PERSONAL EXPENSE TRACKER

PROJECT REPORT

Submitted by

HARSHINE M (19EUCS044)

JANANI A (19EUCS048)

HARINE M (19EUCS039)

ARTHIK S (19EUCS014)

in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE

(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI

MAY 2022

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)
(Approved by AICTE and Affiliated to Anna University, Chennai)
ACCREDITED BY NAAC WITH "A" GRADE

BONAFIDE CERTIFICATE

Certified that this project report titled **Personal Expense Tracker"** is the bonafide work of **HARSHINE M (19EUCS009), JANANI G (19EUCS054), HARINE B (19EUCS055), ARTHIK S M (19EUCS045)** who carried out the project work under mysupervision.

SIGNATURE	SIGNATURE
Dr.K. SASI KALA RANI, M.E., Ph.D.,	**Dr.R.GOWTHAMANI, M.E., Ph.D.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**

Department of Computer Science and Engineering
Sri Krishna College of Engineering and Technology
Kuniamuthur,
Coimbatore

This project report is submitted for the Autonomous Project Viva-Voce examination held on …….....

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere thanks to the management and **Dr.J.JANET, M.E.,Ph.D.,** Principal, Sri Krishna College of Engineering and Technology, Coimbatore for providing us the facilities to carry out this project work.

We are highly indebt to **Dr.K. SASIKALA RANI, M.E.,Ph.D.,** Head of Computer Science and Engineering for her continuous evaluation, valuable suggestions and comments given during the course of the project work.

We are thankful to **Dr.R.GOWTHAMANI, M.E., Ph.D.,,** Project Co-ordinator**,** Department of Computer Science and Engineering for her continuous evaluation, valuable suggestions and comments given during the course of the project work.

We express our deep sense of gratitude to our guide**,** Professor in the department of Computer science and Engineering for her valuable advice, guidance and support during the course of our project work.

By this, we express our heartfelt sense of gratitude and thanks to our beloved parents, family and friends who have all helped in collecting the resources and materials needed for this project and for their support during the study and implementation this project.

TABLE OF CONTENTS

CHAPTER 1

1. INTRODUCTION:

In today's busy and expensive lives we are in a great rush to make money. But at the end of the day we broke off. As we are unknowingly spending money on little and unwanted things. So, we have come over with the idea to track our earnings. Daily Expense Tracker (DET) aims to help everyone who are planning to know their expenses and save from it. DTE is a website in which user can add expenses on daily basis and its table will get generated and at the end based on user expenses report will be generated. User can select date range to calculate his/her expenses come over with the idea to track our earnings. Personal Expense Tracker aims to help everyone who are planning to know their expenses and save from it. Personal Expense Tracker is a website in which user can add expenses on daily basisand at the end, based on user expenses report will be generated. User can select date range to calculate his/her expenses.

1.1 Project Overview :

This website is used to track expenses and control spending beyond limits. while input data of expenses in website, we must select category which spent on and additionally notes can be used to note the details of expenses. By entering those record we can track our expenses. we can generate reports in graphical, pie chat. We can also set limits to particular category which alerts in email when the limits exceed.

1.2 Purpose :

At end of certain period, users does not know where they spent their money and they spend more on needless expenses beyond budgets which leads to financial crisis. To avoid this people needs to track their expenses. While calculating in diary requires lot of manual calculation and lot of time. This is the purpose to go for website application to track expenses.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem :

People can't able to track their expenses and spending more on unnecessary expenses which leads to money crisis. Without tracking people can't know whether they exceed the limit of their budget. Diary notes requires lots of manual calculation and It reduces the interest to track expenses. User frustrated about they can't remember where their money goes and can't handle their cash flow. There is no alerting system aboutexceeding limits.

There can be many disadvantages of using a manual accounting system. Accounting, for any business, can be a complex undertaking. A manual accounting system requires you to understand the accounting process in a way that may be unnecessary with a computerized accounting system. This can be an advantage or a disadvantage, depending on the person doing the bookkeeping; often, a specially trained professional is needed to ensure that accounting is done properly. Unrevealing the complexity of your financial records by hand may be time consuming. Since it takes time to generate reports.

2.2 References:

- https://ijirt.org/Article?manuscript=150860
- https://www.researchgate.net/publication/347972162_Expense_Manager_Application
- https://ijarsct.co.in/Paper391.pdf
- https://irejournals.com/paper-details/1702687
- https://www.irjet.net/archives/V6/i3/IRJET-V6I31110.pdf
- https://www.youneedabudget.com/

