**NALAIYA THIRAN - IBM PROJECT REPORT**

**(19CS406T Professional Readiness for Innovation, Employability and Entrepreneurship)**

**ON**

**PERSONAL EXPENSE TRACKER APPLICATION**

***Submitted by***

**TEAM ID: PNT2022TMID23288**

**BALAJI.N.R. (113219031022)**

**ARAVINDH RAJ.N. (****113219031016)**

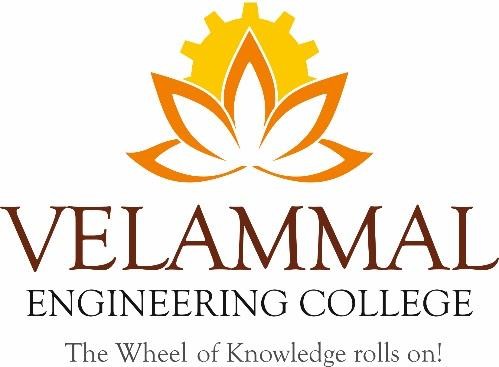
**BALAMURUGAN.V. (113219031023)**

**PRAKASH.S. (113219031108)**

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

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**VELAMMAL ENGINEERING COLLEGE, CHENNAI-66.**

(An Autonomous Institution, Affiliated to Anna University, Chennai)

**2022-2023**

**TABLE OF CONTENTS**

**SI NO TITLE PG NO**

1 **INTRODUCTION 1**

1.1 Project Overview 1

1.2 Purpose 1

2 **LITERATURE SURVEY 2**

2.1 Existing Problem 2

2.2 References 3

2.3 Problem Statement Definition 4

3 **IDEATION & PROPOSED SOLUTION 6**

3.1 Empathy Map Canvas 6

3.2 Ideation & Brainstorming 7

3.3 Proposed Solution 10

3.4 Problem Solution fit 11

4 **REQUIREMENT ANALYSIS 13**

4.1 Functional Requirement 13

4.2 Non-Functional Requirement 13

5 **PROJECT DESIGN 15**

5.1Data Flow Diagrams 15

5.2 Solution & Technical Architechture 15

5.3 User Stories 17

6 **PROJECT PLANNING & SCHEDULING 18**

6.1 Sprint Planning & Estimation 18

6.2 Sprint Delivery Schedule 19

6.3 Reports from JIRA 20

7 **CODING & SOLUTIONING 22**

7.1 Feature 1 22

7.2 Feature 2 24

7.3 Feature 3 30

8 **TESTING 32**

8.1 Test Cases 32

8.2 User Acceptance Testing 32

9 **RESULTS 34**

9.1 Performance Metrices 34

10 **ADVANTAGES & DISADVANTAGES 35**

11 **CONCLUSION 36**

12 **FUTURE SCOPE 36**

13 **APPENDIX 37**

13.1 Source Code 37

13.2 GitHub Project Demo Link 65

**1. INTRODUCTION**

**1.1 PROJECT OVERVIEW**

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 into money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user.  Also, users can get an analysis of their expenditures in graphical forms. They have the 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.

**1.2 PURPOSE**

A comprehensive money management strategy requires clarity and conviction for decision-making. You will need a defined goal and a clear vision for grasping the business and personal finances. That’s when an expense tracking app comes into the picture.

An expense tracking app is an exclusive suite of services for people who seek to handle their earnings and plan their expenses and savings efficiently. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis.

**2. LITERATURE SURVEY**

**2.1 EXISTING PROBLEM**

|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Owner** | **Advantages** | **Disadvantages** |
| ZOHO Expense | ZOHO  Corporation | * Many companies are using Zoho apps. * Ideal for approval of expense. * Free expense tracking. | * Monthly plans   Can  be expensive for a small team.   * lot used for a personal expense tracking. * First-time users may feel over whelmed with the user interface. |
| Spendee | Spendee | * Convenient, one device management of all your financial data,Including   cryptocurrency.   * Affordable and Customizable for however, you choose to budget. * Userfriendly interface | * Takes time to set up. * Occasional account   Commercial issues. |
| Reach-Expense  Tracker | Reach Inc. | * It is simple and easy   to use.   * It calculates budget. | * Design of   application  is one of the  drawback.   * Sometimes it   generates multiple  record of same  Transaction.   * Data cannot be updated without internet   connection. |
| Mint: Budget  and Track Bills | Intuit Inc | * It is free to use and provides high security on data. * Alert and reminder tools. * Free credit   monitoring services.   * Syncs to a diverse set of financial accounts. | * Takes time to set up. * Occasional account connection issues. |

**2.2 REFERENCES**

* **Personal Expense Tracker Application using Mobile application**

**AUTHOR NAME:** Faculty of San Diego State University

https://digitallibrary.sdsu.edu/islandora/object/sdsu%3A3676/datastream/OBJ/view

* **The Economic Times - Tiny habits can convert non-savers into savers: Here's how**

**AUTHOR NAME:** The Economic Times.

https://economictimes.indiatimes.com/wealth/save/tiny-habits-can-convert-non-savers-into-savers-heres-how/articleshow/77164834.cms

* **Role of Cloud in efficient application development**

**AUTHOR NAME:** Security Boulevard

**OBJECTIVE:** This article suggests that App development is more quick in cloud than forming a physical server. Agility, speed and reliablility are higher in cloud development, it also ensures that it meets the customer requirements.

**Reference Link**: https://securityboulevard.com/2020/01/top-3-reasons-why-application-development-in-the-cloud-can-drive-better-products-services-faster/

* **Kubernetes Cluster**

**AUTHOR NAME:** Kubernetes.io

**OBJECTIVE:** [Kubernetes](https://kubernetes.io/docs/concepts/overview/), also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications. Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.

**Refernce Link**: https://kubernetes.io/

* **Expense Manager Application**

**AUTHOR NAME:** Velmurugan A, Albert Maryan J, Niranjana P, Richard Francis

**OBJECTIVE**: Mobile applications are top in user convenience and have overpassed the web applications in terms of popularity and usability. There are variousmobile applications that provide solutions to manage personal and group expense but not many of them provide a comprehensive view of both cases. In this paper, we develop a mobile application developed for the android platform that keeps record of user personal expenses, his/her contribution in group expenditures, top investment options, view of the current stock market, read authenticated financial news and grab the best ongoing offers in the market in popular categories.

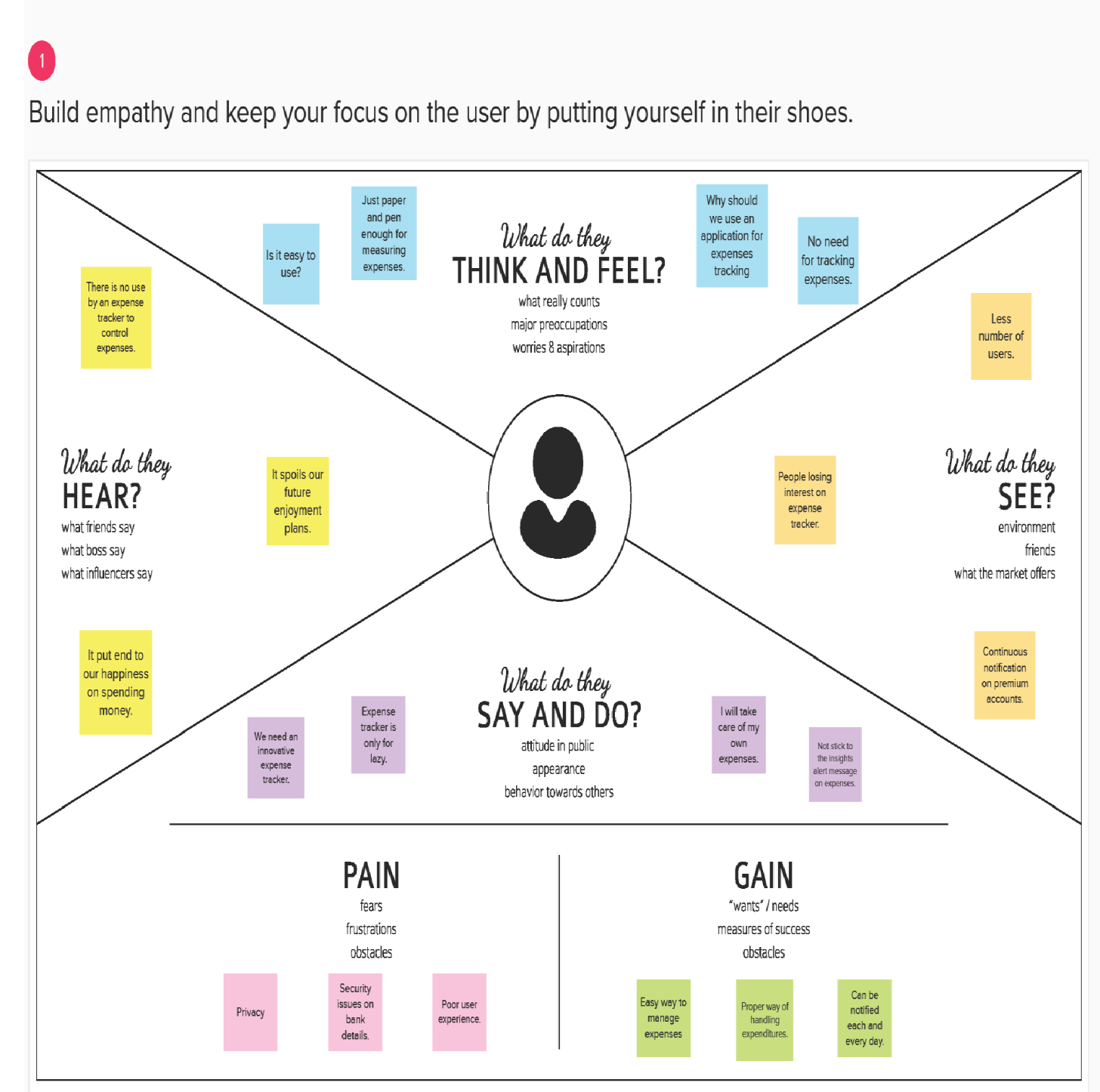
**ReferenceLink**:https://www.researchgate.net/publication/347972162\_Expense\_Manager\_Application.

