Personal Expense Tracker Application

1. INTRODUCTION

1.1 Project Overview

Expense tracker is an android/web based application. This application allows the user to maintain a computerized diary. Expense tracker application which will keep a track of Expenses of a user on a dayto-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also have a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation.

1.2 Purpose

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

The Expense tracker existing system does not provide the user portable device management level, existing system only used on desktop software so unable to update anywhere expenses done and unable to update the location of the expense details disruptive that the proposed system provides. In existing, we need to maintain the Excel sheets, CSV files for the user daily, weekly and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenses easily. To do so a person as to keep a log in a diary or in a computer system, also all the calculations need to be done by the user which may sometimes results in mistakes leading to losses. The existing system is not user friendly because data is not maintained perfectly. But this project will not have any reminder to remain a person in a specific date, so that is the only drawback in which the remainder is not present. This project will be an unpopulated information because it has some disadvantages by not remind a person for each and every month. But it can used to perform calculation on income and expenses.

2.2 References

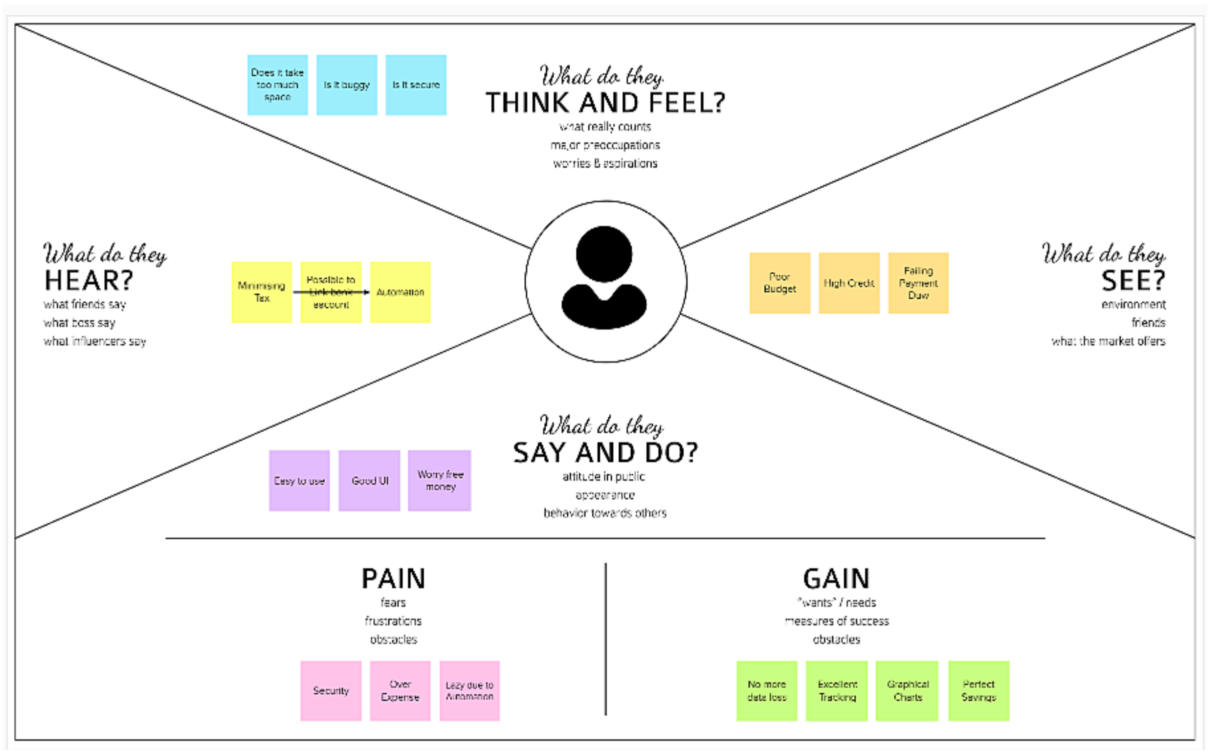
Velmurugan.R, “Expense Tracker Application”, published by IJIRT, March 2021.

2.3 Problem Statement Definition

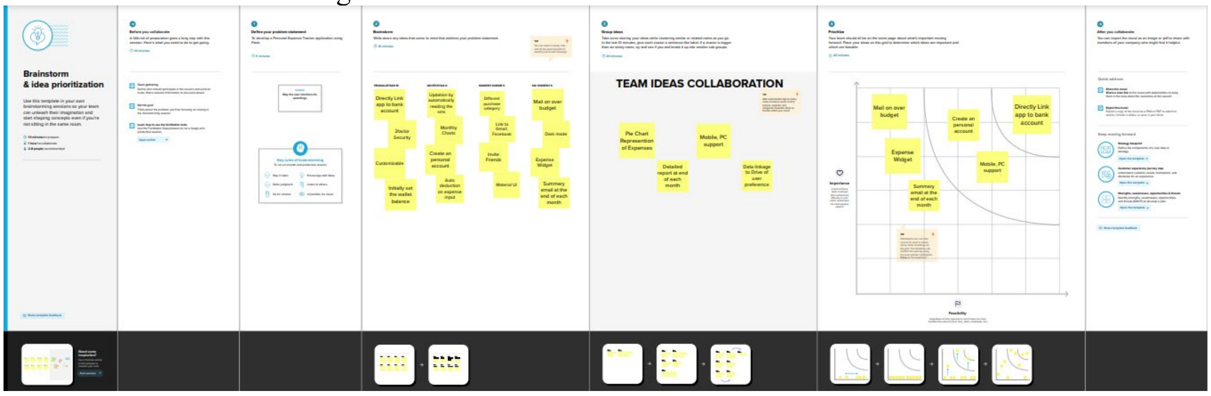
Ideally, a normal person would suffer to keep track of his daily expenses in his mind. This leads to him forget the entire expense history and make him wonder where the money has gone. The proposed expense tracker application makes sure that the expenses are tracked category wise. It also provides monthly, daily and semi-annual expense-based graph analysis of the expense done. The user will also be alerted via an email if the expense goes beyond a certain amount (the amount will be user set) with the help of SendGrid framework.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

