

## **SYNOPSIS**

### **1.INTRODUCTION**

- 1.1 Project Overview
- 1.2 Purpose

### **2.LITERATURE SURVEY**

- 2.1 Existing problem
- 2.2References
- 2.3Problem Statement Definition

### **3.IDEATION & PROPOSED SOLUTION**

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

### **4. REQUIREMENT ANALYSIS**

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

### **5.PROJECT DESIGN**

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

### **6.PROJECT PLANNING & SCHEDULING**

- 6.1 Sprint Planning & Estimation
- 6.2 Reports from JIRA

### **7.CODING & SOLUTIONING**

- 7.1 Html coding

## 7.2 Python coding

## **8.RESULTS**

### 8.1 Signup

### 8.2 Login

### 8.3 Screen layout

### 8.4 Assistant

## **9.ADVANTAGES**

## **10.CONCLUSION**

# 1. INTRODUCTION

## **a. PROJECT OVERVIEW**

If you've ever decided to get a handle on your cash flow, you know that the first step is knowing how you're spending your money. That's something really low on the fun-to-do meter.

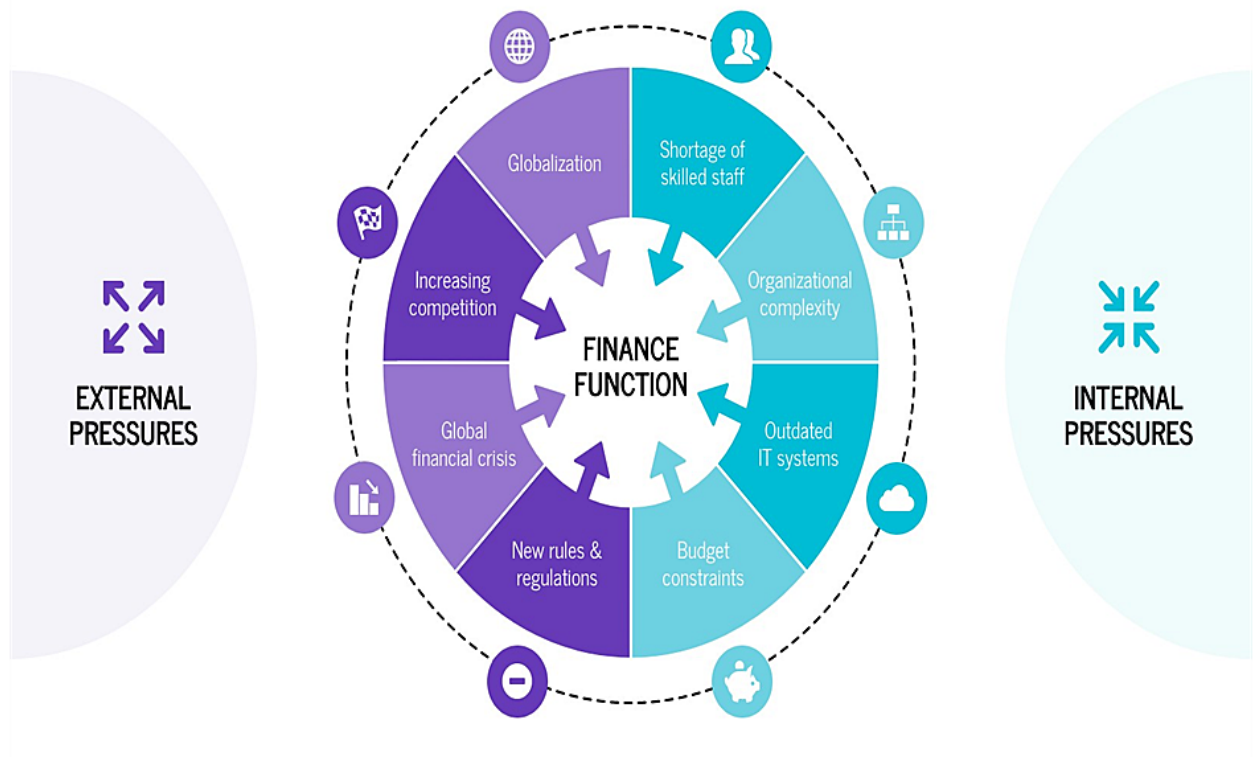
Expense tracker apps help you collect and classify your purchases so that you can identify areas that might be trimmed. Or, in the case of building net worth, places where you can allocate more money, such as savings. You might track expenses for a while just to get an idea of where your money's going, or it might be a stepping stone toward making and following a budget.

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes 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.

## **1.2 PURPOSE**

Personal finance application will ask users to add their expenses and based on their expense's wallet balance will be updated which will be visible to the user. Also, user 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.

## 2.LITERATURE SURVEY



This above study shows that the people prefer and are getting comfortable in managing their finances through a mobile app and evolve from the age-old paper system. They have understood the benefits in leveraging digital tools to get better insights and a bigger look over their finances.

Parents are worried that their children lack financial literacy unlike their global counterparts. They are looking for applications where their children can learn about spending money. This financial illiteracy is prevalent even among elders.

## **2.1 Existing solutions**

### **Money View - Expense Manager App**

Money View App reads all of the transactional SMS messages and provides you with real-time visibility into your finances. This app unearths the hidden financial data that sits idly in SMS logs and makes excellent use of it.

#### **Key features of the App:**

1. Check your bank account balances.
2. See the most recent banking transactions.
3. The Money View app automatically categorises your payments and displays major areas of spending.
4. View weekly and monthly summaries to help you avoid overspending and improve the efficiency of your budget planning.
5. It keeps track of your spending, sends personalised bill-paying reminders, finds relevant savings opportunities, and much more. • Track your financial progress by looking at your spending trends over time.

### **Good budget - Budget & Finance App**

This personal finance manager app acts as a proactive budget planner, assisting you in staying on top of your budget, bills, and finances. The personal finance app was designed for simple, real-time budget and financial tracking, making it one of the best expense tracker apps in India.

#### **Key Features of the App:**

1. Data is backed up automatically and securely to Good budget's website.
2. Disaggregate expense transactions
3. Transactions that are scheduled and envelope fills
4. Save time by using intelligent payee and category suggestions.

5. Transfer funds between Envelopes and Accounts with ease.
6. Match the budget period to the real-life situation.
7. Analyse spending with the Spending by Envelope Report.
8. Use the Income vs. Spending Report to keep track of your cash flow.
9. Export transactions to CSV
10. Carryover any unused funds to the next month as a reward for your incredible self-control.
11. Plan your finances ahead of time to stay on track with your budget.

### **Real byte Money Manager App**

You can use the budget planner and spending tracker to keep track of your personal and business financial transactions, review financial data on a daily/weekly/monthly basis, and manage your assets.

#### **Key Features of the App:**

1. System of double-entry bookkeeping
2. Management of budgets and expenses
3. Management of credit and debit cards
4. Get access to statistics immediately.
5. Bookmarking feature
6. Backup/restore function

### **Monefy - Budget Manager and Expense Tracker App**

Monefy tracks the user's expenses and compares them to the monthly income and the budget planner. Monefy's money manager app keeps your monthly budget in top shape. As a result, it could also serve as the best expense tracker app.

#### **Key Features of the App:**

1. With the intuitive and simple-to-use interface, you can quickly add new

records.

2. Maintain a multi-currency track.
3. Backup and export personal finance data with a single click.
4. Protect your data with passcode protection.
5. View your spending distribution on a simple chart, or get detailed information from the records list.
6. Use a budget tracker to save money.
7. Use your own Google Drive or Dropbox account to safely synchronise.
8. Take control of recurring payments.
9. Create multiple accounts.
10. Use the built-in calculator to crunch numbers.

#### **Wallet - Money, Budget, Finance & Expense Tracker App**

Wallet can automatically track your daily expenses by syncing your bank account, view weekly expense reports, plan your shopping expenses, and share specific features with your loved ones. You can manage your money with a wallet from anywhere and at any time.

#### **Key Features of the App:**

1. Transactions are automatically and securely synced, then intelligently categorised and budgeted.
2. Simple graphs and financial overviews provide actionable insights into the state of your finances, including accounts, credit and debit cards, debts, and cash.
3. Arrange your bills and keep track of their due dates.
4. Examine upcoming payments and how they will affect your cash flow.
5. Selected accounts can be shared with family, friends, or coworkers who need to work together on a budget. Everyone is welcome to contribute from any platform, including Android, iPhone, and the Web.
6. Other features include support for multiple currencies, automatic cloud

sync, receipt and warranty tracking, categories and templates, geo-mapping transactions, hash-tagging, shopping lists, exports to CSV/XLS/PDF, debt management, PIN security, standing orders, notifications, reports, and more.

### **Walnut - All Indian Banks Money Manager App**

Walnut automates and secures the tracking of your monthly expenses. You can stay within your budget, pay your bills on time, and save more money each month by using the Walnut app. They also provide personal loans.

#### **Key Features of the App:**

1. Keep a close eye on your credit card balances.
2. Use BHIM UPI to send money.
3. Locate ATMs that accept cash near you in real-time.
4. Export your information and create expense reports (in PDF & CSV format).
5. Verify the balance of your bank account.
6. Keep track of train, cab, movie, and event reservations, among other things.
7. Search and share information about places you visit with friends and social networks.
8. Report your bank, card, or any other interesting messages directly from the app.