* 1. **PROBLEM STATEMENT DEFINITION**

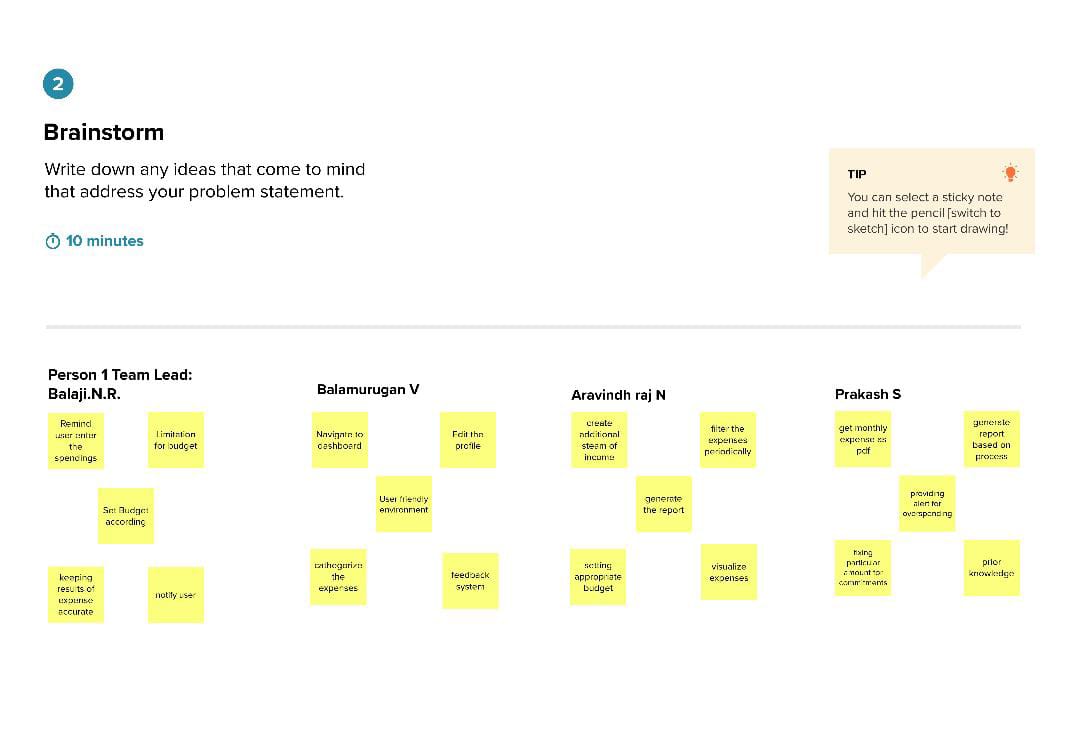
1. Nowadays, saving money is more difficult than earning it. People can earn money in many ways but they can save it in only one way. i.e., controlling or managing the expenditure.
2. Middle class family in India suffers a lot in controlling their expenses, paper and pen method couldn't able to make it happen in this digital world.They are in need of software for tracking their expenditure.
3. People need a software which could track their expense in the daily, weekly, monthly even for yearly basis. This application would give a visualisation method of insights.
4. Alert message is the key to control people from over expense, alert messages are received through an e-mail.
5. Maintaining a good habit is difficult to follow, usually people skip those habits to be in their comfort zone. This is also reflected in controlling their expenditure, usually they snooze their target and they start their habit of over spending.
6. The laziness on entering their expenditures is also the key reason for the failure of the objective.