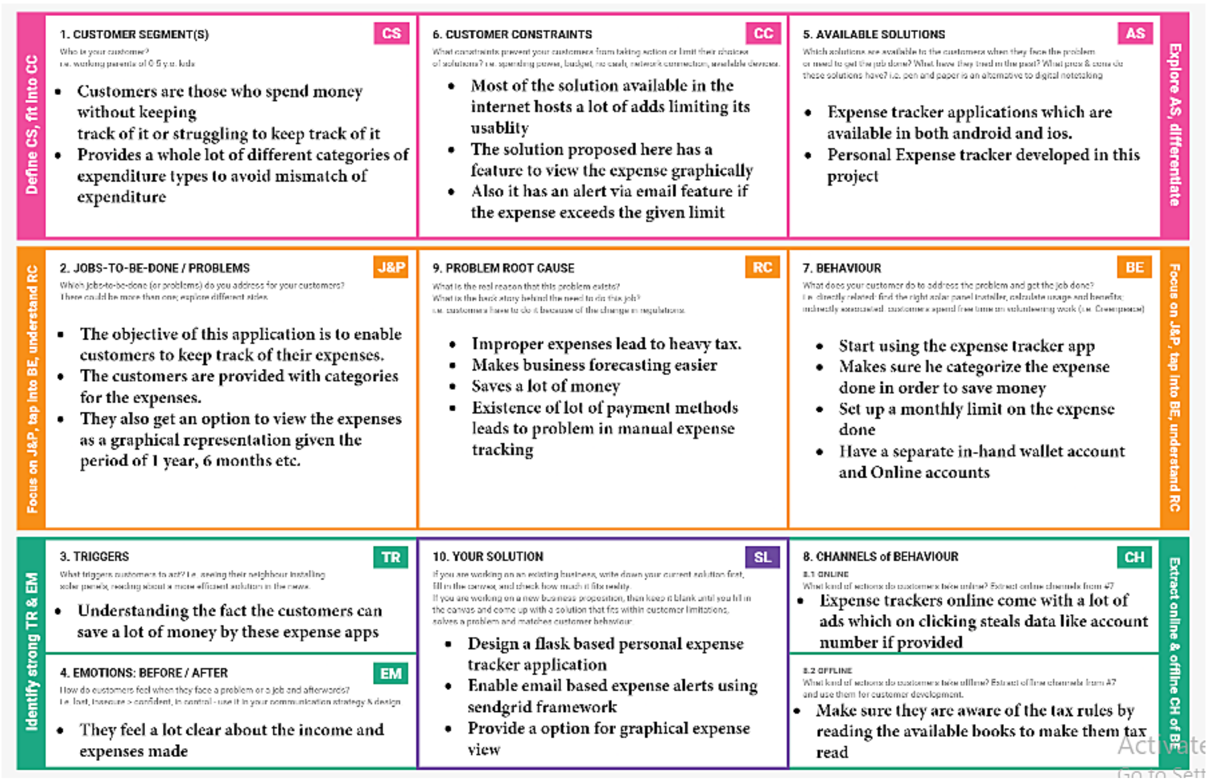


3.3 Proposed Solution

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Your entire financial decision-making process is unable to keep track of it. By assisting you in effectively managing your funds, this software makes your life simpler. A personal finance software will assist you with financial management, accounting, and budgeting in addition to budgeting and accounting.
2	Idea / Solution description	Personal finance includes all of the financial choices and tasks that a finance software streamlines by assisting you in effectively managing your money. A personal finance software will not only assist you with accounting and budgeting, but it will also provide you with

		valuable advice on money management. 3 Novelty / Uniqueness Display the cost
3	Novelty / Uniqueness	Display the costs on a monthly and weekly basis in a pie chart.
4	Social Impact / Customer Satisfaction	People can use it to keep track of their spending and receive alerts when their budget is exceeded.
5	Business Model (Revenue Model)	We can offer the programme on a subscription basis.
6	Scalability of the Solution	Future customers of IBM Cloud will automatically receive storage.

3.4 Problem Solution fit


Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

- Customers are those who spend money without keeping
  track of it or struggling to keep track of it
- Provides a whole lot of different categories of expenditure types to avoid mismatch of expenditure

6. CUSTOMER CONSTRAINTS CC

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

- Most of the solution available in the internet hosts a lot of adds limiting its usablity
- The solution proposed here has a feature to view the expense graphically
- Also it has an alert via email feature if the expense exceeds the given limit

5. AVAILABLE SOLUTIONS AS

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

- Expense tracker applications which are available in both android and ios.
- Personal Expense tracker developed in this project

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS J&P

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

- The objective of this application is to enable customers to keep track of their expenses.
- The customers are provided with categories for the expenses.
- They also get an option to view the expenses as a graphical representation given the period of 1 year, 6 months etc.

9. PROBLEM ROOT CAUSE RC

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

- Improper expenses lead to heavy tax.
- Makes business forecasting easier
- Saves a lot of money
- Existence of lot of payment methods leads to problem in manual expense tracking

7. BEHAVIOUR BE

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

- Start using the expense tracker app
- Makes sure he categorize the expense done in order to save money
- Set up a monthly limit on the expense done
- Have a separate in-hand wallet account and Online accounts

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

Identify strong TR & EM

3. TRIGGERS TR

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

- Understanding the fact the customers can save a lot of money by these expense apps

4. EMOTIONS: BEFORE / AFTER EM

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

- They feel a lot clear about the income and expenses made

10. YOUR SOLUTION SL

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

- Design a flask based personal expense tracker application
- Enable email based expense alerts using sendgrid framework
- Provide a option for graphical expense view

8. CHANNELS of BEHAVIOUR CH

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

- Expense trackers online come with a lot of ads which on clicking steals data like account number if provided

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

- Make sure they are aware of the tax rules by reading the available books to make them tax read

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement	Sub Requirement
FR-1	User Registration	This is a form that collects information fromyou.
FR-2	Login	You will need to enter your username andpassword here.
FR-3	Calendar	The user mustbe able to add the information to their spending in a personalexpense tracking application.
FR-4	Expense Tracker	The expense should be graphically represented in this application's reportformat.

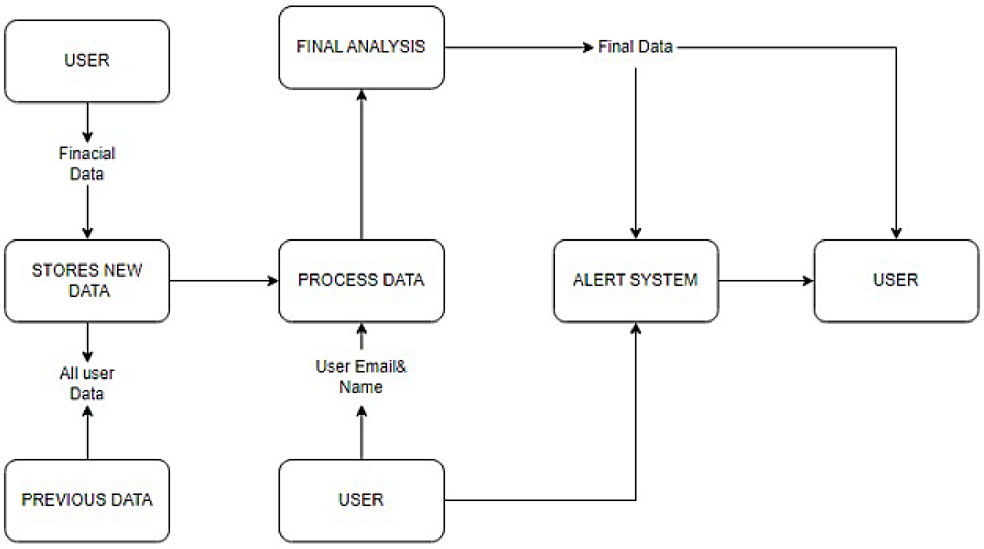
FR-5	Report generation	The report must be representedgraphically.
FR-6	Category	Users of this application will be able to add expense categories.

