IBM Nalaiyathiran

PROJECT REPORT

Personal Expense Tracker Application

Team ID : PNT2022TMID15515

TABLE OF CONTENTS

1.INTRODUCTION

A Personal Expense Tracker Application is a particular form of digital diary that aids in keeping track of all of our cash transitions and moreover offers daily, weekly, monthly, and yearly reports on all financial activities. User receives alerts to keep track of income and expenses that can system for tracking the application. All data is kept in offline mode for easy access at any time and from any location. The Daily Expense Tracker's user interface is incredibly straightforward and appealing, making it simple to grasp and the finest approach to record our financial data.

1.1. PROJECT OVERVIEW

Simply put, personal finance includes all of the financial decisions and actions that a finance software facilitates by assisting you in effectively managing your finances. 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.

Users of personal finance applications will be prompted to enter their costs, after which their wallet balance will be updated and displayed to them. Users can also receive a graphical analysis of their expenses. They can choose to establish a cap on how much can be used in that month, and if the cap is surpassed, the user will receive an email alert.

1.2. PURPOSE

When you keep track of your spending, you can make sure your money is being utilised wisely and you will know where it goes. You can learn why you're in debt and how you got there by keeping track of your spending. You can then use this information to create a debt relief plan that works for you.

You may plan for both short-term and long-term expenses by using a budget to make sure you're not spending more than you're earning. It's a simple, practical solution for folks with all types of income and expenses to maintain order in their finances.

2. LITERATURE SURVEY

Competitive Analysis

There are so many competitors in the market and we took the top 3 applications. We analyzed their features and came up with pros & cons.

	PROS	CONS
INTUIT mint	Free to use Integrate bank account and credit card Customized alerts	Expenses are assigned to wrong category
Goodbudget	Good for beginner Customized alerts	Paid Manually enter the transactions
Spendee	Integrate bank account and credit card + crypto wallets Customized alerts	Paid Expenses are assigned to wrong category Less categorization icons

2.1. EXISTING PROBLEM

The expense tracker existing system does not offer the user portable device management level, is only used on desktop software, and is therefore impossible to update anywhere expenses are done and is unable to update the location of the expense details disrupting that the proposed system provides. The user's daily, weekly, and monthly spending must be maintained in Excel sheets and CSV files at the moment. The ability to conveniently keep track of one's everyday costs does not now have a fully comprehensive answer.To do this, one must maintain a journal in a diary or computer system, and all calculations must be made by the user, which might occasionally result in errors that cause losses. Due to imperfect data maintenance, the

current system is not user friendly. The sole negative where the rest are absent from this endeavor is that there will be no reminder to stay a human on a specified date. This project won't have any information because it doesn't remind people to do anything each month, which has some drawbacks. However, it can be used to calculate income and expenses, so we suggest a new project to solve this issue.

2.2 REFERENCE

[1] D2D Expense Tracker Application Anjali Kumar, Utkarsh Ra, Aman Kumar 2021

[2] Daily Expense Tracker Mobile Application Nuura Najati Binti Mustafa 2021

[3] Daily Expense Tracker Shivam Mehra, Prabhat Parashar 2021

[4] Intuit Mint(Application)

[5] GoodBudget(Application)

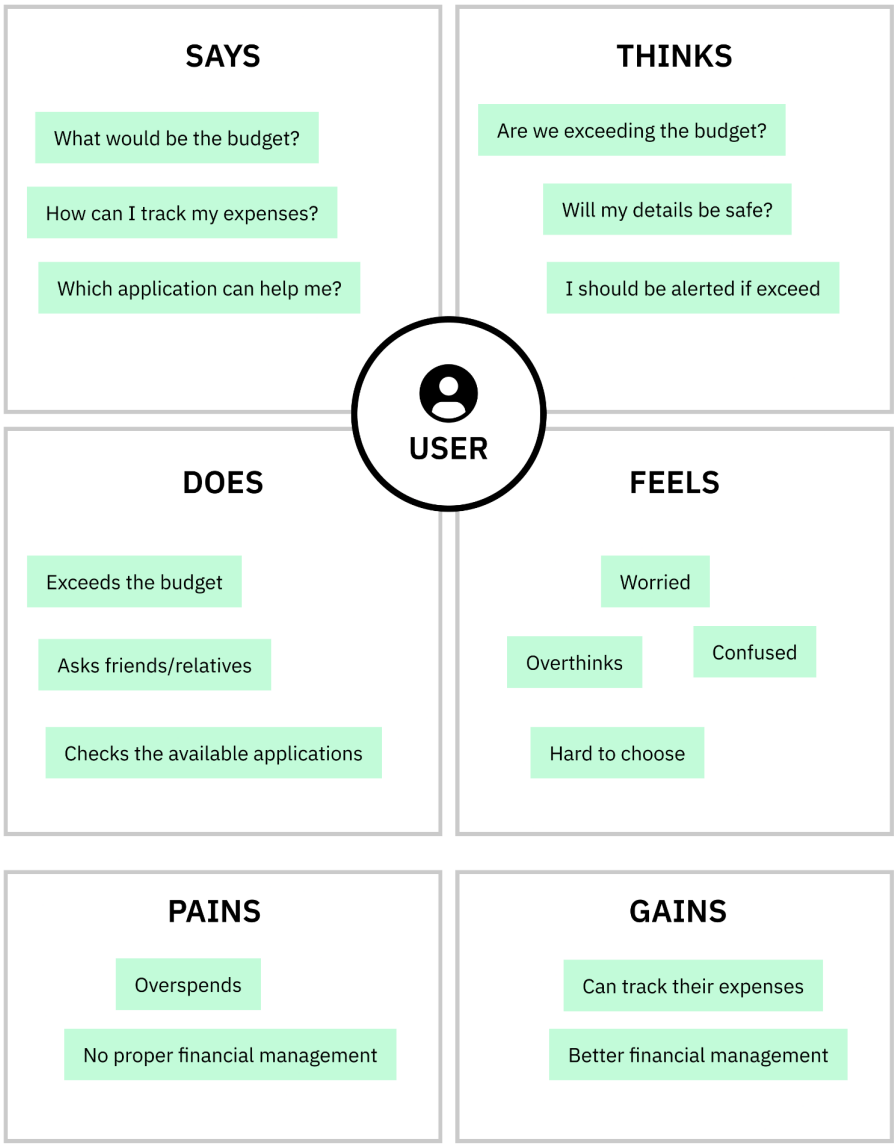
[6] Spendee(Application)

2.3 PROBLEM STATEMENT DEFINITION

In our daily life money is the most important portion and without it we cannot last one day on earth but if we keep on track all financial data then we can overcome this problem. Most of the people cannot track their expenses and income one way they face the money crisis and depression. This situation motivates us to make an android app to track all financial activities. Using the Personal Expense Tracker Application user can be tracking expenses day to day and making life tension free.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2. IDEATION & BRAINSTORMING

Template

Brainstorm & idea prioritization

Personal Expense Tracker

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

Problem

How might we help the users plan their budget?