## **2.2References**

1. <https://www.moneytap.com/blog/best-money-management-apps/>
2. <https://relevant.software/blog/personal-finance-app-like-mint/>
3. <https://www.onmanorama.com/lifestyle/news/2022/01/11/financial-literacy-trend-among-todays-youth-investment.html>
4. <https://www.livemint.com/money/personal-finance/96-indian-parents-feel-their-children-lack-financial-know-how-survey-11661336110855.html>



## 2.3 Problem Statement Definition

Modern education does not focus on finance management. This is primarily due to lack of resources and the Indian value system on giving money to children. Failing to teach this valuable knowledge had left many Indians to recklessly spend their income and fall into vicious cycles of EMI and debt. Many of them are just a month's salary away from bankruptcy.

This issue is tackled by providing a web application for where people can plan their monthly expenses into categories, set alerts and get visual insights from their spending patterns.

1. Who does the problem affect?

Young adults and earning middle class citizens.

2. What is the issue?

Lack of financial literacy among people

3. When does the issue occur?

Primarily when the person moves from college to job and starts earning their own money.

4. Where is the issue occurring?

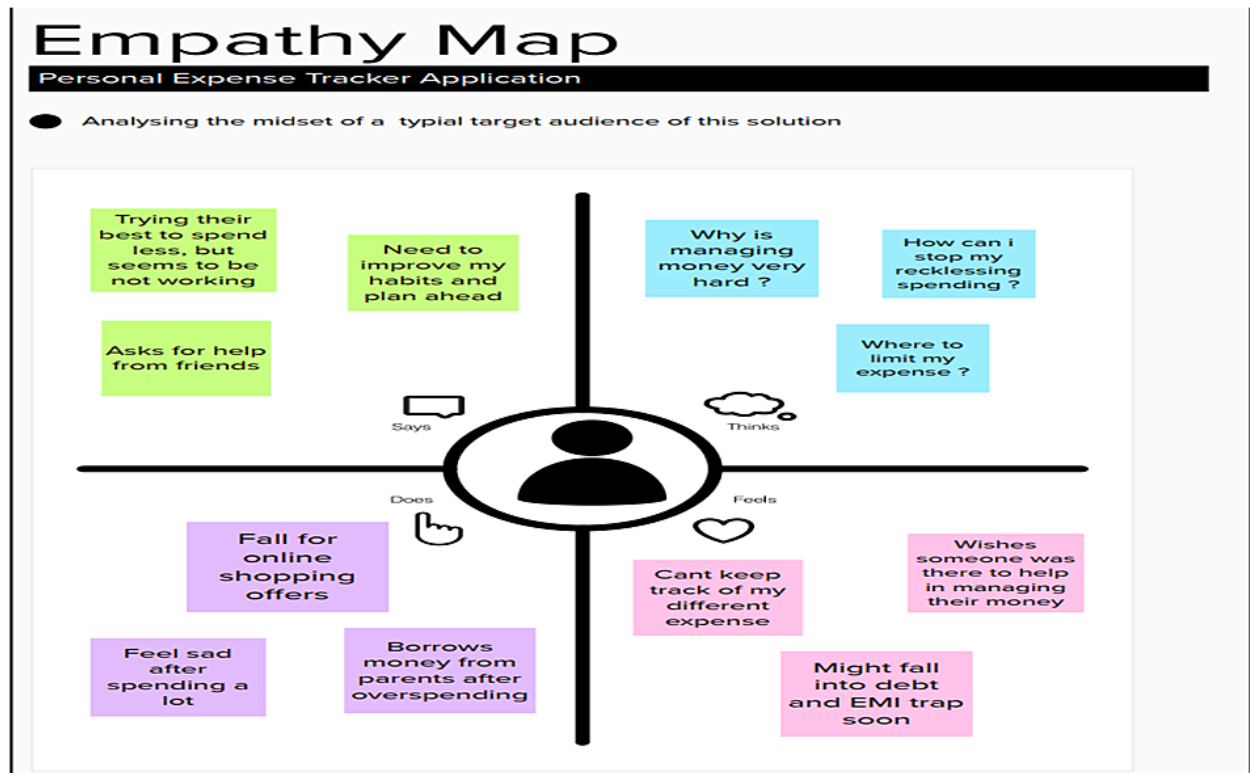
Especially among young engineers who are newly exposed to consumer centric market and services.

5. Why is it important that we fix the problem?

The recent BuyNowPayLater services and Credit apps have made people spend more than what they earn and repay.

## 3.IDEATION & PROPOSED SOLUTION

### 3.1 Empathy Map Canvas



### 3.2 Ideation & Brainstorming



### 3.3 Proposed Solution

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Building a personal finance tracking application that will imbibe good spending habits into students.
2.	Idea / Solution description	To build a web application that is deployed in IBM cloud and leverage mailing service like sendgrid to implement the same
3.	Novelty / Uniqueness	The stats generated with visual graphs are more effective than log books. It also helps in using technology to gain better insights from patterns.
4.	Social Impact / Customer Satisfaction	Better financial knowledge is gained. Gamified approach can be used to give self-satisfaction. Reduced chances of bad debt in future.
5.	Business Model (Revenue Model)	Subscription can be incorporated to access premium tools within the app.
6.	Scalability of the Solution	As the application is containerized for deployment. It can be easily scaled in a cloud service provider like IBM

### 3.4 Problem Solution fit

Define CS, fit into CC	<p><b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span></p> <p>Who is your customer?</p> <p>Predominantly Engineers who are just starting to earn and manage their personal finance. Typically from middle and lower class family, who badly need financial discipline .</p>	<p><b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span></p> <p>What constraints prevent your customers from taking action or limit their choices of solutions?</p> <p>The impulse buying and lacking to awareness to look into bigger picture</p>	<p><b>5. AVAILABLE SOLUTIONS</b> <span>AS</span></p> <p>Which solutions are available to the customers when they face the problem</p> <p>Totally shunning to spend even on necessities under the impression that the spending could result in bad financial position.</p> <p>The existing solutions are otherwise over complicated and designed to extract data from user.</p> <p>Manual physical logging in time consuming</p>	Explore AS, differentiate
------------------------	---	---	---	---------------------------

Focus on J&P, tap into BE, understand RC	<p><b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span></p> <p>Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.</p> <ul style="list-style-type: none"> <li>Logging expenses into categories</li> <li>Show historical stats</li> <li>Generate insightful charts</li> <li>Alert user to imbibe good discipline</li> </ul>	<p><b>9. PROBLEM ROOT CAUSE</b> <span>RC</span></p> <p>What is the real reason that this problem exists?</p> <p>Lack of proper education in financial literacy in school education. More children are not given pocket money to learn by spending/wasting less / saving.</p>	<p><b>7. BEHAVIOUR</b> <span>BE</span></p> <p>What does your customer do to address the problem and get the job done?</p> <p>Get frustrated and fall into debt traps by taking unpayable loans for unnecessary items leading to increase in mental stress</p>	Focus on J&P, tap into BE, understand RC
--	---	--	---	--

<p><b>3. TRIGGERS</b> <span>TR</span></p> <p>What triggers customers to act?</p> <p>Frequent sales in e-commerce platforms and seamless shopping experience online.</p> <p><b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span></p> <p>How do customers feel when they face a problem or a job and afterwards?</p> <p>Dejected and paranoid about the future as they would need relatively more money to provide for a family and to handle unexpected financial needs.</p>	<p><b>10. YOUR SOLUTION</b> <span>SL</span></p> <p>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.</p> <p>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 behavior.</p> <p>Graphical Application with simple UI and to the point clutter free objective. Avoids provision to pay through the app, to minimize the spending and ensure that only necessary spendings are made. The aim is to make the spending process harder throughout the application and keep it clean.</p>	<p><b>8.CHANNELS of BEHAVIOUR</b> <span>CH</span></p> <p><b>8.1 ONLINE</b></p> <p>What kind of actions do customers take online? Extract online channels from #7</p> <ol style="list-style-type: none"> <li>1. Shop from e-commerce</li> <li>2. Subscribe to OTT platforms</li> <li>3. Order food frequently</li> </ol> <p><b>8.2 OFFLINE</b></p> <p>What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.</p> <ol style="list-style-type: none"> <li>1. Shop in malls during sales</li> <li>2. Keep the money somewhere around and forget about /lose it</li> </ol>
---	---	--

## 4. REQUIREMENT ANALYSIS

### 4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Email/Signup Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Add expenses	Enter the everyday expenses Split it into categories (example: food, petrol, movies)
FR-4	Reminder mail	Sending reminder mail on target (for ex: if user wants a reminder when his/her balance reaches some amount (5000)) Sending reminder mail to the user if he/she has not filled that day's expenses.
FR-5	Creating Graphs	Graphs showing every day and weekly expenses. Categorical graphs on expenditure.
FR-6	Add salary	Users must enter the salary at the start of the month.
FR-7	Export CSV	User can export the raw data of their expenditure as CSV