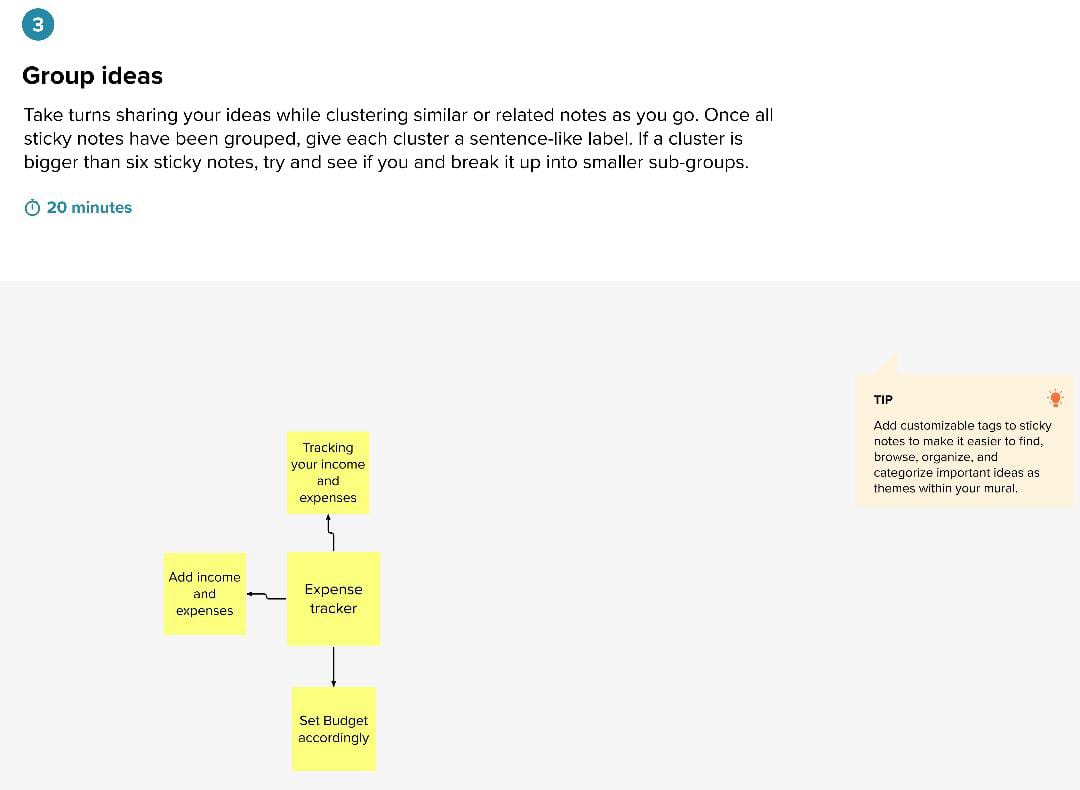
**3. IDEATION AND PROPOSED SOLUTION**

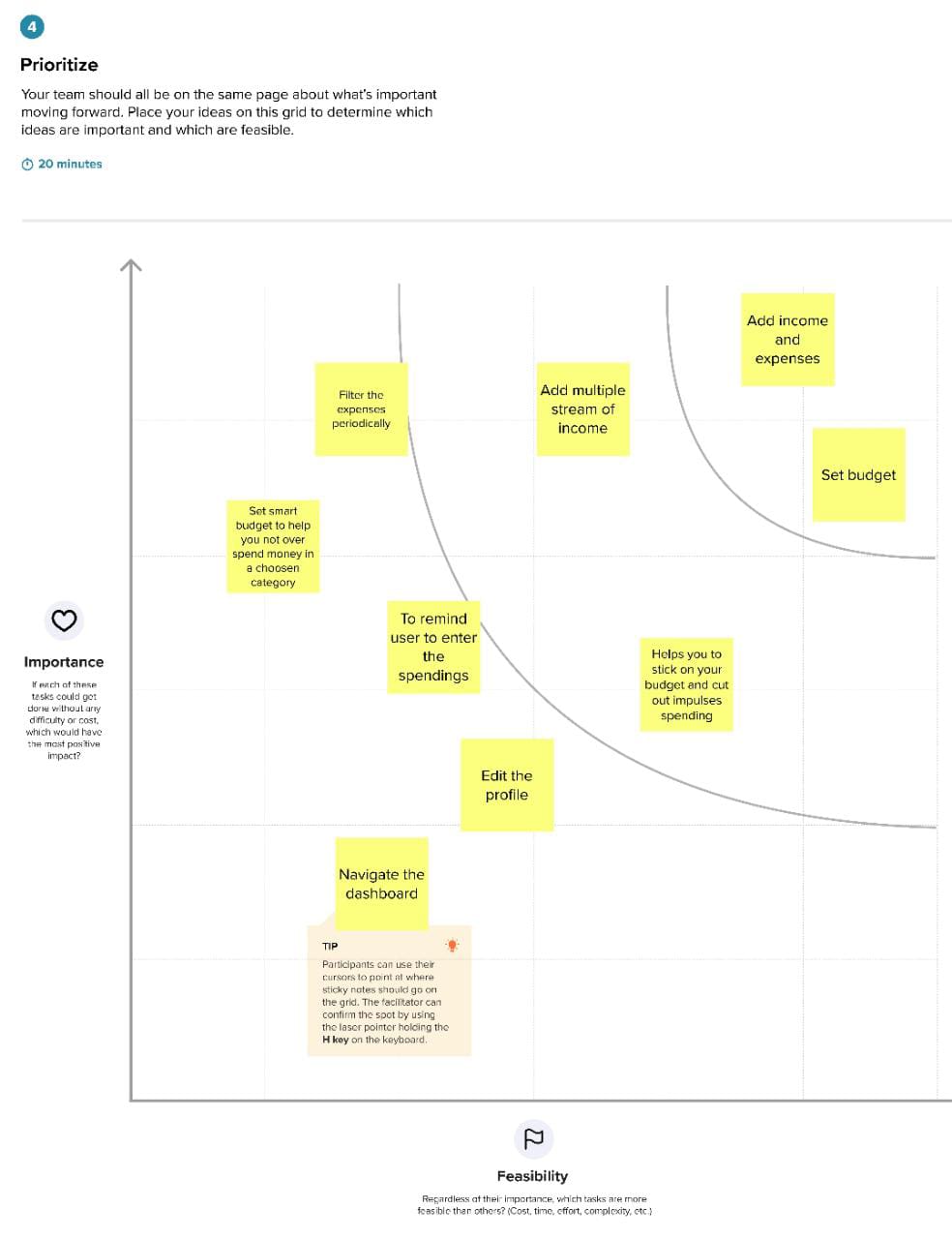
**3.1 EMPATHY MAP CANVAS**

****

**3.2 IDEATION & BRAINSTORMING**

****

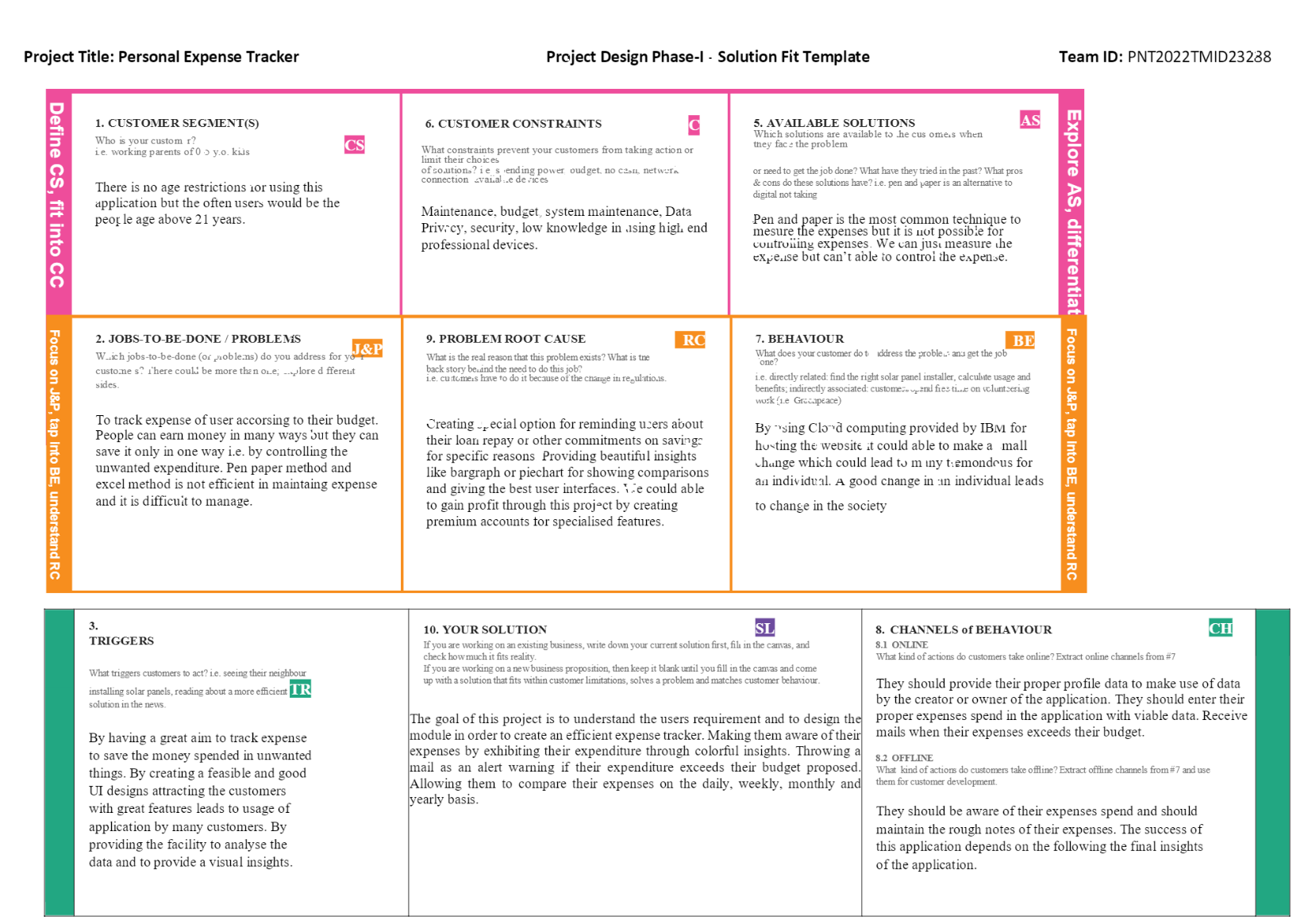
****

****

**3.3 PROPOSED SOLUTION**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | To track expense of user accorsing to their budget. People can earn money in many ways but they can save it only in one way i.e. by controlling the unwanted expenditure. Pen paper method and excel method is not efficient in maintaing expense and it is difficult to manage. |
| 2 | Idea / Solution description | 1. The goal of this project is to understand the users requirement and to design the module in order to create an efficient expense tracker.  2. Making them aware of their expenses by exhibiting their expenditure through colorful insights.  3. Throwing a mail as an alert warning if their expenditure exceeds their budget proposed. Allowing them to compare their expenses on the daily, weekly, monthly and yearly basis. |
| 3 | Novelty / Uniqueness | Creating special option for reminding users about their loan repay or other commitments on savings for specific reasons. Providing beautiful insights like bargraph or piechart for showing. |
| 4 | Social Impact / Customer Satisfaction | By using Cloud computing provided by IBM for hosting the website it could able to make a small change which could lead to many tremondous for an individual. A good change in an individual leads to change in the society. |
| 5 | Business Model (Revenue Model) | We could able to gain profit through this project by creating premium accounts for specialised features. By fixing the amount for premium with moderate rate the user could able to use the application with ease and the developer could able to gain some profit. The profit of the application is based on the best design and user experience of the product. |
| 6 | Scalability of the Solution | • This system can even work more efficiently with large volume of data.  • Implementation of anyone and anywhere using system can be helpful for even a commoner to buy the products.  • Daily and Each time purchase updation of the stock for preventing inventory shrinkage .  • Direct chat system with the retailers and the customers for providing best customer service. |

**3.4 PROBLEM SOLUTION FIT**

****

**4. REQUIREMENT ANALYSIS**

**4.1 FUNCTIONAL REQUIREMENT**

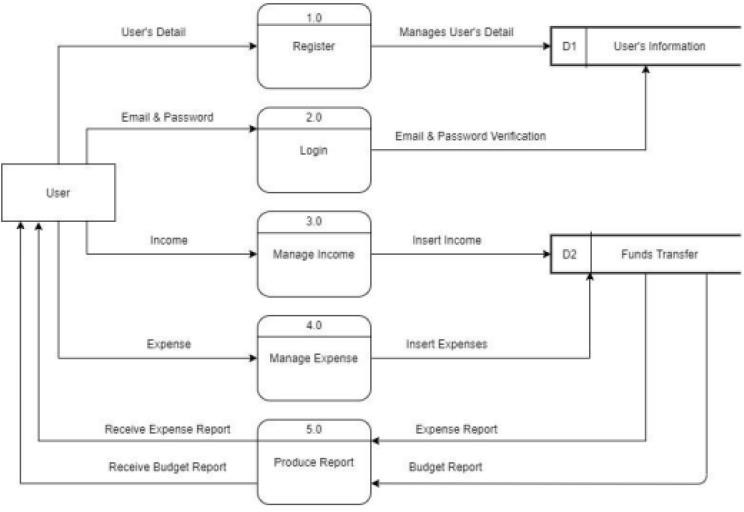
Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User {Registration} | Registration through Form  Registration through |
| FR-2 | User Confirmation | Confirmation via Email  Confirmation via OTP |
| FR-3 | User Login | Login through User name and password. |
| FR-4 | User Financial Accounts | Account Details  Verification of Details. |
| FR-5 | Add Expense | Add expense made which includes date, time and type of expenses. |
| FR-6 | Edit Expenses | User facilitates to edit the expense which they added previously.  Can edit amount, mode of payment or the expense made.  Delete the expense if it is not made. |
| FR-7 | Expenses reach budget | Alert Message through mail. |
| FR-8 | Monitoring of expenses | Using pie-chart user can analyse their expenses on the daily, monthly or even yearly basis. |
| FR-9 | Database | Usage of standard database for storing the data. |

**4.2 NON-FUNCTIONAL REQUIREMENT**

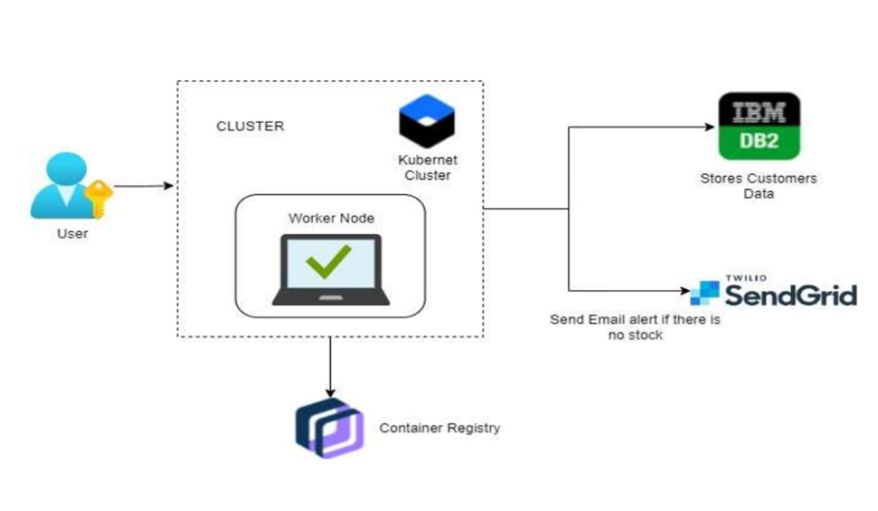
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | Usability | • By using this application, the user can keep track of their expenses and can ensure that user's money is used wisely.  • It can use by wide variety of client as it is very simple to learn and not complex to proceed  • Easy to use, User-friendly and Responsive. |
| NFR-2 | Security | Applications have been developed to help users track and manage their expense related to their own products. The System will ask user to create their accounts by providing essential details. Users can access their accounts by logging into the application. With Registered Mail id only retailers can log into the application. So it provide authentication.  • We are using login for the user and the information will be hashed so that it will be very secure to use |
| NFR-3 | Reliability | • It will be reliable that it can update with very time period so that the accuracy will be good. |
| NFR-4 | Performance | • User can track the record of goods available using the application. Inventory tracking helps to improve inventory management and ensures that having optimal stock available to fulfill orders.Reduces manpower , cost and saves time. Emails will be sent automatically While stocks are not available.Makes the business process more efficient.Improves organizations performance.  • It will be perform fast and secure even at the lower bandwidth. |
| NFR-5 | Availability | • The availability of product is just one way in which an inventory management system creates customer satisfaction. Inventory management systems are designed to monitor product availability, determine purchasing schedules for better customer interaction.  • Prediction will be available for every user but only for premium user news,database and price alert will be alert |
| NFR-6 | Scalability | • Rely on your budgeting app to track, streamline, and automate all the recurrent expenses and remind you on a timely basis.  • It is scalable that we are going to use data in kilobytes so that the quite amount of storage is satisfied |

