Finbud – Personal Expense Tracker Report

1. Introduction:

1.1 Project Overview:

Fin_Bud is a personal finance tracking application. Personal expense tracker is required to maintain budget & get useful insights about the expenses. By understanding what you spend money on and how much you spend, you can see exactly where your cash is going and areas where you can cut back. FinBud categorize your expenses as needs / wants and help you get a good idea of your purchasing behavior.

1.2 Purpose:

Fin Bud 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 expenditure in graphical forms. They have an 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.

2. Literature Survey:

2.1 Existing Problem:

Some of the conventional methods used to tackle this problem in normal circumstances are like making use of a sticky note by normal users, Proficient people deal with this kind problems by using spreadsheet to record expenses and using a ledger to maintain large amounts data by especially by experts. As this shows that it is variable methods used by different people. This makes using this data inconsistent. There are still problems in areas like there is no assurance for data consistency, there are chances of critical inputs can be missed and the manual errors may creep in.

2.2 References:

Mint offers the best-known free budgeting app on the market. It's a great option for anyone looking to improve their spending habits. The app is free, but you may see targeted financial product advertisements. You may sync your financial accounts within the app or manually add transactions. Mint allows you to see all your accounts in one place and keep track of your spending daily. The app automatically organizes your spending so that you can see totals by category at a glance. Mint also offers monthly bill tracking, including payment reminders to avoid late fees.

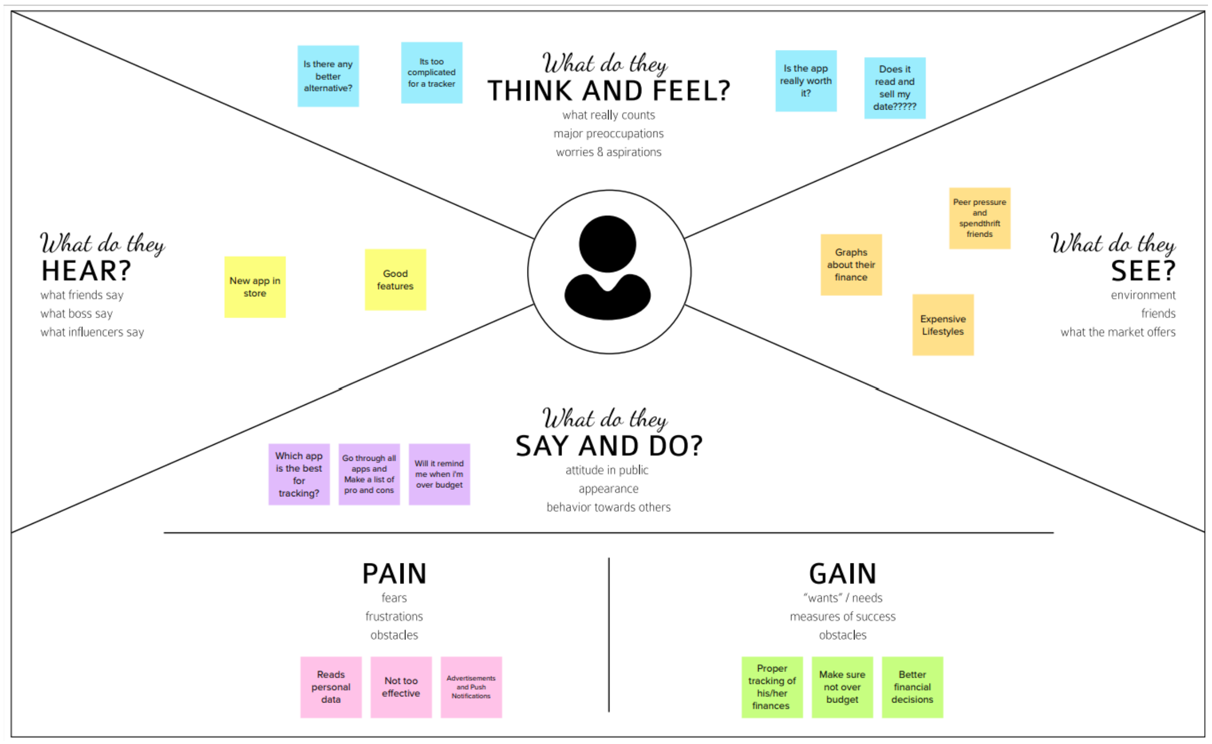
Mobiwik Expense Tracking Application Mobikwik came up with a new feature in their app called Expense Manager. With this feature, you can track and manage your expenditures(expenses), savings, reminders and bill payments. This is a personal budget management app that tracks your expenditures and income and gives you recommendations to make you economically strong.