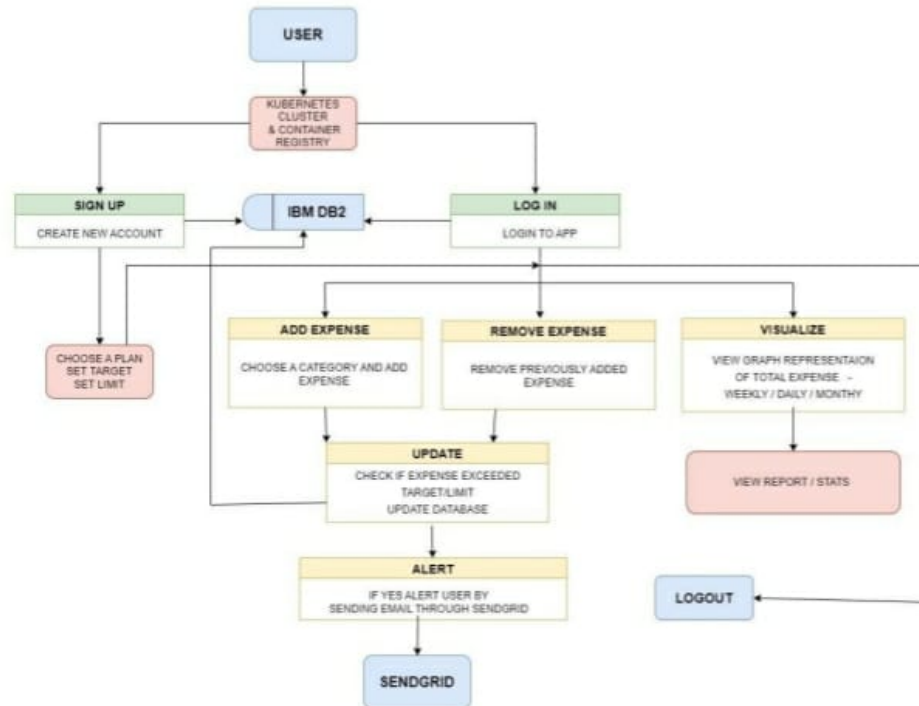
### 4.2 Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	A simple web application which is accessible across devices.
NFR-2	<b>Security</b>	The OAuth Google sign in and email login are secure with hashed and salted secure storage of credentials.
NFR-3	<b>Reliability</b>	Containerized service ensures that new instance can kick up when there is a failure.
NFR-4	<b>Performance</b>	The load is managed through the load balancer used with docker. Thus, ensuring good performance.
NFR-5	<b>Availability</b>	With load balancing and multiple container instances, the service is always available.
NFR-6	<b>Scalability</b>	Docker and Kubernetes are designed to accommodate scaling based on need

## 5.PROJECT DESIGN

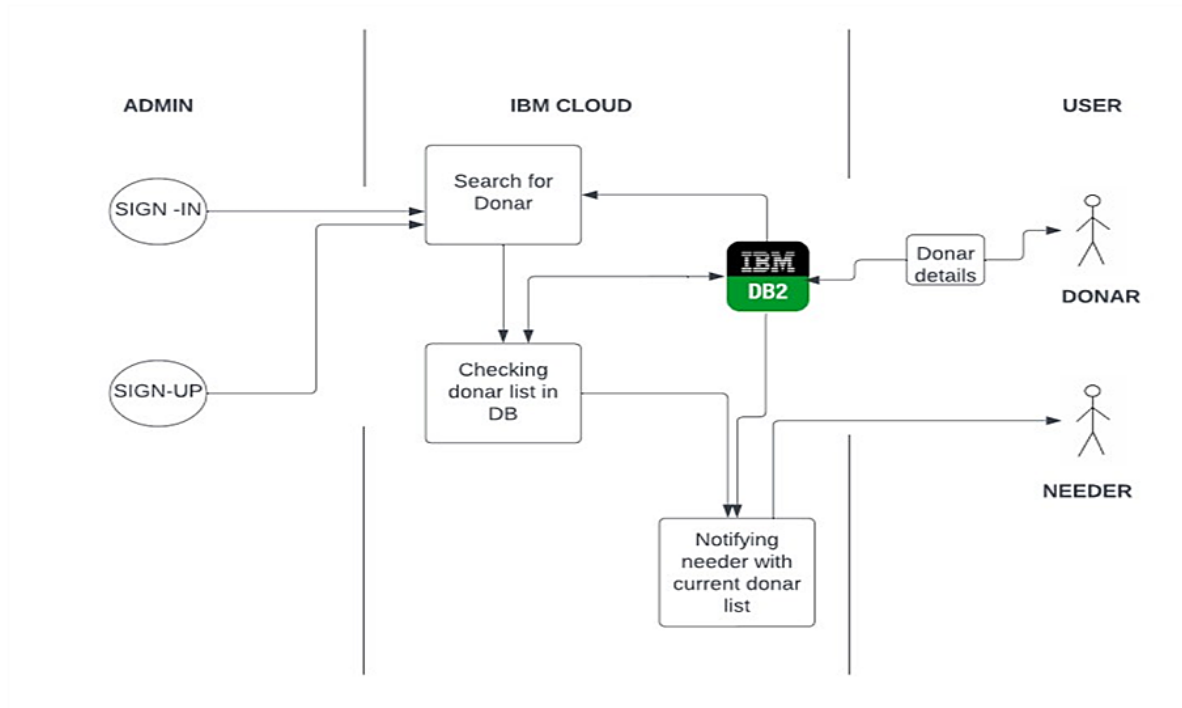
## 5.1 Data Flow Diagrams

Data Flow Diagram of Personal Expense Tracker: DFD Level 2



## 5.2 Solution & Technical Architecture

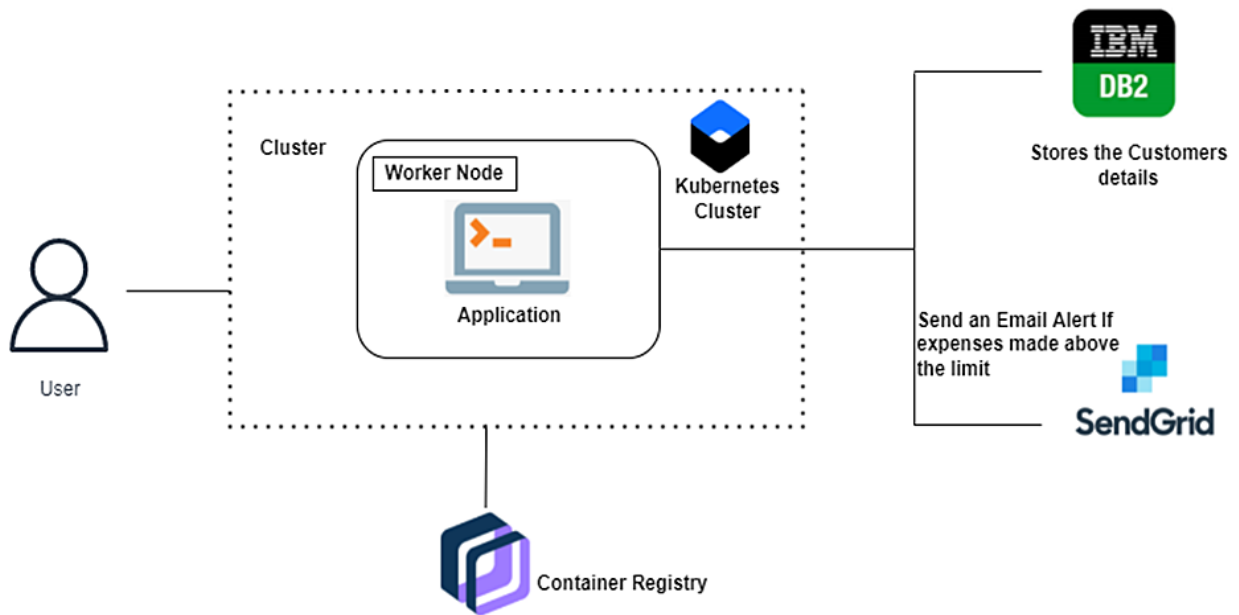




## 5.3 User Stories

User Type	Functional Requirement (Epic)	UserStory Number	User Story / Task
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
	Login	USN-2	As a user, I can log into the application by entering email & password
	Add	USN-3	As a user, I can add in new expenses.
	Remove	USN-4	As a user, I can remove previously added expenses.
	View	USN-5	As a user, I can view my expenses in the form of graphs and get insights.
	Getalert message	USN-6	As a user, I will get alert messages if I exceed my target amount.