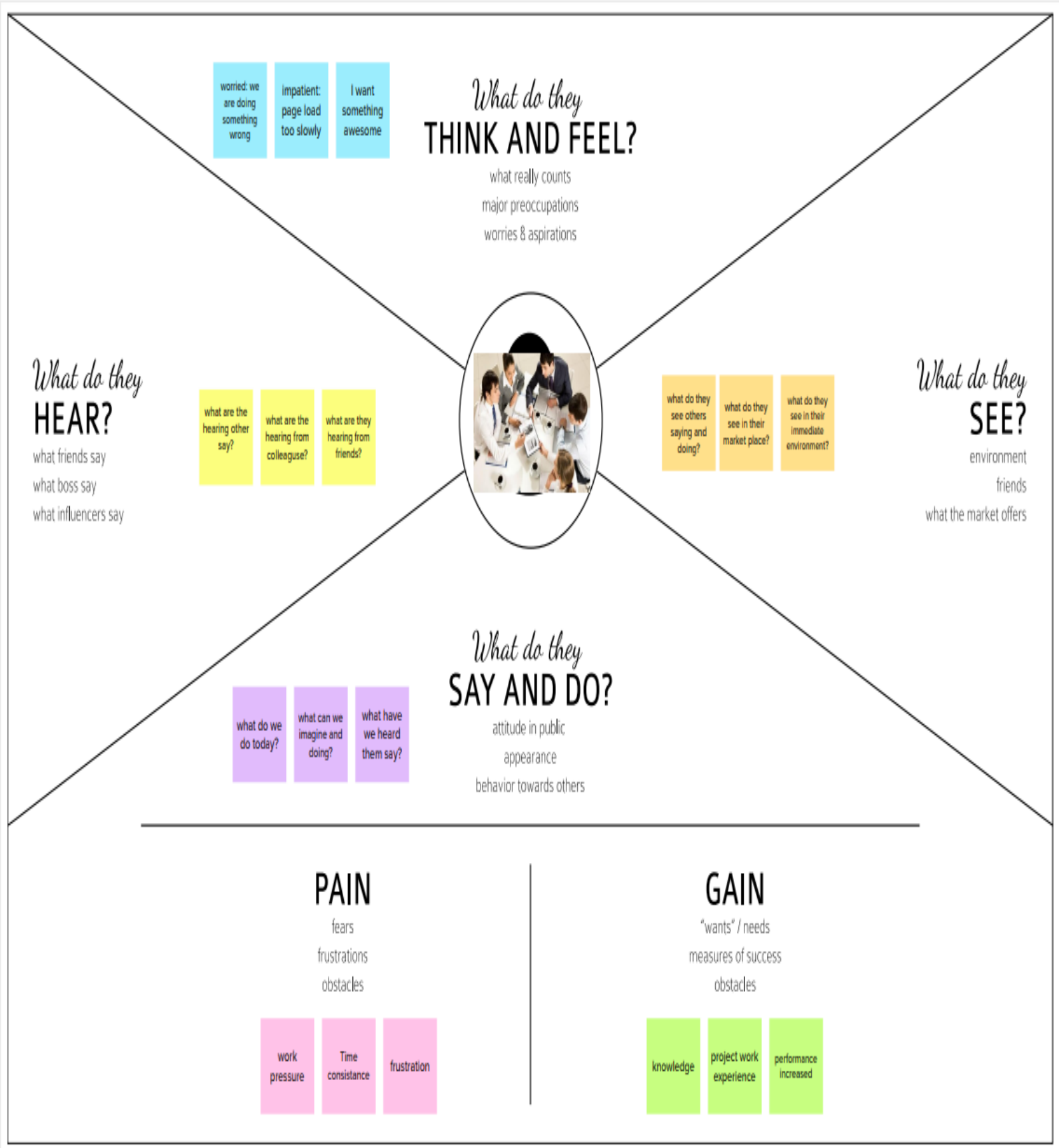
2.3 Problem Statement Definition :

Our project helps the user to keep track their expenses and determine whether they are spending as per their set budget. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Which allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts. It is like automated diary which requires no burden of manual calculation and enables the user to not just keep the control on the expenses but also to generate and save reports. Users can insert and delete transactions. We can compare with past expenses. Customized email alerts are used alerts user when limit exceeds.

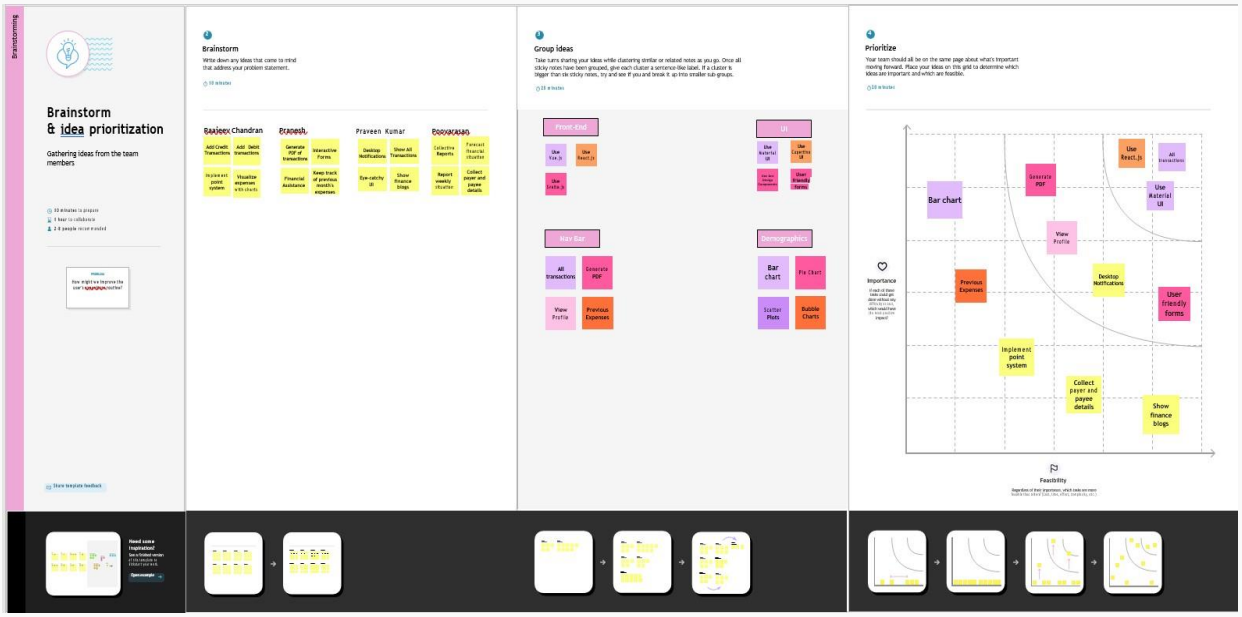
CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1Empathy Map Canvas



3.2 IDEATION & BRAINSTORMING :



3.3 PRO POSED SOLUTION :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	People can’t able to track their expenses and spending more on unnecessary expenses which leads to money crisis. People forget to pay dues on time, sometimes this leads tofine. Diary notes requires lots of manual calculation and It reduces the interest to trackexpenses.
2.	Idea / Solution description	Our project helps the user to keep track their expenses and determine whether they are spending as per their set budget. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Which allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts. User forgotten to input records can be avoided by remainders and alerts are helps to pay dues on time. It is like automated diary which

		requires no burden of manual calculation and enables the user to not just keep the control on the expenses but also to generate and save reports. Users can insert and delete transactions. We can compare with past expenses.
3.	Novelty / Uniqueness	We can set budgets for particular category to track unwanted expenses. we can generate reports as pdf for specific category. Budget setting feature leads people to overconsume some goods, under consume others and control over spending beyond limits.
4.	Social Impact / Customer Satisfaction	This solution controls users on overspending and reduces money crisis due to unwanted expenses. As this tracking expense becomes a habit, people can get a good picture of how much money they need to maintain their lifestyle. Tracking helps people to feel confidence on finance.
5.	Business Model (Revenue Model)	Revenue can be generated by placing advertisement.
6.	Scalability of the Solution	A Future update shall have payment option were we can pay dues and subscription. Linking Bank accounts and also tracking shares. It can be scaled for all types of people from any type of field.

