, H. Sri Lasya, Dr. C.K.Gomathy	2022	This application uses Weekly Budget Planner to track their expenses. Automated message Alert is generated when they cross their budget. UPI linkup to track their online transactions. Weekly and Monthly Analysis are generated in the form of pie chart. App Authentication for security of the user. Income, Expenses, and Wish List are the three data entry choices available to the user	JavaScript, JSX, React and Mangodb.

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Brainstorm & Idea Prioritization:

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Template

Brainstorm & idea prioritization

Personal Expense Tracker Application

10 minutes
1 hour
4 people

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article

1 **Define your problem statement**

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

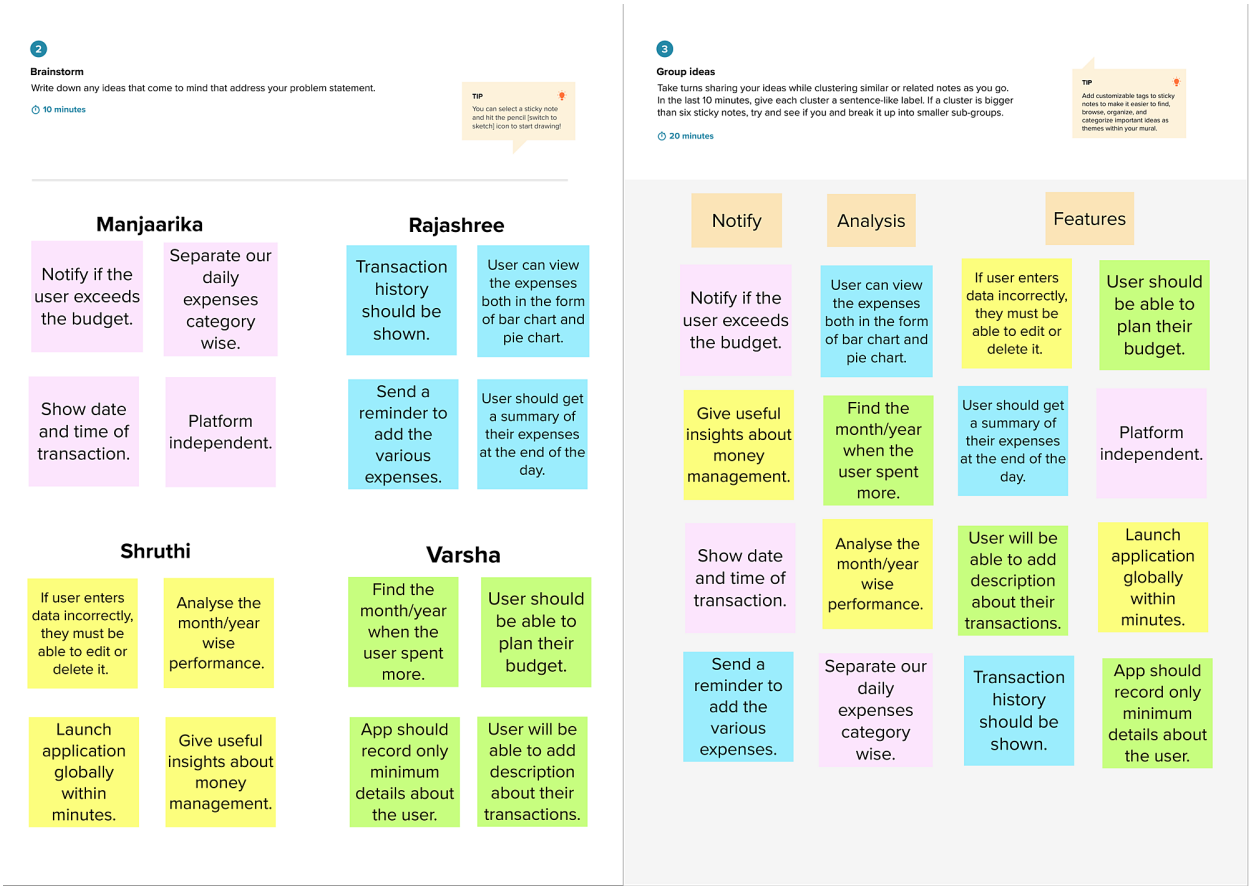
How might we manage our finances efficiently?

Key rules of brainstorming

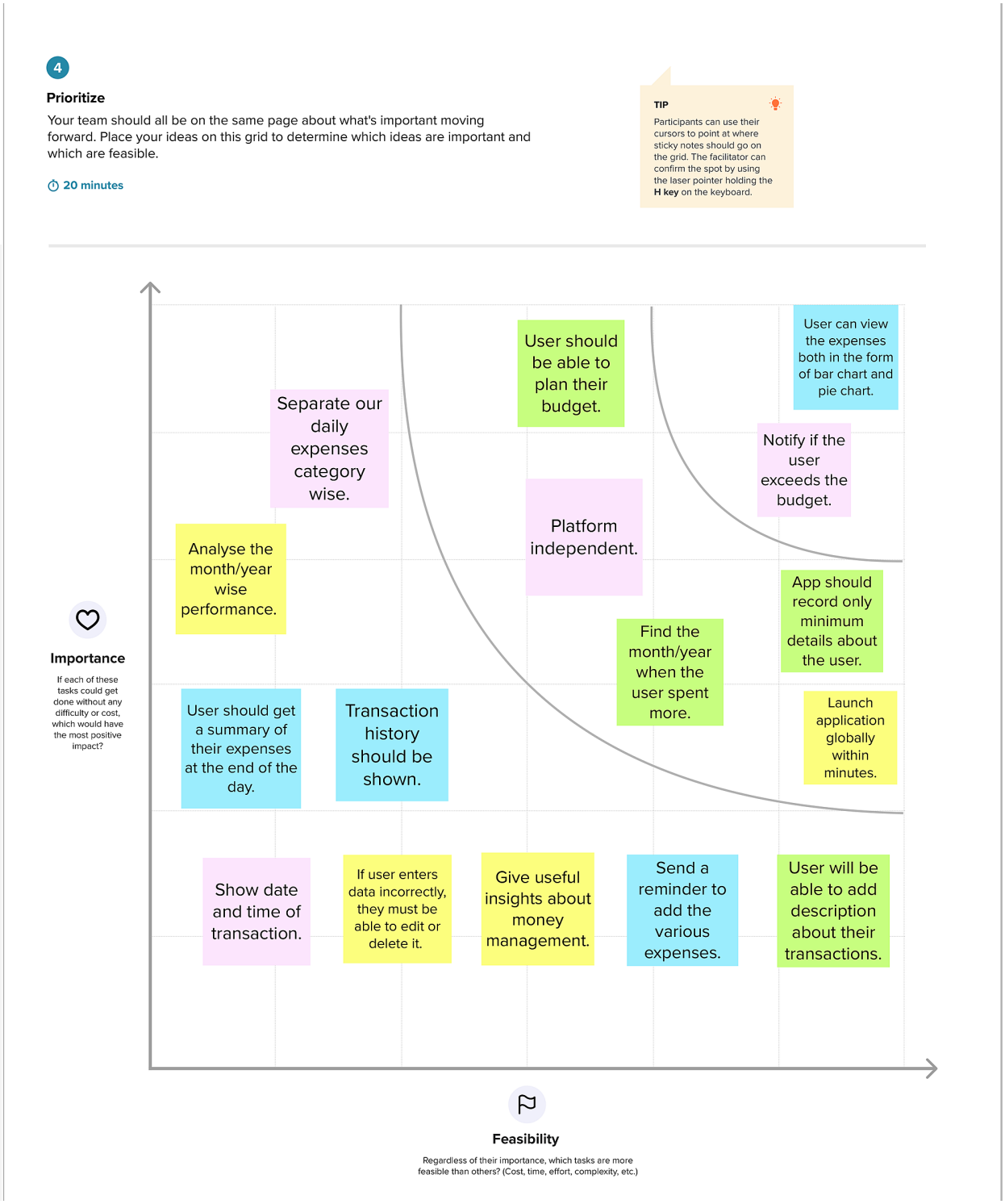
To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To solve complexities involved in keeping track of one's personal expenses. To digitalize the manual process of handling and recording the expenses of a user.
2.	Idea / Solution description	An application is designed to monitor the income and manage the expenses. Using cloud application the system is logged in.
3.	Novelty / Uniqueness	Various new features are present in our app which provides a wholesome experience to the user such as email notifications is sent when the user crosses their expense threshold.
4.	Social Impact / Customer Satisfaction	Using this application eases the tiring and complex process of keeping track of one's expenses and helps them refer their transactions anywhere anytime from their phones.

