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**

1

**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

2

# 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.

3

# TABLE OF CONTENTS CHAPTER NO TITLE ABSTRACT

1. **INTRODUCTION**
   1. Project Overview
   2. Purpose

# LITERATURE SURVEY

* 1. Existing problems
  2. References
  3. Problem Statement Definition

# IDEATION & PROPOSED SOLUTION

* 1. Empathy Map Canvas
  2. Ideation & Brainstorming
  3. Proposed Solution
  4. Problem Solution fit

# REQUIREMENT ANALYSIS

* 1. Functional requirement
  2. Non-Functional requirements

# PROJECT DESIGN

* 1. Data Flow Diagrams
  2. Solution & Technical Architecture
  3. User Stories

# PROJECT PLANNING & SCHEDULING

* 1. Sprint Planning & Estimation
  2. Sprint Delivery Schedule

4

* 1. Reports from JIRA

# CODING & SOLUTIONING

* 1. Feature 1 : Frontend
  2. Feature 2 : Backend

# TESTING

8.1Test Cases

# RESULTS

* 1. Performance Metrics

1. **ADVANTAGES & DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**
4. **APPENDIX**

5

# 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.

### 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.

### 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

### 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.

### References:

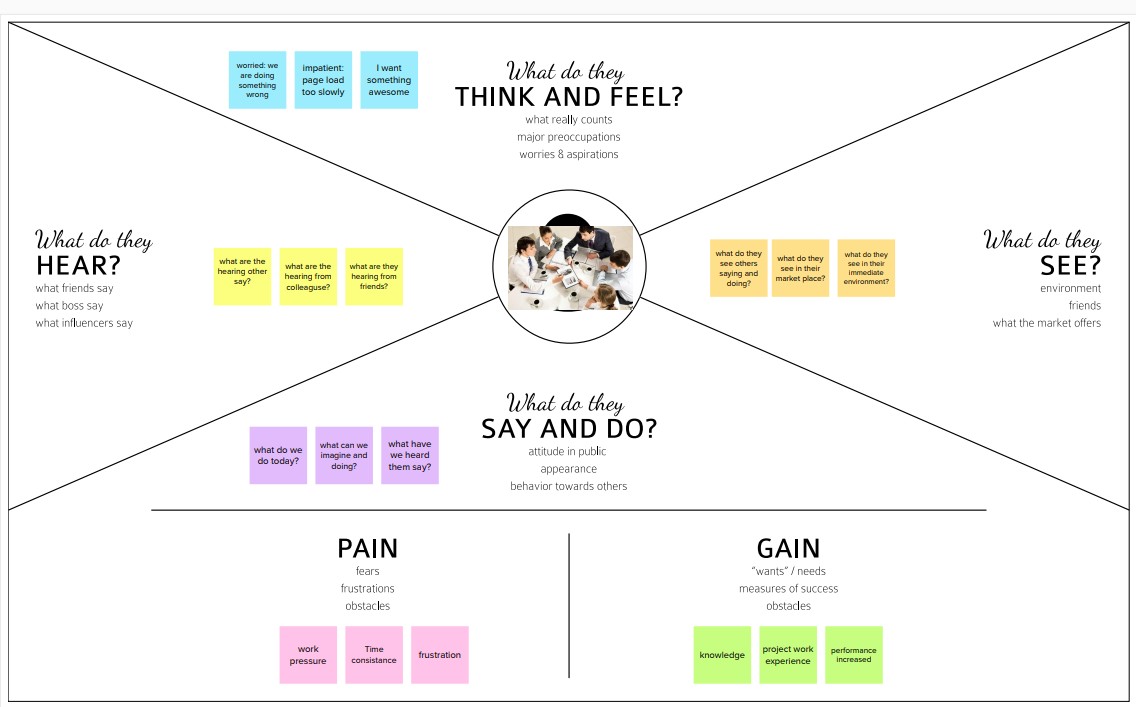
* <https://ijirt.org/Article?manuscript=150860>
* [https://www.researchgate.net/publication/347972162\_Expense\_Manager\_Applicati](https://www.researchgate.net/publication/347972162_Expense_Manager_Application) [on](https://www.researchgate.net/publication/347972162_Expense_Manager_Application)
* <https://ijarsct.co.in/Paper391.pdf>
* <https://irejournals.com/paper-details/1702687>
* <https://www.irjet.net/archives/V6/i3/IRJET-V6I31110.pdf>
* <https://www.youneedabudget.com/>