3.4 Problem Solution fit :

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

Purpose:

- Solve complex problems in a way that fits the state of your customers.
- Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.

- ❑ Sharpen your communication and marketing strategy with the right triggers and messaging.
- ❑ Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
- ❑ **Understand the existing situation in order to improve it for your target group.**

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Be aware of daily expenditures	*Enter amount spent*
FR-2	Generate visually appealing charts	*Notify users periodically to update their expenses*
FR-3	Categorize credit and debit transactions	*Always looks for credit/debit threshold*
FR-4	Prompt to not exceed the threshold amount	*Send email alerts if the user is on the verge of exceeding the threshold*
FR-5	Show ways to minimize expense in the most spentarea	*Constantly look for patterns from previous expenses to improve accuracy*

4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

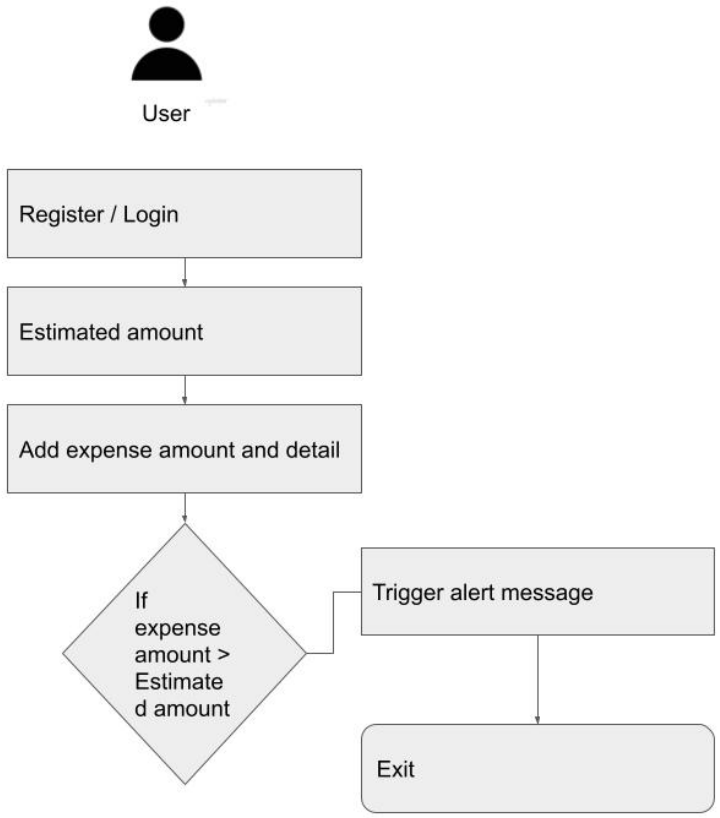
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	*The UI/UX must be visually appealing and pleasing to the senses with proper placements of primitive elements.*
NFR-2	**Security**	*Completely safe and private as user's data is neither shared nor utilized for any other secondary purposes.*
NFR-3	**Reliability**	*The application is guaranteed to give non-erroneous results at most instances.*
NFR-4	**Performance**	*The application is entirely robust to handle the incoming traffic even if there occurs an unexpected surge.*
NFR-5	**Availability**	*The application does not fail to keep track of the expenses that have been entered*

CHAPTER 5

PROJECT DESIGN :

5.1 Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the informationflows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2 Solution Architecture and Technical Architecture :

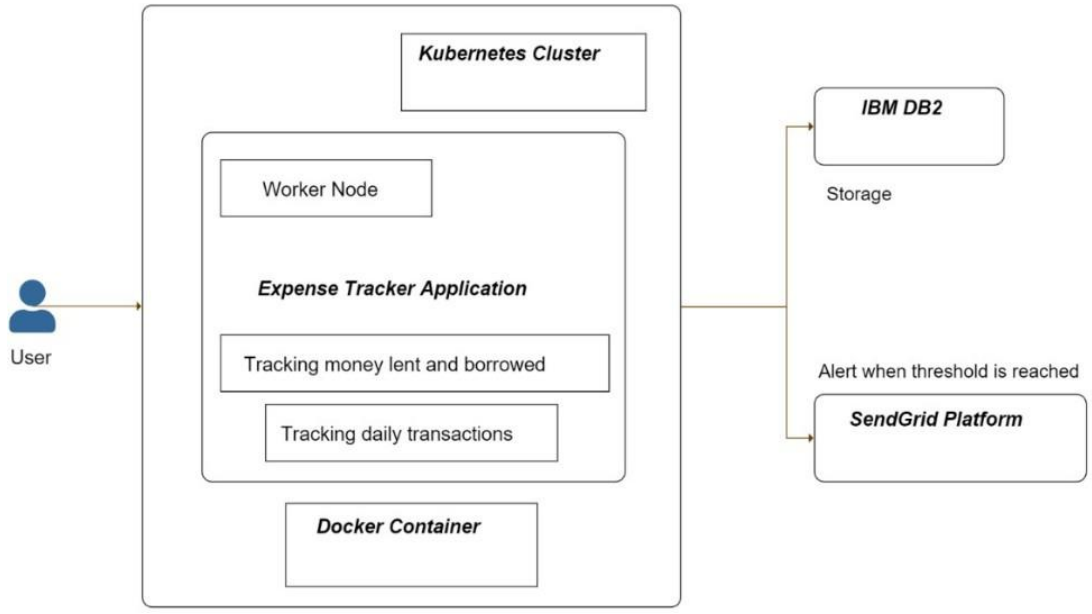
Solution Architecture :

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.

- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Example - Solution Architecture Diagram:



TECHNICAL ARCHITECTURE:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

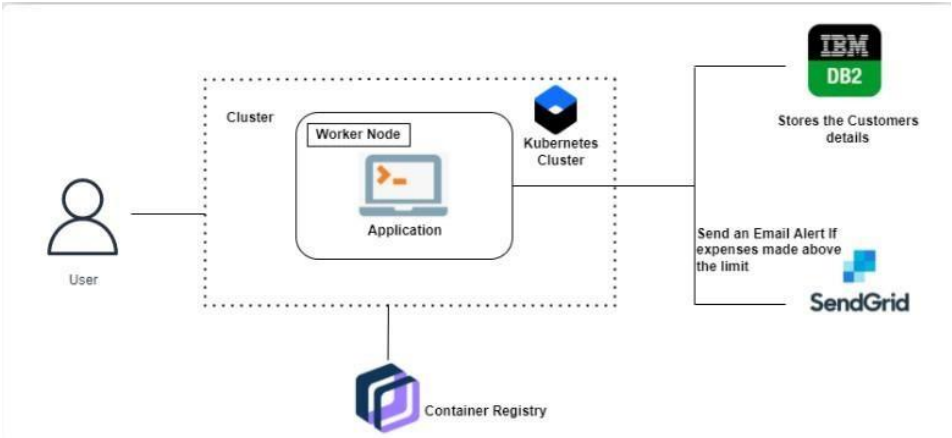


Table-1 : Components & Technologies:

S.No	Component	Technology
1.	User Interface	HTML
2.	Application Logic-1	Python
3.	Application Logic-2	IBM DB2
4.	Microservice	SendGrid

Table-2: Application Characteristics:

S.No	Characteristics	Technology
1.	Open-Source Frameworks	Flask
2.	Performance	It can handle about 100 requests per second

5.3 User Stories :

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my email, password, mobile number, weekly expense, montly salary	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Landing page		As a user, I can view my entire expenses throughout a particular period of time	I can view my expenses	High	Sprint-1
			As a user, I can generate reports based on my previous expenditures.	Report is successfully generated	Medium	Sprint-2
			As a user, I can logout	Successfully logout	High	Sprint-1
			As a user, I can create expense	Expense is successfully added	High	Sprint-1
			As a user, I can edit ,delete, update expense	The corresponding action is made to the expense	High	Sprint-1
			As a user, I can view credit and	The expenses are filtered accordingly	Medium	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story /Task	Acceptance criteria	Priority	Release
			debit expenses separately.			
			As a user, I can set a minimum threshold for my total expenditure either each week or month.	Minimum threshold is set successfully	High	Sprint-1
			As a user, I can view graphically interpreted insights of my expenditures.	Demographics of the expenses are generated	High	Sprint-1
			As a user, I can be aware of the expense that I spend the most on	Know my weak points that prevents user from saving more	Low	Sprint-3

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation :

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create product backlog and sprint 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, mobile number, weekly expense, montly salary	3	High	HARSHINE JANANI
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password	3	High	JANANI
Sprint-1	Landing page	USN-3	As a user, I can view my entire expenses throughout a particular period of time	5	High	ARTHIK
Sprint-2		USN-4	As a user, I can generate reports based on my previous expenditures	5	Medium	HARSHINE
Sprint-2	Logout	USN-5	As a user, I can logout	4	High	HARINE
Sprint-2	Dashboard	USN-6	As a user, I can create expense	6	Medium	JANANI
Sprint-		USN-7	As a user, I can	2	High	ARTHIK

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
2			edit ,delete, update expense			
Sprint-3		USN-8	As a user, I can view credit and debit expenses separately.	6	High	HARSHINE
Sprint-3		USN-9	As a user, I can set a minimum threshold for my total expenditure either each week or month.	6	Low	HARINE
Sprint-3		USN-10	As a user, I can view graphically interpreted insights of my expenditures	6	High	HARINE
Sprint-4		USN-11	As a user, I can be aware ofthe expense that I spend the most on	4	High	JANANI
Sprint-4		USN-12	As a user, I can be able to update my set monthly limit	10	High	ARTHIK HARINE
Sprint-4		USN-13	As a user, I can able to viewmy profile	20	High	HARSHINE JANANI

6.2 Sprint Delivery Schedule :

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

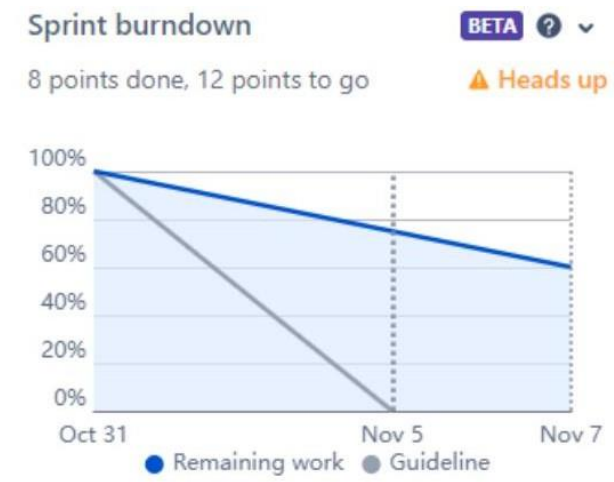
Velocity:

Velocity is a metric that predicts how much work an Agile software development team can successfully complete within a two-week sprint (or similar time-boxed period). Velocity is a useful planning tool for estimating how fast work can be completed and how long it will take to complete a project

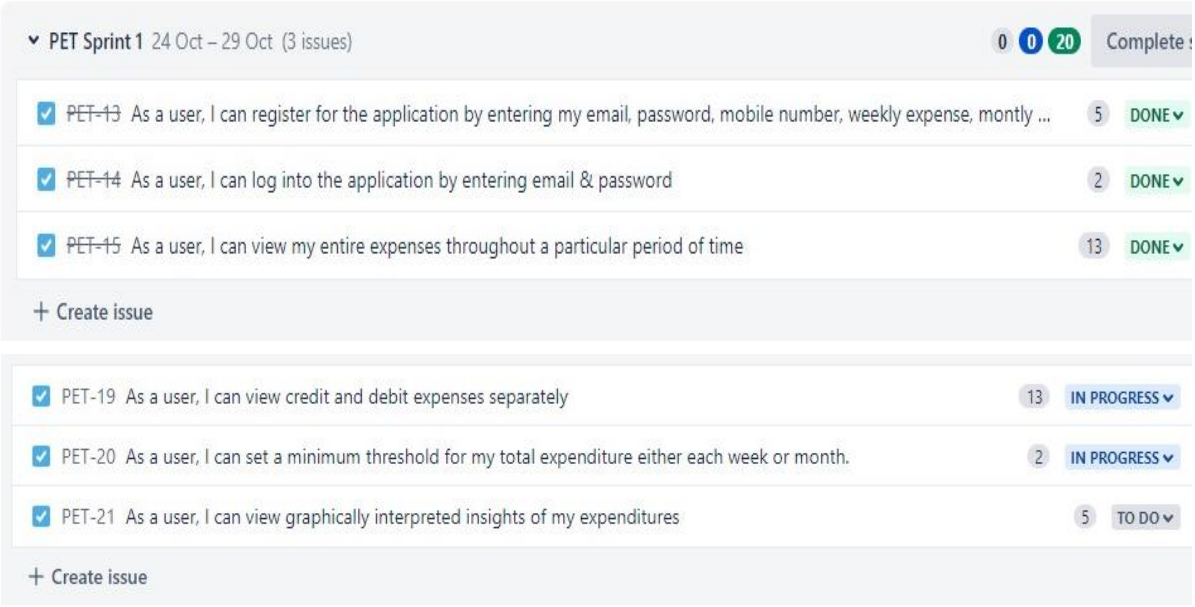
Average velocity = Total story points/ No. of iterations = 80/4 = 20