5.	Business Model (Revenue Model)	The business can profit by including advertisements in the application. The customers can also subscribe to a premium version of the app with a small price to avoid advertisements and enjoy exclusive features .This application is intended for users across all age groups.
6.	Scalability of the Solution	Our application is easily scalable since cloud technology is used. New features can also be added anytime as per requirement.

3.4 Problem- Solution fit

Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) — CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Customers are those who need to keep an accurate record of their money.

Customers who can ensure that money is used wisely.

Customers who wants to categorize the expenses such as food,entertainment, education etc.

6. CUSTOMER CONSTRAINTS — CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Internet hosts lot of ads limiting the application usability.

Adding the expenses made each and every time manually reduces the users.

5. AVAILABLE SOLUTIONS — AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Using Excel spreadsheets to note the expenses and making the calculation where the calculation requires more time and no graphical representation is been provided.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS — J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

A Expense tracking helps in finance management by knowing the income based on expenditure made. This helps to save money.

The objective of this application is to achieve optimal profit, both in short and long run.

People can also view the expenses as a graphical representation and compare the expenses made.

9. PROBLEM ROOT CAUSE — RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Inconvenience to live a life with standardized financial expenses, which may lead into dept traps.

Spending lavishly without keeping records lead to spend beyond income.

It includes stressed and complications to live a economically balanced life.

7. BEHAVIOUR — BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

User can reduce few expenses made unnecessarily.

Sends the Email alert if the expense exceeds the limit.

Keep track of the expenses and view them in a graphical format for detailed analysis.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS — TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Application allows the customers to reduce the lavish expenses made.

4. EMOTIONS: BEFORE / AFTER — EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

They have a better understanding of the income and outgoings.

10. YOUR SOLUTION — SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

Email alert which notifies the user when maximum amount is spent using sendgrid framework.

Application allows to view expenses in graphical from.

8. CHANNELS of BEHAVIOUR — CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

Expense tracker in online come with a lot of ads which have possibilities of stealing data.

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

User should be aware of the tax rules by reading terms and conditions.

Extract online & offline CH of BE

AMALTAMA

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	Registration through Gmail
FR-2	**Historical Data**	Stores previous data and show user a graphical representation based on categories.
FR-3	**Reporting Requirements**	Once the monthly salary reaches the minimum amount it notifies the user in mail.
FR-4	**Transaction Correction and Cancellation**	Users able to edit the transactions made and also can delete or add any transactions at anytime.
FR-5	**Authentication**	Users can set an unique password and login to their account by giving it.
FR-6	**Audit Tracking**	Each and every time calculate the expenses made by user whenever it is uploaded and the calculation is made instantly.

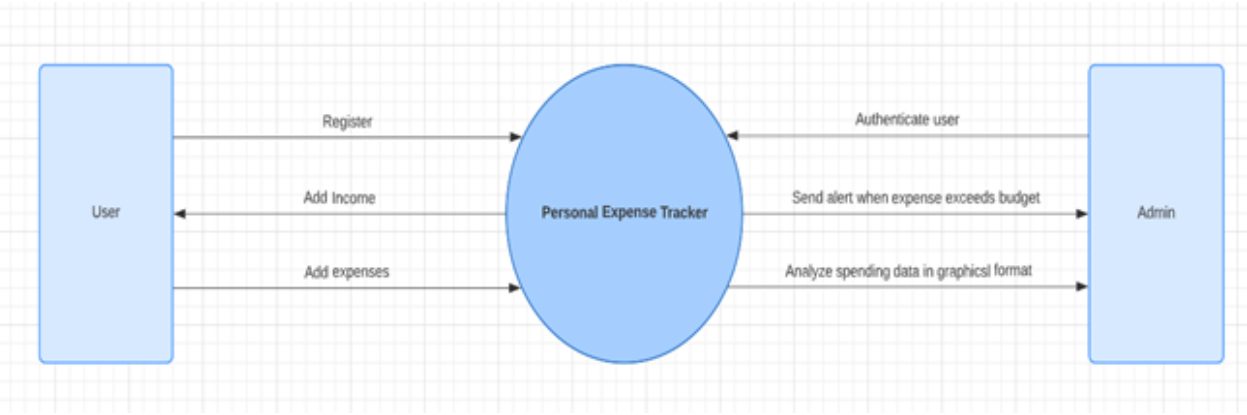
4.2 Non-functional Requirements:

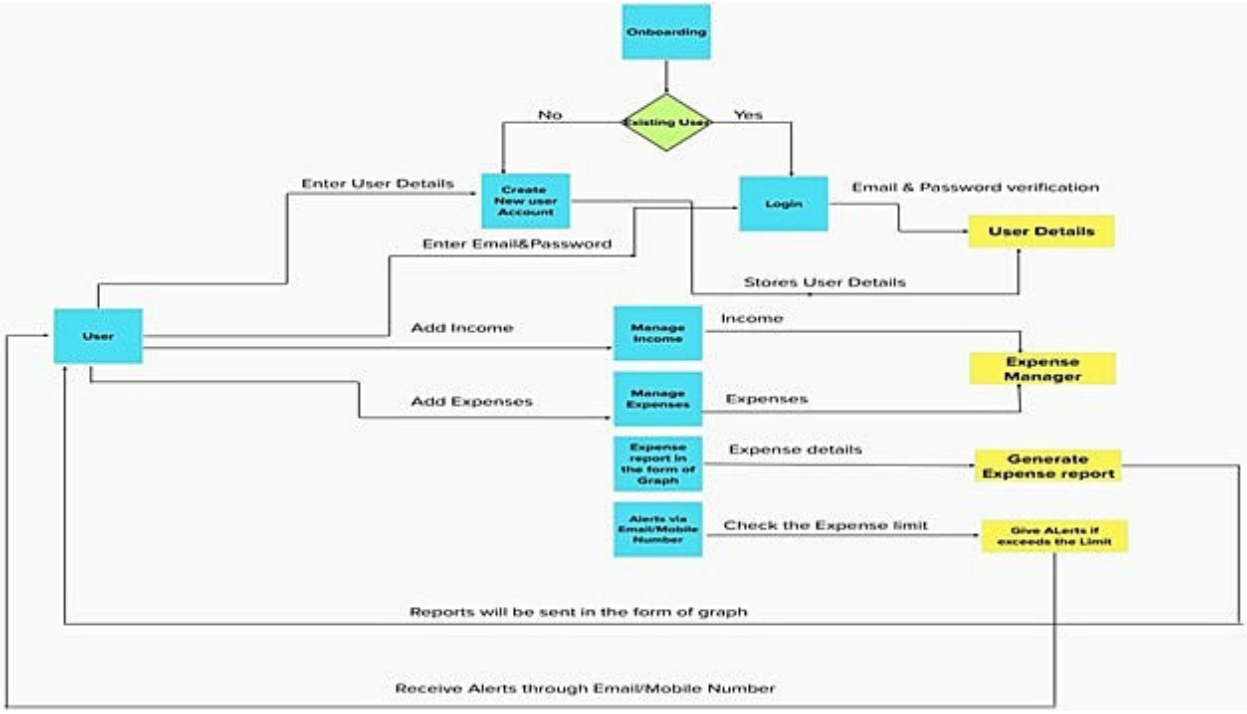
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It is ease to handle the app, navigate and efficient from the user point of view.
NFR-2	**Security**	The application get only name and Mail Id/Phone Number .It doesn't get additional personal information from the user.
NFR-3	**Reliability**	The probability of the system getting fail is very less as the code used in the program is minimum and does not utilize more time and cause run time failure during execution.
NFR-4	**Performance**	The launch time and load time is less and the app size is small .
NFR-5	**Availability**	Available free in play store and premium account requires only minimum amount.
NFR-6	**Scalability**	The app is capable to handle more users and evolving concurrently to the user needs.