How might we help the user's family to track each other spending

How might we help the users to track different kind of expenses?

How might we crosscheck the amount entered is correct or the buget is calculated correct?

how might we manually enter the amount???

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Voleti Varshith

- Can send message notification to the user
- Can integrate with crypto hardware wallets
- Recommendation of some Youtube channels in the dashboard about saving money.
- Better UI and UX for users
- Various themes in the app
- Can integrate with UPI

Subramanian K

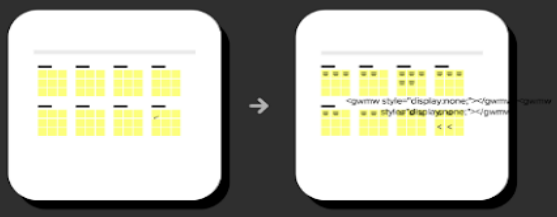
- security
- offer tips to lower expenses
- monitor transcations
- allocate budget based on each location

Nambiraaja T

- Easy Accessbility
- Well Categorizationof the Expenses
- Integrate Any wallets like paytm,Amazon
- Figure out ways to cut back on your spending
- Reduced turnaround time and faster reimbursements
- set budget for daily, weekly, monthly, and yearly

Kavidhasan M

- integrate multipe bank accounts
- allow to enter manually
- List the categories to categorize the spendings
- One account for multiple users(especially when it comes to family)
- Weekly/monthly reports(even comparision of different monthly spendings)
- Customization of categories based on the user needs



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Integration

- Can integrate with UPI
- integrate multipe bank accounts
- Can integrate with crypto hardware wallets
- Integrate Any wallets like Paytm, Amazon

Experience

- Better UI and UX for users
- monitor transcations
- Various themes in the app
- Easy Accessbility
- Reduced turnaround time and faster reimbursements

Alerts

- Can send message notification to the user

Categorization

- List the categories to categorize the spendings
- Customization of categories based on the user needs
- Well Category the Expenses
- allocate budget based on each location

Awareness

- offer tips to lower expenses
- Recommendation of some Youtube channels in the dashboard about saving money.
- Figure out ways to cut back on your spending
- allocate budget based on each location

Customization

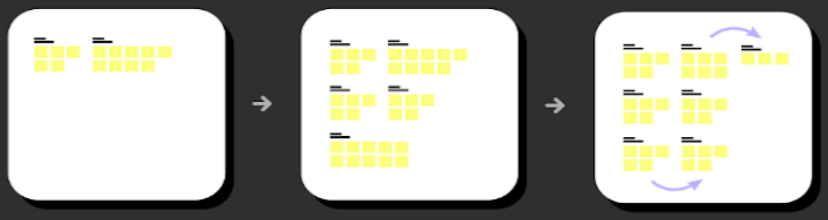
- set budget for daily, weekly, monthly, and yearly

Insights/Reports

- Weekly/monthly reports(even comparision of different monthly spendings)