**5. PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAMS**

**5.2 SOLUTION AND TECHNICAL ARCHITECHTURE**

Figure : Technical Architechture



**5.3 USER STORIES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Balaji N R |
| Sprint-1 |  | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High | Balamurugan V |
| Sprint-1 | Login | USN-3 | As a user, I can register for the application through Gmail | 1 | High | Aravindh Raj N |
| Sprint-1 | Dashboard | USN-4 | As a user, I can log into the application by entering email & password | 2 | High | Prakash S |
| Sprint-2 | Workspace | USN-1 | Workspace for personal expense tracking | 2 | High | Balaji N R |
| Sprint-2 | Charts | USN-2 | Creating various graphs and statistics of customer’s data | 1 | Medium | Balamurugan V |
| Sprint-2 | Connecting to IBM DB2 | USN-3 | Linking database with dashboard | 2 | High | Aravindh raj N |
| Sprint-2 |  | USN-4 | Making dashboard interactive with JS | 2 | High | Prakash S |
| Sprint-3 |  | USN-1 | Wrapping up the server side works of frontend | 1 | Medium | Balaji N R |
| Sprint-3 | Watson Assistant | USN-2 | Creating Chatbot for expense tracking and for calrifying user’s query | 1 | Medium | Aravindh Raj N |
| Sprint-3 | SendGrid | USN-3 | Using SendGrid to send mail to the user about | 1 | Low | Balamurugan V |
| Sprint-3 |  | USN-4 | Integrating both frontend and backend | 2 | High | Prakash S |
| Sprint-4 | Docker | USN-1 | Creating image of website using docker | 2 | High | Balaji N R |
| Sprint-4 | Cloud Registry | USN-2 | Uploading docker image to IBM Cloud registry | 2 | High | Balamurugan V |
| Sprint-4 | kubernetes | USN-3 | Create container using the docker image and hosting the site | 2 | High | Prakash S |
| Sprint-4 | Exposing | USN-4 | Exposing IP/Ports for the site | 2 | High | Aravindh raj N |

**6. PROJECT PLANNING & SCHEDULING**

**6.1 SPRINT PLANNING AND ESTIMATION**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement**  **(Epic)** | **User Story Number** | **User Story/Task** | **Story Points** | **Priority** | **Release** |
| Sprint 1 | Registration | USN – 1 | User can create an account by providing business mail id and password | 5 | High | 1,2,3,4,5 |
| Sprint 2 | Registration Login | USN – 2 | Two step authentication using one time password to provide mail id or phone number | 10 | High | 1,2,3,4,5 |
| Sprint 1 | Login | USN – 3 | Using registered mail Id | 5 | High | 1,2,3,4,5 |
| Sprint 1 | Main dashboard | USN – 4 | User need to complete account settings like giving the details about their inventory and their branches | 10 | High | 1,2,3,4,5 |
| Sprint 2 | Hub maintenance | USN – 5 | User can able to create a separate account for individual hub and he can able to create access policy to share their account with their hub managers | 10 | High | 1,2,3,4,5 |
| Sprint 3 | Hub dashboard login | USN – 6 | Hub mangers can able to login to the account to access their allotted hub details | 10 | High | 1,2,3,4,5 |
| Sprint 3 | Hub dashboard | USN – 7 | Hub mangers can able to add product details and production details. They can also provide access to their allotted space to others. | 10 | High | 1,2,3,4,5 |
| Sprint 4 | Communication system | USN - 8 | User and hub mangers can get the details of the stock moment via mail or chat bot . | 20 | Medium | 1,2,3,4,5 |

**6.2 SPRINT DELIVERY SCHEDULE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 23 Oct 2022 | 28 Oct 2022 | 20 | * 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 30 Oct 2022 | 04 Nov 2022 | 20 | * 05 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 06 Nov 2022 | 11 Nov 2022 | 20 | * 12 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 13 Nov 2022 | 18 Nov 2022 | 20 | * 19 Nov 2022 |

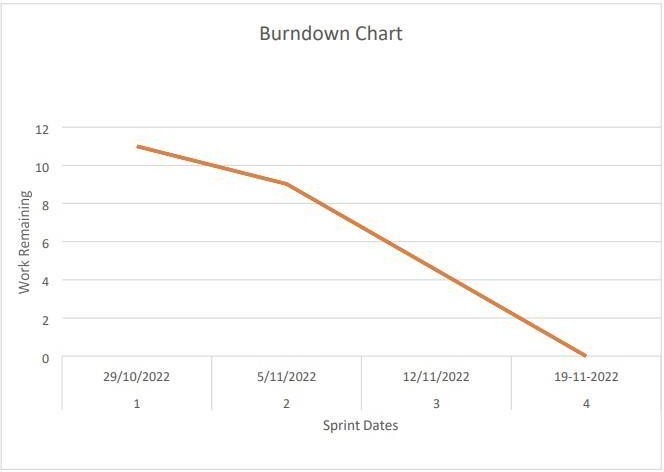
**Velocity:**

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team’s average velocity (AV) per iteration unit (story points per day).

**𝐴𝑉 = sprint duration / velocity = 20/6 = 3.33**

**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**

Jira Software enables teams to make data-driven decisions with agile reports, dashboards, and more. Reports in Jira software offer critical insights for scrum, kanban, and any agile methodology in between.Deliver value to customers faster with real-time insights at your fingertips. Jira Software enables teams to make data-driven decisions with agile reports, dashboards, and more**.**

**Features:**

1.As you plan:

Plan smarter sprints with insights in the backlog. 2.During your sprint:

Stay on target to meet your goals with insights right in the board view. 3.When you ship

Optimize your delivery pipeline with deployment frequency and cycle time insights.

Improve delivery and performance with agile reports:

Reports in Jira software offer critical insights for scrum, kanban, and any agile methodology in between, Reports for scrum teams.

Stay on track of sprint goals and improve retrospectives with data scrum teams can put to use sprint over sprint.

1. Sprint report

Determine overcommittment and excessive scope creep and understand completed work in each sprint.

2. Burndown chart

Track progress towards sprint goals to manage progress and respond accordingly. 3.Release burndown

Track and monitor the projected release date for versions and take action if work is falling behind projected schedule.

4.Velocity chart

Track work from sprint to sprint to helps teams determine the velocity and better estimate the work a team realistically achieve in future sprints.

Optimize kanban flow for continuous delivery

Better predict future performance and spot bottlenecks with agile reports for kanban teams. 1.Cumulative flow diagram

Easily spot blockages by seeing the number of issues that increase in any given state. 2.Control chart

Determine future performance with cycle and lead times for your product, version, or sprint.

**7. CODING AND SOLUTIONING**

**7.1 FEATURE 1**

**Description**

User can add expense by logging into their account

**Source Code**

@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)

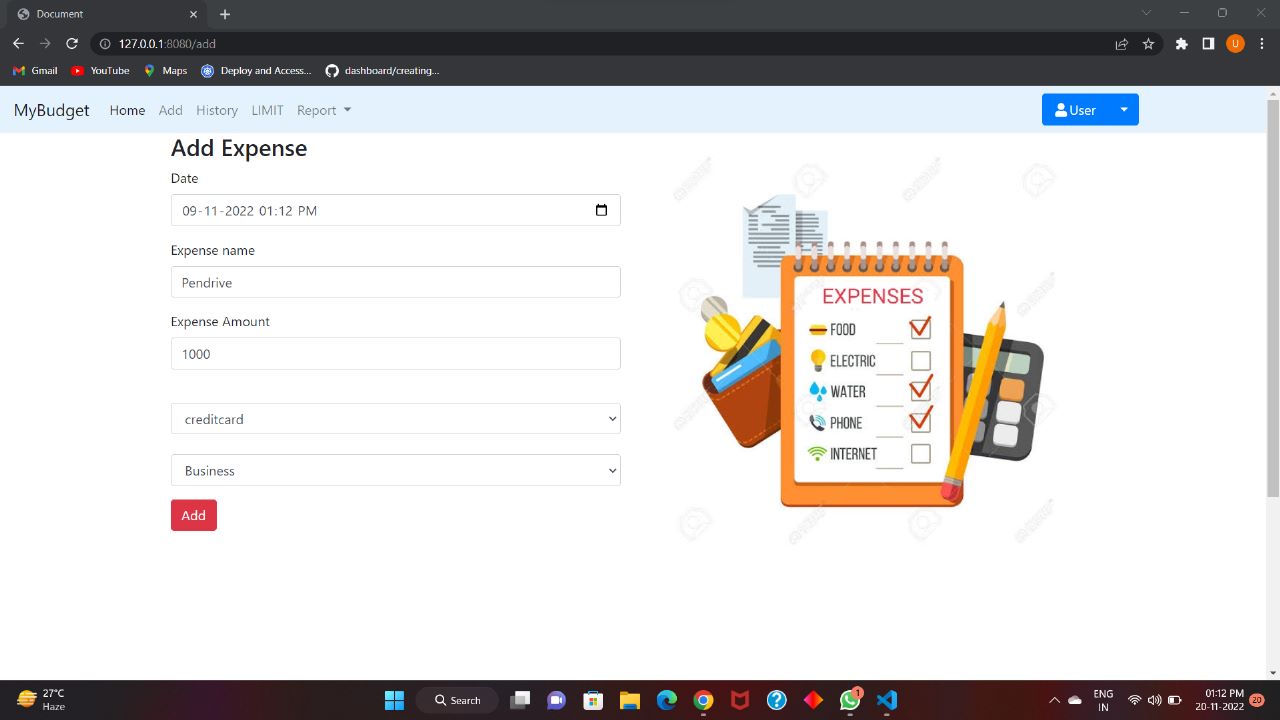
    ibm\_db.bind\_param(stmt, 5, paymode)