5.PROJECT DESIGN

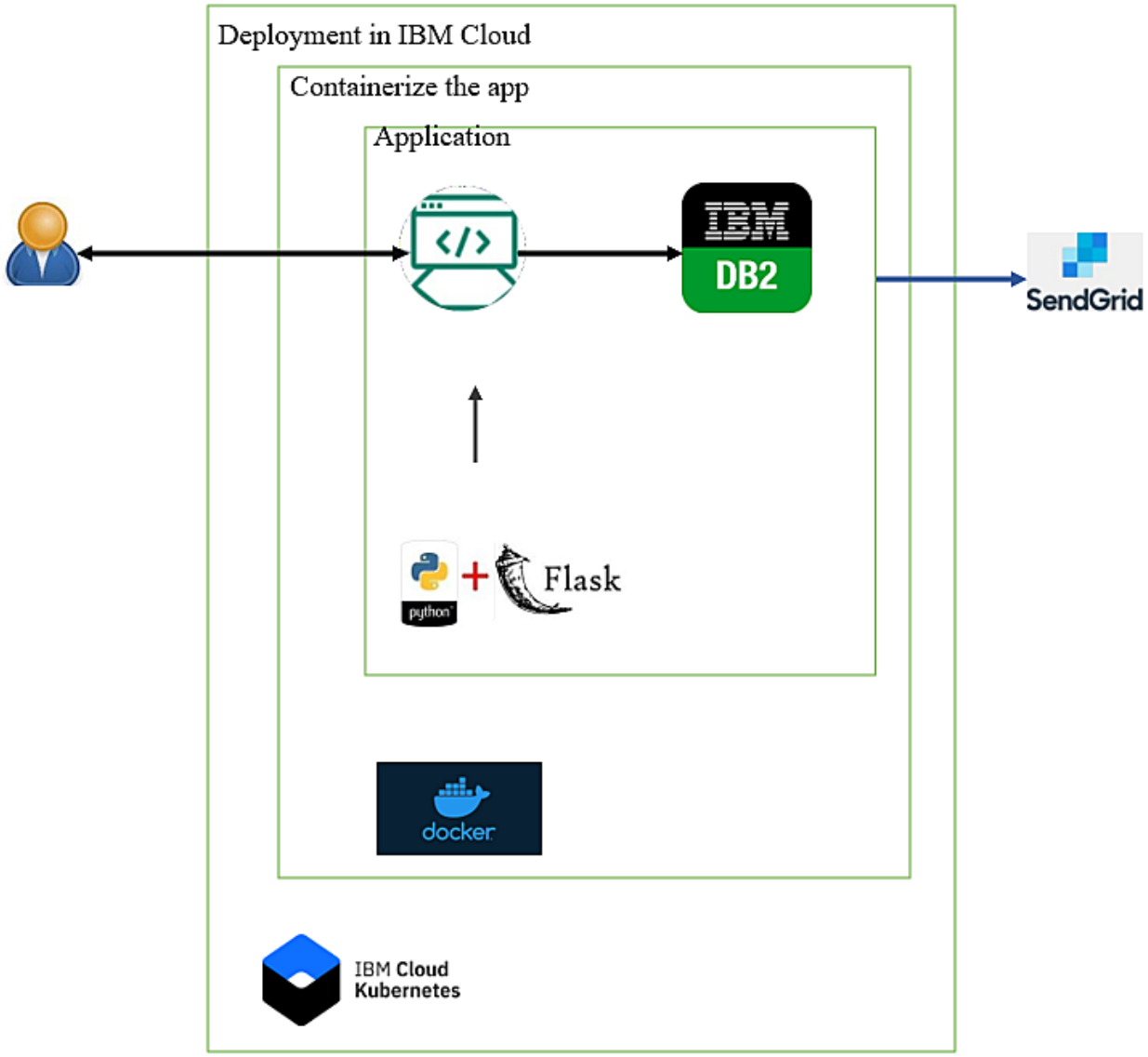
5.1 Data Flow Diagram

DFD for PERSONAL EXPENSE APPLICATION: (Industry Standard)TRACKER





5.2 Solution Architecture Diagram:



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can access the dashboard with Gmail Login	Medium	Sprint-1

	Login	USN-5	As a user, I can log into the application by entering email & password	I can enter into the application by using the registered email and password	High	Sprint-1
	Dashboard	USN-6	As a user, I can enter into the dashboardto add my income and expenditures	I can view my daily, monthly and yearly expenses	High	Sprint-2
Customer Care Executive	Alerts and messages	USN-7	As a Customer Care Executive, I can send Alerts and messages to the user	I can send alerts when the user exceeds the expense limit	High	Sprint-4
	Call Service	USN-8	As a Customer Care Executive, I can alsohelp out the customer at any part of time through the Customer Care Number available in the application	I can help the user to clarify their doubts regarding the usage of application	Medium	Sprint-3
Administrator	Application	USN-9	As an administrator I can upgrade or update the application.	I can fix the bug which arises for the customers and users of the application	High	Sprint-4

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	3 SEPTEMBER 2022.
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	10 SEPTEMBER 2022
Ideation	organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	10 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the scalability of solution ,idea, novelty business model, social impact, etc.	24 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document	01 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	08 OCTOBER 2022

Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	08 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	15 OCTOBER 2022
Data Flow Diagrams	Prepare the functional requirement document.	15 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	15 OCTOBER 2022
PrepareMilestone & Activity List	Prepare the milestones & activity list of the project.	18 OCTOBER 2022
Sprint Delivery Plan	Prepare sprint delivery plan	18 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	IN PROGRESS…

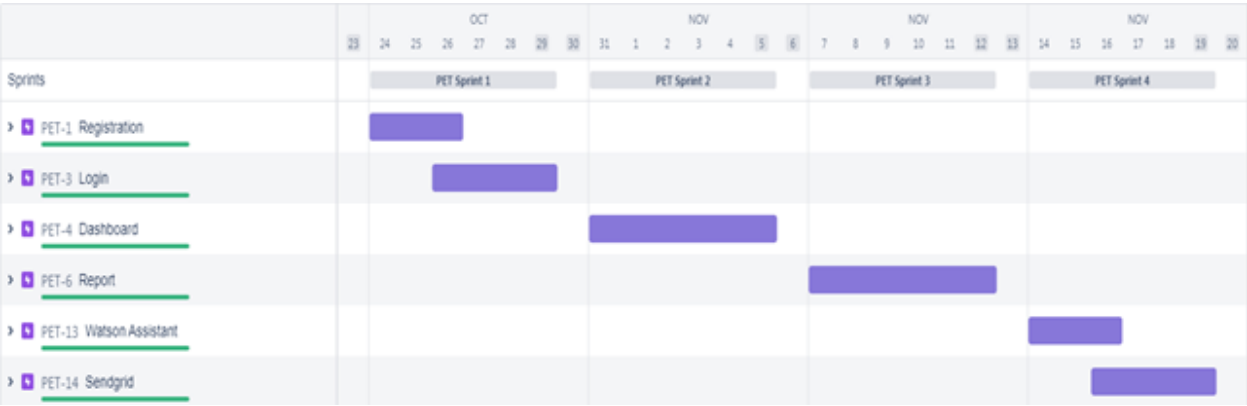
6.2 Sprint delivery schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Varsha, Rajashree
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	2	Low	Manjaarika
Sprint-1		USN-4	As a user, I can log into the application by entering user name & password	1	Low	Rajashree
Sprint-2	Dashboard	USN-5	As a user, I can enter into the dashboard to add transaction and salary	2	High	Shruthi, Varsha

Sprint-2		USN-6	As a user ,I can view the transaction history and salary of a particular month.	2	Medium	Shruthi, Manjaarika
Sprint-2		USN-7	As a user , I can view the current month balance	2	Medium	Varsha
Sprint-3	Report	USN-8	As a user, I can view the category wise transaction for a particular month in the form of pie chart	3	High	Manjaarika, Rajashree

Sprint-3		USN-9	As a user , I can view the comparison between the previous year and current year transaction in the form of bar graph	3	high	varsha, Rajashree
Sprint-4	Watson Assistant	USN-10	As a user, I can clarify the queries using chatbot	2	Low	Shruthi
Sprint-4	SendGrid	USN-11	As a user, I can receive Alerts and messages via email when it exceeds the certain a amount	4	High	Manjaarika, Shruthi

6.3 Reports from JIRA:



7 CODING AND SOLUTION

7.1 FEATURE 1

HOME:

First page of the personal expense tracker application is home page. In This home page two option are available register and login. By clicking register button to signup if already signup click login page to enter the application.