### Problem Statement Definition :

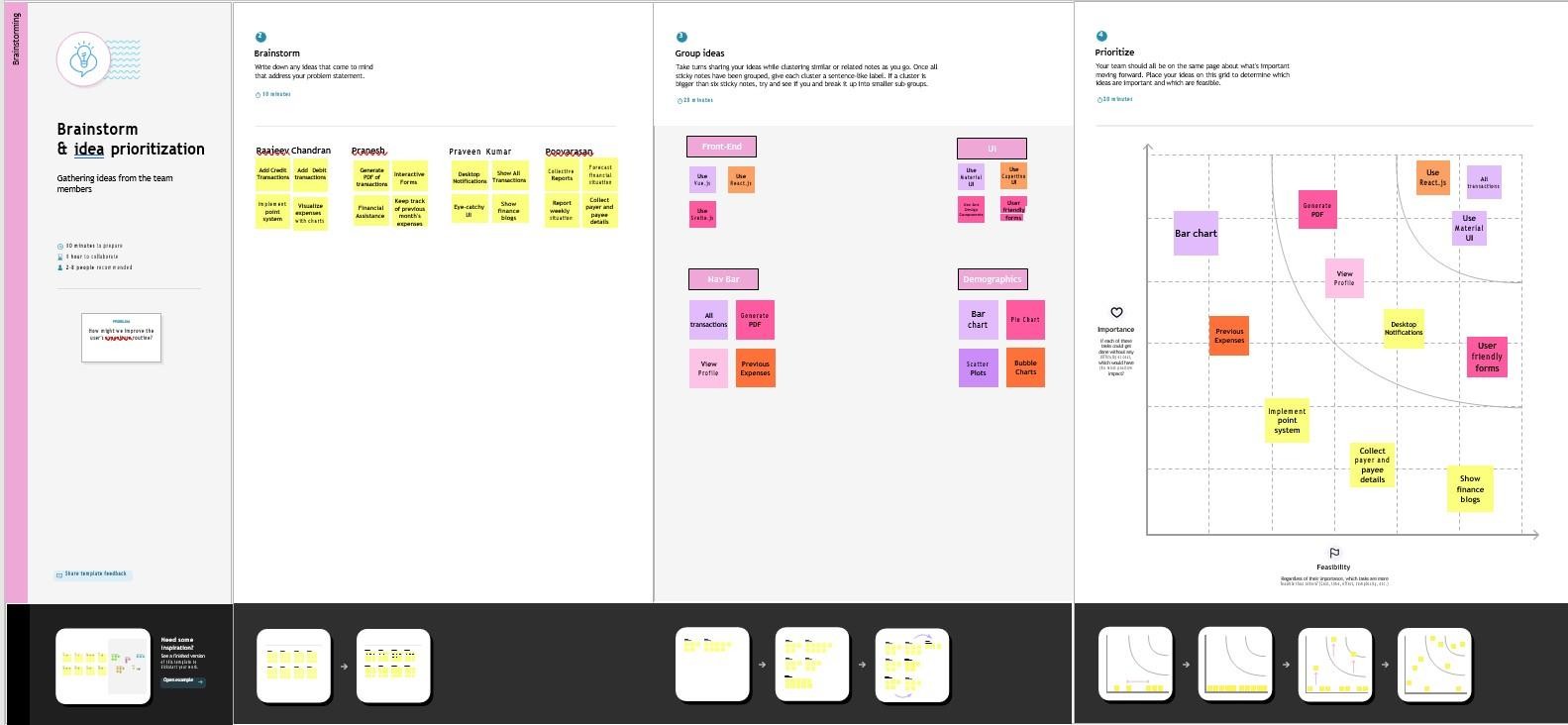
Our project helps the user to keep track their expenses and determine whether they are spending as per their set budget. Potential users need to input the required data such as the expense amount, merchant, category, and date when the expense was made. Which allows users to track their expenses daily, 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**

* 1. Empathy Map Canvas

## IDEATION & BRAINSTORMING :



* 1. **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 Satisfaction | / Customer | 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 advertisement. | be | generated | by | placing |
| 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. | | | | |