Others

- allow to enter manually
- security

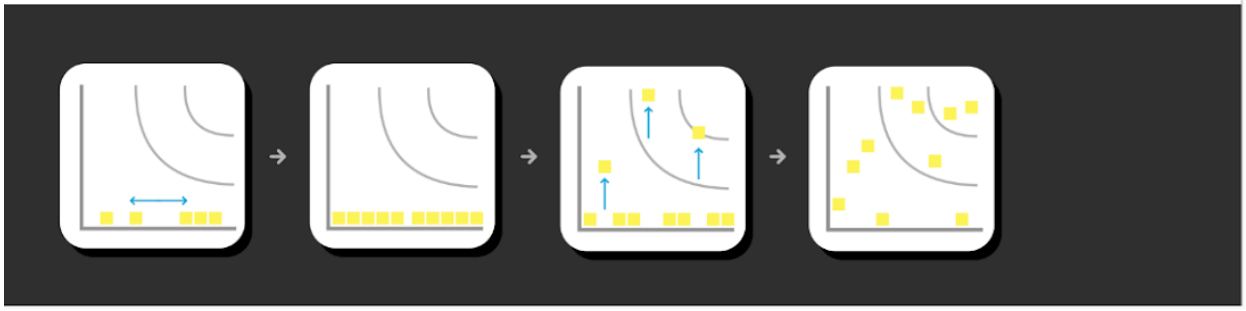


4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

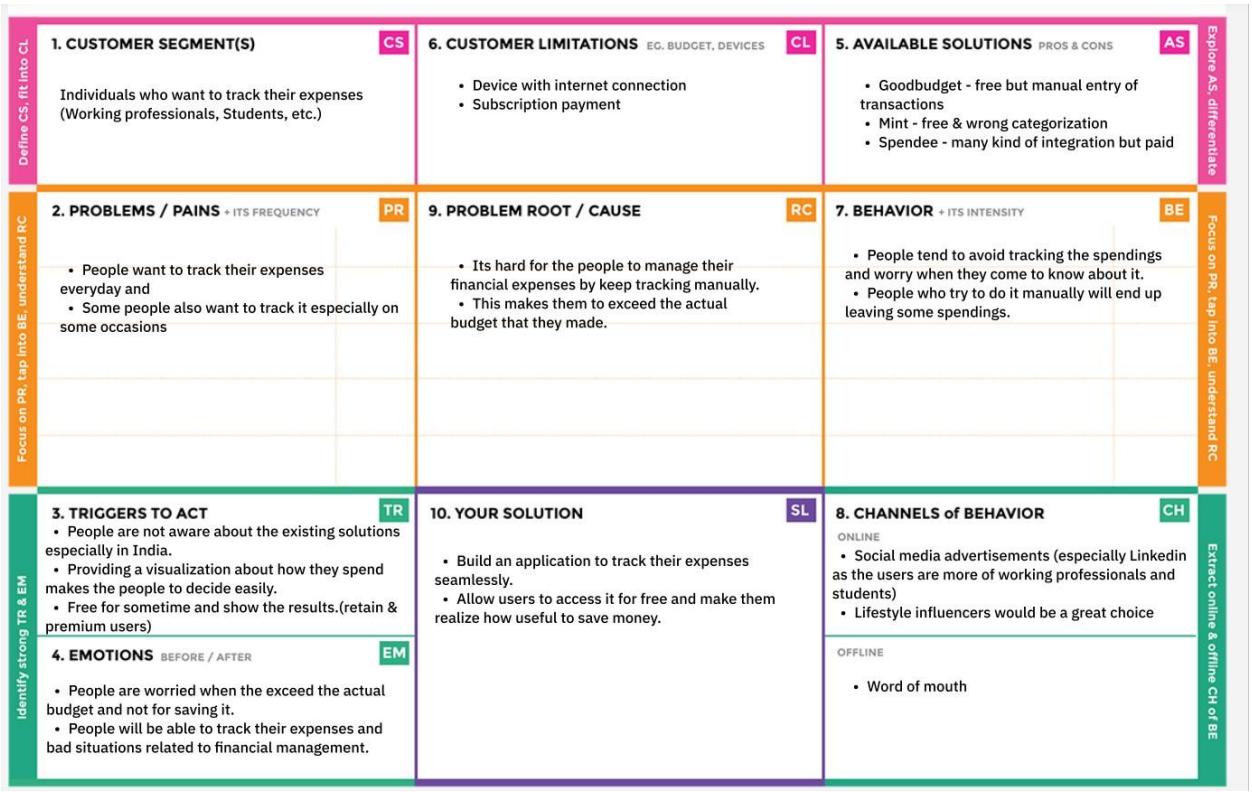
20 minutes



3.3. PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Individuals are dealing hard to track their expenses and exceed their budget.
2.	Idea / Solution description	A web application that helps the users to track their expenses and alerts them if they exceed the limit.
3.	Novelty / Uniqueness	• Joint account - Couple/family can track expenses as a group. • Better visualization of data
4.	Social Impact / Customer Satisfaction	It will help the people to track their expenses and also alerts when you exceed the limit of your budget. This allows the users to take proper financial decisions on spendings.
5.	Business Model (Revenue Model)	• Subscription-based model(per month/year) • Different pricing for different kind of accounts
6.	Scalability of the Solution	This can be scaled in a way that organizations can utilize this application for financial management.

3.4. PROBLEM SOLUTION FIT



4.1 Functional Requirements

Following are the functional requirements of the Proposed solution

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation/ Forgot password	Confirmation via Email
FR-3	Expenses entry	Users can add their expenses

FR-4	Notifications and monthly insights	Send notifications and insights to user's mail (Sendgrid)
FR-5	Dashboard	Graphical insights about their expenses
FR-6	Family account	Multiple users can access single dashboard

Non-functional Requirements:

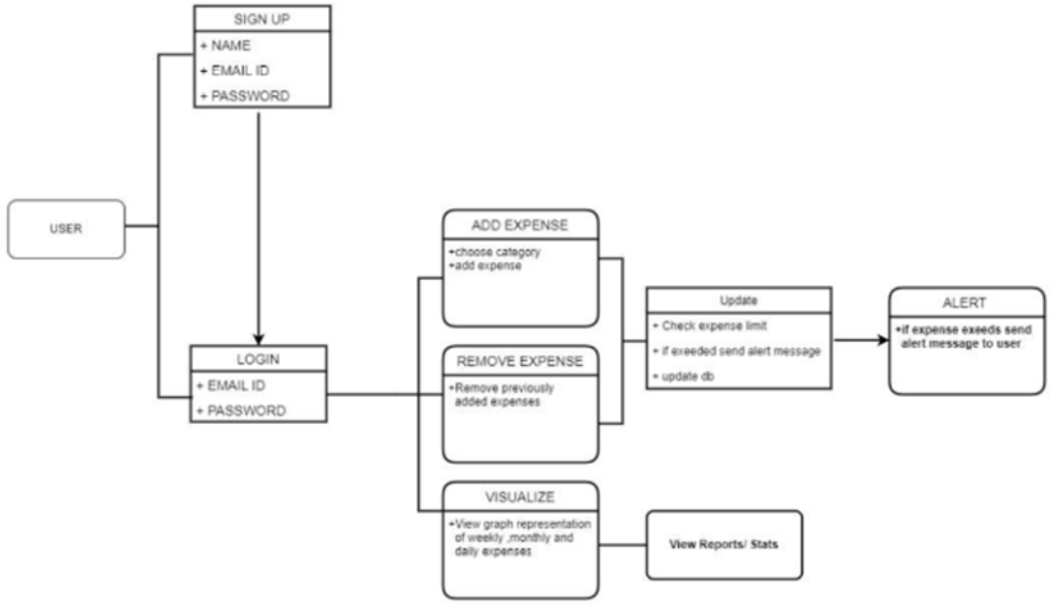
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The application will follow the user-centered way so that the user experience will be seamless
NFR-2	Security	Encrypt data using TLS protocol and the data will be hashed at the server
NFR-3	Reliability	The system must perform without failure in most of the use cases during a month.
NFR-4	Performance	Performance will be constant and can deal more concurrent users
NFR-5	Availability	The application should be accessible to the users for 24 hours.
NFR-6	Scalability	The system must be scalable enough to support many number of visits at the same time while maintaining optimal performance.