-

index.html

```
{% extends 'layout.html' %} {% block body %}
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
.container {
position: relative;
text-align: center;
```

```
color: white;}
.centered {
position:relative;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
}
</style>
</head>
<body>

<div class="container">
<img src="../static/bg.jpg" alt="Snow" style="width:100% ; height: 75vh;">
<div class="centered">
<h2>Log your spendings, earnings & budget with ease to avoid financial crises.</h2>
<center class="sz">
<a class=" btn btn-outline-success btn-lg" href="\register">Register</a>
<a class=" btn btn-outline-success btn-lg" href="\login">Login</a></center></div>
</div>
{% endblock %}
```

REGISTER:

A new user can register themselves by providing user,email,password. By registering as a new user, they can login anytime using username and password

Register.html

```
{% extends 'layout.html' %} {% block body %}

<div class="signUp container text-white">
  {% from "includes/_formhelpers.html" import render_field %}
  <form action="" method="post">
    <div class="green-text form-group mt-3">
      {{render_field(form.first_name, class_="form-control")}}
    </div>
```

```
        <div class="green-text form-group">
            {{render_field(form.last_name, class_="form-control")}}
        </div>
        <div class="green-text form-group">
            {{render_field(form.email, class_="form-control")}}
        </div>
        <div class="green-text form-group">
            {{render_field(form.username, class_="form-control")}}
        </div>
        <div class="green-text form-group">
            {{render_field(form.password, class_="form-control")}}
        </div>
        <div class="green-text form-group">
            {{render_field(form.confirm, class_="form-control")}}
        </div>
        <p class="text-center"><input class="btn btn-info" type="submit" value="Sign Up" /></p>
    </form>
    <p class="account">Have an account?<a class="green-text hover"
href="/login">Login</a></p>
</div>
{% endblock %}
```

app.py

```
class RegistrationForm(Form):
    first_name = StringField("First Name", [validators.Length(min=1, max=100)])
    last_name = StringField("Last Name", [validators.Length(min=1, max=100)])
    username = StringField('Username', [validators.Length(min=4, max=25)])
    email = StringField('Email Address', [validators.Length(min=6, max=35)])
    password = PasswordField(
        "Password",
        [
            validators.DataRequired(),
            validators.EqualTo("confirm", message="Passwords do not match"),
```

```
        ],
    )
    confirm = PasswordField("Confirm Password")

@app.route('/register', methods=['GET', 'POST'])
def register():
    if "logged_in" in session and session["logged_in"] == True:
            flash("You are already logged in", "info")
    form = RegistrationForm(request.form)
    if request.method == 'POST' and form.validate():
        first_name = form.first_name.data
        last_name = form.last_name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.encrypt(str(form.password.data))
        #database
        sql="SELECT * FROM user WHERE email=?"
        prep_stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(prep_stmt,1,email)
        ibm_db.execute(prep_stmt)
        account=ibm_db.fetch_assoc(prep_stmt)
        if account:
            flash("The entered email address has already been taken.Please try using or creating
another one.","info",)
            return redirect(url_for("register"))
        else:
                insert_sql="INSERT INTO user
(FIRST_NAME,LAST_NAME,EMAIL,USER_NAME,PASSWORD) values(?,?,?,?,?)"
                prep_stmt=ibm_db.prepare(conn,insert_sql)
                ibm_db.bind_param(prep_stmt,1,first_name)
                ibm_db.bind_param(prep_stmt,2,last_name)
                ibm_db.bind_param(prep_stmt,3,email)
                ibm_db.bind_param(prep_stmt,4,username)
                ibm_db.bind_param(prep_stmt,5,password)
```

```
            ibm_db.execute(prep_stmt)
            flash(" Registration successfull. Log in to continue !")
            flash('Thanks for registering')
            return redirect(url_for('login'))
    return render_template('register.html', form=form)
```

LOGIN:

A user can login with username and password. If the password of the person is correct, user will be taken to the dashboard. If password or username is incorrect, it shows the notification saying incorrect.

login.html

```
{% extends 'layout.html' %} {% block body %}
<div class="login container text-white">
    {% from "includes/_formhelpers.html" import render_field %}
    <form action="" method="post">
        <div class="green-text form-group mt-5">
            {{render_field(form.username,class_="form-control")}}
        </div>
        <div class="green-text form-group">
            {{render_field(form.password, class_="form-control")}}
        </div>
        <div class="enter">
        <input class="btn btn-info submit" type="submit" value="Login" />
          <h5 class="forgot"><a href="{{ url_for('reset_request') }}" class="green-text link">Forgot
Password?</a></h5>
        </div>
    </form>
      <p class="account">Don't have an account? <a class="green-text hover" href="/register">
SignUp</a></p>
</div>
{% endblock %}
```

app.py

```
class LoginForm(Form):
    username = StringField("Username", [validators.Length(min=4, max=100)])
    password = PasswordField(
        "Password",
        [
            validators.DataRequired(),
        ],
    )


@app.route("/login", methods=["GET", "POST"])
def login():
    if "logged_in" in session and session["logged_in"] == True:
        flash("You are already logged in", "info")
        return redirect(url_for("addTransactions"))
    form = LoginForm(request.form)
    if request.method == "POST" and form.validate():
        username = form.username.data
        password_input = form.password.data
        #database
        sql="SELECT * FROM user WHERE user_name=? or email=?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,username)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            userID = account["ID"]
            password = account["PASSWORD"]
            mailid=account["EMAIL"]
            role = account["ROLE"]
```

```
            if sha256_crypt.verify(password_input, password):
                session["logged_in"] = True
                session["username"] = username
                session["role"] = role
                session["userID"] = userID
                session["mailid"]=mailid
                flash("Logged in successfully!")
                return redirect(url_for("addTransactions"))
            else:
                error = "Invalid Password"
                return render_template("login.html", form=form, error=error)
        else:
            error = "Username not found"
            return render_template("login.html", form=form, error=error)

    return render_template("login.html", form=form)
```

7.2 FEATURE 2

DASHBOARD:

In dashboard usesr can view their transactions , add transactions and salary and also able view the current month charts.

Layout.html

```
<!DOCTYPE html>
<html lang="en">
    <body class="bg-dark">
        {% include 'includes/_navbar.html' %}
        <div class="container bg-dark">
            {% include 'includes/_messages.html' %} {% block body %}{% endblock %}
        </div>
```

Navbar.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark sticky-top   ">
    <img src="../static/money-icon.png" style="width: 40px; height: 40px;" />
```

```
    <a class="navbar-brand" href="/">
        <h2 class="green-text">Expense<span class="textlight">Tracker</span></h2>
    </a><button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav mr-auto"></ul>
        <ul class="navbar-nav ml-auto">
            <li class="nav-item">
                <a class="nav-link" href="/">
                    <h5 class="text-light">Home</h5><span class="sronly"></span></a></li>
            {% if session.logged_in %}
            <li class="nav-item">
                <a class="nav-link" href="/addSalary">
                    <h5 class="text-light">Salary</h5><span class="sr-only"></span></a></li>
            <li class="nav-item">
                <a class="nav-link" href="/addTransactions">
                    <h5 class="text-light">Transactions</h5><span class="sr-only"></span></a></li>
<li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    <span class="text-light h5">History</span>
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
                  <a class="dropdown-item" href="/transactionHistory">Transaction</a>
                  <a class="dropdown-item" href="/salaryHistory">Salary</a>
<li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    <span class="text-light h5">Report</span></a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
                  <a class="dropdown-item" href="category">Category Pie Char</a>
```

```
                <a class="dropdown-item" href="yearly_bar">Comparison Bar Chart</a>
                <a class="dropdown-item" href="daily_line">Daily Line Chart</a></div></li>

            <li class="nav-item">
                <a class="nav-link" href="/logout">
                    <h5 class="text-light">Logout</h5><span class="sr-only"></span></a></li>
            {% else %}<li class="nav-item">
                <a class="nav-link" href="/register">
                    <h5 class="text-light">Sign Up</h5></a></li>
            <li class="nav-item">
                <a class="nav-link" href="/login">
                    <h5 class="text-light">Login</h5><span class="sr-only"></span></a>
            {% endif %}
        </ul>
    </div>
</nav>
```

addtransaction.html

```
{% extends 'layout.html' %} {% block body %}
<div class="add">
    <h2 class="text-light">Add Transactions</h2>
    {% from "includes/_formhelpers.html" import render_field %}



    <form class="form" method="POST" action="">
        <div class="form-group row">
            <div class="form-group col-md-6">
                <input type="number" placeholder="Enter Amount" class="form-control"
name="amount"
                    value="{{request.form.amount}}" />
            </div>
            <div class="form-group category col-md-6">
                <select name="category" id="category" class="form-control">
                    <option value="Miscellaneous" selected="selected">Select Category</option>
                    <option value="Others">Others</option>
```

```
                <option value="Miscellaneous">Miscellaneous</option>
                <option value="Food">Food</option>
                <option value="Transportation">Transportation</option>
                <option value="Groceries">Groceries</option>
                <option value="Clothing">Clothing</option>
                <option value="HouseHold">HouseHold</option>
                <option value="Rent">Rent</option>
                <option value="Bills and Taxes">Bills and Taxes</option>
                <option value="Vacations">Vacations</option>
            </select>
        </div>
        <div class="form-group col-md-10 col-lg-11">
            <input type="text" placeholder="Enter Description" name="description" class="form-
control"
                value="{{request.form.description}}" />
        </div>
        <div class="form-group col-md-2 col-lg-1 btn">
            <button type="submit" class="btn btn-primary">Add</button>
        </div>
    </div>
  </form>
  {% if result != 0%}
  <div class="current-month">
    <h4 class="text-light float-left">
        Expenses Made This Month = <span class="green-text expense">₹
{{totalExpenses}}</span>
    </h4>
    <p class="text-light float-left swipe">Swipe to Edit/Delete</p>
    <!--<a href="category" class="btn btn-warning pie_chart float-right">Category Pie
Chart</a>
    <a href="yearly_bar" class="btn btn-warning bar_chart float-right">Comparison Bar
Chart</a>
    <a href="daily_line" class="btn btn-warning line_chart float-right">Daily Line Chart</a>--
>
```