### 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

### 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* |

### Non-functional Requirements:

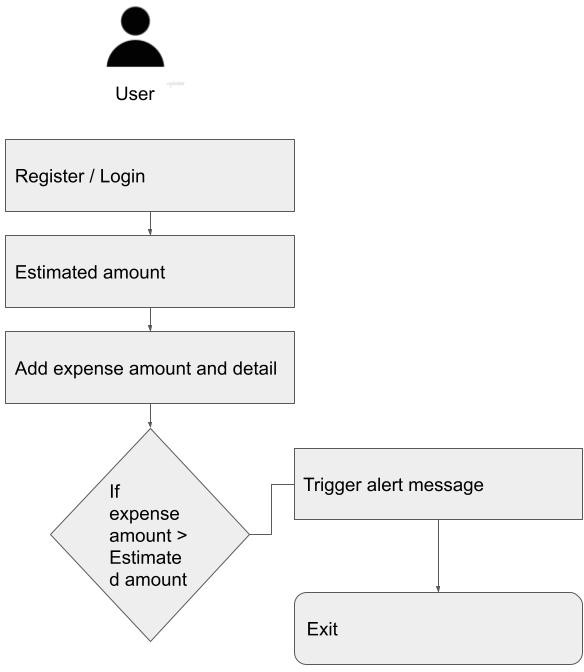
Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **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 :

### 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.



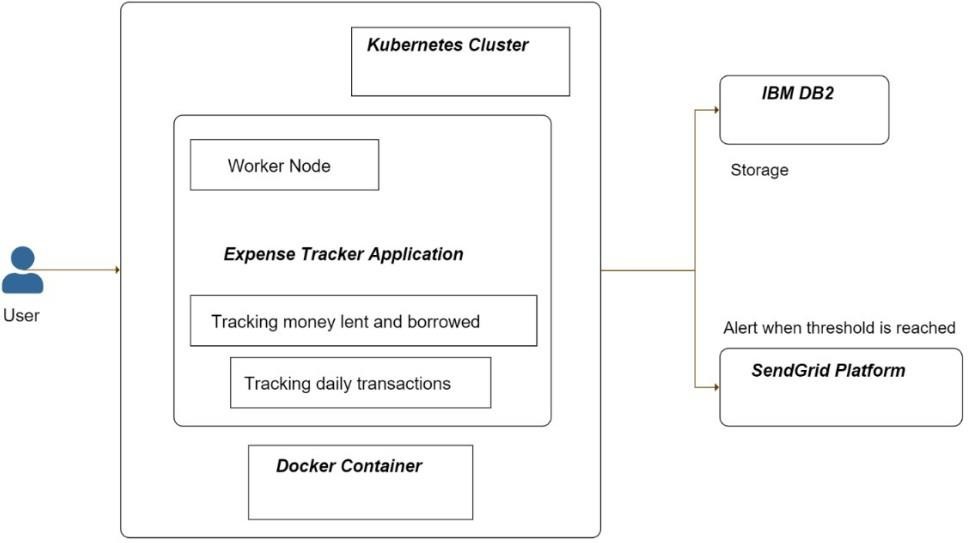
### Solution Architecture and Technical Architecture :

**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:

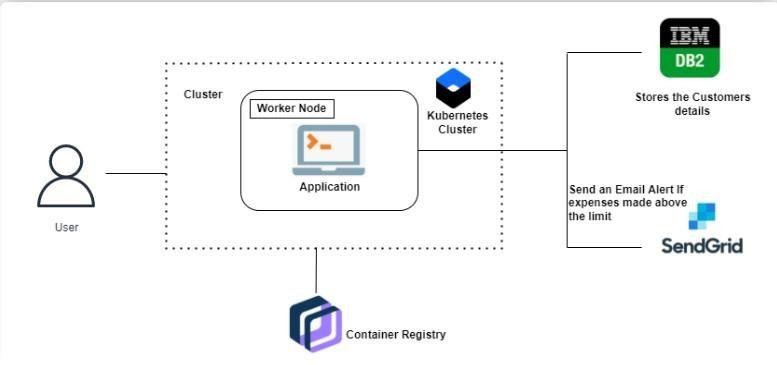
* + - Find the best tech solution to solve existing business problems.
    - Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
    - Define features, development phases, and solution requirements.
    - Provide specifications according to which 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 |

* 1. **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 | I can access | High | Sprint-1 |
| can register | my account / |
| for the | dashboard |
| application by |  |
| entering my |  |
| email, |  |
| password, |  |
| mobile |  |
| number, |  |
| weekly |  |
| expense, |  |
| montly salary |  |
|  | Login | USN-2 | As a user, I | I can access | High | Sprint-1 |
| can log into | my account / |
| the application | dashboard |
| by entering |  |
| email & |  |
| password |  |
|  | Landing page |  | As a user, I | I can view my | High | Sprint-1 |
| can view my | expenses |
| entire |  |
| expenses |  |
| throughout a |  |
| particular |  |
| period of |  |
| time |  |
|  |  |  | As a user, I | Report is | Medium | Sprint-2 |
| can generate | successfully |
| reports based | generated |
| on my |  |
| previous |  |
| expenditures. |  |
|  |  |  | 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 | The | High | Sprint-1 |
| can edit | corresponding |
| ,delete, | action is made |
| update | to the expense |
| expense |  |
|  |  |  | As a user, I  can view credit and | The expenses  are filtered accordingly | Medium | Sprint-2 |