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6.3 Reports from JIRA :



CHAPTER 7

CODING AND SOLUTIONING

7.1 FRONTEND

7.1.1 add_expense_model.jsx:

This is the model which is used for adding new expenses.

```
import "./add_expense_modal.css";
import React, { useState, useEffect } from "react";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import useStore from "../../state";
const AddExpenseModal = ({ handleClose, show }) => {
  const showHideClassName = show ? "modal display-block" : "modal display-none";

  const addExpense = useStore(state => state.addExpense)
  const fetchExpenditureBreakdown = useStore(state => state.fetchExpenditureBreakdown)

  const userId = useStore(state => state.userId)

  const [formData, setFormData] = useState({
    date: "",
    amount: "",
    category: "Food",
    description: "",
    expenseType: "debit",
  });

  let categories = [
    "Food",
    "Automobiles",
    "Entertainment",
    "Clothing",
    "Healthcare",
    "Others",
  ];
  let expenseTypes = ["debit", "credit"];

  const [creatingExpense,setCreatingExpense] = useState(false)

  // const onCategoryChange = (e) => {
```

```
//  setFormData({
//     ...formData,
//     'category': e.target.value,
//  })
//  console.log(formData)
// };

// const onExpenseTypeChange = (e) => {
//  setFormData({
//     ...formData,
//     'expense_type': e.target.value.toLowerCase(),
//  })
//  console.log(formData)

// };

const updateFormData = (event) =>
   setFormData({
      ...formData,
      [event.target.name]: event.target.value,
   });

const { amount, description, date } = formData;

const showErrorToast = (text) => {
   toast.error(text, {
      position: "bottom-right",
      autoClose: 5000,
      hideProgressBar: false,
      closeOnClick: true,
      pauseOnHover: false,
      draggable: true,
      progress: undefined,
      theme: "light",
   });
};

const handleSubmit = async(event) => {
   try{
      event.preventDefault()
      setCreatingExpense(true)
      await addExpense(formData,userId,fetchExpenditureBreakdown)
      setCreatingExpense(false)
      console.log(formData)
      handleClose()
   }catch(e){
      showErrorToast("Something went wrong")
```

```
    }
  };

  return (
    <div className={showHideClassName}>
      <div className="modal-main">
        <div className="close-button-container">
          <button className="close-button" onClick={handleClose}>
            <i className="fa fa-solid fa-close" />
          </button>
        </div>

        <div className="form">
          <h1>Create Expense</h1>
          <form onSubmit={handleSubmit}>
            <h6 style={{ margin: "1em", fontWeight: "bold" }}>Date</h6>
            <input
              value={date}
              className="textfield"
              onChange={(e) => updateFormData(e)}
              type="date"
              name="date"
              required
            />
            <h6 style={{ margin: "1em", fontWeight: "bold" }}>
              {"Amount (₹)"}
            </h6>

            <input
              value={amount}
              className="textfield"
              onChange={(e) => updateFormData(e)}
              type="number"
              name="amount"
              required
            />

            <h6 style={{ margin: "1em", fontWeight: "bold" }}>Category</h6>
                        <select className="textfield" name="category" onChange={e =>
updateFormData(e)}>
              {categories.map((n) => (
                <option className="dropdown-item" value={n}>
                  {n}
                </option>
              ))}
            </select>
```

```
                <h6 style={{ margin: "1em", fontWeight: "bold" }}>Expense Type</h6>
                        <select className="textfield" name="expenseType" onChange={e =>
updateFormData(e)}>
                  {expenseTypes.map((n) => (
                    <option className="dropdown-item" value={n}>
                      {n.charAt(0).toUpperCase() + n.slice(1)}
                    </option>
                  ))}
                </select>

                <h6 style={{ margin: "1em", fontWeight: "bold" }}>Description</h6>
                <input
                  value={description}
                  className="textfield"
                  onChange={(e) => updateFormData(e)}
                  type="text"
                  name="description"
                  // required
                />

                <div className="create-expense-button-container">
                  <button className="create-expense-button" type="submit">{
                      creatingExpense ? "Creating" : "Create"
                  }</button>
                </div>
              </form>
            </div>
          </div>
        </div>
    );
};
export default AddExpenseModal;
```