Administrator	Add / remove user	USN-7	As admin, I can add or remove user details on db2 manually.
		USN-8	As admin, I can add or remove user details on sendgrid



## 6.PROJECT PLANNING & SCHEDULING

### **a.** Sprint Planning & Estimation

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, new password and confirming the same password.	2	High	Mohamed Nishath, Kishorkumar
		USN-2	As a user, I will receive confirmation email once I have registered for the application.	1	Low	
	Login	USN-3	As a user, I can log into the application by entering email and password / Google OAuth.	2	High	Mohan Prasath, Mohammed Shaheer

Sprint - 2	Dashboard	USN-4	Logging in takes the user to their dashboard.	1	Low	Mohamed Nishath
		USN-5	As a user, I will update my salary at the start of each month.	1	Medium	
		USN-6	As a user, I will set a target/limit to keep track of my expenditure.	1	Medium	Kishorkumar
	Workspace	USN-7	Workplace for personal expense tracking	1	Medium	Mohan Prasath
	Charts	USN-8	Graphs to show weekly and everyday expenditure	2	High	Mohammed Shaheer

		USN-9	As a user, I can export raw data as csv file.	1	Medium	Mohan Prasath
--	--	-------	---	---	--------	---------------

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
--------	-------------------------------	-------------------	-------------------	--------------	----------	--------------

Sprint - 3	IBM DB2	USN -10	Linking database with dashboard	2	High	Mohamed nishath
		USN -11	Making dashboard interactive with JS	2	High	Kishorkumar
	Watson Assistant	USN -12	Embedding Chatbot to clarify user's queries	1	Low	Mohammed shaheer
	BCrypt	USN -13	Using BCrypt to store passwords securely	1	Medium	Mohan prasath
	SendGrid	USN -14	Using SendGrid to send mail to the user. (To alert or remind)	1	Medium	Kishorkumar



Sprint - 4	Integration	USN -15	Integrating frontend and backend	2	High	Mohammed shaheer
	Docker	USN -16	Creating Docker image of web app	2	High	Mohan prasath
	Cloud Registry	USN -17	Uploading docker image to IBM cloud registry	2	High	Kishorkumar
	Kubernetes	USN -18	Creating container using docker and hosting the web app	2	High	Mohamed nishath
	Exposing Deployment	USN - 19	Exposing IP/Ports for the site	1	Medium	Mohan prasath

## 6. 2 Reports from JIRA

The screenshot displays the Jira interface for a project named 'Personnel expense tracker'. The top navigation bar includes the Jira logo, 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and a 'Create' button. A search bar is located on the right. The left sidebar shows the 'Projects' section with a 'RECENT' list containing 'Personnel expense tra...'. The main content area features a project board with three columns: 'TO DO' (1 item), 'IN PROGRESS' (3 items), and 'DONE' (3 items). Each column contains task cards with titles, checkboxes, and labels like PET-1 through PET-7. A 'Quickstart' button is visible in the bottom right corner.

**Project: Personnel expense tracker**

**Columns:**

- TO DO 1**
  - Project development phase
    - ☒ PET-7
- IN PROGRESS 3**
  - Deployment of app in cloud
    - ☒ PET-3
  - Ideation phase - 1
    - ☒ PET-4
  - Ideation phase - 2
    - ☒ PET-5
- DONE 3**
  - Setting up the project environment
    - ☒ PET-1
  - Integrating sendgrid service
    - ☒ PET-2
  - Project planning phase
    - ☒ PET-6

**Bottom Bar:** You're in a team-managed project | Give feedback | Quickstart



Projects

RECENT

Personnel expense tra...

View all projects

Personnel expense tracker

SummaryBoardListCalendarNEWTimelineFormsPagesIssuesReportsShortcutsProject settings

Good morning, Mohamed Nishath

Here's where you'll view a summary of Personnel expense tracker's status, priorities, workload, and more.

Project details

3 done

In the last 7 days

7 updated

In the last 7 days

7 created

In the last 7 days

0 due

In the next 7 days

Status overview

View the progress of your project based on the status of each item. For more details, go to the board view.

43%

Done

To Do

In Progress

Done

1

3

3

Total

7

Recent activity

Stay up to date with what's happening across the project.

TODAY

MN

Mohamed Nishath updated the Rank of PET-6 - Project planning phase

21 seconds ago

MN

Mohamed Nishath changed the status to Done on PET-6 - Project planning phase

21 seconds ago

MN

Mohamed Nishath changed the status to In Progress on PET-5 - Ideation phase - 2

28 seconds ago

MN

Mohamed Nishath changed the status to Done on PET-2 -

Priority breakdown

Get a holistic view of how work is being prioritized within your project. To check if the team's focusing on the right work, go to the list view.

4

3

2

1

0

Highest

High

Medium

Low

Lowest

Types of work

View the breakdown of items by their type. For more details, go to the list view.

Type

Distribution

Count

Task

Sub-task

Manage types

100%

0%

Team workload

Assign and share the workload in your team. Go to the list view.

Assignee

Work distribution

Count

Unassigned

Invite teammate

100%

4

Related projects

Use projects to manage all your work in one place and stay aligned with stakeholders.

Create a project

View all projects

Was the information shown in this page useful? Give us feedback

Quickstart

## 7.CODING

### 7.1 Html coding

#### Dashboard:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta charset="UTF-8" />
    <script src="https://cdn.tailwindcss.com"></script>
    <link
      href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
      rel="stylesheet"
    />
    <link href="../static/style.css" rel="stylesheet" />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
      href="https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Staatliches&display=swap"
      rel="stylesheet"
    />
    <title>Dashboard Page</title>
  </head>
  <body class="bg-gray-100">
    {% include 'navbar.html' %}
    <div class="text-center mt-10 mb-10">
      <h2 class="text-2xl">Hey there !... </h2>
      <p class="text-lg font-semibold">
        Welcome to the best platform to track your expense
      </p>
    </div>
```

```

<div>
  <div>
    <div class="flex">
      {% include 'addTransaction.html' %} {% include 'setTrigger.html' %}
    </div>
    {% include 'transactionList.html' %}
  </div>
  <div class="progress">
    <div class="font-bold text-2xl">Complete User Stats</div>
    <div class="w-[30vw] mt-4">
      <div>
        <div class="mb-1 text-lg font-medium text-black">
          Expense Progress
        </div>
        <div class="w-full h-6 bg-gray-300 rounded-full dark:bg-gray-700">
          <div
            class="h-6 bg-red-600 rounded-full dark:bg-red-500"
            style="width: {{percent}}%"
          ></div>
          <span class="mb-1 text-xs float-right pr-2 pt-2">{{left}} </span>
        </div>
      </div>
    </div>
  </div>
</div>
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "4256eb60-32b5-4dae-a877-7178860e8cd6", // The ID of this
integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "361cebc5-c33e-4124-b435-4e9ee8848612", // The ID of your
service instance.
    onLoad: function (instance) {
      instance.render();
    },
  };
  setTimeout(function () {
    const t = document.createElement("script");
    t.src =
      "https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +

```

```

        (window.watsonAssistantChatOptions.clientVersion || "latest") +
        "/WatsonAssistantChatEntry.js";
        document.head.appendChild(t);
    });
</script>
</body>
</html>

```

## Login:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta charset="UTF-8" />
  <script src="https://cdn.tailwindcss.com"></script>
  <link
    href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
    rel="stylesheet"
  />
  <title>Login Page</title>
</head>
<body class="bg-blue-100">
  <div
    class="bg-gradient-to-r from-purple-500 to-pink-500 min-h-screen flex flex-col"
  >
    <div
      class="container max-w-sm mx-auto flex-1 flex flex-col items-center justify-center px-2"
    >
      {% if error %}
      <div
        class="bg-red-100 border-red-200 border-2 rounded-md p-2 mt-2 mb-2 w-full text-center"
      >
        {{error}}
      </div>

```

```

{% endif %}
<form
  class="bg-white px-6 py-8 rounded shadow-md text-black w-full"
  method="POST"
>
  <h2 class="text-center font-bold text-4xl m-4">? Hey user ?</h2>
  <p class="text-center">Already a member lets</p>
  <h1 class="mt-4 mb-8 text-3xl text-center font-bold">Login</h1>
  <input
    type="text"
    class="block border border-gray-light w-full p-3 rounded mb-4"
    name="email"
    placeholder="Email"
  />

  <input
    type="password"
    class="block border border-gray-light w-full p-3 rounded mb-4"
    name="password"
    placeholder="Password"
  />

  <button
    type="submit"
    class="w-full text-center py-3 rounded bg-gradient-to-r from-violet-500 to-fuchsia-500 text-white hover:bg-gradient-to-l from-fuchsia-500 to-violet-500 hover:border-2 hover:border-rose-500 focus:outline-none my-1"
  >
    Login
  </button>
</form>

<div class="text-gray-dark mt-6 flex">
  Don't have an account?
  <a
    class="ml-2 no-underline border-b border-blue text-blue"
    href="/signup"
  >
    <p class="font-bold">Signup</p>
  </a>

```



```
    </div>
  </div>
</div>
</body>
</html>
```

## Signup:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta charset="UTF-8" />
    <script src="https://cdn.tailwindcss.com"></script>
    <link
      href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
      rel="stylesheet"
    />
    <title>Signup Page</title>
  </head>
  <body class="bg-blue-100">
    <div
      class="bg-gradient-to-r from-purple-500 to-pink-500 min-h-screen flex flex-col"
    >
      <div
        class="container max-w-sm mx-auto flex-1 flex flex-col items-center justify-center
px-2"
      >
        {% if error %}
        <div
          class="bg-red-100 border-red-200 border-2 rounded-md p-2 mt-2 mb-2 w-full text-
center"
        >
          {{error}}
        </div>
        {% endif %}
        <form
```

```

class="bg-white px-6 py-8 rounded shadow-md text-black w-full"
method="POST"
>
<h2 class="text-center font-bold text-4xl m-4">Hi there </h2>
<p class="text-center">New to PET lets</p>
<h1 class="mb-8 mt-4 text-3xl text-center font-bold">Sign up</h1>
<input
  type="text"
  class="block border border-gray-light w-full p-3 rounded mb-4"
  name="name"
  placeholder="Name"
/>

<input
  type="text"
  class="block border border-gray-light w-full p-3 rounded mb-4"
  name="email"
  placeholder="Email"
/>

<input
  type="phone"
  class="block border border-gray-light w-full p-3 rounded mb-4"
  name="phone"
  placeholder="Phone"
/>