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

# CHAPTER 6

**PROJECT PLANNING & SCHEDULING**

### 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 |

### 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 |

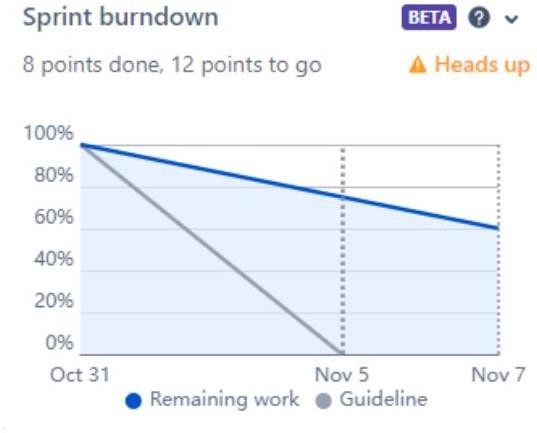
**Velocity:**

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

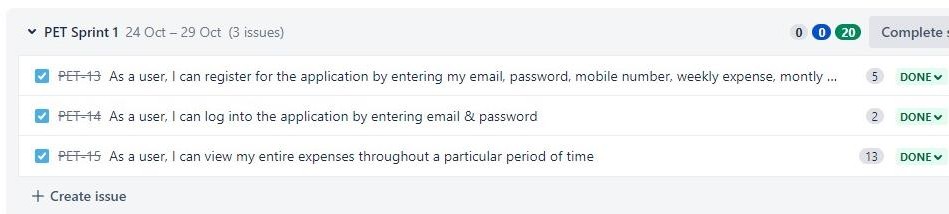
### Average velocity = Total story points/ No. of iterations = 80/4 = 20

**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](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies such as [Scrum.](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/) However, burn down charts can be applied to any project containing measurable progress over time.



* 1. **Reports from JIRA :**



# CHAPTER 7 CODING AND SOLUTIONING

## FRONTEND

### add\_expense\_model.jsx:

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

@app.route("/add") def adding():

return render\_template('add.html')

@app.route('/addexpense',methods=['GET', 'POST']) def addexpense():

date = request.form['date']

expensename = request.form['expensename'] amount = request.form['amount']

paymode = request.form['paymode'] category = request.form['category'] print(date)

p1 = date[0:10] p2 = date[11:13]

p3 = date[14:]

p4 = p1 + "-" + p2 + "." + p3 + ".00"

print(p4)

# cursor = mysql.connection.cursor()

# cursor.execute('INSERT INTO expenses VALUES (NULL, % s, % s, % s, % s, % s, % s)', (session['id']

,date, expensename, amount, paymode, category)) # mysql.connection.commit()

# print(date + " " + expensename + " " + amount + " " + paymode + " " + category)

sql = "INSERT INTO expenses (userid, date, expensename, amount, paymode, category) VALUES (?, ?,

?, ?, ?, ?)"

stmt = ibm\_db.prepare(ibm\_db\_conn, sql) ibm\_db.bind\_param(stmt, 1, session['id'])

ibm\_db.bind\_param(stmt, 2, p4)

ibm\_db.bind\_param(stmt, 3, expensename)

ibm\_db.bind\_param(stmt, 4, amount)

ibm\_db.bind\_param(stmt, 5, paymode)

ibm\_db.bind\_param(stmt, 6, category) ibm\_db.execute(stmt)

print("Expenses added") # email part

param = "SELECT \* FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"

res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) dictionary = ibm\_db.fetch\_assoc(res)

expense = []

while dictionary != False:

temp = [] temp.append(dictionary["ID"]) temp.append(dictionary["USERID"]) temp.append(dictionary["DATE"])

temp.append(dictionary["EXPENSENAME"]) temp.append(dictionary["AMOUNT"]) temp.append(dictionary["PAYMODE"]) temp.append(dictionary["CATEGORY"]) expense.append(temp)

print(temp)

dictionary = ibm\_db.fetch\_assoc(res) total=0

for x in expense: total += x[4]

param = "SELECT id, limitss FROM limits WHERE userid = " + str(session['id']) + " ORDER BY id DESC LIMIT 1"

res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) dictionary = ibm\_db.fetch\_assoc(res)

row = [] s = 0

while dictionary != False:

temp = [] temp.append(dictionary["LIMITSS"]) row.append(temp)

dictionary = ibm\_db.fetch\_assoc(res) s = temp[0]

if total > int(s):

msg = "Hello " + session['username'] + " , " + "you have crossed the monthly limit of Rs. " + s + "/-