4.2 Non-Functional requirements

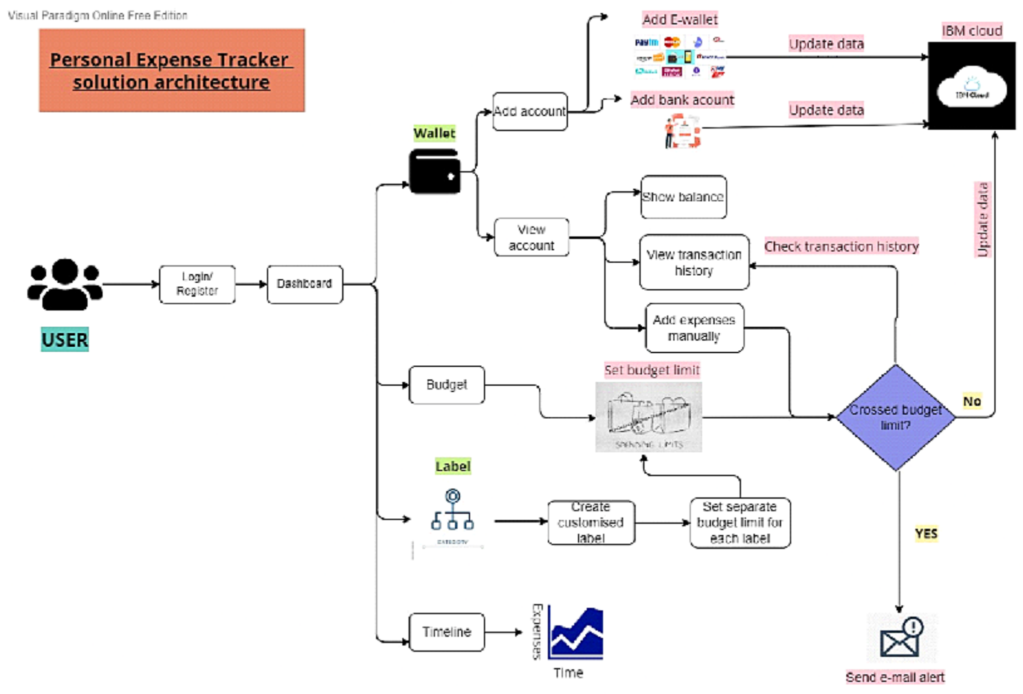
NFR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	keeps an accurate recordof your earnings andoutgoings.
NFR-2	**Security**	adetailed accounting of your income and expenses.
NFR-3	**Performance**	There are categories of expenses as well asan option. Becauseof lightweight database support, the system's throughput is increased.
NFR-4	**Availability**	The application must be completelyoperational at all times.
NFR-5	**Scalability**	The application must always function in its entirety.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

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

	Login	USN-5	As a user,I can log into the appl ication byentering my email & password	I can acces stheapplica tion.	High	Sprint-1
	Dashboard	USN-6	As a user, I can see the expendit ure detailsand the daily expense.	I can view the daily expenses an d add theexpense details.	High	Sprint-1
Customer CareExecutive		USN-7	As a customer careexecutive, I can solvetheproblem that customers face.	I can provide support to customers at any ti me24*7.	Medium	Sprint-1
Administrator	Application	USN-8	As an administrator, I can upgrade orupdate theapp lication.	I can fix any bugs rais edby customers and upgrade theapplic ation.	Medium	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

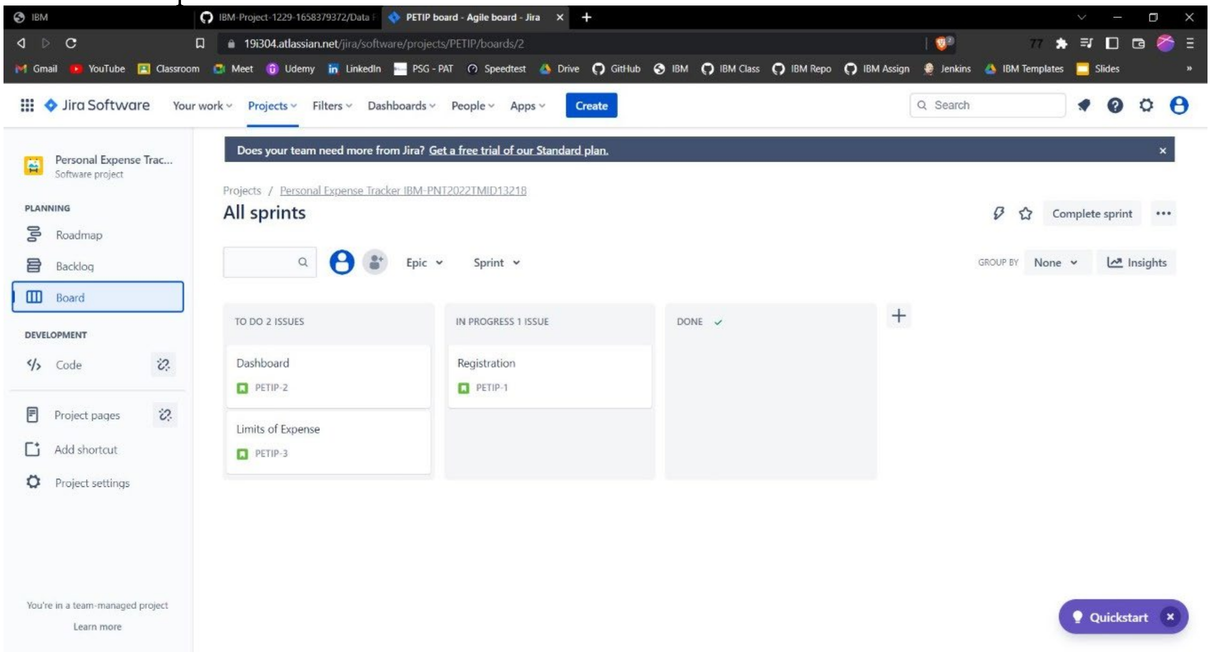
Sprint	**Functional Requirement (Epic)**	**User StoryNumber**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Registration	USN-1	As a user, I can register for the appli cation byentering my email, password, and confirmingmy password.	2	High	MageshKumar
Sprint-1		USN-2	As a user, I will receive confirmation email onceI have registered for the application	1	High	MageshKumar
Sprint-2		USN-3	As a user,I can register for the ap plicationthrough Facebook	2	Low	Adhithiyaa
Sprint-1		USN-4	As a user,I can register for the ap plicationthrough Gmail	2	Medium	Pragalathan
Sprint-1	Login	USN-5	As a user, I can log intothe application bye ntering email & password	1	High	Pragalathan
Sprint-3	Dashboard	USN-6	As a userI can see the expenditure details onthe applicat ion	3	High	Adhithiyaa