<input
  type="password"
  class="block border border-gray-light w-full p-3 rounded mb-4"
  name="password"
  placeholder="Password"
/>

<button
  type="submit"
  class="w-full text-center py-3 rounded bg-gradient-to-r from-violet-500 to-fuchsia-
500 text-white hover:bg-gradient-to-l from-fuchsia-500 to-violet-500 hover:border-2
hover:border-rose-500 focus:outline-none my-1"
>

```

```

        Create Account
    </button>
</form>

<div class="text-gray-dark mt-6 flex">
    Already have an account?
    <a
        class="ml-2 no-underline border-b border-blue text-blue"
        href="/login"
    >
        <p class="font-bold">Log in</p>
    </a>
</div>
</div>
</div>
</body>
</html>

```

### Navbar:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta charset="UTF-8" />
    <script src="https://cdn.tailwindcss.com"></script>
    <link
        href="https://cdn.jsdelivr.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css"
        rel="stylesheet"
    />
    <title>Login Page</title>
</head>
<body class="bg-blue-100">
    <nav
        class="flex items-center justify-between flex-wrap bg-gradient-to-r from-purple-500
to-pink-500 p-6"
    >

```

```

<div class="flex items-center flex-shrink-0 text-white mr-6">
  <span class="font-semibold text-xl tracking-tight"
    >Personal Finance Tracker</span>
  </div>
<div class="w-full block flex-grow lg:flex lg:items-center lg:w-auto">
  <div class="text-sm lg:flex-grow">
    <a
      href="/dashboard"
      class="block mt-4 text-white lg:inline-block lg:mt-0 text-teal-200 mr-4"
    >
      Dashboard
    </a>
    <a
      href="/graph"
      class="block mt-4 text-white lg:inline-block lg:mt-0 text-teal-200 mr-4"
    >
      Graph
    </a>
    <a
      href="/configure"
      class="block mt-4 text-white lg:inline-block lg:mt-0 text-teal-200"
    >
      Configure
    </a>
  </div>
  <button
    onclick="window.location.href='logout'"
    type="submit"
    class="w-full float-right w-32 text-center py-3 rounded bg-blue-700 text-white
    hover:bg-blue-800 focus:outline-none"
  >
    Login Out
  </button>
</div>
</nav>
</body>
</html>

```

## 7.2 Python coding

**app.py:**

```
from flask import Flask, render_template, request, redirect, url_for, session
import requests
import flash
import ibm_db
from flask_login import UserMixin
from ibm_db import exec_immediate
from flask_session import Session
import re
import connect
```

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'abf6703a27964e61953307e963426a04'
```

```
dbname = "bludb"
username = "txs49042"
password = "cNEKMzATxRu70nLU"
hostname = "125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
cert = "DigiCertGlobalRootCA.crt"
port = 30426
protocol = "TCPIP"
conn = connect.conn
```

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

```
@app.route('/login', methods=['GET', 'POST'])  
  
def login():  
    if request.method == 'POST':  
        name = request.form['name']  
        password = request.form['password']  
        sql = "SELECT * FROM registration WHERE name=? AND password=?, (name,  
password)"  
        stmt = ibm_db.exec_immediate(conn, sql)  
        res = conn.fetch_assoc(stmt)  
        return render_template('login.html')
```

```
@app.route('/signup', methods=["GET", "POST"])  
  
def Signup():  
    if request.method == 'POST':  
        arr = []  
        try:  
            name = request.form['name']
```

```
password = request.form['password']
phone = request.form['phone']
email = request.form['email']

sql = "insert into registration(name, email, password, phone) values (?, ?, ?, ?),
(name, email, password, phone)"

stmt = ibm_db.exec_immediate(conn, sql)
conn.fetch_assoc(stmt)

flash("Record added successfully", "success")

except:

    flash("Error in insert", "danger")

finally:

    return redirect(url_for("dashboard"))

    conn.close()

return render_template("signup.html")
```

```
@app.route('/dashboard')

def dashboard():

    return render_template("dashboard.html")
```

```
@app.route('/graph')

def graph():

    return render_template("graph.html")
```

```
@app.route('/transactionList')
def transactionList():
    return render_template("transactionList.html")
```

```
@app.route('/addTransactionList')
def addTransactionList():
    return render_template("addTransaction.html")
```

```
@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for("login"))
```

```
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=5000, debug=True)
```

### **connect.py:**

```
from flask import Flask, render_template, request, redirect, url_for, session
import requests
import flash
import ibm_db
```



```
from flask_login import UserMixin
from ibm_db import exec_immediate
from flask_session import Session
import re
import connect

app = Flask(__name__)
app.config['SECRET_KEY'] = 'abf6703a27964e61953307e963426a04'

dbname = "bludb"
username = "txs49042"
password = "cNEKMzATxRu70nLU"
hostname = "125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
cert = "DigiCertGlobalRootCA.crt"
port = 30426
protocol = "TCPIP"
conn = connect.conn

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

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        name = request.form['name']
        password = request.form['password']
        sql = "SELECT * FROM registration WHERE name=? AND password=?, (name,
password)"
        stmt = ibm_db.exec_immediate(conn, sql)
        res = conn.fetch_assoc(stmt)
        return render_template('login.html')
```

```
@app.route('/signup', methods=["GET", "POST"])
def Signup():
    if request.method == 'POST':
        arr = []
        try:
            name = request.form['name']
            password = request.form['password']
            phone = request.form['phone']
            email = request.form['email']
            sql = "insert into registration(name, email, password, phone) values (?, ?, ?, ?),
(name, email, password, phone)"
            stmt = ibm_db.exec_immediate(conn, sql)
            conn.fetch_assoc(stmt)
```

```
        flash("Record added successfully", "success")
    except:
        flash("Error in insert", "danger")
    finally:
        return redirect(url_for("dashboard"))
        conn.close()
return render_template("signup.html")
```

```
@app.route('/dashboard')
def dashboard():
    return render_template("dashboard.html")
```

```
@app.route('/graph')
def graph():
    return render_template("graph.html")
```

```
@app.route('/transactionList')
def transactionList():
    return render_template("transactionList.html")
```

```
@app.route('/addTransactionList')
```

```
def addTransactionList():  
    return render_template("addTransaction.html")
```

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

```
def logout():  
    session.clear()  
    return redirect(url_for("login"))
```

```
if __name__ == '__main__':  
    app.run(host="0.0.0.0", port=5000, debug=True)
```

**conn.py:**

```
import ibm_db  
  
from ibm_db import tables  
from ibm_db import fetch_assoc  
from ibm_db import exec_immediate  
from app import db  
  
dbname = "bludb"  
username = "txs49042"  
password = "cNEKMzATxRu70nLU"  
hostname = "125f9f61-9715-46f9-9399-  
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"  
cert = "DigiCertGlobalRootCA.crt"
```

```
port = 30426
```

```
protocol = "TCPIP"
```

```
# Establish connection
```

```
def establish():
```

```
    try:
```

```
        conn = db.connect(
```

```
f"DATABASE={dbname};HOSTNAME={hostname};PORT={port};PROTOCOL={protocol};UID={username};PWD={password}; SECURITY=SSL; SSLServerCertificate={cert};",
```

```
        "", "")
```

```
        print("Connected to database")
```

```
        return conn
```

```
    except:
```

```
        print("Error connecting to database")
```

```
# To insert a new user
```

```
def insertuser(conn1, name, email, user, passw):
```

```
    sql = "INSERT INTO users(name,email,username,password) VALUES ('{}','{}','{}','{}').format(
```

```
        name, email, user, passw)
```

```
    try:
```

```
stmt = db.exec_immediate(conn1, sql)
print("Number of affected rows: ", db.num_rows(stmt))
except:
    print("cannot insert user to database")
```

# To check if user exists with given email

```
def useremail_check(conn, email):
    sql = "SELECT * FROM users WHERE email='{}' ".format(email)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_assoc(stmt)
    if results == False:
        return True
    else:
        return False
```

# To check if user exists with given username and password

```
def user_check(conn, email, passwd):
    sql = "SELECT * FROM users WHERE email='{}' AND password='{}' ".format(
        email, passwd)
    stmt = db.exec_immediate(conn, sql)
    results = db.fetch_both(stmt)
```

return results

# Set basic details of each user

```
def setuser(conn, money, budget, goal, email, pwd):
```

```
    sql = "UPDATE USERS SET(pocketmoney,budget,monthlygoal) = ('{}','{}','{}') WHERE  
email='{}' AND password='{}'".format(
```

```
        money, budget, goal, email, pwd)
```

```
    try:
```

```
        stmt = db.exec_immediate(conn, sql)
```

```
        print("Number of affected rows: ", db.num_rows(stmt))
```

```
    except:
```

```
        print("Error inserting data to database")
```

# Insert a new transaction of a user

```
def inserttransac(conn, id, amt, des, cat):
```

```
    sql = "INSERT INTO TRANSACTIONS(user_id,amount,description,category)  
VALUES('{}','{}','{}','{}')".format(
```

```
        id, amt, des, cat)
```

```
    try:
```

```
        stmt = db.exec_immediate(conn, sql)
```

```
        print("Number of affected rows: ", db.num_rows(stmt))
```

```
    except:
```

```
print("Error inserting data to database")
```

```
# To get the total mount spent for the month
```

```
def gettotalsum(conn, id):
```

```
    sql = "SELECT SUM(amount) as SUM FROM transactions WHERE MONTH(date) =  
MONTH(CURRENT DATE) AND YEAR(date) = YEAR(CURRENT DATE) AND  
user_id='{}'.format(id)
```