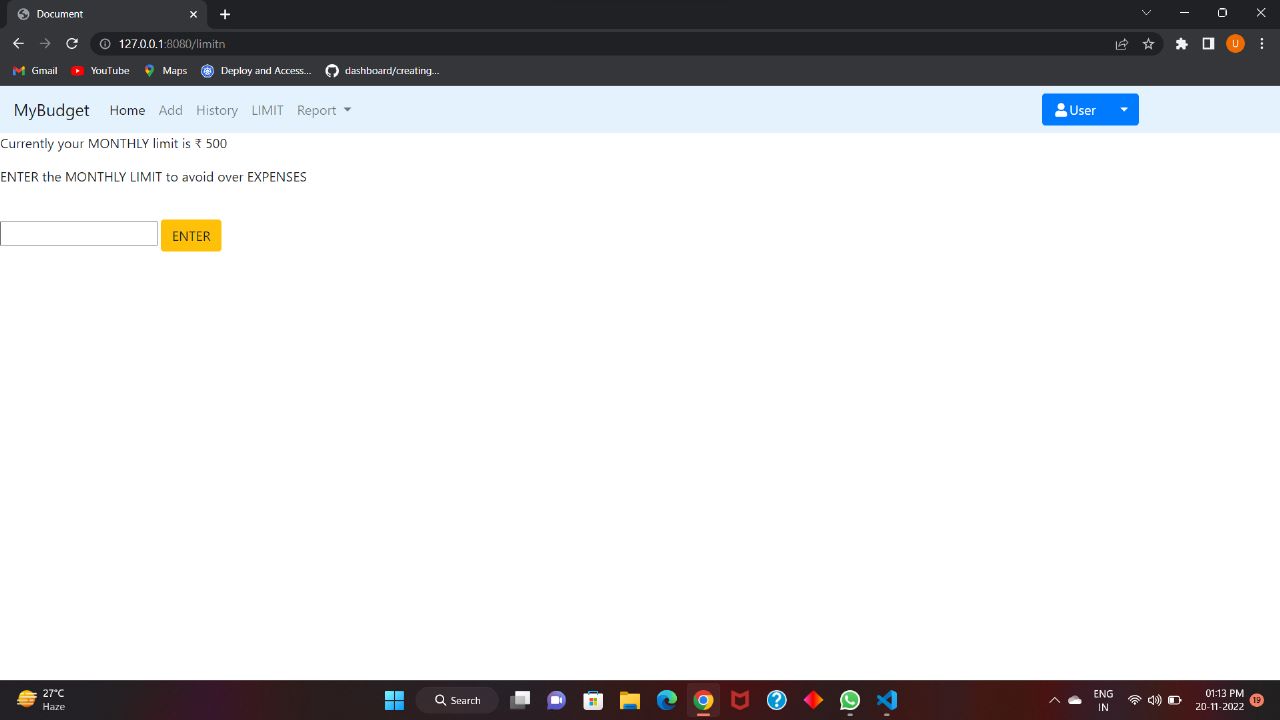
    ibm\_db.bind\_param(stmt, 6, category)

    ibm\_db.execute(stmt)

    print("Expenses added")

**Screenshots**





**7.2 FEATURE 2**

**Description**

Limit the user expense according to the budget.

**Source Code**

**app.py**

@app.route("/today")

def today():

      param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"

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

      dictionary1 = ibm\_db.fetch\_assoc(res1)

      texpense = []

      while dictionary1 != False:

          temp = []

          temp.append(dictionary1["TN"])

          temp.append(dictionary1["AMOUNT"])

          texpense.append(temp)

          print(temp)

          dictionary1 = ibm\_db.fetch\_assoc(res1)

    #   cursor = mysql.connection.cursor()

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

    #   expense = cursor.fetchall()

      param = "SELECT \* FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(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)

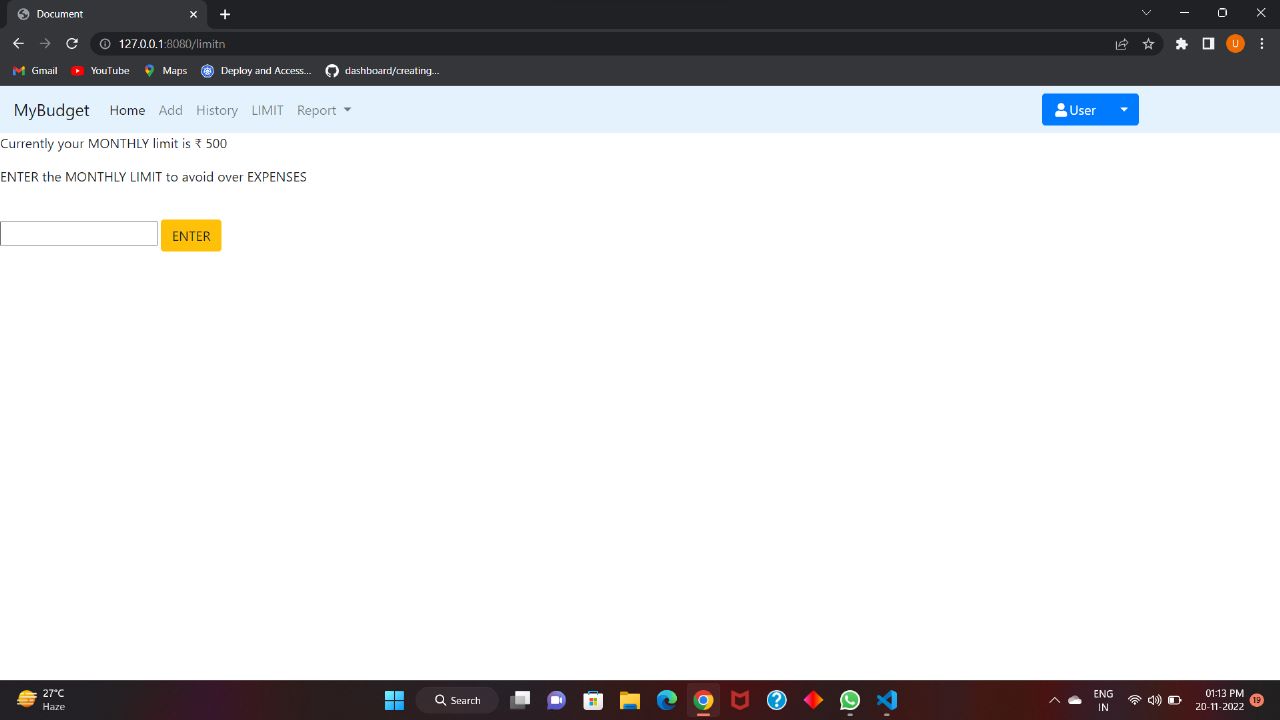
      print(t\_other)

      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 )



@app.route("/limit" )

def limit():

       return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])

def limitnum():

     if request.method == "POST":

         number= request.form['number']

         sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"

         stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

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

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

         ibm\_db.execute(stmt)

         return redirect('/limitn')

@app.route("/limitn")

def limitn():

    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 = " /-"

    while dictionary != False:

        temp = []

        temp.append(dictionary["LIMITSS"])

        row.append(temp)

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

        s = temp[0]

    return render\_template("limit.html" , y= s)

**Update Expenses:**

@app.route('/update/<id>', methods = ['POST'])

def update(id):

  if request.method == 'POST' :

      date = request.form['date']

      expensename = request.form['expensename']

      amount = request.form['amount']

      paymode = request.form['paymode']

      category = request.form['category']

    #   cursor = mysql.connection.cursor()

    #   cursor.execute("UPDATE `expenses` SET `date` = % s , `expensename` = % s , `amount` = % s, `paymode` = % s, `category` = % s WHERE `expenses`.`id` = % s ",(date, expensename, amount, str(paymode), str(category),id))

    #   mysql.connection.commit()

      p1 = date[0:10]

      p2 = date[11:13]

      p3 = date[14:]

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

      sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?, category = ? WHERE id = ?"

      stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

      ibm\_db.bind\_param(stmt, 1, p4)

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

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

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

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

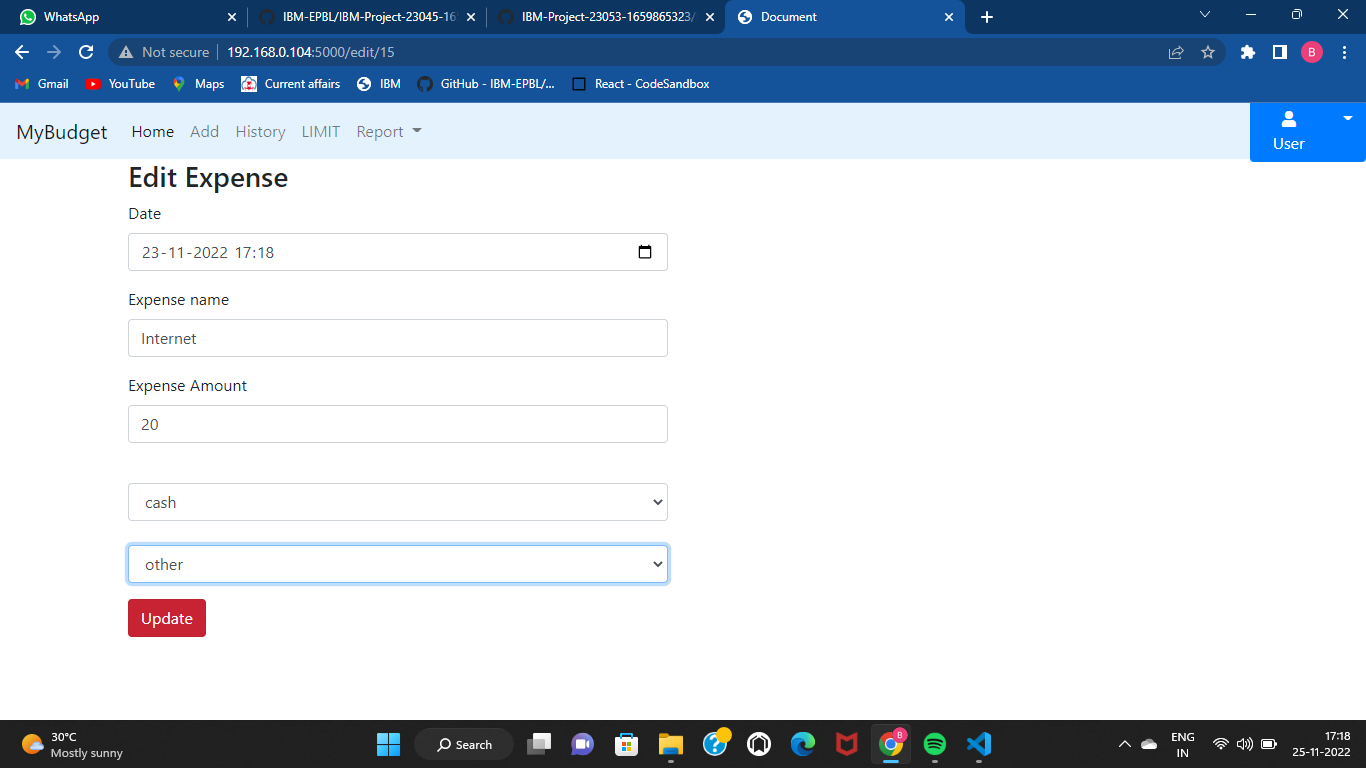
      ibm\_db.bind\_param(stmt, 6, id)