7.1.2 expenditure_breaksown.jsx

This is the model through which we can see detailed analysis of our expenses

```
import React, { useEffect, useState } from "react";
import "bootstrap/dist/css/bootstrap.css";
import "../../pages/home/home.css";
import LoadingDots from "../loader/loading_dots";
import useStore from "../../state";

const ExpenditureBreakdown = () => {
  const userId = useStore(state => state.userId)
  const expenditureBreakdown = useStore(state => state.expenditureBreakdown)
```

```
                                const fetchingExpenditureBreakdown =
useStore(state=>state.fetchingExpenditureBreakdown)
  const fetchExpenditureBreakdown = useStore(state => state.fetchExpenditureBreakdown)
  const gradients = [
    "card-purple-blue",
    "card-salmon-pink",
    "card-blue-green",
    "card-purple-pink",
  ];

  useEffect( () => {
    async function fetch(){
        if(!fetchingExpenditureBreakdown && Object.keys(expenditureBreakdown).length
=== 0){
        await fetchExpenditureBreakdown(fetchingExpenditureBreakdown,userId)
      }
    }

     fetch();
  });

  return (
    <div className="container-fluid">
      <div className="row row-cols-4">
        {Object.entries(expenditureBreakdown)
          .slice(0, 4)
          .map(([key, val], i) => (
            <div className="col-12 col-sm-6 col-md-3">
              <div
                className={`card ${gradients[i % 4]} text-white mb-3 mb-md-0`}
              >
                  <div className="card-body d-flex justify-content-between align-items-
end">
                    <div className="card-number">
                      <div className="h3 m-0">
                        { `₹ ${val === null ? '0' : val.toLocaleString('en-US')}`}
                      </div>
                      <small>
                        <strong>
                          {key === null ? '-'
                          : key
                            .replace(/([A-Z])/g, " $1")
                            .charAt(0)
                            .toUpperCase() +
                            key.replace(/([A-Z])/g, " $1").slice(1)}
                        </strong>
                      </small>
                    </div>
```

```
                    <div className="card-description text-right">
                      <small> </small>
                      <br />
                      <small> </small>
                    </div>
                  </div>
                </div>
              </div>
            ))}
          {Object.entries(expenditureBreakdown)
            .slice(4)
            .map(([key, val], i) => (
              <div className="col-12 col-sm-6 col-md-3 mt-3">
                <div
                  className={`card ${gradients[i % 4]} text-white mb-3 mb-md-0`}
                >
                    <div className="card-body d-flex justify-content-between align-items-
end">
                    <div className="card-number">
                      <div className="h3 m-0">
                        {
                          key === "mostSpentOn" || key == "leastSpentOn" ?
                          val : (val === undefined ? '-' : `₹ ${val.toLocaleString('en-US')}`)
                        }

                      </div>
                      <small>
                        <strong>
                          {key
                            .replace(/([A-Z])/g, " $1")
                            .charAt(0)
                            .toUpperCase() +
                            key.replace(/([A-Z])/g, " $1").slice(1)}
                        </strong>
                      </small>
                    </div>
                    <div className="card-description text-right">
                      <small> </small>
                      <br />
                      <small> </small>
                    </div>
                  </div>
                </div>
              </div>
            ))}
        </div>
      </div>
```

```
    );
};

export default ExpenditureBreakdown;
```

7.1.3 expense_charts.jsx

This is the model for creating charts for easily analysing the category wise expense details

```
import React, { PureComponent } from "react";
import {
  Radar,
  RadarChart,
  PolarGrid,
  Legend,
  PolarAngleAxis,
  PolarRadiusAxis,
  ResponsiveContainer,
  PieChart,
  Pie,
  Sector,
  Cell,
} from "recharts";

const data = [
  {
    subject: "Food",
    A: 120,
    B: 110,
    fullMark: 3000,
  },
  {
    subject: "Automobiles",
    A: 98,
    B: 130,
    fullMark: 3000,
  },
  {
    subject: "Entertainment",
    A: 86,
    B: 130,
    fullMark: 3000,
  },
  {
    subject: "Clothing",
```

```
    A: 99,
    B: 100,
    fullMark: 3000,
  },
  {
    subject: "Healthcare",
    A: 85,
    B: 90,
    fullMark: 3000,
  },
  {
    subject: "Others",
    A: 65,
    B: 85,
    fullMark: 3000,
  },
];

const pieData = [
  { name: "Group A", value: 200 },
  { name: "Group B", value: 300 },
  { name: "Group C", value: 300 },
  { name: "Group D", value: 200 },
  { name: "Group E", value: 600 },
  { name: "Group F", value: 200 },
];

const COLORS = ["#0088FE", "#00C49F", "#FFBB28", "#FF8042"];

const RADIAN = Math.PI / 180;
const renderCustomizedLabel = ({
  cx,
  cy,
  midAngle,
  innerRadius,
  outerRadius,
  percent,
  index,
}) => {
  const radius = innerRadius + (outerRadius - innerRadius) * 0.5;
  const x = cx + radius * Math.cos(-midAngle * RADIAN);
  const y = cy + radius * Math.sin(-midAngle * RADIAN);

  return (
    <text
      x={x}
      y={y}
```

```
        fill="white"
        textAnchor={x > cx ? "start" : "end"}
        dominantBaseline="central"
      >
        {`${(percent * 100).toFixed(0)}%`}
      </text>
    );
};

export default class ExpenseCharts extends PureComponent {
  static demoUrl =
    "https://codesandbox.io/s/radar-chart-specified-domain-mfl04";

  render() {
    return (
      <div style={{ display: "flex", flexDirection: "row", marginTop: "20px" }}>
        <ResponsiveContainer width={"99%"} height={500}>
          <RadarChart cx="50%" cy="50%" outerRadius="80%" data={data}>
            <PolarGrid />
            <PolarAngleAxis dataKey="subject" />
            <PolarRadiusAxis angle={30} domain={[0, 150]} />
            <Radar
              name="Food"
              dataKey="A"
              stroke="#8884d8"
              fill="#8884d8"
              fillOpacity={0.6}
            />
                    {/* <Radar name="Lily" dataKey="B" stroke="#82ca9d" fill="#82ca9d"
fillOpacity={0.6} /> */}
            <Legend />
          </RadarChart>
        </ResponsiveContainer>
        <div style={{ width: "100%", height: 450 }}>
          <ResponsiveContainer>
            <PieChart>
              <Pie dataKey="value" data={pieData} fill="#8884d8" label />
            </PieChart>
          </ResponsiveContainer>
        </div>
      </div>
    );
  }
}
```

7.1.5 app_layout.jsx

This is the layout class for sidebar. It loads the sidebar component

```
import { Outlet } from "react-router-dom";
import Sidebar from "./sidebar/sidebar";

const AppLayout = () => {
  return (
    <div
      style={{
        padding: "20px 0px 0px 320px",
      }}
    >
      <Sidebar />
      <Outlet />
    </div>
  );
};

export default AppLayout;
```