```
        <h4 class="text-light float-right">
            Balance = <span class="green-text expense">₹ {{balances}}</span>
        </h4>
    </div>
    <div class="table-responsive">
        <table class="table table-striped text-light">
            <thead>
                <tr>
                    <th>Date</th>
                    <th>Amount</th>
                    <th>Category</th>
                    <th>Description</th>
                    <th></th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                {% for transaction in transactions %}
                <tr>
                    <td>{{transaction.DATE}}</td>
                    <td>{{transaction.AMOUNT}}</td>
                    <td>{{transaction.CATEGORY}}</td>
                    <td>{{transaction.DESCRIPTION}}</td>
                    <td><a href="editCurrentMonthTransaction/{{transaction.ID}}" class="btn btn-
primary pull-right">Edit</a>
                    </td>
                    <td>
                        <button type="button" class="btn btn-danger delete-transaction" data-
toggle="modal"
                            data-target="#exampleModalCenter" data-id="{{transaction.ID}}"
                            data-url="{{url_for('deleteCurrentMonthTransaction', id=transaction.ID)}}">
                            Delete
                        </button>
                    </td>
```

```
            </tr>
            {% endfor %}
        </tbody>



    </table>
  </div>
</div>
<div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog"
  aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLongTitle">Confirmation</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        Are you sure you want to delete this transaction?
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
        <form class="modal-form" method="POST">
          <input type="hidden" name="_method" value="DELETE" />
          <input type="submit" value="Delete" class="btn btn-danger" />
        </form>
      </div>
    </div>
  </div>
</div>
```

{%endif%} {% endblock %}

app.py

```
@app.route("/addTransactions", methods=["GET", "POST"])
@is_logged_in
def addTransactions():
    if request.method == "POST" :
        amount = request.form["amount"]
        description = request.form["description"]
        category = request.form["category"]

        if(amount==""):
            flash("please enter the amount", "success")
            return redirect(url_for("addTransactions"))
        if(description==""):
            description="--"
        sql="SELECT SUM(amount) as AMT FROM transactions WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,session["userID"])
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        totalExpense = account["AMT"]

        sql="SELECT SUM(amount) as AMT FROM SALARY WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,session["userID"])
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        salary = account["AMT"]
```

```
        if salary==None and totalExpense==None:
            balance=0
        elif salary==None:
            balance=-totalExpense
        elif totalExpense==None:
            balance=salary
        else:
            balance=salary-totalExpense
        print(balance)
        print(amount)
        if (balance > int(amount)):
          sql="INSERT INTO transactions(user_id, amount, description,category)
VALUES(?,?,?,?)"
          stmt=ibm_db.prepare(conn,sql)
          ibm_db.bind_param(stmt,1,session["userID"])
          ibm_db.bind_param(stmt,2,amount)
          ibm_db.bind_param(stmt,3,description)
          ibm_db.bind_param(stmt,4,category)
          ibm_db.execute(stmt)
          flash("Transaction Successfully Recorded", "success")
          return redirect(url_for("addTransactions"))
        else:
            flash("Sorry you have not enough balance", "success")
            return redirect(url_for("addTransactions"))
    else:
        salary=0
        sql="SELECT SUM(amount) as AMT FROM transactions WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,session["userID"])
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        totalExpense = account["AMT"]
```

```
    sql="SELECT SUM(amount) as AMT FROM SALARY WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session["userID"])
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    salary = account["AMT"]

    if salary==None and totalExpense==None:
        balance=0
    elif salary==None:
        balance=-totalExpense
    elif totalExpense==None:
        balance=salary
    else:
        balance=salary-totalExpense

    if(balance<1000 and salary!=None):
      def sendMailUsingSendGrid(API,from_email,to_emails,subject,html_content):
        message=Mail(from_email,to_emails,subject,html_content)
        print(message)
        try:
          sg = SendGridAPIClient(API)
          response = sg.send(message)
          print(response.status_code)
          print(response.body)
          print(response.headers)
        except Exception as e:
          print(e.message)

      API=settings.get("APIKEY",None)
      from_email='personalexpensetrackerapp@gmail.com'
```

```
        to_email=session["mailid"]
        print(API)
        print(from_email)
        print(to_email)

        subject="hello0"
        html_content="Msg"

        sendMailUsingSendGrid(API,from_email,to_email,subject,html_content)

      list=[]
      # get the month's transactions made by a particular user
      sql="SELECT * FROM transactions WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ? ORDER BY date DESC"
      stmt=ibm_db.prepare(conn,sql)
      ibm_db.bind_param(stmt,1,session["userID"])
      ibm_db.execute(stmt)
      transactions=ibm_db.fetch_assoc(stmt)

      if transactions:
        while transactions!=False:
          list.append(transactions)
          transactions = ibm_db.fetch_assoc(stmt)

        for transaction in list:
            if datetime.datetime.now() - transaction["DATE"] < datetime.timedelta(days=0.5):
                transaction["DATE"] = timeago.format(transaction["DATE"],
datetime.datetime.now())
            else:
                transaction["DATE"] = transaction["DATE"].strftime("%d %B, %Y")
        return
render_template("addTransactions.html",totalExpenses=totalExpense,balances=balance,transacti
ons=list)
```

```
        else:
            return render_template("addTransactions.html", result=transactions)
    return render_template("addTransactions.html")
```

-

DATABASE SCHEMA

```
DROP TABLE IF EXISTS `user`
CREATE TABLE user
(
  id int NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY(START WITH
1,INCREMENT BY 1),
  first_name varchar (100) DEFAULT NULL,
  last_name varchar (100) DEFAULT NULL,
  email varchar (100) DEFAULT NULL,
  usernam varchar (100) DEFAULT NULL,
  password varchar (100) DEFAULT NULL,
  role varchar (100) DEFAULT 'user'
)
-
CREATE TABLE transactions
(
  id int NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY(START WITH
1,INCREMENT BY 1),
  user_id int DEFAULT NULL,
  amount int NOT NULL DEFAULT '0',
  description varchar (255) DEFAULT NULL,
  category varchar (255) DEFAULT NULL,
  date timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE NO ACTION
)