```
    try:
```

```
        stmt = db.exec_immediate(conn, sql)
```

```
        res = db.fetch_both(stmt)
```

```
        return res
```

```
    except:
```

```
        print("Error while fetching")
```

```
# To get all transactions of a user
```

```
def getalltransac(conn, id):
```

```
    sql = "SELECT * FROM transactions WHERE MONTH(date) = MONTH(CURRENT DATE)  
AND YEAR(date) = YEAR(CURRENT DATE) AND user_id='{}' ORDER BY date  
DESC".format(id)
```

```
    try:
```

```
        stmt = db.exec_immediate(conn, sql)
```

```
        res = db.fetch_both(stmt)
```



```

if (res == False):
    return []
else:
    res['DATE'] = res['DATE'].date()
    dict = [{'id': res['ID'], 'date': res['DATE'], 'amt': res['AMOUNT'],
            'cat': res['CATEGORY'], 'des': res['DESCRIPTION']}]
    res = db.fetch_both(stmt)
    while (res != False):
        res['DATE'] = res['DATE'].date()
        dict.append({'id': res['ID'], 'date': res['DATE'], 'amt': res['AMOUNT'],
                    'cat': res['CATEGORY'], 'des': res['DESCRIPTION']})
        res = db.fetch_both(stmt)
    return dict
except:
    print("Error while fetching")

```

# To delete a transaction

```

def deletetrans(conn, id):
    sql = "DELETE FROM transactions WHERE id={}".format(id)
    try:
        stmt = db.exec_immediate(conn, sql)
        print("Number of affected rows: ", db.num_rows(stmt))
    except:

```

```
print("Error deleting data from database")
```

```
# To update a transaction
```

```
def updateTrans(conn, id, amt, des):
```

```
    sql = "UPDATE transactions SET AMOUNT='{}',DESCRIPTION = '{}' WHERE  
ID='{}' ".format(
```

```
        amt, des, id)
```

```
    try:
```

```
        stmt = db.exec_immediate(conn, sql)
```

```
        print("Number of affected rows: ", db.num_rows(stmt))
```

```
    except:
```

```
        print("Successfully updated transaction")
```

```
def get_budget(conn, id):
```

```
    sql = "SELECT POCKETMONEY FROM users WHERE ID='{}' ".format(id)
```

```
    stmt = db.exec_immediate(conn, sql)
```

```
    results = db.fetch_assoc(stmt)
```

```
    return results
```

**utils.py:**

```
import uuid

from connect import execDB, execReturn

positive_money = ['Salary', 'Festival Bonus']

negative_money = ['EMI', 'Food', 'Transportation', 'Groceries',
                  'Clothing', 'Electronic', 'Entertainment', 'Rent', 'Vacations']
```

```
def addUser(name, email, password):
```

```
    print(name, email, password)
```

```
    sql_fd = f"SELECT * FROM user WHERE email='{email}'"
```

```
    r = execReturn(sql_fd)
```

```
    if r != []:
```

```
        return "Email Exists"
```

```
    sql_st = f"INSERT INTO user(name , email , password ) values ( '{name}' , '{email}' , '{password}' )"
```

```
    r = execDB(sql_st)
```

```
    return "User registered successfully"
```

```
def getPassword(email):
```

```
    sql_fd = f"SELECT password FROM user WHERE email='{email}'"
```

```
    r = execReturn(sql_fd)
```

```
    # print(r[0])
```

```
return r[0]['PASSWORD'].strip()
```

```
def fetchFinanceRecord(email):
```

```
    sql_fd = f"SELECT * FROM finance WHERE email='{email}' order by date desc"
```

```
    r = execReturn(sql_fd)
```

```
    return r
```

```
def getIncomeExpend(email):
```

```
    sql_fd1 = f"SELECT SUM(AMOUNT) FROM finance WHERE AMOUNT>0 AND  
email='{email}'"
```

```
    sql_fd2 = f"SELECT SUM(AMOUNT) FROM finance WHERE AMOUNT<0 AND  
email='{email}'"
```

```
    r1 = execReturn(sql_fd1)
```

```
    r2 = execReturn(sql_fd2)
```

```
    print(r1, r2)
```

```
    return {"income": r1[0]['1'], "expend": -r2[0]['1']}
```

```
def createFinanceRecord(email, category, amount, description, date):
```

```
    amount = int(amount)
```

```
    if category in negative_money:
```

```
        amount = -amount
```

```
    print("FINANCE", email, amount, category, description, date)
```

```
    sql_st = f"INSERT INTO finance(id,email , amount , category , description , date ) values
```

```
( '{uuid.uuid1()}', '{email}', {amount}, '{category}', '{description}', '{date}' )"
r = execDB(sql_st)
print(r)
return "Record created successfully"
```

### credentials.py:

```
{
  "connection": {
    "cli": {
      "arguments": [
        [
          "-u",
          "txs49042",
          "-p",
          "cNEKMzATxRu70nLU",
          "--ssl",
          "--sslCAFile",
          "1cbbb1b6-3a1a-4d49-9262-3102a8f7a7c8",
          "--authenticationDatabase",
          "admin",
          "--host",
          "125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426"
        ]
      ],
    },
  },
}
```

```
"bin": "db2",

"certificate": {

    "certificate_base64":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURFakNDQWZxZ0F3SUJBZ0lKQVA1S0R
3ZTNCTkxiTUeWRONTcUdTSWlZRFFFQkN3VUFNQjR4SERBYUJnTIYKQkFNTUUwbENUU0JE
Ykc5MVpDQkVZWVJoWW1GelpYTXdlaGNOTWpBd01qSTVNRFF5TVRBeVdoY05NekF3TW
pJMgpNRFF5TVRBeVdqQWVNUnd3R2dZRFZRUUREQk5KUWswZ1EyeHZkV1FnUkdGMFIX
SmhjMlZ6TUIlQklqQU5CZ2txCmhraUc5dzBCQVFFRkFBTONBUThBTUIlQkNnS0NBUEVBd
XUvbitpWW9xdkdGNU8xSGpEalpsK25iYjE4UkR4ZGwKTzRUL3FoUGMxMTREY1FUK0pIRX
dhdG13aGljTGxaQnF2QWFMb1hrbmhqSVFOMG01L0x5YzdBY291VXNmSGR0QwpDVGcr
SUsxbjBrdDMrTHM3d1dTakxqVE96N3M3MlZUSU5yYmx3cnRIRUlzM1JWTKv6SkNHYW5L
SXdZMWZVSUtrCldNMlR0SDI5cnFsSGN0Z2pIUlFmRkVTRmlYaHJiODhSQmd0amlva0xtVG
pCaTFBeEVadWNobWZ2QVRmNEN0Y3EKY21QcHNqdDBPTnI0YnhJMVRyUWxEemNiN1h
MSFBrWW91SUprdnVzMUZvaTEySmRNM1MrK3labFZPMUZmZkU3bwpKMjhUdGJoZ3JG
OGtIU0NMSkVjTTF5Sz3FPZG90Vm5QOC9EOWZhamNNN0lWd2V4a0lSOTNKR1FJREFRQUJ
vMU13CIVUQWRCZ05WSFE0RUZnUVVlQ3JZanFJQzc1VUpxVmZEMDh1ZWdqeDZiUmN3S
HdZRFZSMGpCQmd3Rm9BVWVDclkKanFJQzc1VUpxVmZEMDh1ZWdqeDZiUmN3RHdZRF
ZSMFRBUUgVqkFVd0F3RUlvekFOQmdrcWhraUc5dzBCQVfzRgpBQU9DQVFFQUkyRTBUO
Ut3MIN3RjJ2MXBqaHV4M0lkWWV2SGFVSkRMb0tPd0hSRnFSOHgxZ2dRcGVFcFBnMk5S
Ckx3R08yek85SWZUMmhLaWd1d2orWnJ5SGxxcHlxQ0pLOHJEU28xZUVPEklyWmE2S1Yr
QTVscEttMWdjv3VHYzMkk1UrVTFzTDdlUjd3ZFFuVjU0TVU4aERvNi9sVHRMRVB2Mnc3VI
NPSIFDK013ejgrTFJMdjVHSW5BNlJySWNhKwozM0wxNnB4ZEttd1pLYThWcnBnMXJ3QzR
nY3dIYUhyMUNEWE42K0JlbzhvWG5YWkh6UG91cldYS1BoaGdXZ2J5CkNDcUdIK0NWNn
Q1eFg3b05NS3VNSUNqRVZndnNLWnRqeTQ5VW5iNVZZbHQ0b1J3dTFlbGdzRDNjekltbjlL
REQKNHB1REFvYTZyMktZZE4xVksuN3F3VG1TbDITU05RPT0KLS0tLS1FTkQgQ0VSVElGSU
NBVEUtLS0tLQo=",

    "name": "1cbbb1b6-3a1a-4d49-9262-3102a8f7a7c8"

},

"composed": [

    "db2 -u txs49042 -p cNEKMzATxRu70nLU --ssl --sslCAFile 1cbbb1b6-3a1a-4d49-
9262-3102a8f7a7c8 --authenticationDatabase admin --host 125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426"