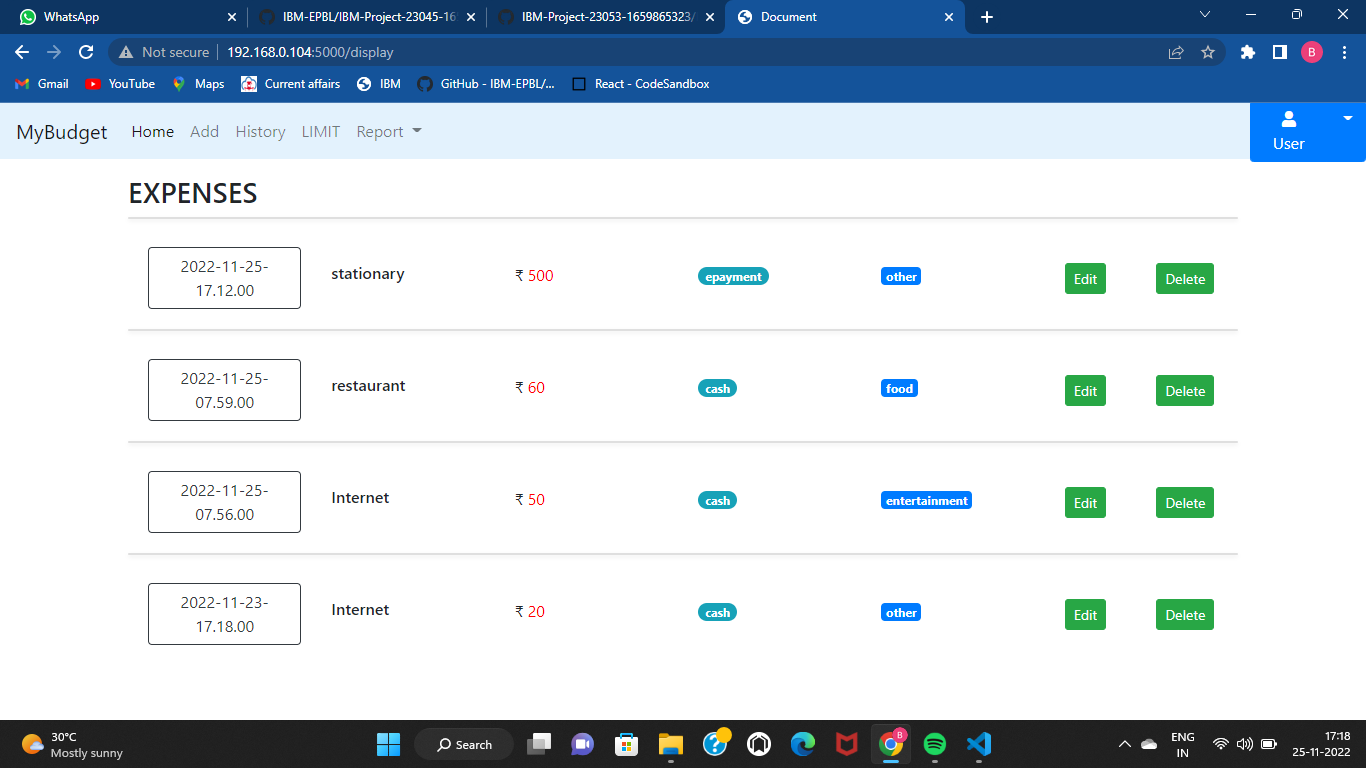
      ibm\_db.execute(stmt)

      print('successfully updated')

      return redirect("/display")

**Screenshots**





**7.3 FEATURE 3**

 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. " + str(s) + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Experte"

        #sendmail(msg,session['email'])

        sg = sendgrid.SendGridAPIClient(api\_key='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')

        from\_email = Email("balajinrcse2022@gmail.com")

        cusmail = session['email']

        to\_email = To(cusmail)

        content = Content("text/html", msg)

        subject = "Limit alert !!! - Experte"

        mail = Mail(from\_email, to\_email, subject, content)

        mail\_json = mail.get()

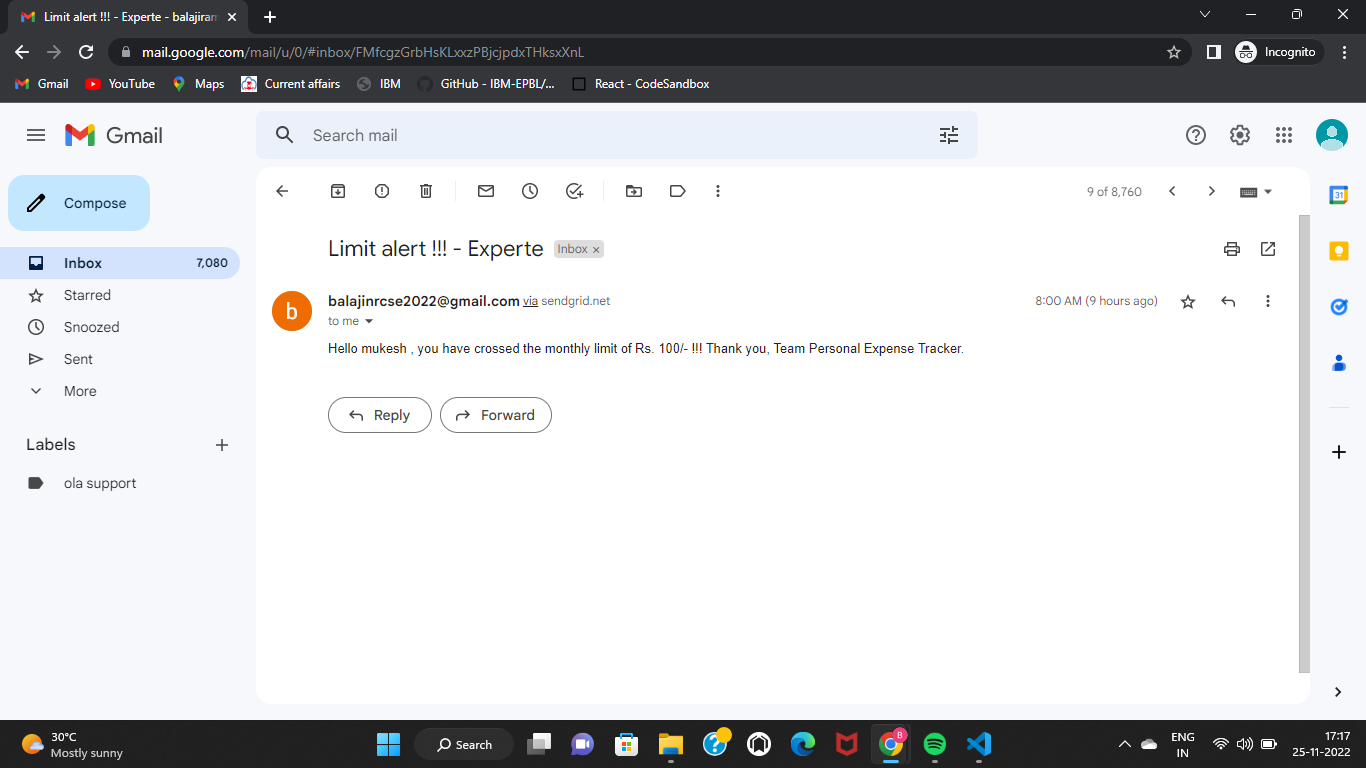
        response = sg.client.mail.send.post(request\_body=mail\_json)

        print(response.status\_code)

        print(response.headers)

    return redirect("/display")