!!!" + "\n" + "Thank you, " + "\n" + "Team Personal Expense Tracker." sendmail(msg,session['email'])

return redirect("/display")

#DISPLAY---graph @app.route("/display") def display():

print(session["username"],session['id']) # cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM expenses WHERE userid = % s AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))

# expense = cursor.fetchall()

param = "SELECT \* FROM expenses WHERE userid = " + str(session['id']) + " ORDER BY date DESC" res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

dictionary = ibm\_db.fetch\_assoc(res) expense = []

while dictionary != False:

temp = [] temp.append(dictionary["ID"]) temp.append(dictionary["USERID"]) temp.append(dictionary["DATE"])

temp.append(dictionary["EXPENSENAME"]) temp.append(dictionary["AMOUNT"]) temp.append(dictionary["PAYMODE"]) temp.append(dictionary["CATEGORY"]) expense.append(temp)

print(temp)

dictionary = ibm\_db.fetch\_assoc(res)

return render\_template('display.html' ,expense = expense)

### expenditure\_breaksown.jsx

This is the model through which we can see detailed analysis of our expenses param1 = "SELECT MONTH(date) as mn, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) GROUP BY MONTH(date) ORDER BY MONTH(date)"

res1 = ibm\_db.exec\_immediate(ibm\_db\_conn, param1) dictionary1 = ibm\_db.fetch\_assoc(res1)

texpense = []

while dictionary1 != False:

temp = [] temp.append(dictionary1["MN"]) temp.append(dictionary1["TOT"]) texpense.append(temp) print(temp)

dictionary1 = ibm\_db.fetch\_assoc(res1) # cursor = mysql.connection.cursor()

# cursor.execute('SELECT \* FROM expenses WHERE userid = % s AND YEAR(DATE(date))= YEAR(now()) AND date ORDER BY `expenses`.`date` DESC',(str(session['id'])))

# expense = cursor.fetchall()

param = "SELECT \* FROM expenses WHERE userid = " + str(session['id']) + " AND YEAR(date) = YEAR(current timestamp) ORDER BY date DESC"

res = ibm\_db.exec\_immediate(ibm\_db\_conn, param) dictionary = ibm\_db.fetch\_assoc(res)

expense = []

while dictionary != False:

temp = [] temp.append(dictionary["ID"]) temp.append(dictionary["USERID"]) temp.append(dictionary["DATE"])

temp.append(dictionary["EXPENSENAME"]) temp.append(dictionary["AMOUNT"])

temp.append(dictionary["PAYMODE"]) temp.append(dictionary["CATEGORY"]) expense.append(temp)

print(temp)

dictionary = ibm\_db.fetch\_assoc(res) total=0

t\_food=0 t\_entertainment=0 t\_business=0 t\_rent=0

t\_EMI=0

t\_other=0

for x in expense:

total += x[4]

if x[6] == "food": t\_food += x[4]

elif x[6] == "entertainment": t\_entertainment += x[4]

elif x[6] == "business": t\_business += x[4]

elif x[6] == "rent": t\_rent += x[4] elif x[6] == "EMI": t\_EMI += x[4]

elif x[6] == "other": t\_other += x[4]

print(total) print(t\_food) print(t\_entertainment)

print(t\_business) print(t\_rent) print(t\_EMI) print(t\_other)

return render\_template("today.html", texpense = texpense, expense = expense, total = total , t\_food = t\_food,t\_entertainment = t\_entertainment,

t\_business = t\_business, t\_rent = t\_rent, t\_EMI = t\_EMI, t\_other = t\_other )

### 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;

## BACKEND

### app.py

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

# -\*- coding: utf-8 -\*- """

Spyder Editor

This is a temporary script file. """

from ﬂask import Flask, render\_template, request, redirect, session # from ﬂask\_mysqldb import MySQL

# import MySQLdb.cursors import re

from ﬂask\_db2 import DB2 import ibm\_db

import ibm\_db\_dbi

from sendemail import sendgridmail,sendmail # from gevent.pywsgi import WSGIServer import os

app = Flask( name )

app.secret\_key = 'a'