7.2 BACKEND

7.2.1 app.py

This file has the required endpoints running on Flask server. The data will be stored and fetched from DB2 from here.

```
from flask import Flask, request
from flask_cors import CORS, cross_origin
import ibm_db
import json
import uuid
import datetime
from datetime import datetime, timedelta, date
import calendar

app = Flask(_name_)
cors = CORS(app)
try:
  print("Connecting")
        conn=ibm_db.connect('DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-
21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;SECURITY=S
```

```
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ncv90043;PWD=l6SXXm78Bc5lPuP0',''
,'')
   print("Successfully connected")
except Exception as e:
   print(ibm_db.conn_errormsg())
@app.route('/')
@cross_origin()
def hello():
   return 'hello'

@app.route('/login', methods = ['POST'])
@cross_origin()
def login():
   email = request.form['email']
   password = request.form['password']
   try:
        stmt = ibm_db.exec_immediate(conn, "select * from users where email = '%s' and
password = '%s'" % (email,password))
      result = ibm_db.fetch_assoc(stmt)
      if result:
         response = app.response_class(
         response=json.dumps({"user_id":result['USER_ID']}),
         status=200,
         mimetype='application/json'
         )
         return response
      else:
         response = app.response_class(
         response=json.dumps('User Not Found'),
         status=404,
         mimetype='application/json'
         )
      return response
   except Exception as e:
      print(e)
      response = app.response_class(
         response=json.dumps({"message":str(e)}),
         status=400,
         mimetype='application/json'
      )
      return response



@app.route('/register', methods= ['POST'])
@cross_origin()
def register():
```

```
    name = request.form['name']
    email = request.form['email']
    password = request.form['password']
    limit = request.form['monthly_limit']
    try:
        id = "".join([n for n in str(uuid.uuid4())[:8] if n.isdigit()])
            stmt = ibm_db.exec_immediate(conn, "select * from users where email = '%s'" %
(email))
        print("num rows is ",ibm_db.num_rows(stmt))
        if ibm_db.fetch_assoc(stmt):
            response = app.response_class(
            response=json.dumps({"message":'Email already exists'}),
            status=409,
            mimetype='application/json'
            )
            print("already exists")
            return response
        print("new email")
        stmt = ibm_db.exec_immediate(conn, "INSERT into users values('%s','%s','%s','%s','%s')"
% (int(id),name,email,password,limit))
        print("Number of affected rows: ", ibm_db.num_rows(stmt))
            stmt = ibm_db.exec_immediate(conn, "SELECT * from users where email = '%s' and
password = '%s'" % (email,password))
        result = ibm_db.fetch_assoc(stmt)
        response = app.response_class(
            response=json.dumps({"user_id":result["USER_ID"]}),
                status=200,
            mimetype='application/json'
        )
        return response
    except Exception as e:
        print(e)
        response = app.response_class(
            response=json.dumps({"user_id":None}),
            status=400,
            mimetype='application/json'
        )
        return response

@app.route('/add', methods = ['POST'])
@cross_origin()
def add_expense():
    date = request.form['date']
    amount = request.form['amount']
    category_id = request.form['category_id']
    description = request.form['description']
    expense_type = request.form['expense_type']
```

```
    user_id = request.headers['user_id']
    print("user_id got is ",user_id)
    try:
        id = "".join([n for n in str(uuid.uuid4())[:8] if n.isdigit()])
         stmt = ibm_db.exec_immediate(conn, "SELECT expense_id from FINAL TABLE (INSERT
INTO expense values ('%s','%s','%s','%s','%s','%s'))" %
(int(id),date,amount,category_id,description,expense_type))
        expense_id = ibm_db.fetch_assoc(stmt)['EXPENSE_ID']
            ibm_db.exec_immediate(conn, "INSERT INTO user_expense VALUES ('%s','%s')" %
(user_id,expense_id))
        response = app.response_class(
            response=json.dumps({'message':'expense added successfully'}),
            status=200,
            mimetype='application/json'
        )
        return response
    except Exception as e:
        print(e)
        response = app.response_class(
            response=json.dumps(str(e)),
            status=400,
            mimetype='application/json'
        )
        return response

@app.route('/categories', methods = ['GET'])
@cross_origin()
def get_categories():
    try:
        stmt = ibm_db.exec_immediate(conn, 'SELECT * from category')
        result = ibm_db.fetch_assoc(stmt)
        categories=[]
        while result != False:

categories.append({'category_id':result['CATEGORY_ID'],'category_name':result['CATEGORY
_NAME']})
            result = ibm_db.fetch_assoc(stmt)
        response = app.response_class(
            response=json.dumps(categories),
            status=200,
            mimetype='application/json'
        )
        return response
    except Exception as e:
        response = app.response_class(
            response=json.dumps(str(e)),
```

```
            status=400,
            mimetype='application/json'
        )
        return response

@app.route('/expenses', methods = ['GET'])
@cross_origin()
def get_expenses():
    user_id = request.headers['user_id']
    type = None
    if request.args:
        type = request.args['type']
    try:
            sql = "SELECT e.expense_id, e.amount, e.date, c.category_name, e.expense_type,
e.description FROM expense e INNER JOIN user_expense u ON e.expense_id=u.expense_id
FULL JOIN category c ON e.category_id = c.category_id  where u.user_id = %s" % user_id
        if type:
            sql +=" AND e.expense_type = '%s'" % type
        sql+=" ORDER BY e.date DESC"

        stmt = ibm_db.exec_immediate(conn, sql )
        expense = ibm_db.fetch_assoc(stmt)
        expenses = []
        while expense !=False:
            exp = {k.lower(): v for k, v in expense.items()}
            date = exp['date']
            exp['date'] = date._str_()
            expenses.append(exp)
            expense = ibm_db.fetch_assoc(stmt)
        response = app.response_class(
            response=json.dumps(expenses),
            status=200,
            mimetype='application/json'
        )
        return response
    except Exception as e:
        response = app.response_class(
            response=json.dumps(str(e)),
            status=400,
            mimetype='application/json'
        )
        return response

@app.route('/update-monthly-limit/<monthly_limit>', methods = ['PUT'])
def update_limit(monthly_limit):
    user_id = request.headers['user_id']
    try:
```

```
        sql_update = "UPDATE users SET MONTHLY_LIMIT = '%s' where user_id = %s" %
(monthly_limit,user_id)
      stmt = ibm_db.exec_immediate(conn, sql_update)
      response = app.response_class(
         response=json.dumps({'message':'Updated Successfully'}),
         status=200,
         mimetype='application/json'
      )
      return response
   except Exception as e:
      response = app.response_class(
         response=json.dumps(str(e)),
         status=400,
         mimetype='application/json'
      )
      return response


@app.route('/chart', methods = ['GET'])
def chart():
   user_id =request.headers['user_id']
   try:
      month_start , month_end = get_month_start_and_end()
                                    categories_map        =        {1:'Food',        2:'Automobiles',
3:'Entertainment',4:'Clothing',5:'Healthcare',6:'Others'}
       sql_cat1 = "SELECT SUM(e.amount) FROM expense e INNER JOIN user_expense u ON
e.expense_id=u.expense_id RIGHT JOIN category c ON e.category_id = c.category_id where
u.user_id = %s and e.category_id = 1 and e.date between '%s' And '%s'" %
(user_id,month_start,month_end)
       sql_cat2 = "SELECT SUM(e.amount) FROM expense e INNER JOIN user_expense u ON
e.expense_id=u.expense_id FULL JOIN category c ON e.category_id = c.category_id where
u.user_id = %s and e.expense_type = 'debit' and e.category_id = 2 and e.date between '%s'
And '%s'" % (user_id,month_start,month_end)
       sql_cat3 = "SELECT SUM(e.amount) FROM expense e INNER JOIN user_expense u ON
e.expense_id=u.expense_id FULL JOIN category c ON e.category_id = c.category_id where
u.user_id = %s and e.expense_type = 'debit' and e.category_id = 3 and e.date between '%s'
And '%s'" % (user_id,month_start,month_end)
       sql_cat4 = "SELECT SUM(e.amount) FROM expense e INNER JOIN user_expense u ON
e.expense_id=u.expense_id FULL JOIN category c ON e.category_id = c.category_id where
u.user_id = %s and e.expense_type = 'debit' and e.category_id = 4 and e.date between '%s'
And '%s'" % (user_id,month_start,month_end)
       sql_cat5 = "SELECT SUM(e.amount) FROM expense e INNER JOIN user_expense u ON
e.expense_id=u.expense_id FULL JOIN category c ON e.category_id = c.category_id where
u.user_id = %s and e.expense_type = 'debit' and e.category_id = 5 and e.date between '%s'
And '%s'" % (user_id,month_start,month_end)
       sql_cat6 = "SELECT SUM(e.amount) FROM expense e INNER JOIN user_expense u ON
e.expense_id=u.expense_id FULL JOIN category c ON e.category_id = c.category_id where
```

```
u.user_id = %s and e.expense_type = 'debit' and e.category_id = 6 and e.date between '%s'
And '%s'" % (user_id,month_start,month_end)
      queries = [sql_cat1,sql_cat2,sql_cat3,sql_cat4,sql_cat5,sql_cat6]

      chart_data = {}
      for index in range(0,6):
         stmt = ibm_db.exec_immediate(conn, queries[index])
         result = ibm_db.fetch_assoc(stmt)
         if result['1']:
            print(result)
            chart_data[categories_map[index+1]] = result['1']
         else:
            chart_data[categories_map[index+1]] = 0
      response = app.response_class(
            response=json.dumps(chart_data),
            status=200,
            mimetype='application/json'
            )
      return response
   except Exception as e:
      response = app.response_class(
         response=json.dumps(str(e)),
         status=400,
         mimetype='application/json'
         )
      return response

if__name__== '_main_':
   app.run(debug = True)
```