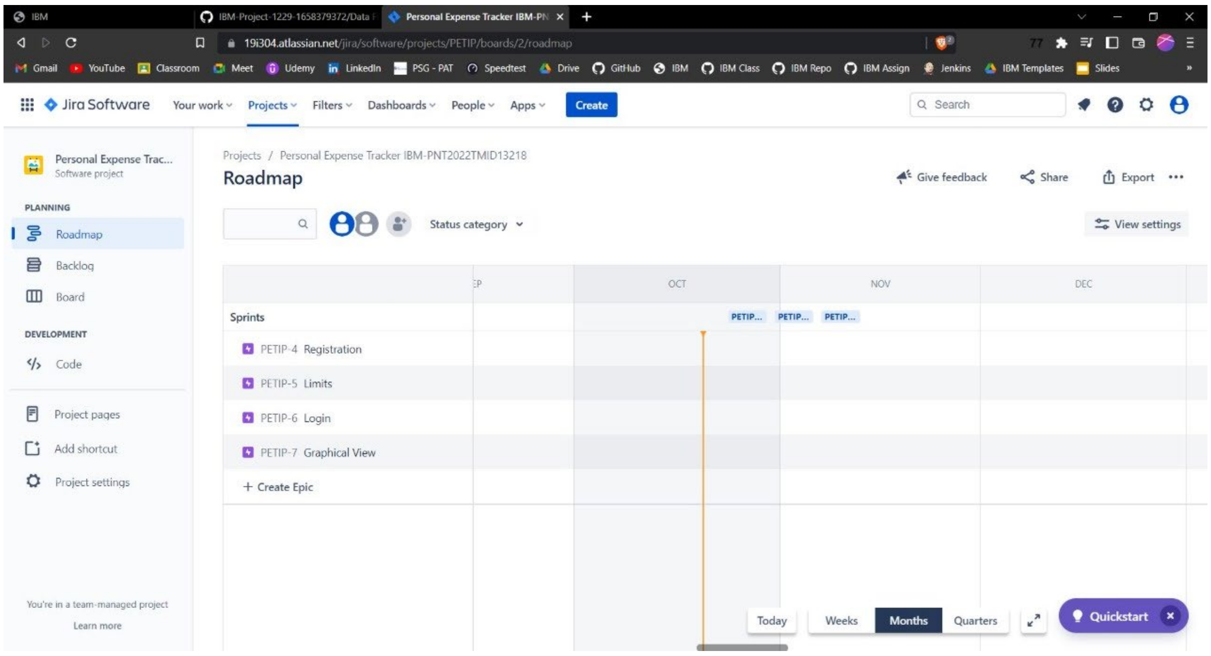
Sprint-3	Limits	USN-6	As a userI can set my monthlyexpense limit sothat I receive a mail on exceeding that	4	High	Sai Vignesh
Sprint-4	Reports	USN-6	As a user I can view the graphical formof myexpenses category wise	5	Medium	Pragalathan

6.2 Sprint Delivery Schedule

Sprint	Total StoryPoints	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	2	6 Days	31 Oct 2022	05 Nov 2022	2	05 Nov 2022
Sprint-3	7	6 Days	07 Nov 2022	12 Nov 2022	7	12 Nov 2022
Sprint-4	5	6 Days	14 Nov 2022	19 Nov 2022	5	19 Nov 2022

6.3 Reports from JIRA





7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1 (Login and Registration Page):

Flask Route File for Login and Registration Page:

```
#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 = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        sql = "SELECT * FROM REGISTER WHERE USERNAME =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account 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:
            sql1="INSERT INTO REGISTER(USERNAME,PASSWORD,EMAIL) VALUES(?,?,?)"
            stmt1 = ibm_db.prepare(conn, sql1)

            ibm_db.bind_param(stmt1,1,username)
            ibm_db.bind_param(stmt1,2,password)
            ibm_db.bind_param(stmt1,3,email)
            ibm_db.execute(stmt1)
            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']
        sql = "SELECT * FROM REGISTER WHERE USERNAME =? AND PASSWORD =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if account:
            session['loggedin'] = True
            session['id'] = account["ID"]
            userid=  account["ID"]
            session['username'] = account["USERNAME"]
            session['email']=account["EMAIL"]

            return redirect('/home')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
```

7.2 Feature 2 (Adding Expense and its CRUD Operations):

This feature enables the user to add expense by stating the date and time of expense, the expense category etc.. Also it enables the user to edit the expenses by clicking on the edit button or can even delete them.

CODE:

```
@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']
  time=request.form['time']

  sql = "INSERT INTO
EXPENSES(USERID,DATE,EXPENSENAME,AMOUNT,PAYMENTMODE,CATEGORY,TIME)
VALUES(?,?,?,?,?,?,?)"
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt,1,session['id'])
  ibm_db.bind_param(stmt,2,date)
  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.bind_param(stmt,7,time)
  ibm_db.execute(stmt)


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

  sql1 = "SELECT * FROM EXPENSES WHERE USERID=? AND
MONTH(date)=MONTH(DATE(NOW()))"
  stmt1 = ibm_db.prepare(conn, sql1)
  ibm_db.bind_param(stmt1,1,session['id'])
  ibm_db.execute(stmt1)
  list2=[]
  expense1 = ibm_db.fetch_tuple(stmt1)
  while(expense1):
    list2.append(expense1)
    expense1 = ibm_db.fetch_tuple(stmt1)
  total=0
  for x in list2:
     total += x[4]

  sql2 = "SELECT EXPLIMIT FROM LIMITS ORDER BY LIMITS.ID DESC LIMIT 1"
  stmt2 = ibm_db.prepare(conn, sql2)
  ibm_db.execute(stmt2)
  limit=ibm_db.fetch_tuple(stmt2)

  if(total>limit[0]):
    mail_from = '19i304@psgtech.ac.in'
    mail_to = session['email']

    msg = MIMEMultipart()
    msg['From'] = mail_from
    msg['To'] = mail_to
    msg['Subject'] = 'Expense Alert Limit'
    mail_body = """
    Dear User, You have exceeded the specified monthly expense Limit!!!!

    """
    msg.attach(MIMEText(mail_body))

    try:
        server = smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
        server.ehlo()
        server.login('apikey',
'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')
        server.sendmail(mail_from, mail_to, msg.as_string())
        server.close()
        print("mail sent")
    except:
        print("issue")

  return redirect("/display")

#DISPLAY---graph

@app.route("/display")
def display():
  print(session["username"],session['id'])

  sql = "SELECT * FROM EXPENSES WHERE USERID=?"
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt,1,session['id'])
  ibm_db.execute(stmt)
  list1=[]
  row = ibm_db.fetch_tuple(stmt)
  while(row):
    list1.append(row)
    row = ibm_db.fetch_tuple(stmt)
  print(list1)

  total=0
  t_food=0
  t_entertainment=0
  t_business=0
  t_rent=0
  t_EMI=0
  t_other=0

  for x in list1:
    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]

    return render_template('display.html' ,expense = list1,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)

#delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
     print(id)
     sql = "DELETE FROM expenses WHERE  id =?"
     stmt = ibm_db.prepare(conn, sql)
     ibm_db.bind_param(stmt,1,id)
     ibm_db.execute(stmt)

     return redirect("/display")

#UPDATE---DATA

@app.route('/edit/<id>', methods = ['POST', 'GET' ])
def edit(id):
    sql = "SELECT * FROM expenses WHERE  id =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.execute(stmt)
    row=ibm_db.fetch_tuple(stmt)
    print(row)
    return render_template('edit.html', expenses = row)

@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']
      time=request.form["time"]

      sql = "UPDATE expenses SET date =? , expensename =? , amount =?, paymentmode =?,
category =?, time=? WHERE expenses.id =? "
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt,1,date)
      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,time)
      ibm_db.bind_param(stmt,7,id)
      ibm_db.execute(stmt)

      print('successfully updated')
      return redirect("/display")
```

7.3 Feature 3 (Sendgrid, Kubernetes):

This feature enables alerts the user if the specified expense limit is exceeded via an automated email by sendgrid.