5. PRODUCT DESIGN

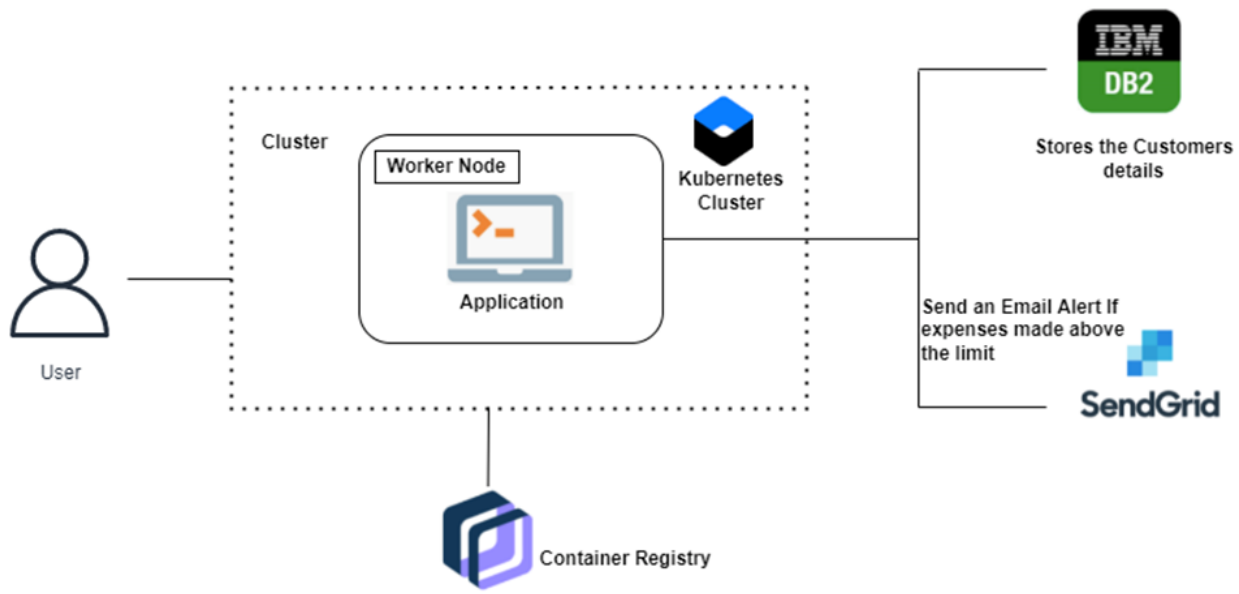
5.1 Data Flow Diagrams

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



5.2 Technical Architecture

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



5.3 User Stories

Use the below template to list all the user stories of the product

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-3	As a user, I can see the insights of my spending (pie-chart) and perform other operations like	I can get insights about my spending	High	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			adding, updating and removing budgets.			
	Profile	USN-4	As a user, I can view details of me and my group and update it.	Update my details	Low	Sprint-3
	Group	USN-5	As a user, I can form group and track the expenses as a group	I can track as joint account	Low	Sprint-4
	Alerts	USN-6	As a user, I should receive alerts if I exceed my budget	I can receive alerts	High	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High	Voleti Varshith, Kavidhasan M
Sprint 1	Login	USN-2	As a user, I can log into the application by entering email & password	10	High	Voleti Varshith, Kavidhasan M
Sprint 2	Dashboard	USN-3	As a user, I can see the insights of my spending (pie-chart) and perform other operations like adding, updating and removing budgets.	20	High	Voleti Varshith, Kavidhasan M
Sprint 4	Profile	USN-4	As a user, I can view details of me and my group and update it.	10	Low	Subramanian K, Nambiraaja T
Sprint 4	Group	USN-5	As a user, I can form group and track the expenses as a group	10	Low	Subramanian K, Nambiraaja T
Sprint 3	Alerts	USN-6	As a user, I should receive alerts if I exceed my budget	20	High	Subramanian K, Nambiraaja T

Project Tracker, Velocity & Burndown Chart:

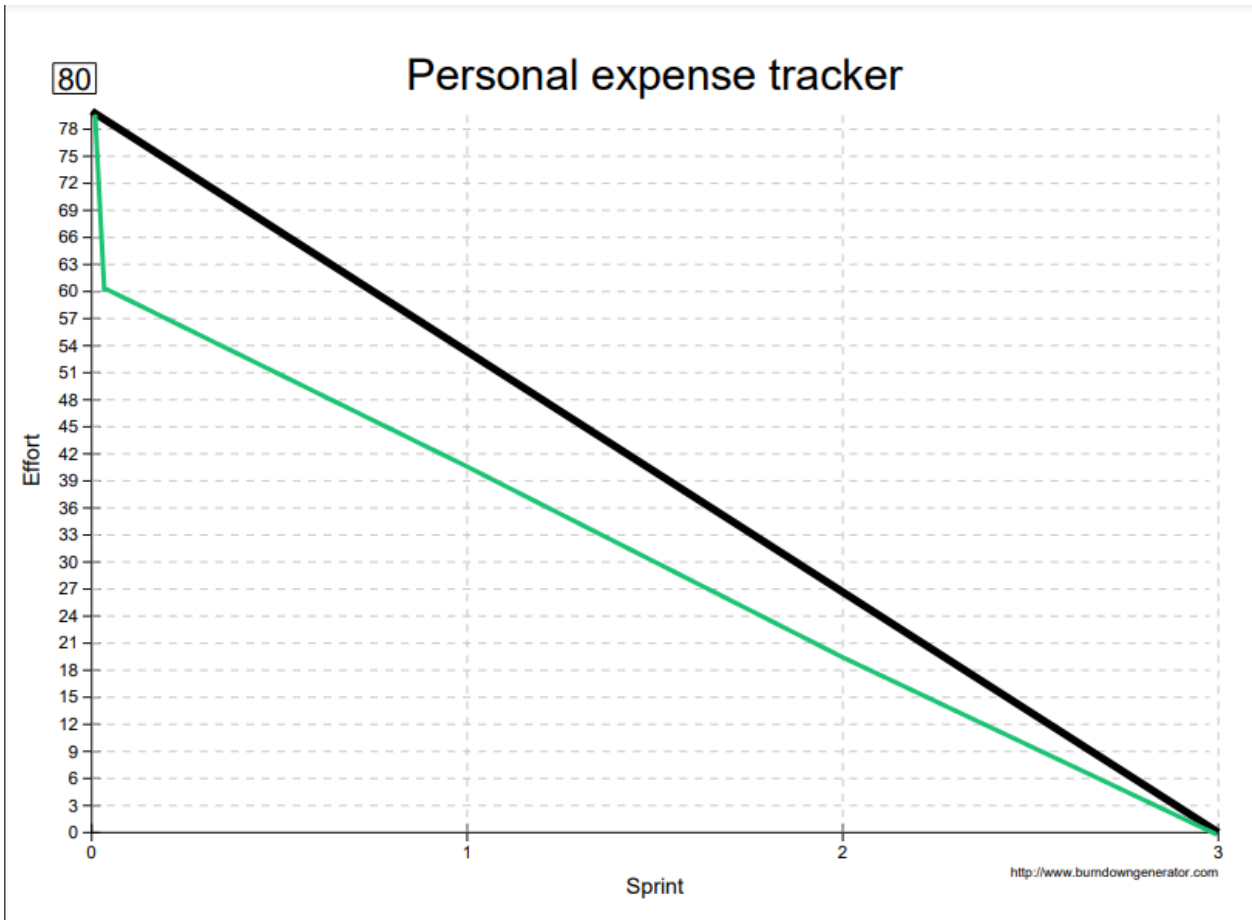
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	30 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	09 Nov 2022

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-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	10	19 Nov 2022