CHAPTER 8

TESTING

8.1 TEST CASES :

s. no	Test Case id	Feature Type	component	Test description	Input test Data	Actual output	Expected output	remarks
1	TC – RG 01	Func-tional	Register page	register for the application by entering my name, email, password, monthly limit	User1 User1@gmail.com ***** 10000	Registration successful	Registration successful	pass
2	TC – SI 01	Func-tional	Login page	log into the application by entering email & password	User1@gmail.com *****	Login successful	Login sucessfull	pass
3	TC – ST 01	UI	Stats page	view my entire expenses throughout a particular period of time		Expenses are displayed For particular time	Expenses are displayed For particular time	pass
4	TC – DB 01	UI	Dash-board	Display graph in dashboard		Graph is displayed	Graph is displayed	pass
5	TC – ST 02	Func-tional	Stats page	generate reports based on my previous expenditures		Reports generated in graphical form	Reports generated in graphical form	pass
6	TC – SI 02	Func-tional	Dash-board	can logout		Go to sign page	Sign in page displayed	pass

7	TC – ST 03	Func-tional	Stats page	create expense	14-11-2022 100 Food Debit Night food	Expenses created	Expenses created	pass
8	TC – ST 04	Func-tional	Stats page	can edit ,delete, update expense		Expenses updated	Updated of expenses	pass
9	TC – ST 05	UI	Stats page	can view credit and debit expenses separately.		Expenses are listed separately	Expenses are listed separately	pass
10	TC – ST 06	UI	Stats page	aware of the expense that I spend the most on		Expenses are listed for particular category	Expenses are listed for particular category	pass
11	TC – PG 01	Func-tional	Profile page	able to update my set monthly limit		Monthly limit updated	Monthly limit updated	pass
12	TC – PG 01	UI	Profile page	able to view my profile		Profile details displayed	Profile details displayed	pass

8.3 USER ACCEPTANCE TESTING

CHAPTER 9

RESULTS

9.1 Performance Metrics :

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

Advantages :

- Which allows users to track their expenses daily, weekly, monthly, and yearly in terms of summary, bar graphs, and pie-charts.
- Separate view for credit and debit transactions
- no burden of manual calculations
- generate and save reports.
- You can insert, delete records
- You can track expenses by categories like food, automobile, entertainment, education etc..
- You can track expenses by time, weekly, month, year etc..
- Setting monthly limits and we can update it later
- Customized email alerts when limit exceeds.

Disadvantages :

- User have entry every records manually
- The category divided may be blunder or messy
- Can’t able to customized user defined categories

CHAPTER 11

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 changetheir way of spending money. In short, this application will help its users to overcome the wastage of money.

12.FUTURE SCOPE :

- In further days, there will be mails and payment embedded with the app. Also, backup details will be recorded on cloud.
- Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spend .
- Alerts for paying dues and remainders to record input at particular user defined time.

13. APPENDIX :

Source code link : https://github.com/IBM-EPBL/IBM-Project-16601-1659618371