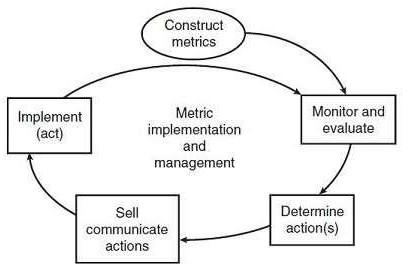
**Screenshots**



**8. TESTING**

**8.1 TEST CASES**

**PERFORMANCE TESTING:**



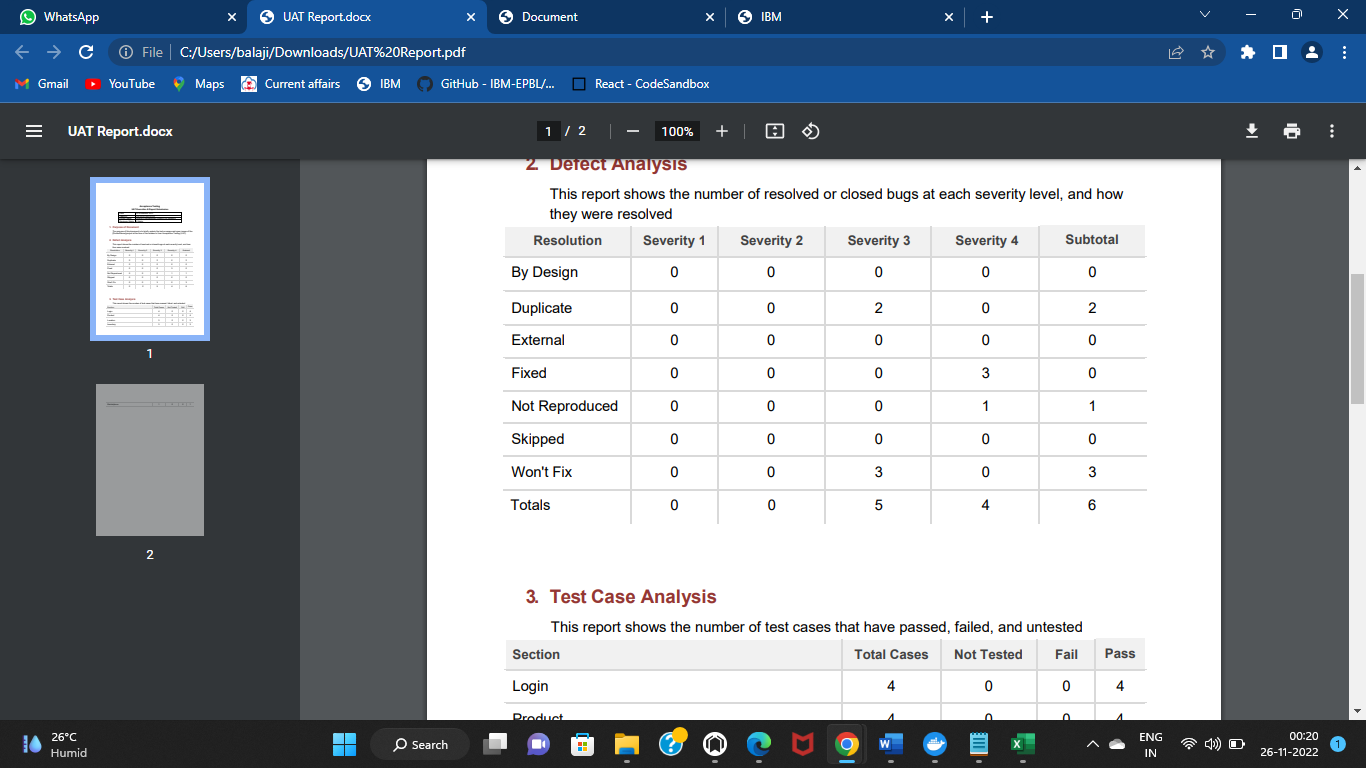
**8.2 User Acceptance Testing:**

**1.Purpose of Document:**

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

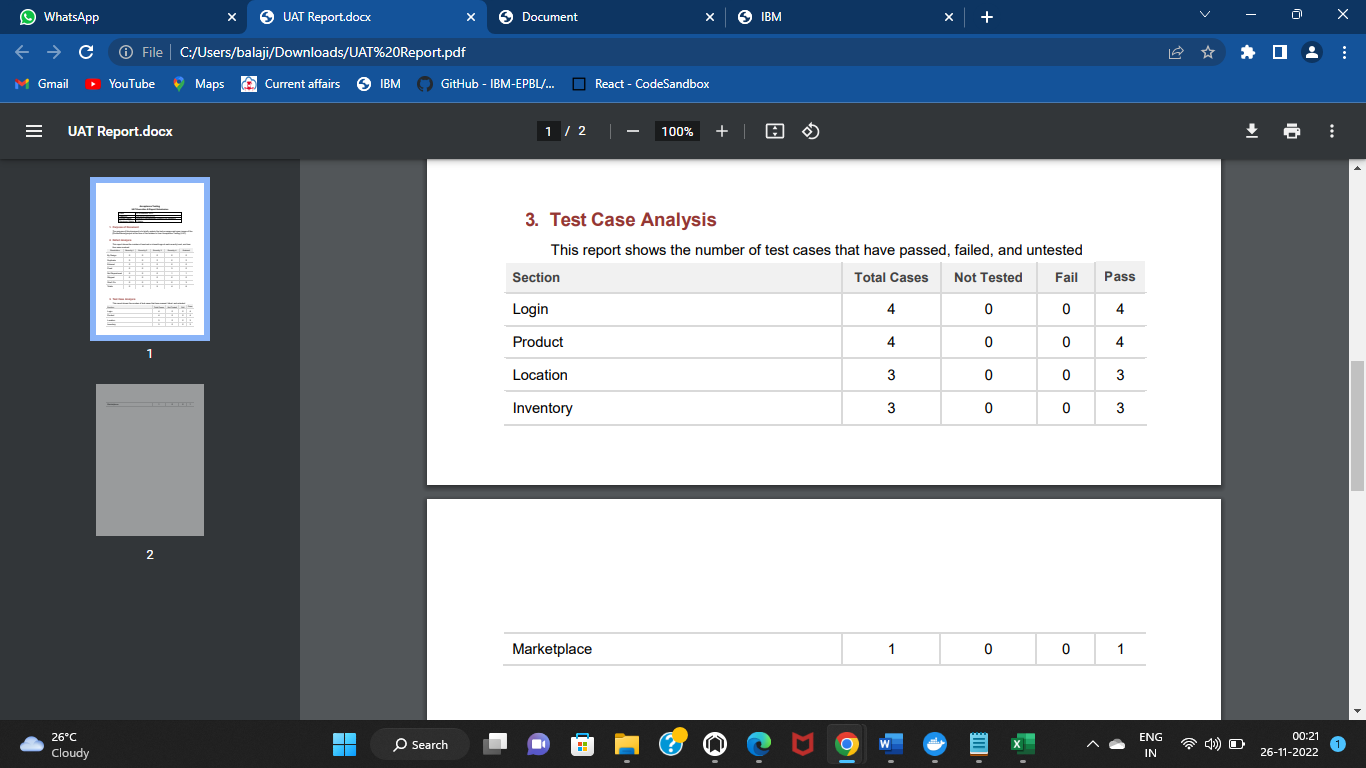
**2. Defect Analysis:**

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

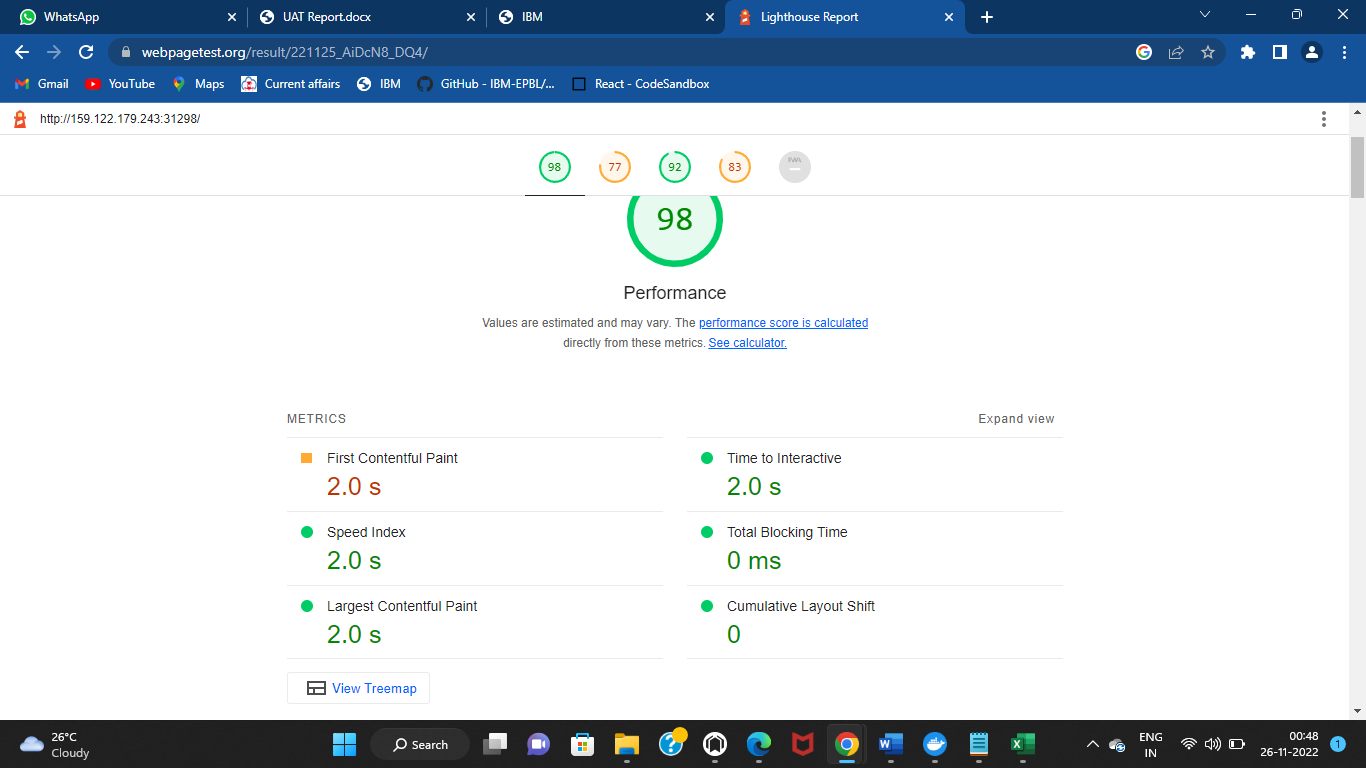


**3. Test Case Analysis:**

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



1. **RESULTS**
   1. **Performance Metrics:**



# 10.ADVANTAGES & DISADVANTAGES

## Advantages

1. **You will spend mindfully**

When you write down every expense it helps you spend more mindfully and prevents you from splurging. It makes you responsible with your spending.

## Making financial control

When you track your expenses, you take complete control over your finances. At any one time, you will know exactly how much money is sitting in your bank account, and how much you can spend.

## Identify problem areas

As you track your spending over time, you'll get a better idea of what's happening with your cash. Many of your daily expenses may seem really, but once you add up everything you spend on dining out, coffee, lottery tickets, or whatever your indulgence is, you may be shocked to find out how much your habits actually cost.

## Make a better budget

By tracking your expenses it will help you make clear budgets for your monthly spends. After you set up a budget, which is a monthly plan for spending that takes into account your income and expenses, tracking expenses daily is essential to keeping you on that budget

## Tracking your financial progress

Tracking your expenses on a day-to-day basis helps you to see your progress on the road to your financial goals.

Tracking your expenses on a day-to-day basis helps you to see your progress on the road to your financial goals.

## Keeping finances organized

Disorganized finances lead to financial problems. It is easier to stay organized than it is to organize a messy financial situation.

## Improving financial security

It helps you track your bank accounts. What if somebody steals your debit card information and starts spending your money? If you have a track on your spends you avoid these risks.

## Encourages and increases savings

When you track your expenses you are likely to find wasteful expenses you can eliminate. This will help you encourage and increase your savings. By eliminating wasteful expenses it opens up the opportunity to redirect that money into savings.