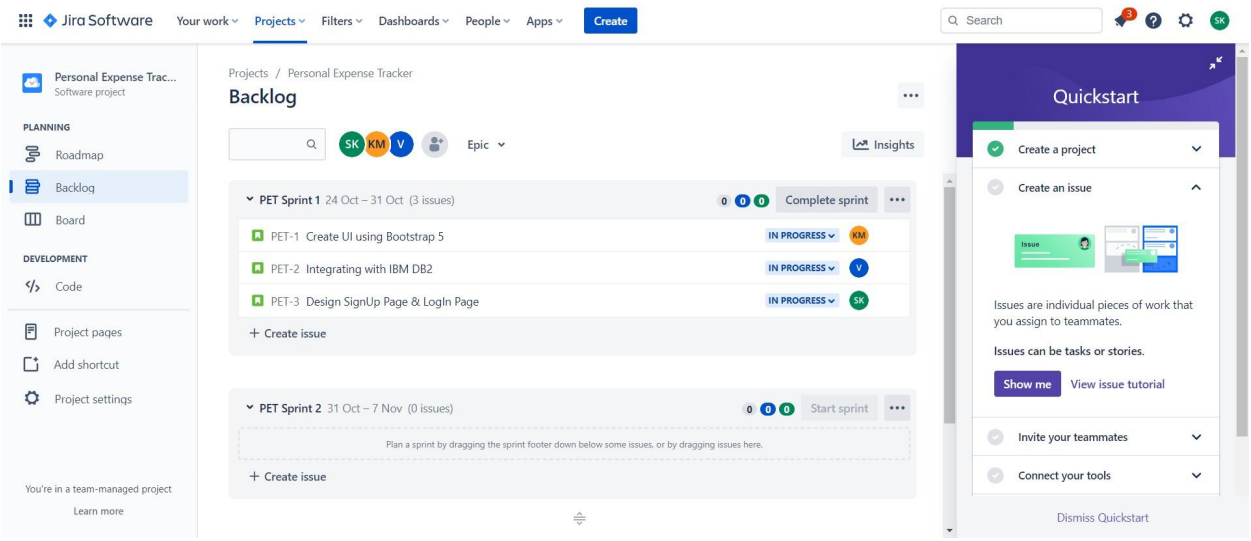
Velocity

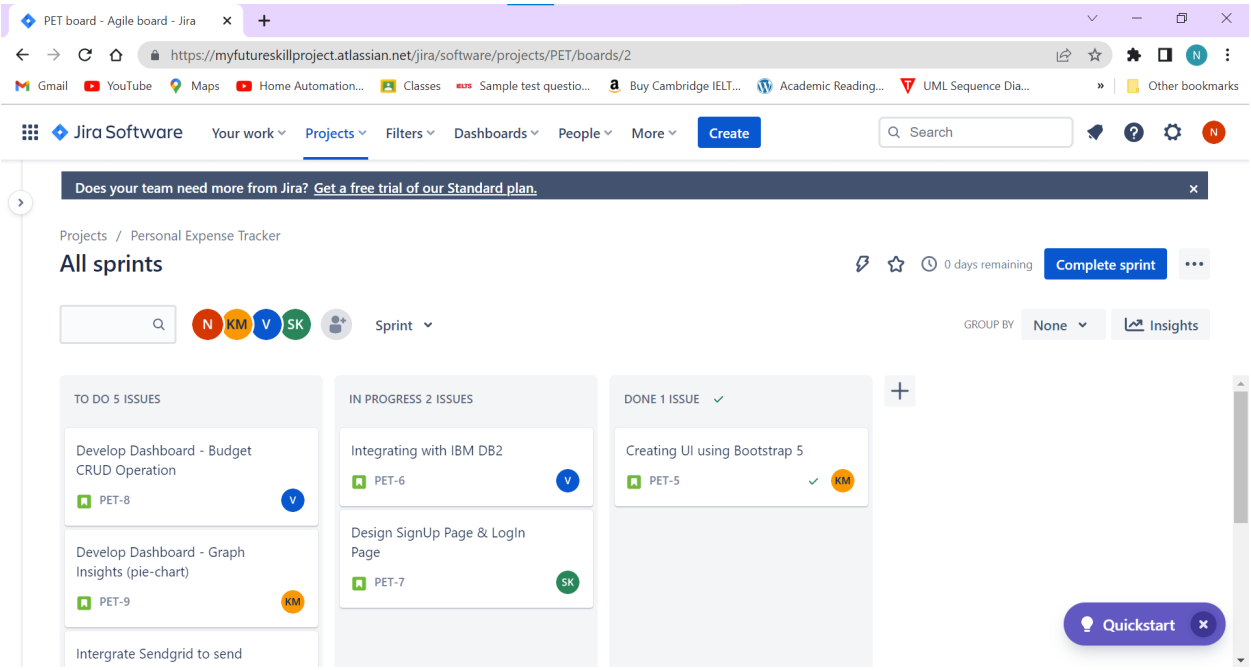
Average Velocity **=** $\frac{20}{6} = 3.33$