2.3 Problem Statement:

An app to track the daily expenses, not only help in saving money but also help in setting financial goals for the future. Need of an app to understand where our money is being spent every day & to set some cutbacks and such to help reduce expenditure. This project is developed to work more efficiently in comparison to other trackers and avoid manual calculation.

3. Ideation & Proposed Solution:

3.1 Empathy Map Canvas:



What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Is there any better alternative?
Its too complicated for a tracker
Is the app really worth it?
Does it read and sell my date?????
What do they HEAR?
what friends say
what boss say
what influencers say
New app in store
Good features
What do they SEE?
environment
friends
what the market offers
Peer pressure and spendthrift friends
Graphs about their finance
Expensive Lifestyles
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Which app is the best for tracking?
Go through all apps and Make a list of pro and cons
Will it remind me when i'm over budget
PAIN
fears
frustrations
obstacles
Reads personal data
Not too effective
Advertisements and Push Notifications
GAIN
"wants" / needs
measures of success
obstacles
Proper tracking of his/her finances
Make sure not over budget
Better financial decisions

3.2 Ideation & Brainstorming:

2

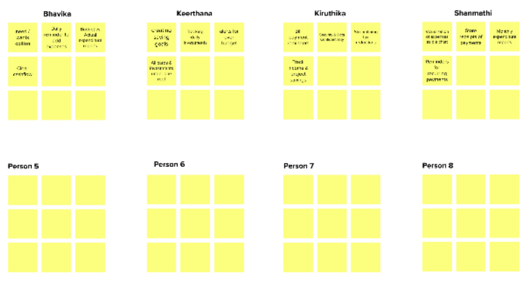
Brainstorm

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

10 minutes

Add on Features

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!



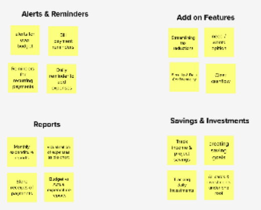
Bhavika
need / wants option
Daily reminder to add expenses
Budget vs Actual expenditure reports
Clear cashflow
Keerthana
creating saving goals
Tracking daily investments
alerts for over budget
All cards & investments under one roof
Kiruthika
Bill payment reminders
Security & Data Confidentiality
Streamlining tax reductions
Track income & project savings
Shanmathi
Visualisation of expenses as pie chart
Store receipts of payments
Monthly expenditure reports
Reminders for recurring payments
Person 5
Person 6
Person 7
Person 8

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, 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



Alerts & Reminders
alerts for over budget
Bill payment reminders
Reminders for recurring payments
Daily reminder to add expenses
Add on Features
Streamlining tax reductions
need / wants option
Security & Data Confidentiality
Clear cashflow
Reports
Monthly expenditure reports
Visualisation of expenses as pie chart
Store receipts of payments
Budget vs Actual expenditure reports
Savings & Investments
Track income & project savings
creating saving goals
Tracking daily investments
All cards & investments under one roof

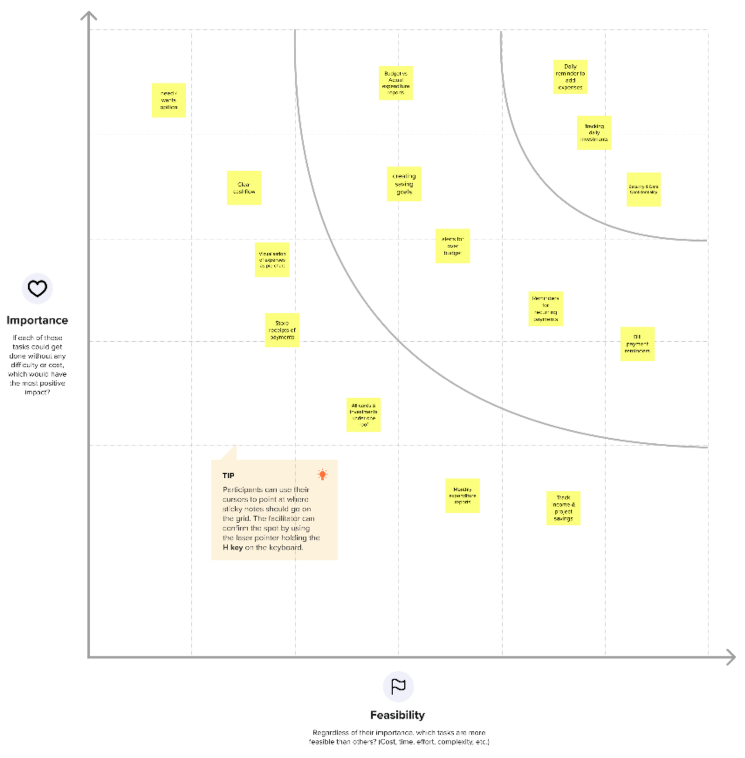
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