# app.config['MYSQL\_HOST'] = 'remotemysql.com' # app.config['MYSQL\_USER'] = 'D2DxDUPBii'

# app.config['MYSQL\_PASSWORD'] = 'r8XBO4GsMz' # app.config['MYSQL\_DB'] = 'D2DxDUPBii'

"""

dsn\_hostname = "3883e7e4-18f5-4afe-be8c- fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"

dsn\_uid = "sbb93800"

dsn\_pwd = "wobsVLm6ccFxcNLe" dsn\_driver = "{IBM DB2 ODBC DRIVER}" dsn\_database = "bludb"

dsn\_port = "31498" dsn\_protocol = "tcpip" dsn = (

"DRIVER={0};"

"DATABASE={1};"

"HOSTNAME={2};"

"PORT={3};"

"PROTOCOL={4};"

"UID={5};"

"PWD={6};"

).format(dsn\_driver, dsn\_database, dsn\_hostname, dsn\_port, dsn\_protocol, dsn\_uid, dsn\_pwd) """

# app.config['DB2\_DRIVER'] = '{IBM DB2 ODBC DRIVER}'

app.config['database'] = 'bludb'

app.config['hostname'] = '3883e7e4-18f5-4afe-be8c- fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'

app.config['port'] = '31498' app.config['protocol'] = 'tcpip' app.config['uid'] = 'sbb93800' app.config['pwd'] = 'wobsVLm6ccFxcNLe' app.config['security'] = 'SSL'

try:

mysql = DB2(app) conn\_str='database=bludb;hostname=3883e7e4-18f5-4afe-be8c-

fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;port=31498;protocol=tcpip;\ uid=sbb93800;pwd=wobsVLm6ccFxcNLe;security=SSL'

ibm\_db\_conn = ibm\_db.connect(conn\_str,'','') print("Database connected without any error !!")

except:

print("IBM DB Connection error : " + DB2.conn\_errormsg()) # app.config['']

# mysql = MySQL(app)

#HOME--PAGE

@app.route("/home") def home():

return render\_template("homepage.html")

@app.route("/") def add():

return render\_template("home.html")

#SIGN--UP--OR--REGISTER

@app.route("/signup") def signup():

return render\_template("signup.html")

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

msg = ''

print("Break point1")

if request.method == 'POST' :

username = request.form['username'] email = request.form['email'] password = request.form['password']

print("Break point2" + "name: " + username + "------" + email + " " + password)

try:

print("Break point3")

connectionID = ibm\_db\_dbi.connect(conn\_str, '', '') cursor = connectionID.cursor()

print("Break point4") except:

print("No connection Established") # cursor = mysql.connection.cursor() # with app.app\_context():

# print("Break point3")

# cursor = ibm\_db\_conn.cursor() # print("Break point4") print("Break point5")

sql = "SELECT \* FROM register WHERE username = ?" stmt = ibm\_db.prepare(ibm\_db\_conn, sql) ibm\_db.bind\_param(stmt, 1, username) ibm\_db.execute(stmt)

result = ibm\_db.execute(stmt) print(result)

account = ibm\_db.fetch\_row(stmt) print(account)

param = "SELECT \* FROM register WHERE username = " + "\'" + username + "\'" res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

print(" ")

dictionary = ibm\_db.fetch\_assoc(res) while dictionary != False:

print("The ID is : ", dictionary["USERNAME"])

dictionary = ibm\_db.fetch\_assoc(res)

# dictionary = ibm\_db.fetch\_assoc(result) # cursor.execute(stmt)

# account = cursor.fetchone() # print(account)

# while ibm\_db.fetch\_row(result) != False:

# # account = ibm\_db.result(stmt)

# print(ibm\_db.result(result, "username")) # print(dictionary["username"]) print("break point 6")

if account:

msg = 'Username already exists !'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !' else:

sql2 = "INSERT INTO register (username, email,password) VALUES (?, ?, ?)" stmt2 = ibm\_db.prepare(ibm\_db\_conn, sql2)

ibm\_db.bind\_param(stmt2, 1, username)

ibm\_db.bind\_param(stmt2, 2, email)

ibm\_db.bind\_param(stmt2, 3, password) ibm\_db.execute(stmt2)

# cursor.execute('INSERT INTO register VALUES (NULL, % s, % s, % s)', (username, email,password)) # mysql.connection.commit()

msg = 'You have successfully registered !'