CREATE TABLE salary
(
```

```
  id int NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY(START WITH
1,INCREMENT BY 1),
  user_id int DEFAULT NULL,
  amount int NOT NULL DEFAULT '0',
  description varchar (255) DEFAULT NULL,
  date timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE NO ACTION
)
```

8. TESTING

8.1 Test Cases

Test case ID	Test Scenario	Expected Result	Status
Tracker_TC_OO1	Verify users are able to see the Register and log in button.	Login/Signup button should display	Pass
Tracker_TC_OO2	Verify the UI elements in the Login/Register button.	Login and Register page is viewed.	Pass
Tracker_TC_OO3	Verify user is able to log into application with Valid credentials	Users should navigate to the user account transaction page.	Pass
Tracker_TC_OO4	Verify user is able to log into application with InValid credentials	Application should show an 'Incorrect email or password ' message.	Pass

Tracker_TC_OO5	Verify the UI elements in Login page	Application should show below UI elements: a.email text box b.password text box c.Login button d.New customer? Sign up link e.Forgot password? Recovery password link	Pass
Tracker_TC_OO6	Verify if the input details given by the user during registration are valid	Application should not show 'Incorrect email or password ' validation message. And register the account.	Pass
Tracker_TC_OO7	Verify if the warning is shown when input details given by the user during registration are invalid	Application should show an 'Incorrect email or phone number' message.	Pass
Tracker_TC_OO8	Verify if the expenses field accepts only numbers.	Application should show an 'Incorrect input ' message.	Pass
Tracker_TC_OO9	Verify if data entered by the user, as expenses are displayed.	Application should display the entered data.	Pass
Tracker_TC_O10	Verify if data entered by a user can be modified and deleted.	Application should display the updated data.	Pass
Tracker_TC_O11	Verify if the income field accepts only numbers.	Application should show 'Incorrect input ' message	Pass
Tracker_TC_O12	Verify if data entered by the user as income is displayed.	Application should display the entered data.	Pass
Tracker_TC_O13	Verify if data entered by a user can be modified and deleted.	Application should display the updated data.	Pass
Tracker_TC_O14	Verify if user is able to see no matches found messages when no results are matching with the entered month or year	Application should show 'No results found message .	Pass
Tracker_TC_O15	Verify if user is able to see the history for the month entered	Application should display the required history.	Pass

8.2 User Acceptance Testing

1. **Purpose of Document**

   The purpose of this document is to briefly explain the test coverage and open issues of the Personal Expense Tracker Application, the time of the release to User Acceptance Testing (UAT).

1. **Defect Analysis**

   This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	3	2	5	12

Duplicate	1	0	0	1	2
External	2	`1	0	1	4
Fixed	5	2	3	4	14
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	10	6	5	11	32

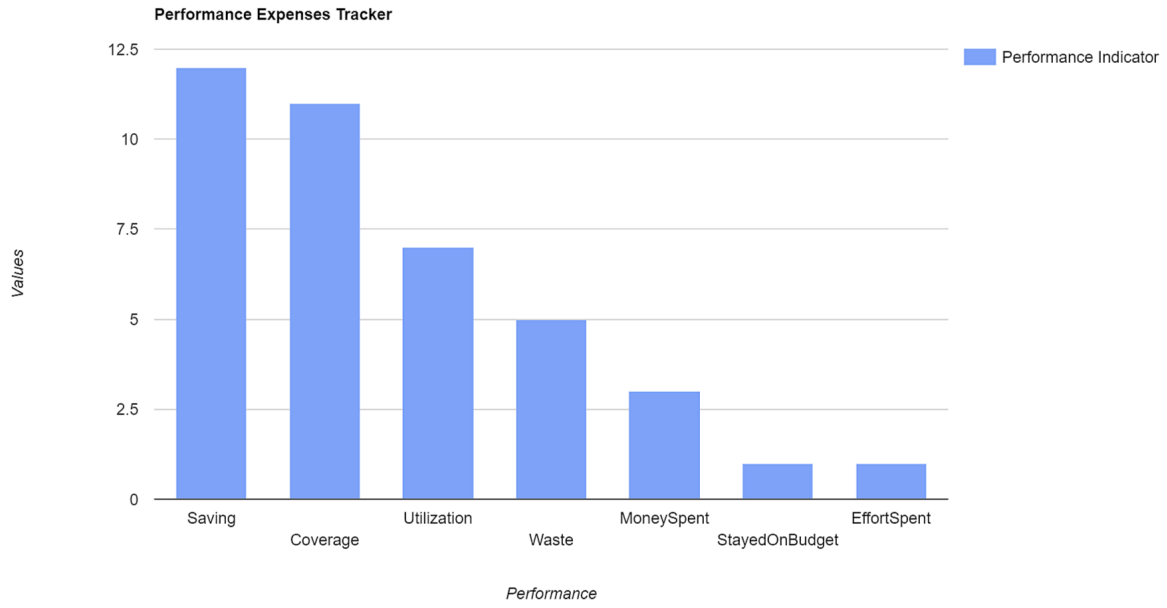
1. **Test Case Analysis**

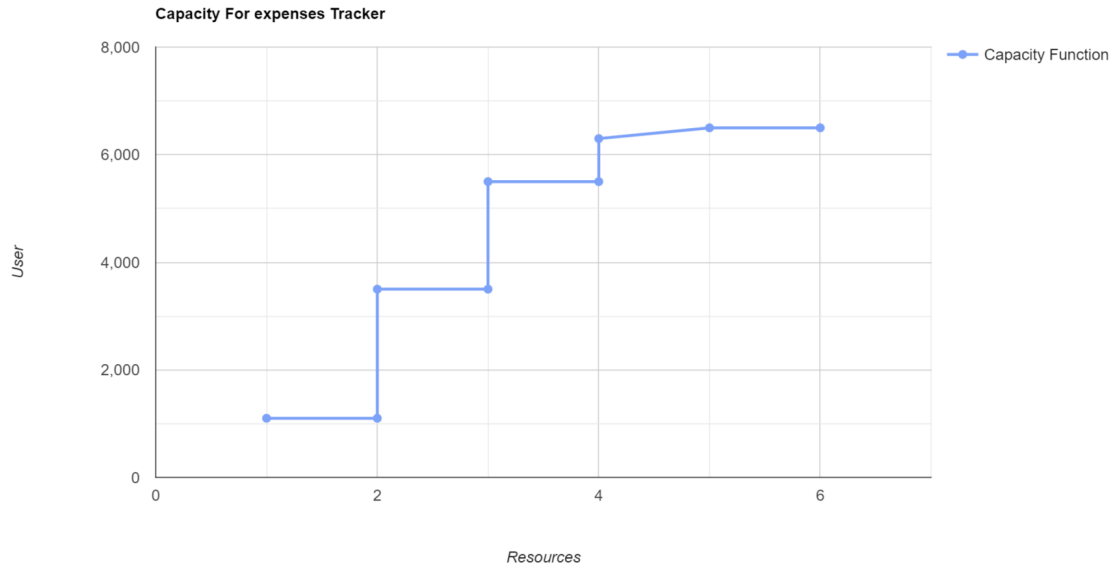
This report shows the number of test cases that have passed, failed, and untested

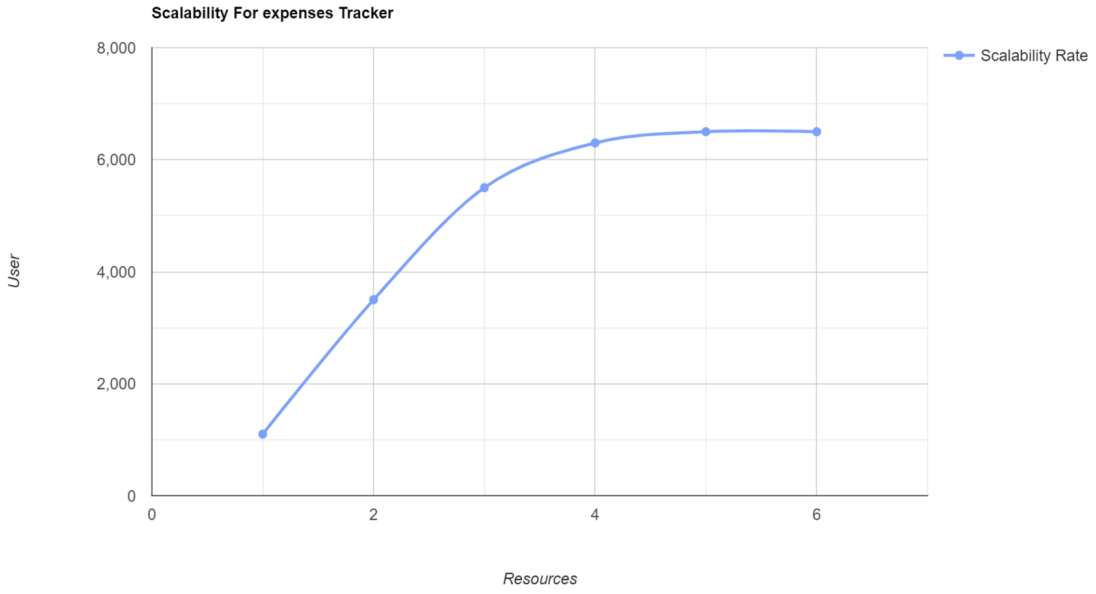
Section	Total Cases	Not Tested	Fail	Pass
Login	5	0	0	5
Register	4	0	0	4
Dashboard	8	0	0	8
Report Chart	2	0	0	2

9.RESULTS

9.1 PERFORMANCE METRICS


Performance Expenses Tracker
Performance Indicator
12.5
10
7.5
5
2.5
0
Values
Saving
Coverage
Utilization
Waste
MoneySpent
StayedOnBudget
EffortSpent
Performance


Capacity For expenses Tracker
Capacity Function
8,000
6,000
4,000
2,000
0
User
0
2
4
6
Resources


Scalability For expenses Tracker
8,000
6,000
4,000
2,000
0
User
0
2
4
6
Resources
Scalability Rate

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. With a daily expense manager, you will be able to allocate money to different priorities and this will also help you cut down on unnecessary spending. As a result, you will be able to save and be able to keep worry at bay.
2. A daily money tracker helps you budget your money so that you use it wisely.
3. you will be able to allocate money to different priorities.
4. Tracking Your Expenses Can Reveal Spending Issues and this will also help you cut down on unnecessary spending.
5. It Helps You Meet Your Financial Objectives.

DISADVANTAGES:

1. To get the full benefit of a budgeting app, you'll probably want to link your financial accounts to it.
2. However, this means that if the app gets hacked or your password gets stolen, all of those other important accounts will be at risk.
3. If you lose any money in the hack, or your credit card gets compromised, you could be on the hook because you violated the terms and conditions of your banking and card

holder agreements.

11.Conclusion

In this paper, After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and make them aware about their daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of the amount of expenses and wish to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money.

The new system has overcome most of the limitations of the existing system and works according to the design specification given. The project we have developed is work more efficient than the other income and expense tracker. The project successfully avoids the manual calculation for avoiding calculating the income and expense per month.

The modules are developed efficiently and also in an attractive manner. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system. The newly

developed system consumes less processing time and all the details are updated and processed immediately. Since the screen provides online help messages and is very user friendly, any user will get familiarized with its usage.

12. FUTURE SCOPE

1. In further days, there will be mails and pay mode embedded with the app.
2. Also, backup details will be recorded on the database.
3. History can be set to view all the details in the app even if the particular data is deleted from the database.
4. Statistics could be prepared based on the Income, Expense details of the user.
5. Sharing files via Bluetooth, WhatsApp can be allowed.
6. Printing the details of the particular income or expense details can be made.
7. Some of the extra components are like enabling users to register to the application using existing email or social network account, it will synchronize the users profile data to the application

13.APPENDIX

SOURCE CODE

Layout.html