Burndown Chart



6.3 Reports from JIRA

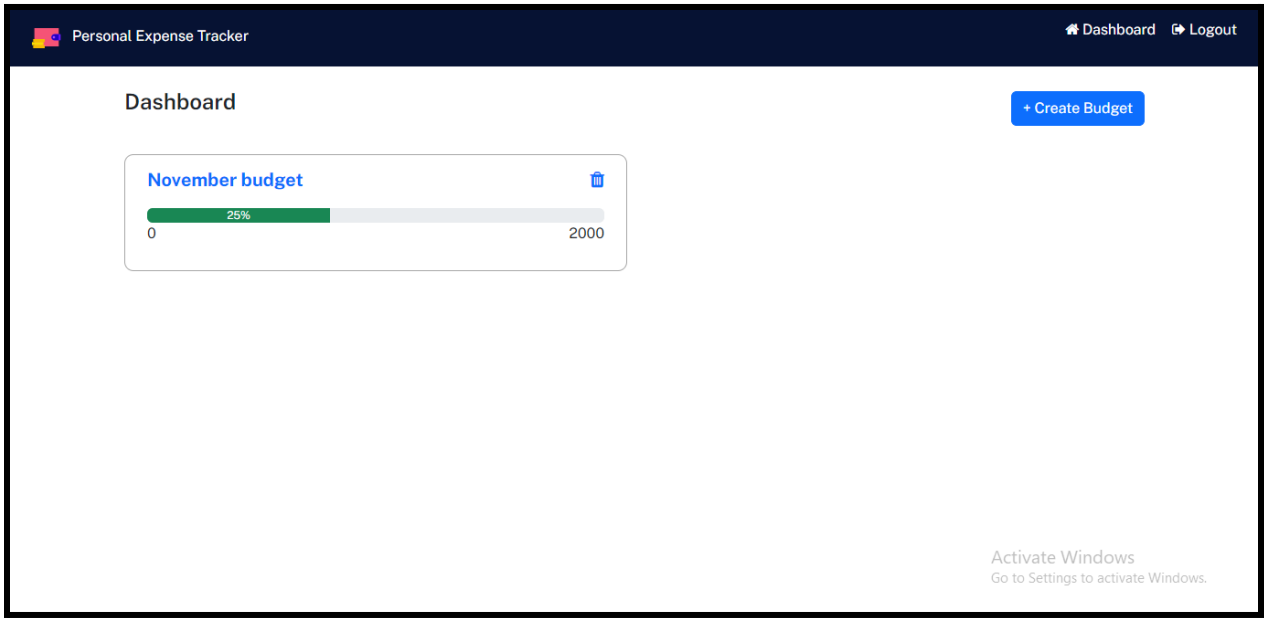




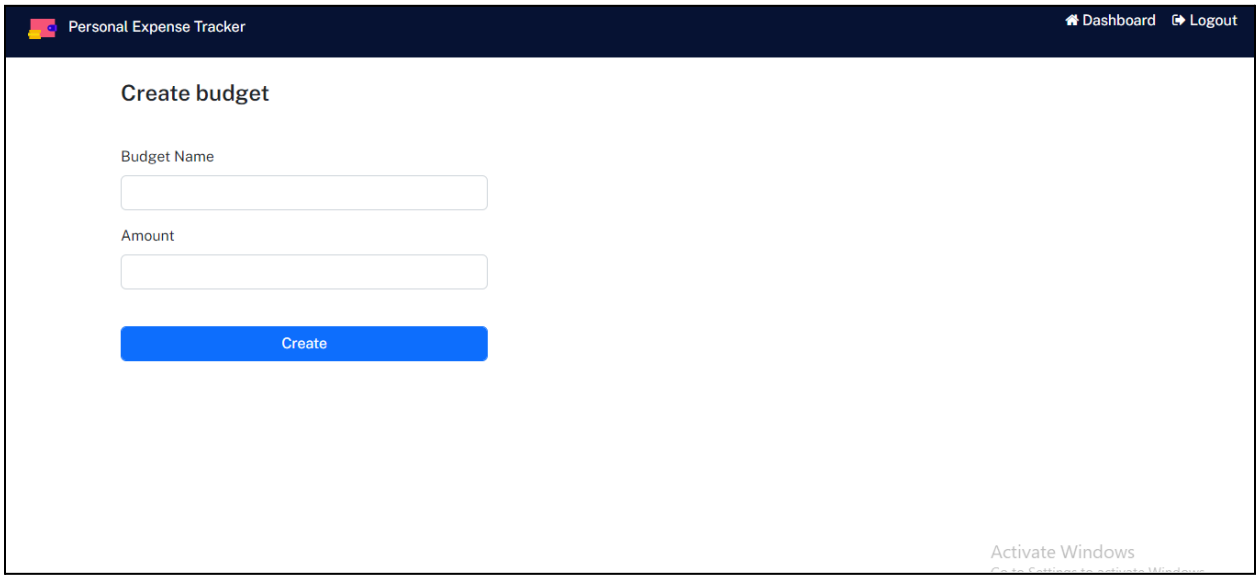
7.Coding & Solutioning

7.1 Feature 1 (Spending progress, Create budget, & Add transactions)

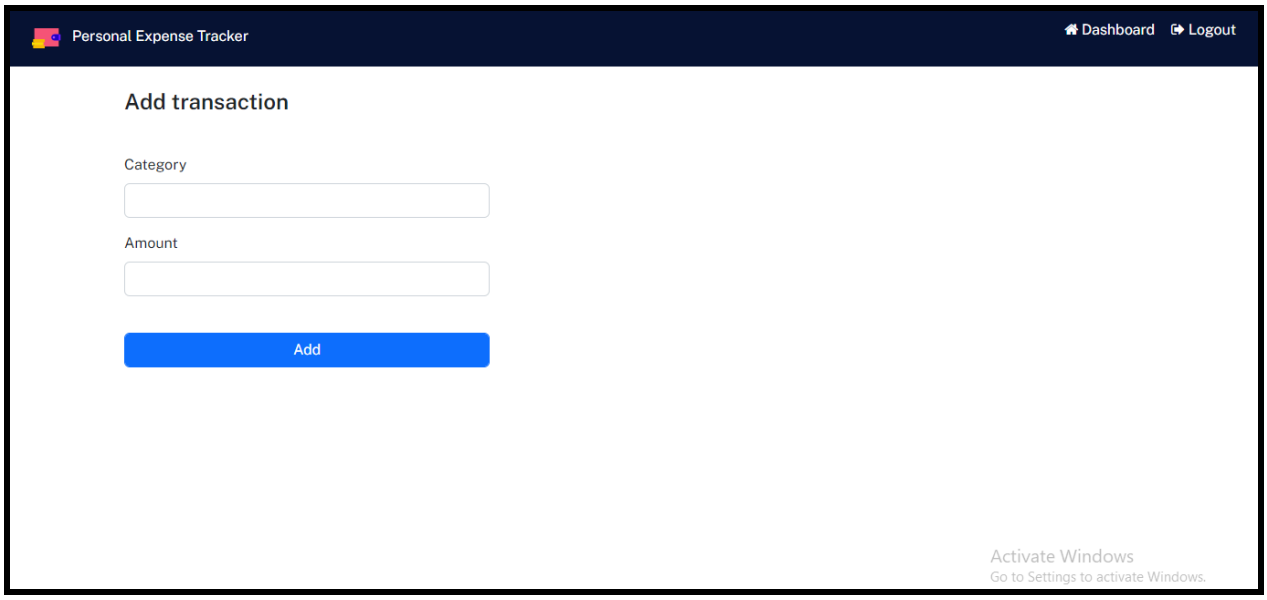
Spending Progress(Dashboard)



Create Budget



Add Transactions(Manual)

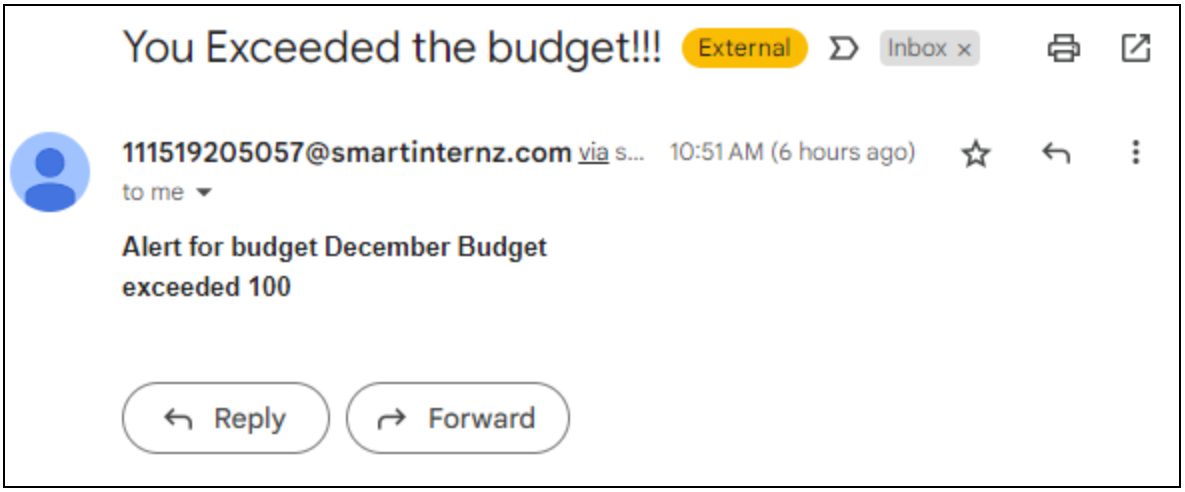


7.2 Feature 2 (Graph Insights)

Doughnut Chart(Remaining amount and other spendings)