return render\_template('signup.html', msg = msg)

# CHAPTER 8 TESTING

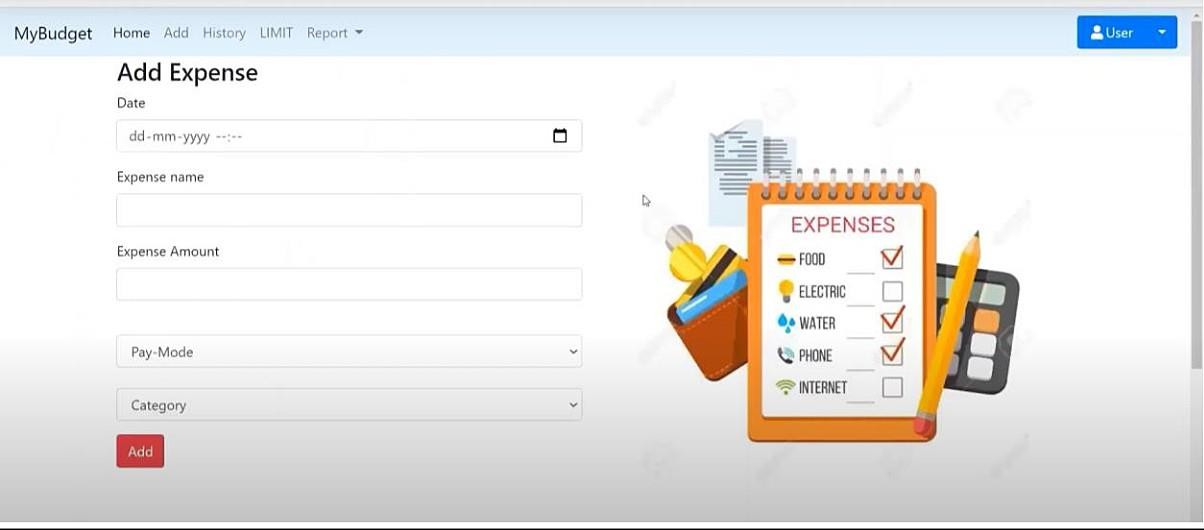
* 1. **TEST CASES :**

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

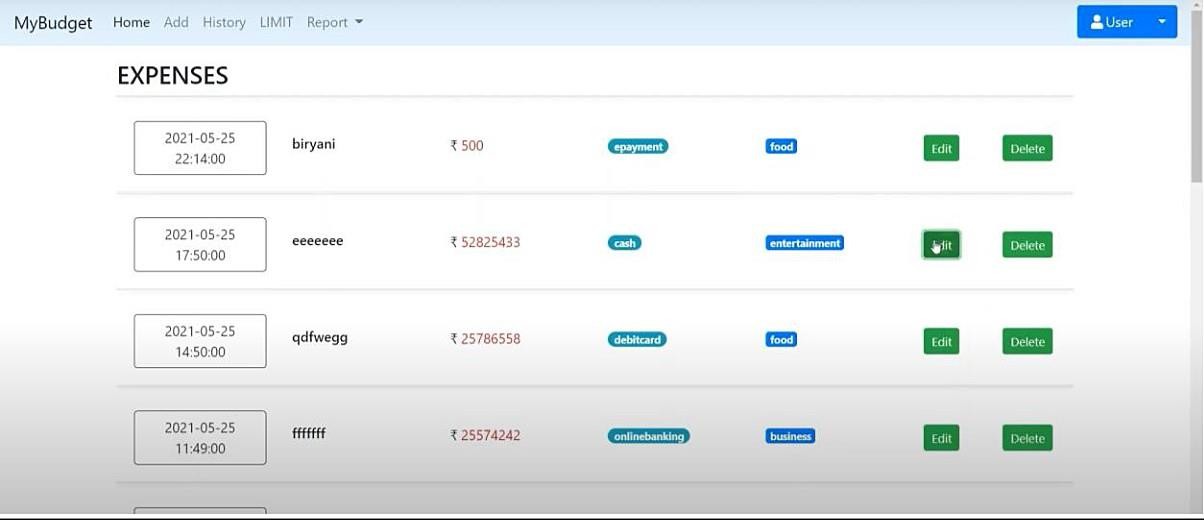
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | TC – ST 03 | Func- tional | Stats page | create expense | 14-11-2022  100  Food  Debit Night food | Expenses created | Expenses created | pass |
| 8 | TC – ST 04 | Func- tional | Stats page | can edit  ,delete, 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 |