],
```

```
"environment": {},
"type": "cli"
},
"db2": {
  "authentication": {
    "method": "direct",
    "password": "cNEKMzATxRu70nLU",
    "username": "txs49042"
  },
  "certificate": {
    "certificate_base64":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURFakNDQWZxZ0F3SUJBZ0lKQVA1S0R3ZTNCTkxiTUEwR0NTcUdTZWl3RFFFQkN3VUFNQjR4SERBYUJnTIYKQkFNTUwEENUU0JEYkc5MVpDQkVZWFFJoWW1GelpYTXdlaGNOTWpBd01qSTVNRF5TVRBeVdoY05NekF3TWpJMgpNRFF5TVRBeVdqQWVNUnd3R2dZRFZRUUREQk5KUWswZ1EyeHZkV1FnUkdGMFIXSmhjMlZ6TUlJQklqQU5CZ2txCmhraUc5dzBCQVFFRkFBT0NBUThBTUIJQkNnS0NBUEVBdXUvbitpWW9xdkdGNU8xSGpEalpsK25iYjE4UkR4ZGwKTzRUL3FoUGMxMTREY1FUK0pIRXdhdG13aGljTGxaQnF2QWFMb1hrbmhqSVFOMG01L0x5YzdBY291VXNmSGR0QwpDVGcrSUxsbjBrdDMrTHM3d1dTakxqVE96N3M3MlZUSU5yYmx3cnRIRUlvM1JWTKV6SkNHYW5LSXdZMWZVSUtrClNMIR0SDl5cnFsSGN0Z2plUjFmRkVTRmlYaHJiODhSQmd0amlva0xtVGpCaTFBeEVadWNobWZ2QVRmNEN0Y3EKY21QcHNqdDBPTnI0YnhJMVRyUWxEemNiN1hMSFBrWW91SUprdnVzMUZvaTEySmRNM1MrK3labFZPMUZmZkU3bwpKMjhUdGJoZ3JGOGtIU0NMSkVjTTF5Z3FPZG90Vm5QOC9EOWZhamNNN0lWd2V4a0lSOTNKR1FJREFRQUJvMU13CIVUQWRCZ05WSFE0RUZnUVVlQ3JZanFJQzc1VUpxVmZEMDh1ZWdqeDZiUmN3SHdZRFZSMGpCQmd3Rm9BVWVDclkKanFJQzc1VUpxVmZEMDh1ZWdqeDZiUmN3RHdZRFZSMFRBUUgVqkFVd0F3RUlvekFOQmdrcWhraUc5dzBCQVFFZGpBQU9DQVFFQUkyRTBUOUt3MIN3RjJ2MXBqaHV4M0lkWWV2SGFVSkRmb0tPd0hSRnFSOHgxZ2dRcGVFcFBNMk5SCkx3R08yek85SWZUMmhLaWd1d2orWnJ5SGxxcHlxQ0pLOHJEU28xZUVPEklyWmE2S1YrQTVscEttMWdjv3VHYzMkk1UrVTFzTDDlUjd3ZFFuVjU0TVU4aERvNi9sVHRMRVB2Mnc3VINPSIFDK013ejgrTFJMdjVHSW5BNlJySWNkKwozM0wxNnB4ZEttd1pLYThWcnBnMXJ3QzRnY3dIYUhYMUNEWE42K0JlbzhvWG5YWkh6UG91cldYS1BoaGdXZ2J5CkNDcUdIK0NWNnQ1eFg3b05NS3VNSUNqRVZndnNLWnRqeTQ5VW5iNVZZbHQ0b1J3dTFlbGdzRDNjekltbjJL
```

REQKNHB1REFvYTZyMktZZE4xVkxuN3F3VG1TbDITU05RPT0KLS0tLS1FTkQgQ0VSVEIGSU  
NBVEUtLS0tLQo=",

"name": "1cbbb1b6-3a1a-4d49-9262-3102a8f7a7c8"

},

"composed": [

"db2://txs49042:cNEKMzATxRu7OnLU@125f9f61-9715-46f9-9399-  
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb?authSo  
urce=admin&replicaSet=replset"

],

"database": "bludb",

"host\_ros": [

"125f9f61-9715-46f9-9399-  
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31048"

],

"hosts": [

{

"hostname": "125f9f61-9715-46f9-9399-  
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud",

"port": 30426

}

],

"jdbc\_url": [

"jdbc:db2://125f9f61-9715-46f9-9399-  
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb:user=<  
userid>;password=<your\_password>;sslConnection=true;"

],

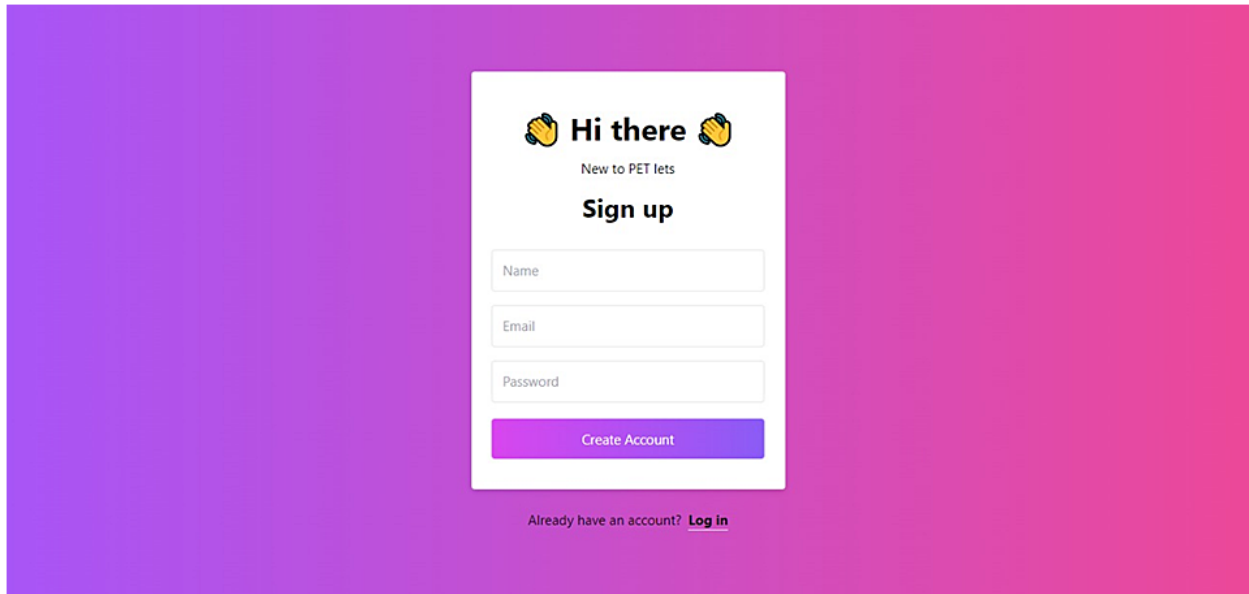
"path": "/bludb",