7.3 Feature 3 (Email Alerts)

Email Alerts on exceeding the budget


You Exceeded the budget!!!
External
Inbox ×
111519205057@smartinternz.com via s...
10:51 AM (6 hours ago)
to me
Alert for budget December Budget exceeded 100
Reply
Forward

8.Testing

8.1 Test Cases

Test Case ID	Feature Type	Component	Test Scenario	Steps to execute	Test Data	Expected Result	Actual Result	Status	Comments	Bug
Login&Register TC-01	UI	LoginPage	Verify UI elements in the LoginPage	Go to site -- Verify UI elements like email field, password field, Login button, new user signup link	http://127.0.0.1:5000/login	Following elements should be shown in the interface: email field, password field, Login button, new user signup link	Working as expected	Pass		
Login&Register TC-02	Functional	LoginPage	Verify the user is able to login	Go to page -- Enter valid email and password	email: uit19120@rmd.ac.in --- password : testing123	Should direct to dashboard	Working as expected	Pass		
Login&Register TC-03	UI	SignupPage	Verify UI elements in the SignupPage	Go to site -- Verify UI elements like Firstname and lastname field,email field,username field, password field, Signup button, existing user login link	http://127.0.0.1:5000/signup	Following elements should be shown in the interface: Firstname and lastname field,email field,username field, password field, Signup button, existing user	Working as expected	Pass		

						login link				
Login&Register TC-04	Functional	SignupPage	Verify the user is able to create an account	Go to page -- Fill up the fields in the Signup form	Enter valid details in the signup form	Should direct to loginpage	Working as expected	Pass		

Dashboard TC-01	UI	Dashboard	Verify UI elements in the Dashboard	Go to site -- Login -- Go to Dashboard and verify UI elements like create budget button, created budgets and delete button for deleting budgets	http://127.0.0.1:5000/dashboard email: uit19120@rmd.ac.in --- password : testing123	Following elements should be shown in the interface: create budget button, created budgets and delete button for deleting budgets	Working as expected	Pass		
Dasboard TC-02	Functional	Dashboard	Verify the user is able to create budget	Go to page -- Login -- Go to Dashboard -- Click on create budget button -- Enter the budget name and amount -- submit	Budget name : November Budget -- Amount : 10000	Should create a budget in the dashboard	Working as expected	Pass		

Budget TC-01	UI	BudgetPage	Verify UI elements in the Budget page	Go to site -- Login -- Go to Dashboard -- Click on a budget that is created and verify UI elements like graph(pie-chart), Total amount,	email: uit19120@rmd.ac.in --- password : testing123	Following elements should be shown in the interface: graph(pie-chart), Total amount, Remaining amount, Categories and money spent on that, and Add	Working as expected	Pass		

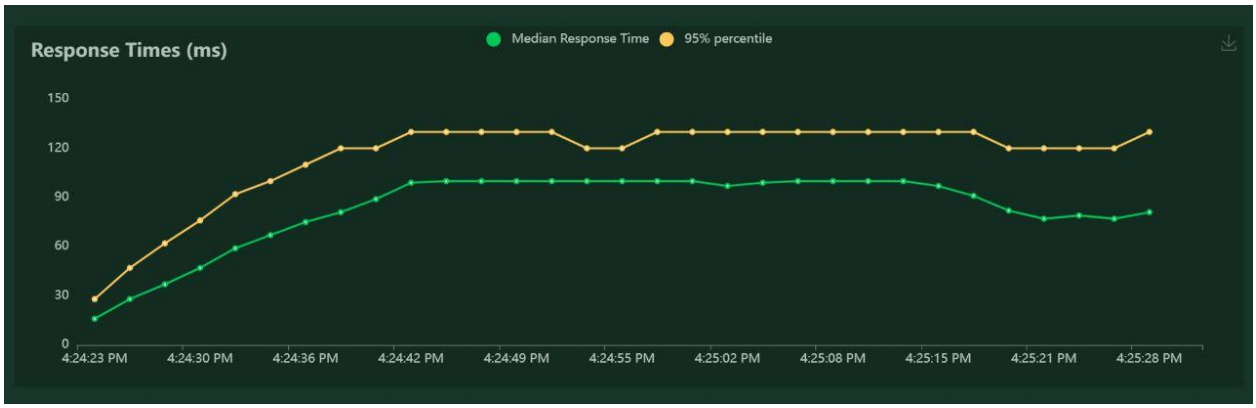
				Remaining amount, Categories and money spent on that, and Add transaction button		transaction button				
Budget TC-02	Functional	BudgetPage	Verify the user is able to add transactions in each budget with category	Go to page -- Login -- Go to Dashboard -- Click on any budget -- Click on add transaction button	Category name : Vehicle -- Amount : 2000	Should add a new transaction record and also it should affect the pie-chart	Working as expected	Pass		
Budget TC-03	Functional	BudgetPage	Verify the graph values are correct	Go to page -- Login -- Go to Dashboard -- Click on any budget -- Click on add transaction button	Category name : Vehicle -- Amount : 2000	The pie-chart should get updated with the values after adding new transactions	Working as expected	Pass		
Budget TC-04	Functional	BudgetPage	Verify the user is able to add transactions to the same category	Go to page -- Login -- Go to Dashboard -- Click on any budget -- Click on add transaction button -- Try adding a transaction to a same category	Category name : Vehicle -- Amount : 2000	The amount should get added if the category exists already	Not working as expected	Fail	It adds as a new category	BUG 01

9.RESULTS

9.1 Performance Metrics

Performance testing was conducted with this application using a performance testing tool called Locust. The following are the performance charts of the application,

Response Times Chart



Total Request per Second Chart

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

- The users will be able to track their expenses easily.
- Avoid papers and calculations
- Better understanding of their spending behavior
- Avoid overspending

10.2 Disadvantages

- Good internet connection is needed
- Manual way of adding transactions will break the user experience

11. CONCLUSION

A spending plan (also called a budget) is simply a plan you create to help you meet expenses and spend money the way you want to spend it. A good spending plan can help you stop “spending leaks”; in other words, it can keep you from spending money without thinking. It can help you make sure you have money to pay bills on time, even when your bills and income change each month.