# CHAPTER 9 RESULTS

### ADDING EXPENSES:

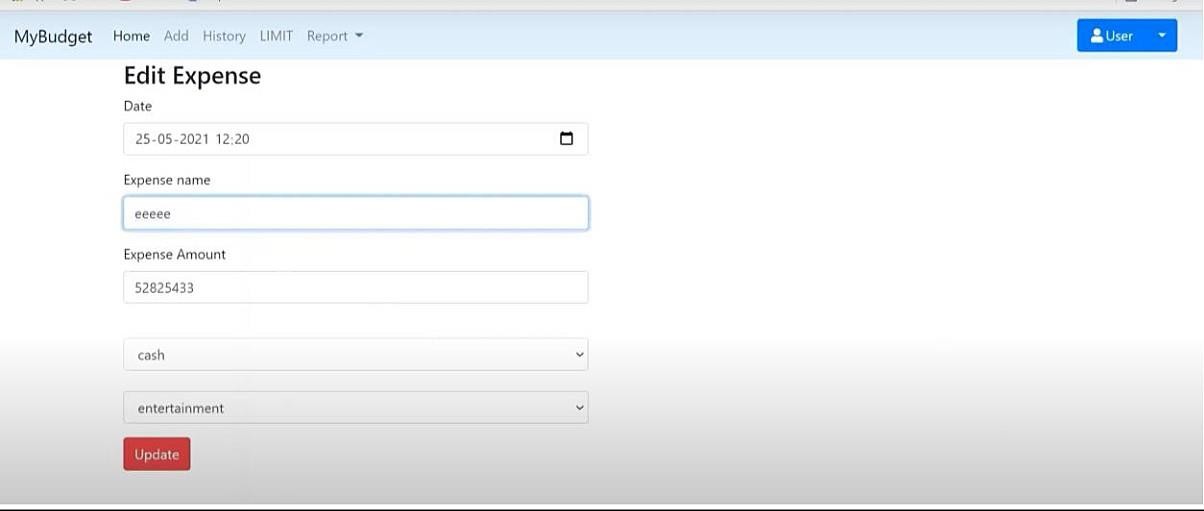


**EXPENSE HISTORY:**

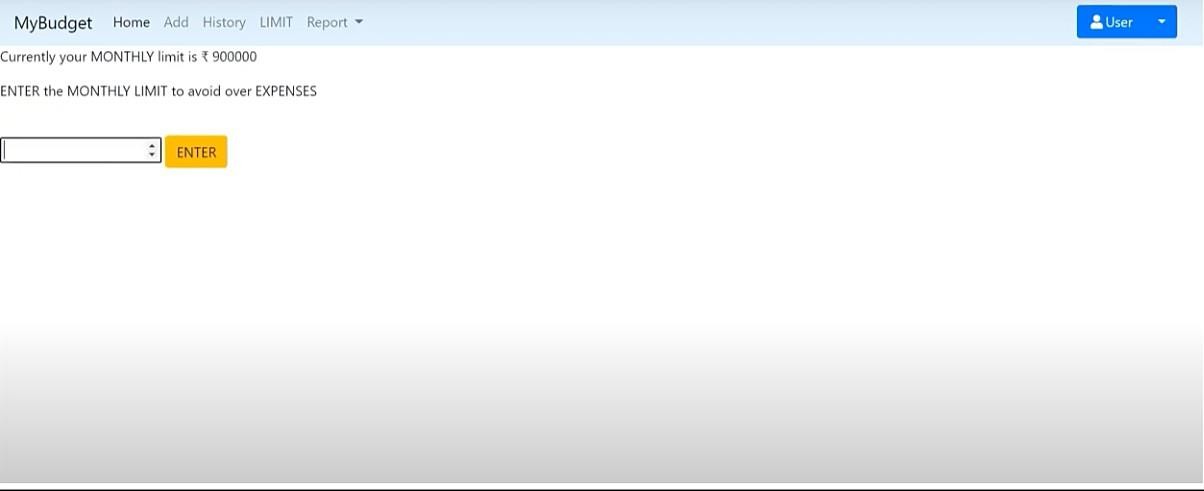


### 

### EDITING EXPENSE:



**SETTING LIMIT:**



### EXPENDITURE REPORT – PIECHART

### 

# 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

## 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.

## 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.

## APPENDIX :

**Source code link :** [**https://github.com/IBM-EPBL/IBM-Project-16601-1659618371**](https://github.com/IBM-EPBL/IBM-Project-16601-1659618371)