Limits File:

```
#limit
@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,EXPLIMIT) VALUES(?,?)"
       stmt = ibm_db.prepare(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():

   sql = "SELECT EXPLIMIT FROM LIMITS ORDER BY LIMITS.ID DESC LIMIT 1"
   stmt = ibm_db.prepare(conn, sql)
   ibm_db.execute(stmt)
   row=ibm_db.fetch_tuple(stmt)

   return render_template("limit.html" , y= row)
```

SENDGRID:

```
mail_from = '19i304@psgtech.ac.in'
      mail_to = session['email']

      msg = MIMEMultipart()
      msg['From'] = mail_from
      msg['To'] = mail_to
      msg['Subject'] = 'Expense Alert Limit'
      mail_body = """
      Dear User, You have exceeded the specified monthly expense Limit!!!!
```

```
    """
    msg.attach(MIMEText(mail_body))

    try:
       server = smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
       server.ehlo()
       server.login('apikey',
'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')
       server.sendmail(mail_from, mail_to, msg.as_string())
       server.close()
       print("mail sent")
    except:
       print("issue")
```

KUBERNETES DEPLOYMENT FILE:

```
apiVersion: apps/v1

kind: Deployment

metadata:
   name: expensetracker
   labels:
      app: expensetracker

spec:
   selector:
      matchLabels:
         app: expensetracker
   replicas: 1

   template:
      metadata:
         labels:
            app: expensetracker

      spec:
         containers:
            - name: expensetracker

              image: icr.io/personal_expense/expensetracker

              imagePullPolicy: Always

              ports:
                 - containerPort: 5000
              env:
                 - name: DISABLE_WEB_APP
                   value: "false"
```

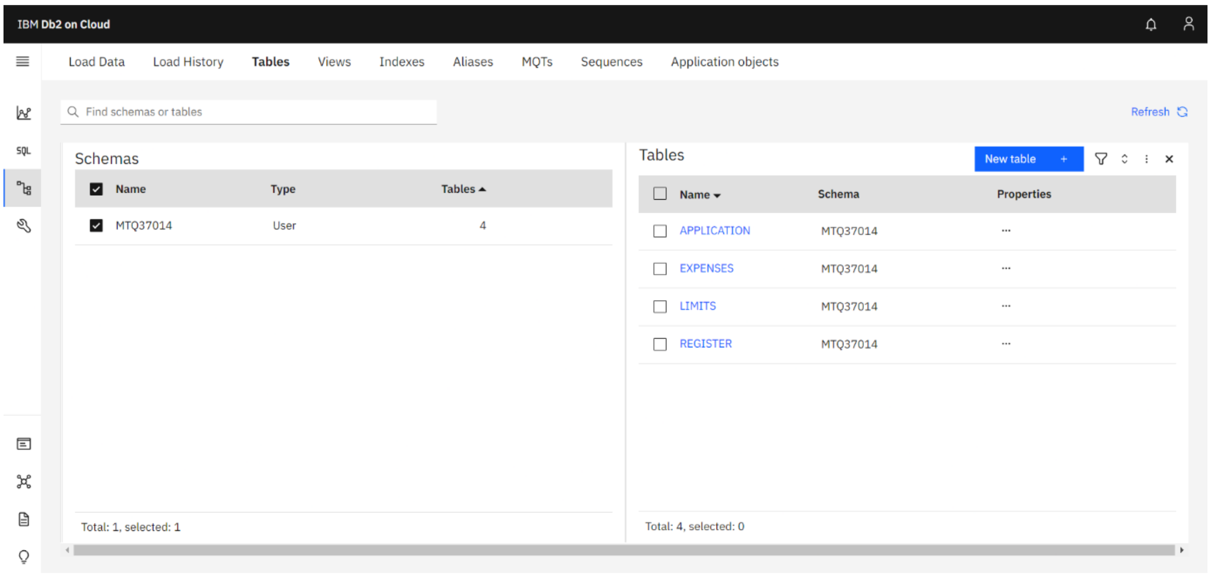
7.4 Database Schema:

Code to connect with IBM DB2:

To connect with IBM DB2 , Windows SSL certificate was required which is downloaded from the IBM DB2 resource.

```
app.secret_key = 'a'

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=mtq37014;PWD=W4Sam6RCrj9zDrfD;",'','')
```



Schemas

Name	Type	Tables
MTQ37014	User	4

Total: 1, selected: 1

Tables

Name	Schema	Properties
APPLICATION	MTQ37014	···
EXPENSES	MTQ37014	···
LIMITS	MTQ37014	···
REGISTER	MTQ37014	···