```
"query_options": {
  "authSource": "admin",
  "replicaSet": "replset"
},
"replica_set": "replset",
"scheme": "db2",
"type": "uri"
}
},
"instance_administration_api": {
  "deployment_id": "crn:v1:bluemix:public:dashdb-for-transactions:us-
south:a/985cff4f31bc43048994b3f3b9558779:cdb71d18-4cbf-4e72-99fd-
acdd7d81c646::",
  "instance_id": "crn:v1:bluemix:public:dashdb-for-transactions:us-
south:a/985cff4f31bc43048994b3f3b9558779:cdb71d18-4cbf-4e72-99fd-
acdd7d81c646::",
  "root": "https://api.db2.cloud.ibm.com/v5/ibm"
}
}
```

## 8.RESULTS

## 8.1 Signup



The image shows a signup form centered on a background with a purple-to-pink gradient. The form is a white card with rounded corners. At the top, it says "Hi there" with a clapping hands emoji on either side, followed by "New to PET lets" in a smaller font. Below this is the heading "Sign up". There are three input fields: "Name", "Email", and "Password". At the bottom of the form is a purple button with the text "Create Account". Below the form, there is a link that says "Already have an account? [Log in](#)".

Hi there 🖐️🖐️

New to PET lets

**Sign up**

Name

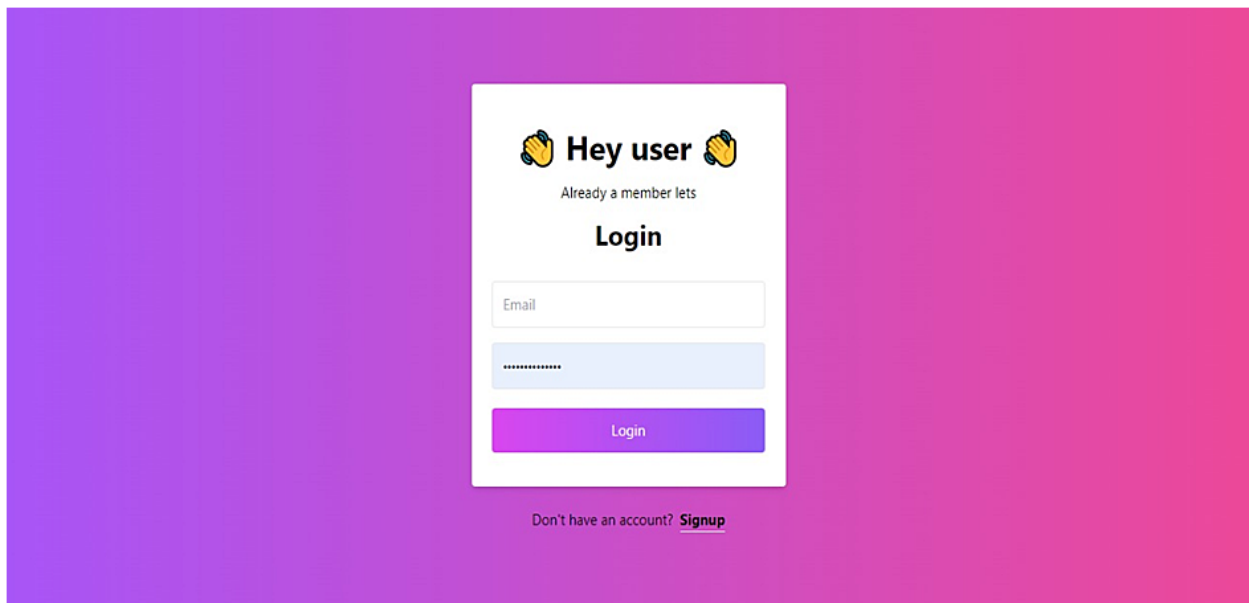
Email

Password

Create Account

Already have an account? [Log in](#)

## 8.2 Login



The image shows a login form centered on the same purple-to-pink gradient background. The form is a white card with rounded corners. At the top, it says "Hey user" with a clapping hands emoji on either side, followed by "Already a member lets" in a smaller font. Below this is the heading "Login". There are two input fields: "Email" and a password field represented by a light blue rectangle with a series of dots. At the bottom of the form is a purple button with the text "Login". Below the form, there is a link that says "Don't have an account? [Signup](#)".

Hey user 🖐️🖐️

Already a member lets

**Login**

Email

\*\*\*\*\*

Login

Don't have an account? [Signup](#)

## 8.3 Screen layout

Dashboard Page

127.0.0.1:5000/dashboard

Maps News Gmail Translate YouTube Fiverr / nishath\_nic... PayPal: Wallet Amazon.in Associat... ProductionCrate ~... How to Become a F... Other bookmarks

Personal Finance Tracker

Dashboard Graph Configure

Login Out

Hey there !...

Welcome to the best platform to track your expense

Add one from below:

Amount

475

Category

EMI

Description

Write description here...

Add

Added expenses will pop here

2022/11/26 Salary

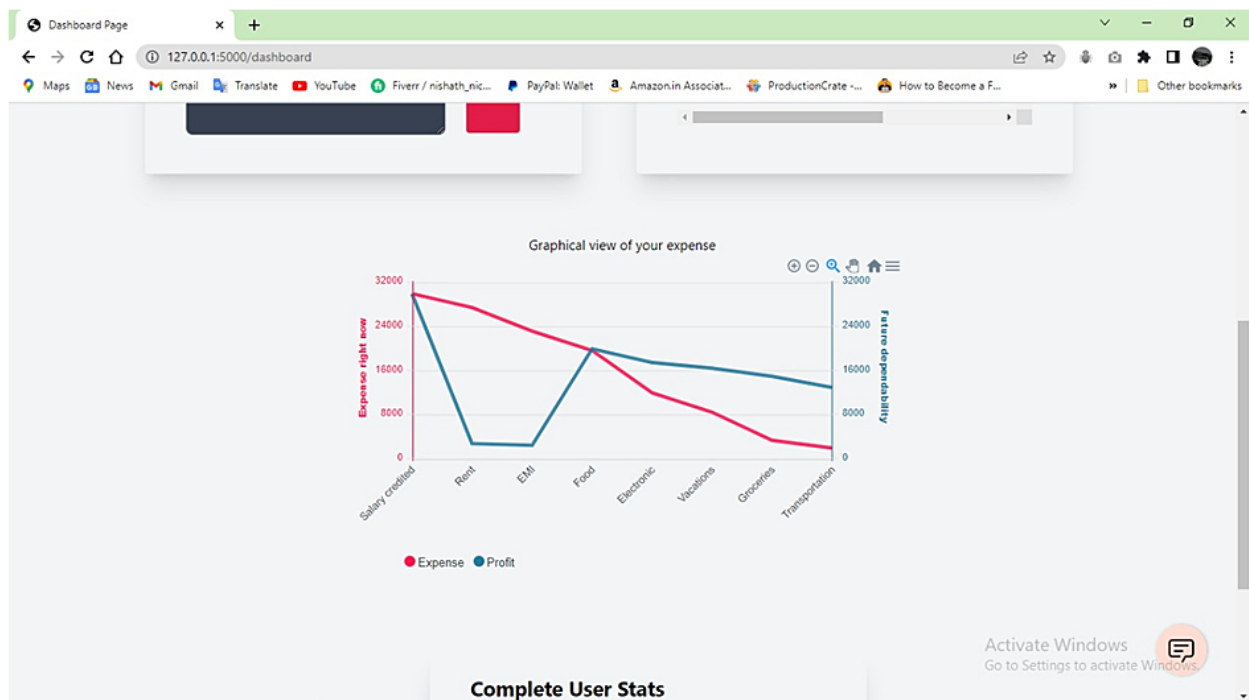
2022/11/26 EMI

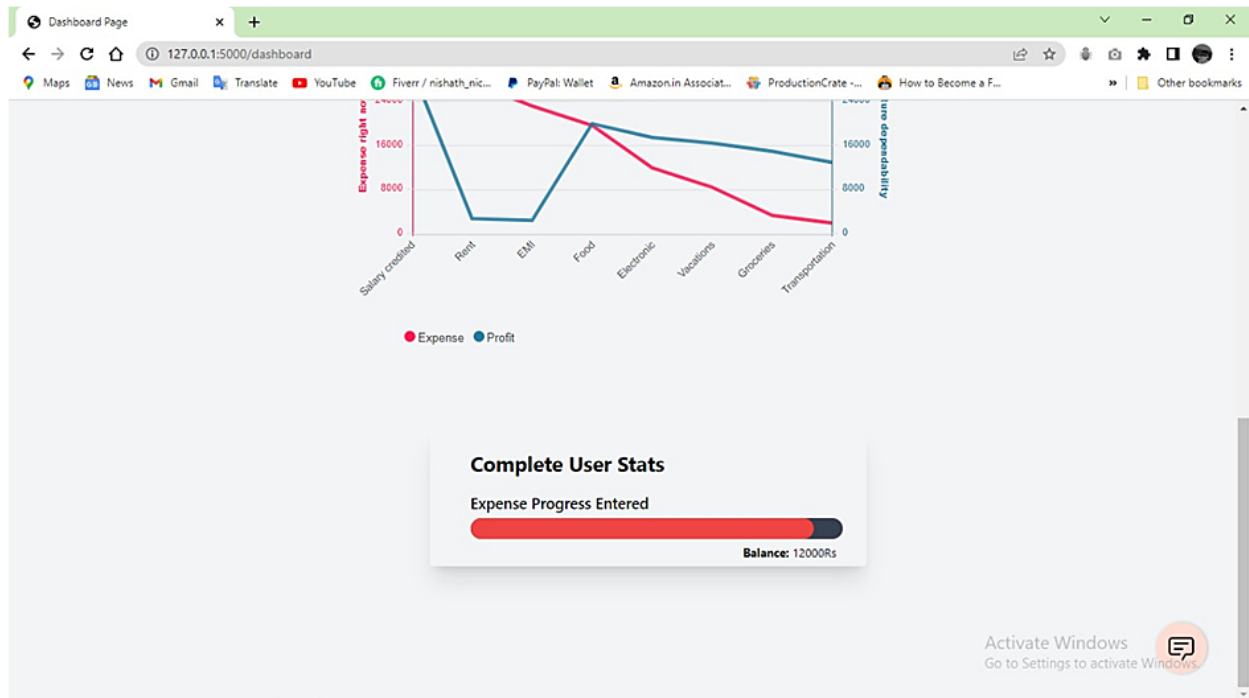
2022/11/26 Rent

2022/11/26 Vacations

Activate Windows  
Go to Settings to activate Windows

Graphical view of your expense





## 8.4 Assistant

**Personal Finance Tracker** | Dashboard | Graph | Configure

Hey there !...  
Welcome to the best platform to track your expense

**Add one from below:**

Amount:  Category:

Description:

Added expenses will pop here

- 2022/11/26
- 2022/11/26
- 2022/11/26
- 2022/11/26

**Chat Assistant:**

Hi! I'm a virtual assistant. How can I help you today?

- I want to check my expences
- I want an financial mentor
- Example: See how I can help

Built with IBM Watson®

## **9.ADVANTAGES**

- Puts You Back in Control It's a terrible feeling not knowing where your money goes.
- Identifies Budget Categories to Re-evaluate.
- Shows You Your Bad Spending Habits.
- Keeps You From Overspending.
- Helps You Avoid Debt.
- Helps You Cut Out Impulse Spending.
- Keeps You Focused on Your Goals.
- Reduces Stress.

## **10.CONCLUSION**

If you are not using an expense tracker, you are missing out on the ability to manage your finances wisely and effortlessly. You will end up spending money without even realising it, and your daily expenses will go through the roof. On the other hand, if you use a money manager app, you will be aware when and why you are spending money and how much you spend.