```
<!DOCTYPE html>
<html lang="en">
    <body class="bg-dark">
        {% include 'includes/_navbar.html' %}
        <div class="container bg-dark">
            {% include 'includes/_messages.html' %} {% block body %}{% endblock %}
        </div></body></html>
```

Navbar.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark sticky-top   ">
    <img src="../static/money-icon.png" style="width: 40px; height: 40px;" />
    <a class="navbar-brand" href="/">
        <h2 class="green-text">Expense<span class="textlight">Tracker</span></h2>
    </a><button class="navbar-toggler" type="button" data-toggle="collapse" data-
```

```
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav mr-auto"></ul>
        <ul class="navbar-nav ml-auto">
            <li class="nav-item">
                <a class="nav-link" href="/">
                    <h5 class="text-light">Home</h5><span class="sronly"></span></a></li>
            {% if session.logged_in %}
            <li class="nav-item">
                <a class="nav-link" href="/addSalary">
                    <h5 class="text-light">Salary</h5><span class="sr-only"></span></a></li>
            <li class="nav-item">
                <a class="nav-link" href="/addTransactions">
                    <h5 class="text-light">Transactions</h5><span class="sr-only"></span></a></li>
<li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    <span class="text-light h5">History</span>
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
                  <a class="dropdown-item" href="/transactionHistory">Transaction</a>
                  <a class="dropdown-item" href="/salaryHistory">Salary</a>
<li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    <span class="text-light h5">Report</span></a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
                  <a class="dropdown-item" href="category">Category Pie Char</a>
                  <a class="dropdown-item" href="yearly_bar">Comparison Bar Chart</a>
                  <a class="dropdown-item" href="daily_line">Daily Line Chart</a></div></li>
```

```
        <li class="nav-item">
            <a class="nav-link" href="/logout">
                <h5 class="text-light">Logout</h5><span class="sr-only"></span></a></li>
        {% else %}<li class="nav-item">
            <a class="nav-link" href="/register">
                <h5 class="text-light">Sign Up</h5></a></li>
        <li class="nav-item">
            <a class="nav-link" href="/login">
                <h5 class="text-light">Login</h5><span class="sr-only"></span></a>
        {% endif %}
    </ul>
  </div>
</nav>
```

index.html

```
{% extends 'layout.html' %} {% block body %}
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
.container {
position: relative;
text-align: center;
color: white;}
.centered {
position:relative;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
}
</style>
</head>
<body>
```

```
<div class="container">
<img src="../static/bg.jpg" alt="Snow" style="width:100% ; height: 75vh;">
<div class="centered">
<h2>Log your spendings, earnings & budget with ease to avoid financial crises.</h2>
<center class="sz">
<a class=" btn btn-outline-success btn-lg" href="\register">Register</a>
<a class=" btn btn-outline-success btn-lg" href="\login">Login</a></center></div>
</div>
{% endblock %}

    {% extends 'layout.html' %} {% block body %}

<div class="signUp container text-white">
   {% from "includes/_formhelpers.html" import render_field %}
   <form action="" method="post">
      <div class="green-text form-group mt-3">
         {{render_field(form.first_name, class_="form-control")}}
      </div>
      <div class="green-text form-group">
         {{render_field(form.last_name, class_="form-control")}}
      </div>
      <div class="green-text form-group">
         {{render_field(form.email, class_="form-control")}}
      </div>
      <div class="green-text form-group">
         {{render_field(form.username, class_="form-control")}}
      </div>
      <div class="green-text form-group">
         {{render_field(form.password, class_="form-control")}}
      </div>
      <div class="green-text form-group">
         {{render_field(form.confirm, class_="form-control")}}
      </div>
      <p class="text-center"><input class="btn btn-info" type="submit" value="Sign Up" /></p>
```

```
    </form>
    <p class="account">Have an account?<a class="green-text hover"
href="/login">Login</a></p>
</div>
{% endblock %}
```

login.html

```
{% extends 'layout.html' %} {% block body %}
<div class="login container text-white">
    {% from "includes/_formhelpers.html" import render_field %}
    <form action="" method="post">
        <div class="green-text form-group mt-5">
            {{render_field(form.username,class_="form-control")}}
        </div>
        <div class="green-text form-group">
            {{render_field(form.password, class_="form-control")}}
        </div>
        <div class="enter">
        <input class="btn btn-info submit" type="submit" value="Login" />
          <h5 class="forgot"><a href="{{ url_for('reset_request') }}" class="green-text link">Forgot
Password?</a></h5>
        </div>
    </form>
      <p class="account">Don't have an account? <a class="green-text hover" href="/register">
SignUp</a></p>
</div>
{% endblock %}
```

addtransaction.html

```
{% extends 'layout.html' %} {% block body %}
<div class="add">
```

```
<h2 class="text-light">Add Transactions</h2>
{% from "includes/_formhelpers.html" import render_field %}


<form class="form" method="POST" action="">
    <div class="form-group row">
        <div class="form-group col-md-6">
            <input type="number" placeholder="Enter Amount" class="form-control"
name="amount"
                value="{{request.form.amount}}" />
        </div>
        <div class="form-group category col-md-6">
            <select name="category" id="category" class="form-control">
                <option value="Miscellaneous" selected="selected">Select Category</option>
                <option value="Others">Others</option>
                <option value="Miscellaneous">Miscellaneous</option>
                <option value="Food">Food</option>
                <option value="Transportation">Transportation</option>
                <option value="Groceries">Groceries</option>
                <option value="Clothing">Clothing</option>
                <option value="HouseHold">HouseHold</option>
                <option value="Rent">Rent</option>
                <option value="Bills and Taxes">Bills and Taxes</option>
                <option value="Vacations">Vacations</option>
            </select>
        </div>
        <div class="form-group col-md-10 col-lg-11">
            <input type="text" placeholder="Enter Description" name="description" class="form-
control"
                value="{{request.form.description}}" />
        </div>
        <div class="form-group col-md-2 col-lg-1 btn">
            <button type="submit" class="btn btn-primary">Add</button>
        </div>
```

```
        </div>
    </form>
    {% if result != 0%}
    <div class="current-month">
        <h4 class="text-light float-left">
            Expenses Made This Month = <span class="green-text expense">₹
{{totalExpenses}}</span>
        </h4>
        <p class="text-light float-left swipe">Swipe to Edit/Delete</p>
        <!--<a href="category" class="btn btn-warning pie_chart float-right">Category Pie
Chart</a>
        <a href="yearly_bar" class="btn btn-warning bar_chart float-right">Comparison Bar
Chart</a>
        <a href="daily_line" class="btn btn-warning line_chart float-right">Daily Line Chart</a>--
>
        <h4 class="text-light float-right">
            Balance = <span class="green-text expense">₹ {{balances}}</span>
        </h4>
    </div>
    <div class="table-responsive">
        <table class="table table-striped text-light">
            <thead>
                <tr>
                    <th>Date</th>
                    <th>Amount</th>
                    <th>Category</th>
                    <th>Description</th>
                    <th></th>
                    <th></th>
                </tr>
            </thead>
            <tbody>
                {% for transaction in transactions %}
                <tr>
```

```
                <td>{{transaction.DATE}}</td>
                <td>{{transaction.AMOUNT}}</td>
                <td>{{transaction.CATEGORY}}</td>
                <td>{{transaction.DESCRIPTION}}</td>
                <td><a href="editCurrentMonthTransaction/{{transaction.ID}}" class="btn btn-
primary pull-right">Edit</a>
                </td>
                <td>
                    <button type="button" class="btn btn-danger delete-transaction" data-
toggle="modal"
                        data-target="#exampleModalCenter" data-id="{{transaction.ID}}"
                        data-url="{{url_for('deleteCurrentMonthTransaction', id=transaction.ID)}}">
                        Delete
                    </button>
                </td>
            </tr>
            {% endfor %}
        </tbody>



    </table>
  </div>
</div>
<div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog"
  aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLongTitle">Confirmation</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
```

```
            Are you sure you want to delete this transaction?
        </div>
        <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
            <form class="modal-form" method="POST">
                <input type="hidden" name="_method" value="DELETE" />
                <input type="submit" value="Delete" class="btn btn-danger" />
            </form>
        </div>
      </div>
   </div>
</div>

   {%endif%} {% endblock %}
```