12. FUTURE SCOPE

1) This can be developed into a mobile application so that the users find it easy to use as mobile applications then a web application.

2) Summary can be provided to the users based on the spending behavior in the application.

3) The users can be allowed to integrate their bank accounts, crypto wallets, etc to avoid wasting time on manual way of adding transactions.

4) This can also be utilized by small business owners.

13. APPENDIX

13.1. SOURCE CODE

App.py

```
from flask import
Flask,render_template,request,flash,redirect,url_for
from flask_sqlalchemy import SQLAlchemy
from datetime import date
from flask_bcrypt import Bcrypt
from flask_login import
LoginManager,login_user,current_user,login_required,logout_user,curre
nt_user
app = Flask(__name__)
login_manager = LoginManager(app)
app.config['SECRET_KEY'] =
'c\xae_O#H\xbdjTD\xed\xcf\x9e\x0f\xa3,\xbb\xcd:\x08\x05\xb8>\x18'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)
login_manager.login_view = "login"
from .models import User,Category,Budget

@app.route("/dashboard")
@login_required
def dashboard():
    if request.method=="POST":
        bud = Budget()
    return render_template("dashboard.html",budgets=
Budget.query.order_by(Budget.id.desc()).filter_by(user=current_user.i
d))

@app.route("/")
def hello():
    return render_template("home.html")

@app.route("/login",methods=["POST","GET"])
def login():
```

```
    if current_user.is_authenticated:
        return redirect(url_for("dashboard"))
    if request.method == "POST":
        email = request.form.get("email",'').lower()
        password = request.form.get("password",'')
        user = User.query.filter_by(email=email).first()
        obj = Bcrypt()
        redirect_to = request.args.get("next","/dashboard")
        if user and obj.check_password_hash(user.password,password):
            login_user(user,remember=True)
            return redirect(redirect_to)
        else:
            flash("Invalid credentials",'error')
    return render_template("login.html")

@app.route("/signup",methods=["GET","POST"])
def signup():
    from .validators import UserValidator
    if current_user.is_authenticated:
        return redirect(url_for("dashboard"))
    if request.method=="POST":
        # Validation code for signup page
        firstName = request.form.get("firstName",'')
        lastName = request.form.get("lastName","")
        userName = request.form.get("userName",'')
        email = request.form.get("email",'').lower()
        password = request.form.get("password",'')
        msg = UserValidator()
        msg =
msg.validate(firstName,lastName,userName,email,password)
        if msg==True:
            obj = Bcrypt()
            pwd =
obj.generate_password_hash(password).decode("utf-8")
            user =
User(firstName=firstName,lastName=lastName,email=email,userName=userN
ame,password=pwd)
            db.session.add(user)
```

```
            db.session.commit()
            flash("Account created successfully",'success')
            return redirect(url_for("login"))
        else:
            flash(msg,'error')
    return render_template("signup.html")

@app.route("/logout",methods=["POST","GET"])
def logout():
    logout_user()
    return redirect(url_for("login"))

@app.route("/create-budget",methods=["POST","GET"])
@login_required
def createBudget():
    if request.method=="POST":
        from .models import Budget
        budget =
Budget(name=request.form.get("name"),amount=request.form.get("amount"
),user=User.query.filter_by(id=current_user.id).first().id,is_active=
True)
        db.session.add(budget)
        db.session.commit()
        totalCategory = Category(amount =
int(request.form.get("amount")),category="Total",budget=budget.id)
        db.session.add(totalCategory)
        db.session.commit()
        flash("Created budget successfully")
        return redirect("dashboard")
    return render_template("addbudget.html")

@app.route("/budget/<id>/",methods=["GET"])
def budgetID(id):
    budget = Budget.query.get(id)
    amount = []
    category = []
    for i in budget.categories:
        amount.append(i.amount)
```

```
        category.append(i.category)
    import json
    return
render_template("budgetID.html",budget=Budget.query.get(id),amount =
json.dumps(amount),category = json.dumps(category))

@app.route("/budget/<id>/delete")
def deleteBudget(id):
    budget = Budget.query.get(id)
    db.session.delete(budget)
    db.session.commit()
    flash("Budget deleted successfully")
    return redirect(url_for("dashboard"))

@app.route("/add-category/<id>/",methods=["POST","GET"])
def addCategory(id):
    if request.method=="POST":
        category =
Category(budget=id,category=request.form.get("category"),amount=int(r
equest.form.get("amount")))
        total_category =
Category.query.filter_by(budget=id,category="Total").first()
        total_category.amount =
total_category.amount-int(request.form.get("amount"))
        if(total_category.amount<=0):

sendMail(Budget.query.get(id).name,-total_category.amount,current_use
r.email)
        db.session.add(category)
        db.session.add(total_category)
        db.session.commit()
        return redirect(url_for("budgetID",id=id))
    return render_template("addcategory.html")

@app.route("/delete-category/<bid>/<cid>/",methods=["GET","POST"])
def deleteCategory(bid,cid):
    category = Category.query.filter_by(budget=bid,id=cid).first()
```

```
    total_category =
Category.query.filter_by(category="Total",budget=bid).first()
    total_category.amount = total_category.amount + (category.amount)
    db.session.delete(category)
    db.session.commit()
    return redirect(url_for("budgetID",id=bid))

def sendMail(name,amount,email):
    print(name,amount)
    import os
    from sendgrid import SendGridAPIClient
    from sendgrid.helpers.mail import Mail
    message = Mail(
    from_email='111519205057@smartinternz.com',
    to_emails=email,
    subject='You Exceeded the budget!!!',
    html_content="<strong>Alert for budget {}<strong><br>exceeded
{}".format(name,amount))
    try:
        import os
        from dotenv import load_dotenv
        load_dotenv()
        sg = SendGridAPIClient(os.getenv('SENDGRID_API_KEY'))
        response = sg.send(message)
    except Exception as e:
        print(e.message)
```

13.2. GITHUB AND PROJECT DEMO LINK

GitHub Link : https://github.com/IBM-EPBL/IBM-Project-33658-1660225031

Project Demo Link :
https://drive.google.com/file/d/1TwKgK_RoYVDCAYJ8lJV7_Lh4KSgXNYQW/view?usp=sharing