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



TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

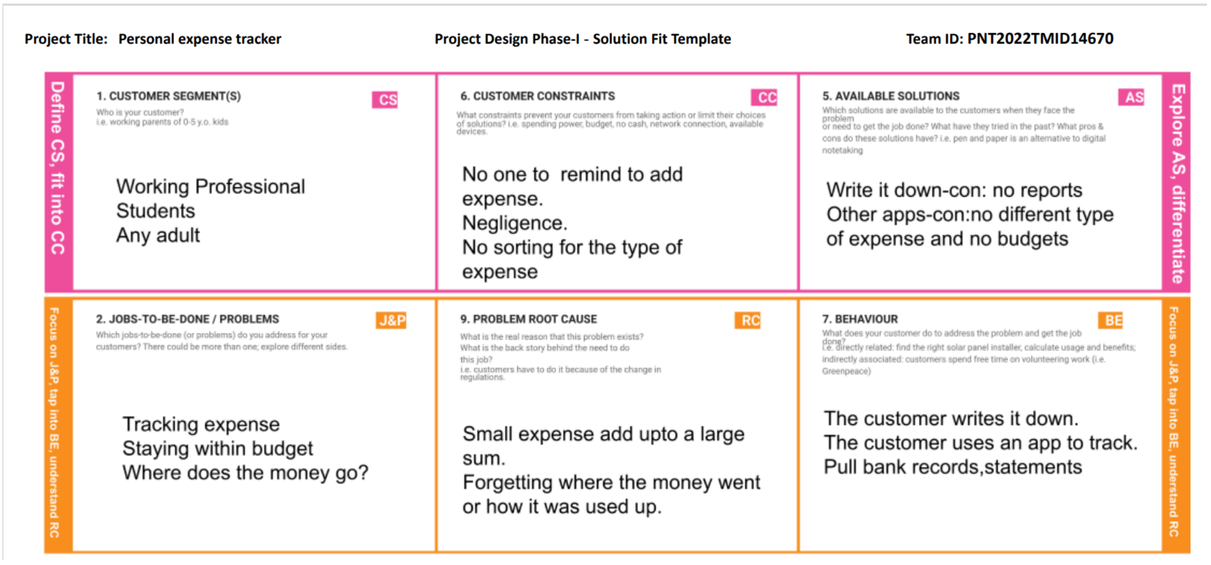
Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Tracking all expenses incurred and drawing proper conclusions from various visualization
2.	Idea / Solution description	Create a web application that will allow users to input their daily expenses and have useful visualizations and reports sent to user.
3.	Novelty / Uniqueness	The app will have an additional category of want and need. All the spendings have to be under one. 'Add a budget' section too.
4.	Social Impact / Customer Satisfaction	The customer might be pleased to have his spending in check and have an app to remind him when he is over budget and if he misses daily entry.
5.	Business Model (Revenue Model)	Revenue can be generated by adding a consulting feature. The user shall pay for consultation over his financial records. Some ads can be sprinkled across the app.
6.	Scalability of the Solution	This project is highly feasible and can later on be further updated with other additional features as well .

3.4 Problem Solution Fit



Project Title: Personal expense tracker

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID14670

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

Working Professional
Students
Any adult

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.

No one to remind to add expense.
Negligence.
No sorting for the type of expense

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

Write it down-con: no reports
Other apps-con:no different type of expense and no budgets

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.

Tracking expense
Staying within budget
Where does the money go?

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.