Total: 4, selected: 0

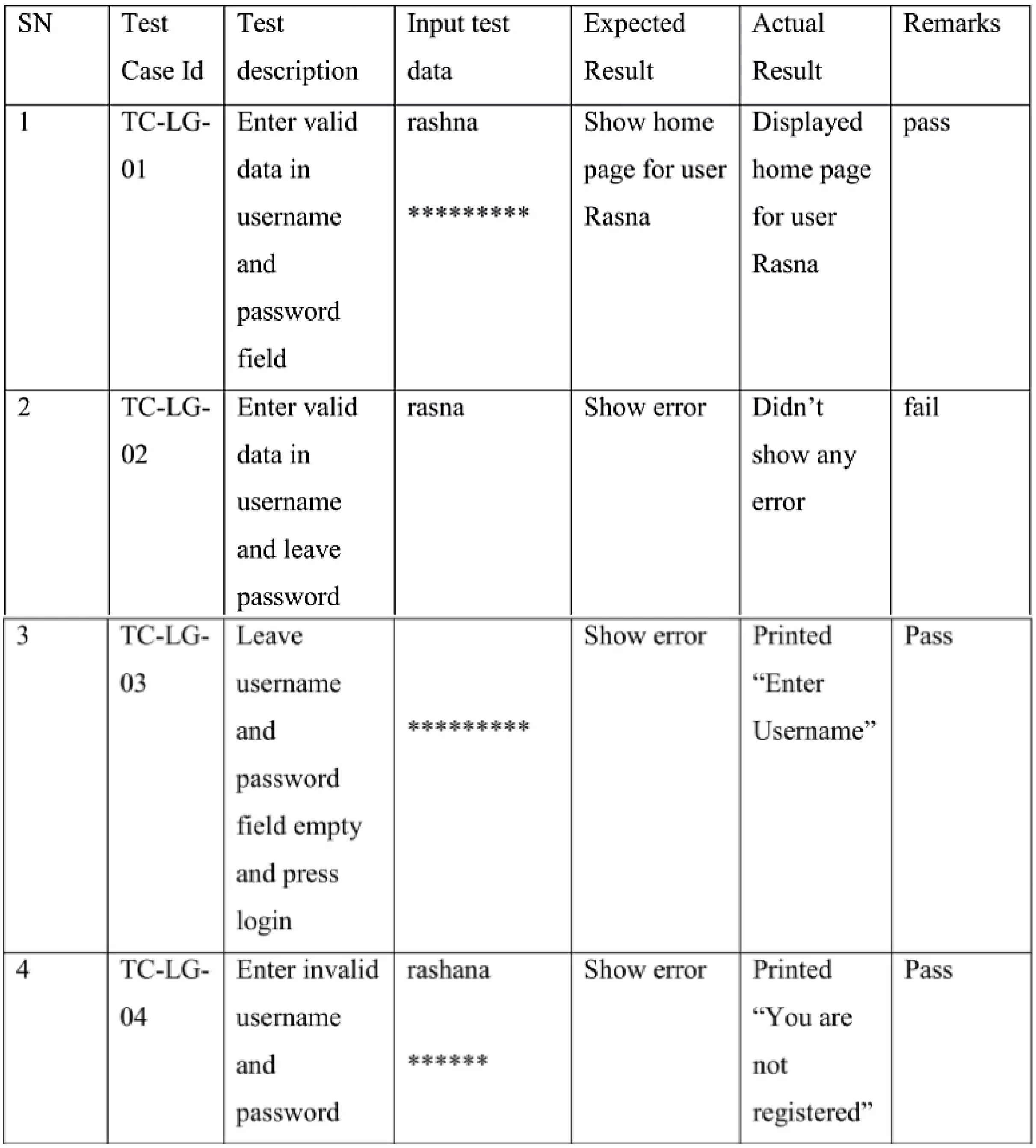
8. TESTING

8.1 Test Cases

Test case for Installation

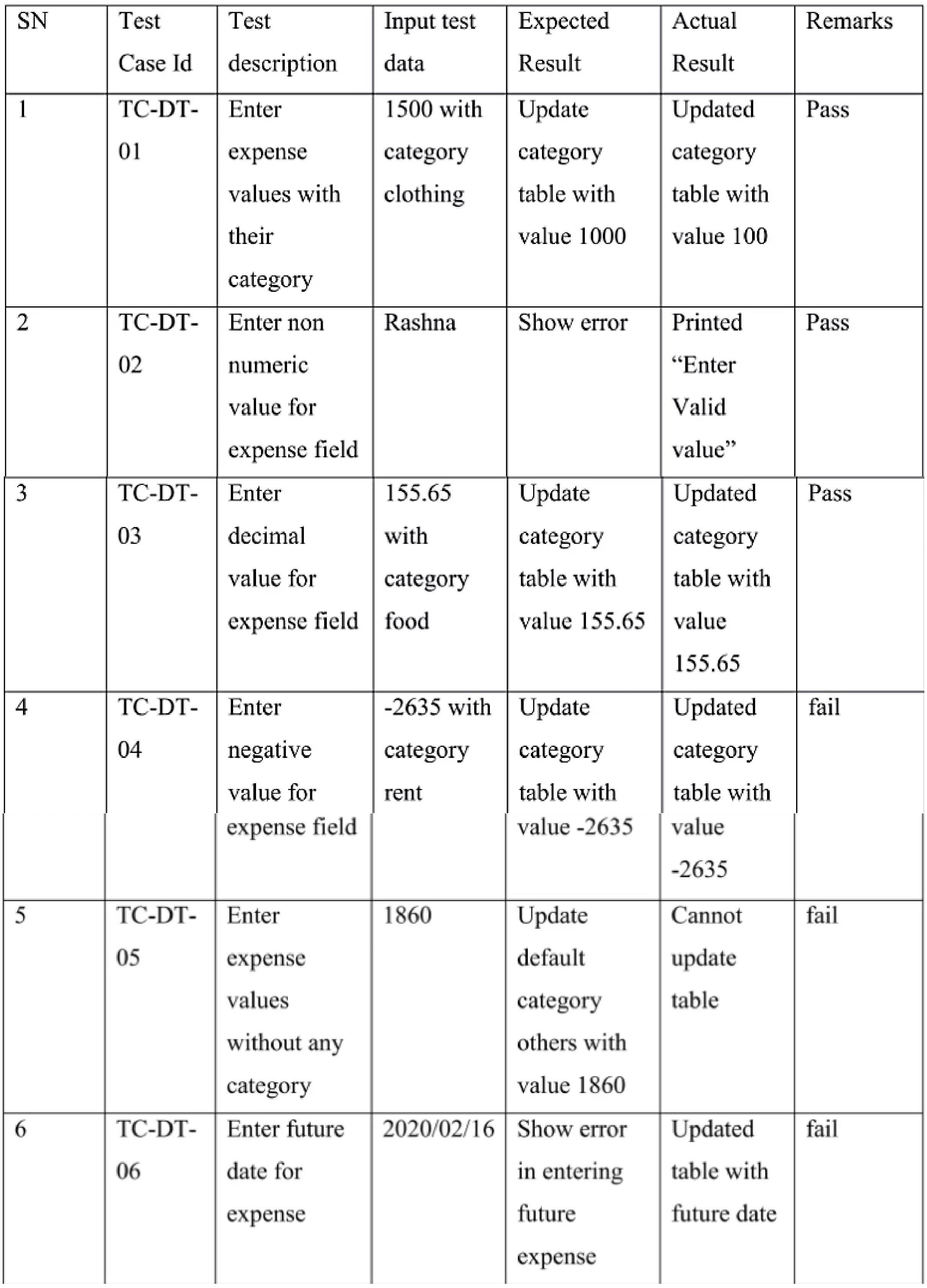
SN	Test Case Id	Test description	Input test data	Expected Result	Actual Result	Remarks
1	TC-INS-01	Install DET app in android phone	Transfer DET app	Open application with its home page	Application executed with home page	Pass

Test case for Login



SN	Test Case Id	Test description	Input test data	Expected Result	Actual Result	Remarks
1	TC-LG-01	Enter valid data in username and password field	rashna *********	Show home page for user Rasna	Displayed home page for user Rasna	pass
2	TC-LG-02	Enter valid data in username and leave password	rasna	Show error	Didn't show any error	fail
3	TC-LG-03	Leave username and password field empty and press login	*********	Show error	Printed "Enter Username"	Pass
4	TC-LG-04	Enter invalid username and password	rashana ******	Show error	Printed "You are not registered"	Pass