## Avoids debt

Tracking your expenses can be a powerful motivator to steer clear of debt. When you are in debt, and not tracking your day-to-day expenses, it’s easy to let the amount of debt you are paying each month slip through the cracks, unnoticed. But, once you start tracking every dollar that leaves your bank account, you will start to add up the debt payments, and it can be eye-opening.

## Disadvantages

1. Your information may be less secure, and probably being used and sold. If the service is free, then the product is you.
2. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere.
3. Now, you shouldn’t have to worry about credit card fraud, these companies are large enough and secure enough that you’ll never have to worry about something like that.

# 11.CONCLUSION

The personal expense tracker application successfully avoids the manual calculation for avoiding calculating the income and expense per month. Monitoring your everyday expenses can set aside you cash, yet it can likewise help you set your monetary objectives for what’s to come. On the off chance that you know precisely where your sum is going much of a stretch see where a few reductions and bargains can be made. Expense Tracker project is for keeping our day-to-day expenditures will helps us to keep record of our money daily. The project what we have created is work more proficient than the other income and expense tracker. The project effectively keeps away from the manual figuring for trying not to ascertain the pay and cost each month. It’s a user- friendly application.

# 12.FUTURE SCOPE

It will have various options to keep record (for example Food, Travelling Fuel, Salary etc). Automatically it will keep on sending notifications for our daily expenditure. In today’s busy and expensive life, we are in a great rush to make money, but at the end of the month we broke off. As we are unknowingly spending money on title and unwanted things. So, we have come over with the plan to follow our profit.

# 13.APPENDIX

# 13.1 SOURCE CODE:

# App.py

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

# import re

# from flask\_db2 import DB2

# import ibm\_db

# import ibm\_db\_dbi

# from sendgrid import \*

# import os

# app = Flask(\_\_name\_\_)

# app.secret\_key = 'a'

# 

# """

# dsn\_hostname = "ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"

# dsn\_uid = "vmk08423"

# dsn\_pwd = "3KfJl6HGDtPdbIWy"

# dsn\_driver = "{IBM DB2 ODBC DRIVER}"

# dsn\_database = "bludb"

# dsn\_port = "31321"

# 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'] = '8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud'

# app.config['port'] = '30120'

# app.config['protocol'] = 'tcpip'

# app.config['uid'] = 'fsd14997'

# app.config['pwd'] = '7JR2ia5UzeAseRvL'

# app.config['security'] = 'SSL'

# try:

# mysql = DB2(app)

# conn\_str='DATABASE=bludb;HOSTNAME=8e359033-a1c9-4643-82ef-8ac06f5107eb.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;SECURITY=SSL;PORT=30120;PROTOCOL=TCPIP;UID=fsd14997;PWD=7JR2ia5UzeAseRvL'

# 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")

# 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)

# 

# 

# 

# 

# #LOGIN--PAGE

# 

# @app.route("/signin")

# def signin():

# return render\_template("login.html")

# 

# @app.route('/login',methods =['GET', 'POST'])

# def login():

# global userid

# msg = ''

# 

# 

# if request.method == 'POST' :

# username = request.form['username']

# password = request.form['password']

# # cursor = mysql.connection.cursor()

# # cursor.execute('SELECT \* FROM register WHERE username = % s AND password = % s', (username, password ),)

# # account = cursor.fetchone()

# # print (account)

# 

# sql = "SELECT \* FROM register WHERE username = ? and password = ?"

# stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

# ibm\_db.bind\_param(stmt, 1, username)

# ibm\_db.bind\_param(stmt, 2, password)

# result = ibm\_db.execute(stmt)

# print(result)

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

# print(account)

# 

# param = "SELECT \* FROM register WHERE username = " + "\'" + username + "\'" + " and password = " + "\'" + password + "\'"

# res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

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

# # sendmail("hello sakthi","sivasakthisairam@gmail.com")

# if account:

# session['loggedin'] = True

# session['id'] = dictionary["ID"]

# userid = dictionary["ID"]

# session['username'] = dictionary["USERNAME"]

# session['email'] = dictionary["EMAIL"]

# 

# return redirect('/home')

# else:

# msg = 'Incorrect username / password !'

# 

# 

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

# 

# #ADDING----DATA

# @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. " + str(s) + "/- !!!" + "\n" + "Thank you, " + "\n" + "Team Experte"

# #sendmail(msg,session['email'])

# sg = sendgrid.SendGridAPIClient(api\_key='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')

# from\_email = Email("balajinrcse2022@gmail.com")

# cusmail = session['email']

# to\_email = To(cusmail)

# content = Content("text/html", msg)

# subject = "Limit alert !!! - Experte"

# 

# mail = Mail(from\_email, to\_email, subject, content)

# mail\_json = mail.get()

# response = sg.client.mail.send.post(request\_body=mail\_json)

# 

# print(response.status\_code)

# print(response.headers)

# 

# return redirect("/display")

# #DISPLAY---graph

# @app.route("/display")

# def display():

# print(session["username"],session['id'])

# 

# 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)

# @app.route('/delete/<string:id>', methods = ['POST', 'GET' ])

# def delete(id):

# param = "DELETE FROM expenses WHERE id = " + id

# res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

# print('deleted successfully')

# return redirect("/display")

# #UPDATE---DATA

# @app.route('/edit/<id>', methods = ['POST', 'GET' ])

# def edit(id):

# # cursor = mysql.connection.cursor()

# # cursor.execute('SELECT \* FROM expenses WHERE id = %s', (id,))

# # row = cursor.fetchall()

# param = "SELECT \* FROM expenses WHERE id = " + id

# res = ibm\_db.exec\_immediate(ibm\_db\_conn, param)

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

# row = []

# 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"])

# row.append(temp)

# print(temp)

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

# print(row[0])

# return render\_template('edit.html', expenses = row[0])

# @app.route('/update/<id>', methods = ['POST'])

# def update(id):

# if request.method == 'POST' :

# 

# date = request.form['date']

# expensename = request.form['expensename']

# amount = request.form['amount']

# paymode = request.form['paymode']

# category = request.form['category']

# p1 = date[0:10]

# p2 = date[11:13]

# p3 = date[14:]

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

# sql = "UPDATE expenses SET date = ? , expensename = ? , amount = ?, paymode = ?, category = ? WHERE id = ?"

# stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

# ibm\_db.bind\_param(stmt, 1, p4)

# ibm\_db.bind\_param(stmt, 2, expensename)

# ibm\_db.bind\_param(stmt, 3, amount)

# ibm\_db.bind\_param(stmt, 4, paymode)

# ibm\_db.bind\_param(stmt, 5, category)

# ibm\_db.bind\_param(stmt, 6, id)

# ibm\_db.execute(stmt)

# print('successfully updated')

# return redirect("/display")

# 

# @app.route("/limit" )

# def limit():

# return redirect('/limitn')

# @app.route("/limitnum" , methods = ['POST' ])

# def limitnum():

# if request.method == "POST":

# number= request.form['number']

# sql = "INSERT INTO limits (userid, limitss) VALUES (?, ?)"

# stmt = ibm\_db.prepare(ibm\_db\_conn, sql)

# ibm\_db.bind\_param(stmt, 1, session['id'])

# ibm\_db.bind\_param(stmt, 2, number)

# ibm\_db.execute(stmt)

# 

# return redirect('/limitn')

# 

# 

# @app.route("/limitn")

# def limitn():

# 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 = " /-"

# while dictionary != False:

# temp = []

# temp.append(dictionary["LIMITSS"])

# row.append(temp)

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

# s = temp[0]

# 

# return render\_template("limit.html" , y= s)

# #REPORT

# @app.route("/today")

# def today():

# param1 = "SELECT TIME(date) as tn, amount FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(current timestamp) ORDER BY date DESC"

# res1 = ibm\_db.exec\_immediate(ibm\_db\_conn, param1)

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

# texpense = []

# while dictionary1 != False:

# temp = []

# temp.append(dictionary1["TN"])

# temp.append(dictionary1["AMOUNT"])

# texpense.append(temp)

# print(temp)

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

# 

# # cursor = mysql.connection.cursor()

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

# # expense = cursor.fetchall()

# param = "SELECT \* FROM expenses WHERE userid = " + str(session['id']) + " AND DATE(date) = DATE(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 )

# 

# @app.route("/month")

# def month():

# # cursor = mysql.connection.cursor()

# # cursor.execute('SELECT DATE(date), SUM(amount) FROM expenses WHERE userid= %s AND MONTH(DATE(date))= MONTH(now()) GROUP BY DATE(date) ORDER BY DATE(date) ',(str(session['id'])))

# # texpense = cursor.fetchall()

# # print(texpense)

# param1 = "SELECT DATE(date) as dt, SUM(amount) as tot FROM expenses WHERE userid = " + str(session['id']) + " AND MONTH(date) = MONTH(current timestamp) AND YEAR(date) = YEAR(current timestamp) GROUP BY DATE(date) ORDER BY DATE(date)"

# res1 = ibm\_db.exec\_immediate(ibm\_db\_conn, param1)

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

# texpense = []

# while dictionary1 != False:

# temp = []

# temp.append(dictionary1["DT"])

# temp.append(dictionary1["TOT"])

# texpense.append(temp)

# print(temp)

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

# 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

# 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 )

# 

# @app.route("/year")

# def year():

# # cursor = mysql.connection.cursor()

# # cursor.execute('SELECT MONTH(date), SUM(amount) FROM expenses WHERE userid= %s AND YEAR(DATE(date))= YEAR(now()) GROUP BY MONTH(date) ORDER BY MONTH(date) ',(str(session['id'])))

# # texpense = cursor.fetchall()

# # print(texpense)

# 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 )

# #log-out

# @app.route('/logout')

# def logout():

# session.pop('loggedin', None)

# session.pop('id', None)

# session.pop('username', None)

# session.pop('email', None)

# return render\_template('home.html')

# port = os.getenv('VCAP\_APP\_PORT', '8080')

# if \_\_name\_\_ == "\_\_main\_\_":

# app.secret\_key = os.urandom(12)

# app.run(host='0.0.0.0', port=5000, debug=True)

# 13.2 GitHub Repository for Templates:

# <https://github.com/IBM-EPBL/IBM-Project-23053-1659865323>

# Project Demo Link:

# <https://drive.google.com/file/d/1E1JwnWW6kr_vGRbg4TyiiJWerRiR1NvN/view?usp=sharing>