-

app.py

```
from mailbox import Mailbox
import ibm_db
from flask import Flask, render_template, request, flash, redirect, url_for,
session, logging, jsonify,Response,
from wtforms import Form, StringField, PasswordField, TextAreaField,
IntegerField, validators,
from wtforms.validators import DataRequired
from passlib.hash import sha256_crypt
from functools import wraps
import timeago, datetime
from wtforms.fields import EmailField
from itsdangerous import URLSafeTimedSerializer as Serializer
from flask_mail import Mail, Message
import plotly.graph_objects as go
import configparser
```

```
import ssl
ssl._create_default_https_context=ssl._create_unverified_context
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

app=Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

class RegistrationForm(Form):
    first_name = StringField("First Name", [validators.Length(min=1, max=100)])
    last_name = StringField("Last Name", [validators.Length(min=1, max=100)])
    username = StringField('Username', [validators.Length(min=4, max=25)])
    email = StringField('Email Address', [validators.Length(min=6, max=35)])
    password = PasswordField("Password",
        [ validators.DataRequired(),
       validators.EqualTo("confirm", message="Passwords do not match"),],)
    confirm = PasswordField("Confirm Password")

@app.route('/register', methods=['GET', 'POST'])
def register():
    if "logged_in" in session and session["logged_in"] == True:
            flash("You are already logged in", "info")
    form = RegistrationForm(request.form)
    if request.method == 'POST' and form.validate():
        first_name = form.first_name.data
        last_name = form.last_name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.encrypt(str(form.password.data))
        #database
        sql="SELECT * FROM user WHERE email=?"
```

```
        prep_stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(prep_stmt,1,email)
        ibm_db.execute(prep_stmt)
        account=ibm_db.fetch_assoc(prep_stmt)
        if account:
            flash("The entered email address has already been taken.Please try using or creating
another one.","info",)
            return redirect(url_for("register"))
        else:
                insert_sql="INSERT INTO user
(FIRST_NAME,LAST_NAME,EMAIL,USER_NAME,PASSWORD) values(?,?,?,?,?)"
                prep_stmt=ibm_db.prepare(conn,insert_sql)
                ibm_db.bind_param(prep_stmt,1,first_name)
                ibm_db.bind_param(prep_stmt,2,last_name)
                ibm_db.bind_param(prep_stmt,3,email)
                ibm_db.bind_param(prep_stmt,4,username)
                ibm_db.bind_param(prep_stmt,5,password)
                ibm_db.execute(prep_stmt)
                flash(" Registration successfull. Log in to continue !")
                flash('Thanks for registering')
                return redirect(url_for('login'))
    return render_template('register.html', form=form)

class LoginForm(Form):
    username = StringField("Username", [validators.Length(min=4, max=100)])
    password = PasswordField(
        "Password",
        [
            validators.DataRequired(),
        ],
    )

@app.route("/login", methods=["GET", "POST"])
def login():
```

```
if "logged_in" in session and session["logged_in"] == True:
    flash("You are already logged in", "info")
    return redirect(url_for("addTransactions"))
form = LoginForm(request.form)
if request.method == "POST" and form.validate():
    username = form.username.data
    password_input = form.password.data
    #database
    sql="SELECT * FROM user WHERE user_name=? or email=?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,username)
    ibm_db.bind_param(stmt,2,username)
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
        userID = account["ID"]
        password = account["PASSWORD"]
        mailid=account["EMAIL"]
        role = account["ROLE"]
        if sha256_crypt.verify(password_input, password):
            session["logged_in"] = True
            session["username"] = username
            session["role"] = role
            session["userID"] = userID
            session["mailid"]=mailid
            flash("Logged in successfully!")
            return redirect(url_for("addTransactions"))
        else:
             error = "Invalid Password"
             return render_template("login.html", form=form, error=error)
    else:
        error = "Username not found"
        return render_template("login.html", form=form, error=error)
```

```
    return render_template("login.html", form=form)

@app.route("/addTransactions", methods=["GET", "POST"])
@is_logged_in
def addTransactions():
     if request.method == "POST" :
        amount = request.form["amount"]
        description = request.form["description"]
        category = request.form["category"]

        if(amount==""):
            flash("please enter the amount", "success")
            return redirect(url_for("addTransactions"))
        if(description==""):
            description="--"
        sql="SELECT SUM(amount) as AMT FROM transactions WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,session["userID"])
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        totalExpense = account["AMT"]

        sql="SELECT SUM(amount) as AMT FROM SALARY WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,session["userID"])
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        salary = account["AMT"]
```

```
        if salary==None and totalExpense==None:
            balance=0
        elif salary==None:
            balance=-totalExpense
        elif totalExpense==None:
            balance=salary
        else:
            balance=salary-totalExpense
        print(balance)
        print(amount)
        if (balance > int(amount)):
          sql="INSERT INTO transactions(user_id, amount, description,category)
VALUES(?,?,?,?)"
          stmt=ibm_db.prepare(conn,sql)
          ibm_db.bind_param(stmt,1,session["userID"])
          ibm_db.bind_param(stmt,2,amount)
          ibm_db.bind_param(stmt,3,description)
          ibm_db.bind_param(stmt,4,category)
          ibm_db.execute(stmt)
          flash("Transaction Successfully Recorded", "success")
          return redirect(url_for("addTransactions"))
        else:
            flash("Sorry you have not enough balance", "success")
            return redirect(url_for("addTransactions"))
    else:
        salary=0
        sql="SELECT SUM(amount) as AMT FROM transactions WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
        stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,session["userID"])
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
        totalExpense = account["AMT"]
```

```
    sql="SELECT SUM(amount) as AMT FROM SALARY WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session["userID"])
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    salary = account["AMT"]

    if salary==None and totalExpense==None:
        balance=0
    elif salary==None:
        balance=-totalExpense
    elif totalExpense==None:
        balance=salary
    else:
        balance=salary-totalExpense

    if(balance<1000 and salary!=None):
      def sendMailUsingSendGrid(API,from_email,to_emails,subject,html_content):
        message=Mail(from_email,to_emails,subject,html_content)
        print(message)
        try:
          sg = SendGridAPIClient(API)
          response = sg.send(message)
          print(response.status_code)
          print(response.body)
          print(response.headers)
        except Exception as e:
          print(e.message)

      API=settings.get("APIKEY",None)
      from_email='personalexpensetrackerapp@gmail.com'
```

```
        to_email=session["mailid"]
        print(API)
        print(from_email)
        print(to_email)

        subject="hello0"
        html_content="Msg"

        sendMailUsingSendGrid(API,from_email,to_email,subject,html_content)

      list=[]
      # get the month's transactions made by a particular user
      sql="SELECT * FROM transactions WHERE MONTH(date) =
MONTH(CURRENT_TIMESTAMP) AND YEAR(date) = YEAR(CURRENT_TIMESTAMP)
AND user_id = ? ORDER BY date DESC"
      stmt=ibm_db.prepare(conn,sql)
      ibm_db.bind_param(stmt,1,session["userID"])
      ibm_db.execute(stmt)
      transactions=ibm_db.fetch_assoc(stmt)

      if transactions:
        while transactions!=False:
          list.append(transactions)
          transactions = ibm_db.fetch_assoc(stmt)

        for transaction in list:
            if datetime.datetime.now() - transaction["DATE"] < datetime.timedelta(days=0.5):
                transaction["DATE"] = timeago.format(transaction["DATE"],
datetime.datetime.now())
            else:
                transaction["DATE"] = transaction["DATE"].strftime("%d %B, %Y")
        return
render_template("addTransactions.html",totalExpenses=totalExpense,balances=balance,transacti
ons=list)
```

```
    else:
        return render_template("addTransactions.html", result=transactions)
    return render_template("addTransactions.html")
```

GITHUB AND PROJECT DEMO LINK

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-31755-1660204720

PROJECT DEMO LINK:

https://drive.google.com/file/d/1dGrnJr7khR-NBi_d2I9KGOzhBJaU3XDd/view?usp=share_link