Test case for Data entry

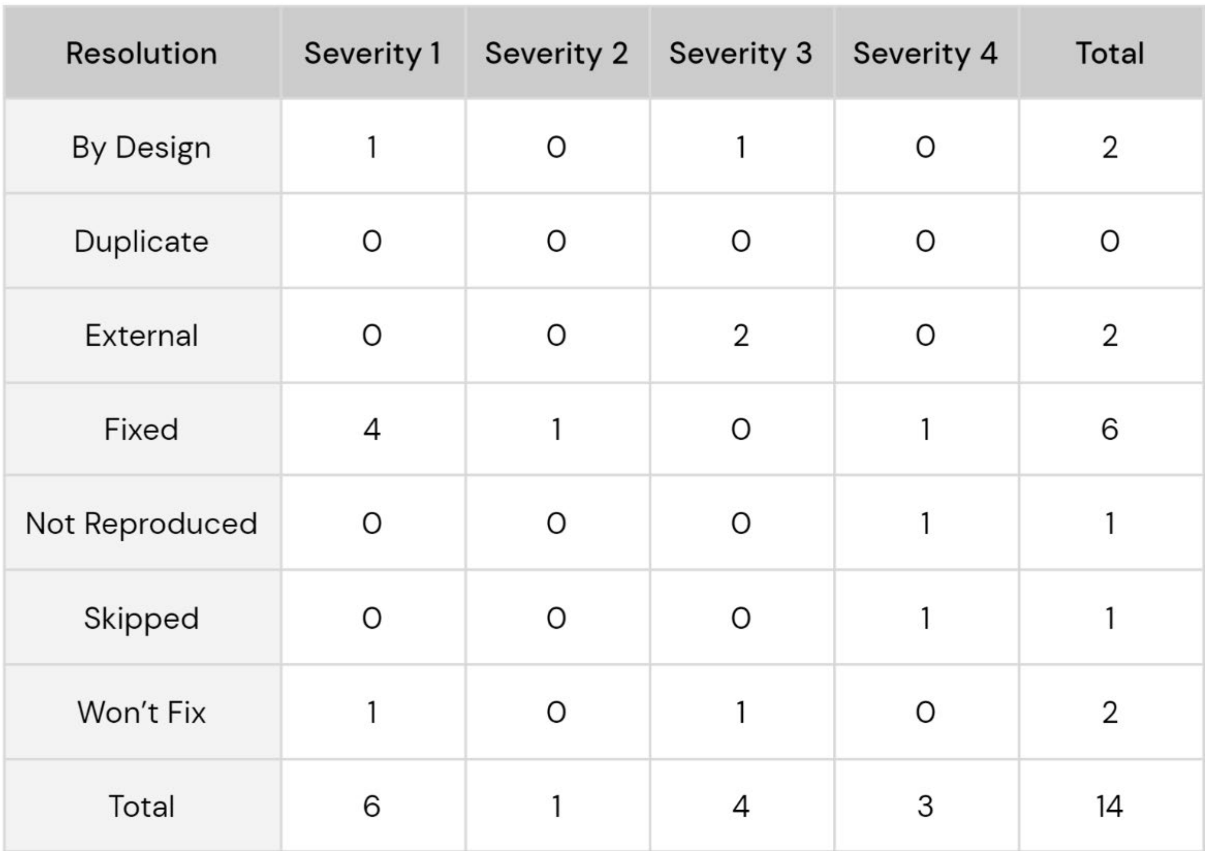


SN	Test Case Id	Test description	Input test data	Expected Result	Actual Result	Remarks
1	TC-DT-01	Enter expense values with their category	1500 with category clothing	Update category table with value 1000	Updated category table with value 100	Pass
2	TC-DT-02	Enter non numeric value for expense field	Rashna	Show error	Printed "Enter Valid value"	Pass
3	TC-DT-03	Enter decimal value for expense field	155.65 with category food	Update category table with value 155.65	Updated category table with value 155.65	Pass
4	TC-DT-04	Enter negative value for expense field	-2635 with category rent	Update category table with value -2635	Updated category table with value -2635	fail
5	TC-DT-05	Enter expense values without any category	1860	Update default category others with value 1860	Cannot update table	fail
6	TC-DT-06	Enter future date for expense	2020/02/16	Show error in entering future expense	Updated table with future date	fail

8.2 User Acceptance Testing

Purpose of UAT:

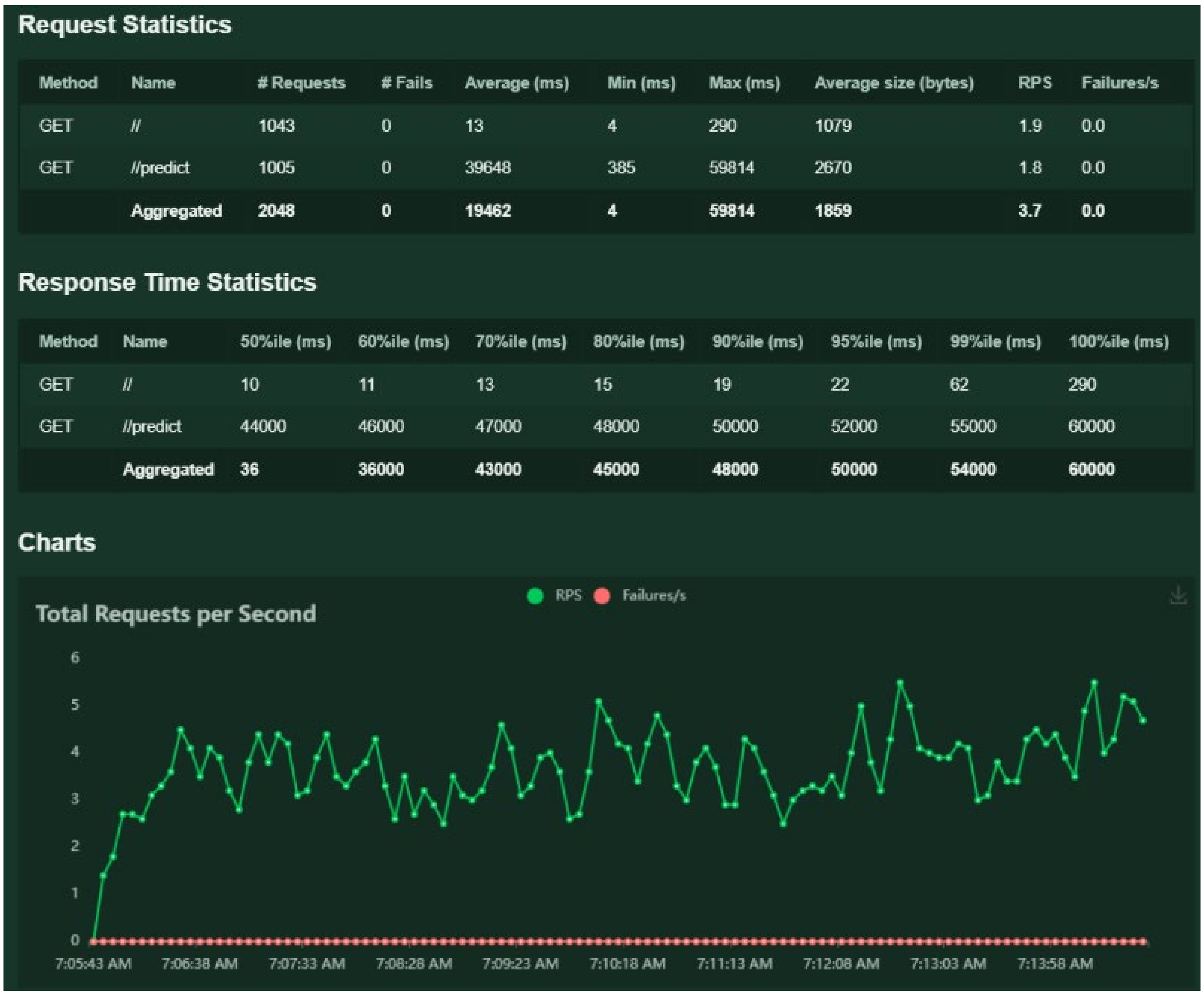
The purpose of UAT is to explain the test coverage and open issues of the personal expense tracker application at the time of release to User Acceptance Testing(UAT).



Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Total
By Design	1	0	1	0	2
Duplicate	0	0	0	0	0
External	0	0	2	0	2
Fixed	4	1	0	1	6
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	1	1
Won’t Fix	1	0	1	0	2
Total	6	1	4	3	14

9. RESULTS

9.1 Performance Metrics



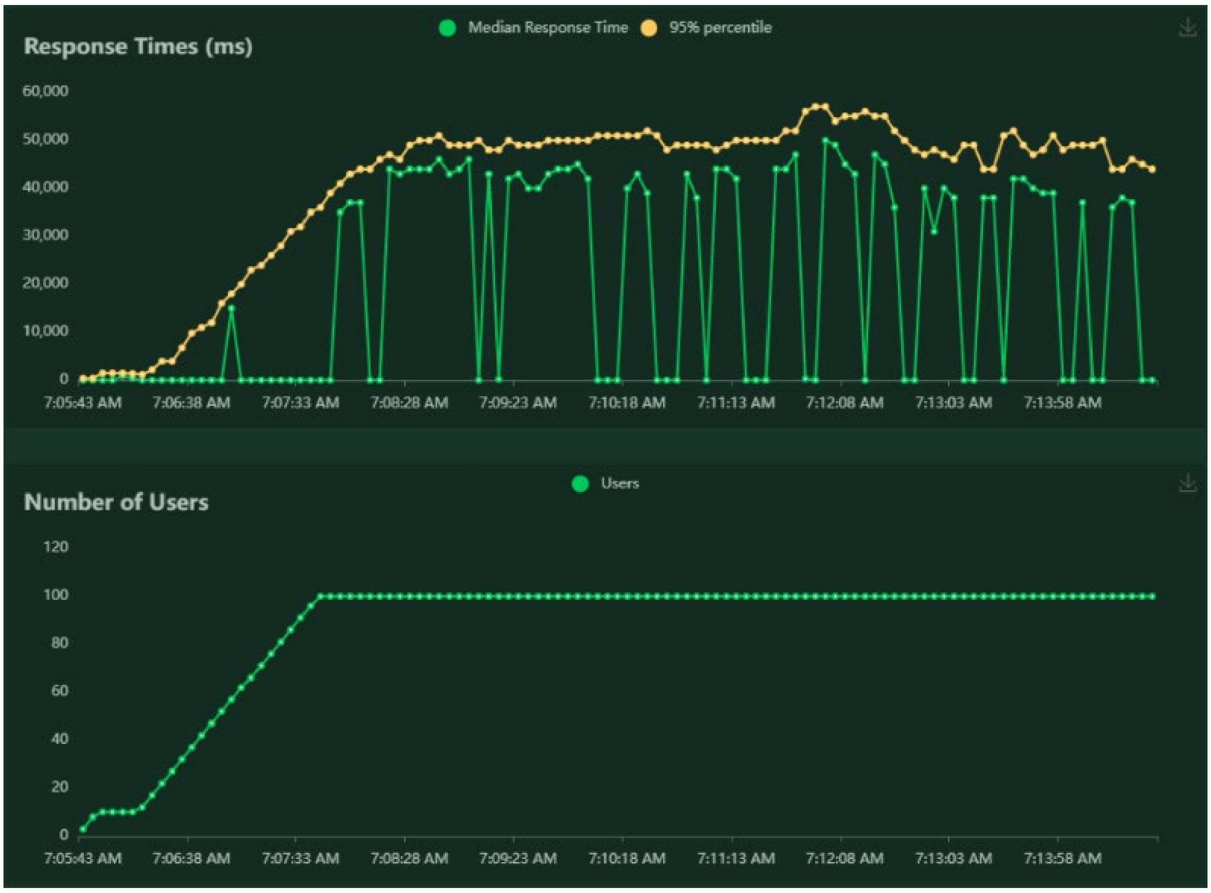
Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	1043	0	13	4	290	1079	1.9	0.0
GET	//predict	1005	0	39648	385	59814	2670	1.8	0.0
	Aggregated	**2048**	**0**	**19462**	**4**	**59814**	**1859**	**3.7**	**0.0**

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	10	11	13	15	19	22	62	290
GET	//predict	44000	46000	47000	48000	50000	52000	55000	60000
	Aggregated	**36**	**36000**	**43000**	**45000**	**48000**	**50000**	**54000**	**60000**

Charts



10. ADVANTAGES & DISADVANTAGES

Advantages:

- Saves time on manually calculating the expenses
- Improves workflow with quick approvals and reimbursements
- Eliminates errors while uploading data and calculations
- Prevents frauds relating to unreasonable expenses
- Helps in claiming tax returns and reinforcing compliance
- Enables negotiating for volume discounts with hotel chains ,travel companies etc.,

Disadvantage:

-

11. CONCLUSION

Thus, we have developed such type of web application which help the users to reduces their effort of handling daily expenses. That the application will have various components of updating and viewing users' expenditures. As part of research, we considered adding certain components to the application to make it more useful to the user. Some of the extra Components are like enabling users to register to the application using existing email or social network account, it will synchronize the users profile data to the application

12. FUTURE SCOPE

The Future Enhancements of the application can be allowed to support in all the upcoming android/web versions. History can be set to view all the details in the app even if the particular data is deleted from the database. Statistics could be prepared based on the Income, Expense details of the user. Sharing files via Bluetooth, WhatsApp can be allowed. Printing the details of the particular income or expense details can be made. Some of the extra components are like enabling users to register to the application using existing email or social network account, it will synchronize the users profile data to the application

13. APPENDIX

Source Code

GitHub & Project Demo Link