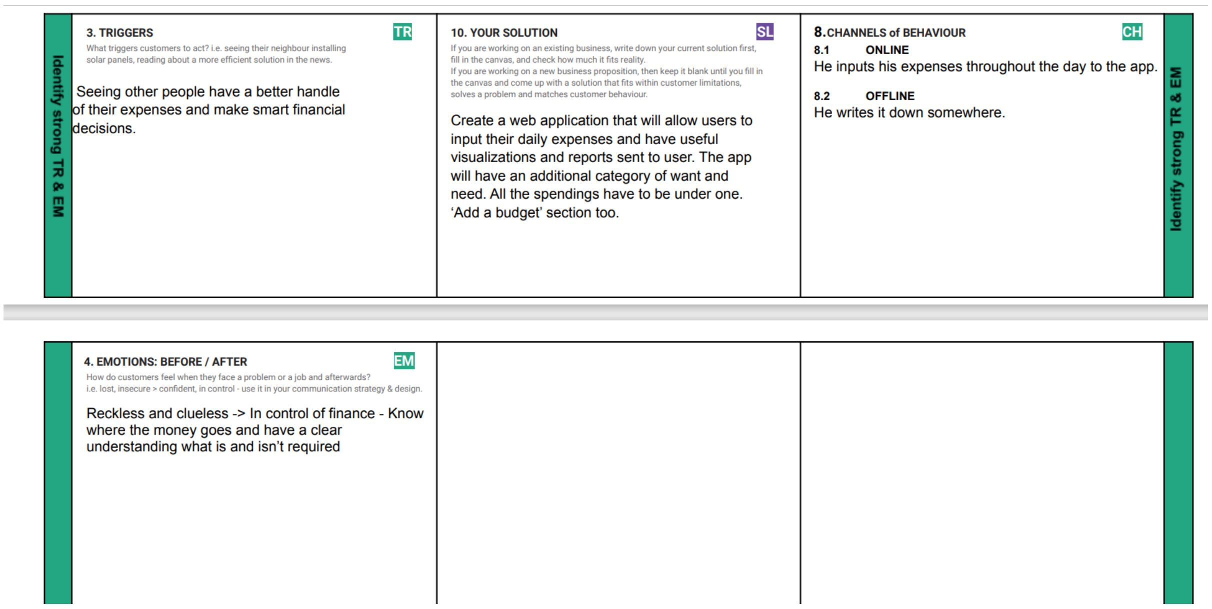
Small expense add upto a large sum.
Forgetting where the money went or how it was used up.

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)

The customer writes it down.
The customer uses an app to track.
Pull bank records,statements

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


Identify strong TR & EM
TR
SL
CH
EM
Identify strong TR & EM

3. TRIGGERS

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

Seeing other people have a better handle of their expenses and make smart financial decisions.

10. YOUR SOLUTION

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.

Create a web application that will allow users to input their daily expenses and have useful visualizations and reports sent to user. The app will have an additional category of want and need. All the spendings have to be under one. 'Add a budget' section too.

8. CHANNELS of BEHAVIOUR

8.1 ONLINE
He inputs his expenses throughout the day to the app.

8.2 OFFLINE
He writes it down somewhere.

4. EMOTIONS: BEFORE / AFTER

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.

Reckless and clueless -> In control of finance - Know where the money goes and have a clear understanding what is and isn't required

4. Requirement Analysis:

4.1 Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Add/Delete Transaction	-
FR-4	Data Representation as Charts	Transactions entered
FR-5	Overview With Filter	Overall transactions
FR-6	Password	-
FR-8	Reminder	Reminding through notification or mail

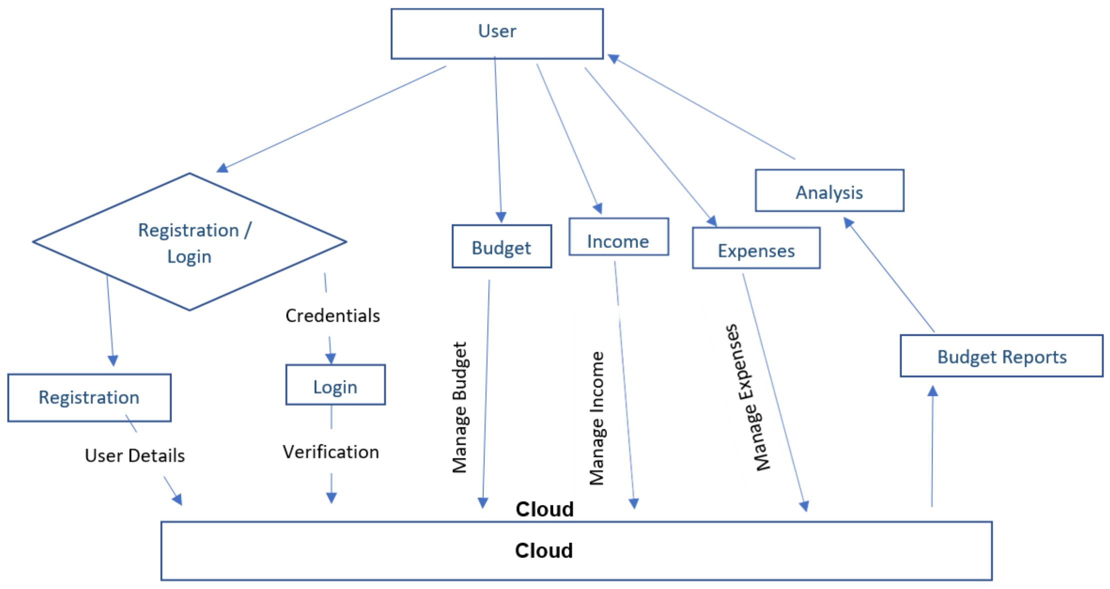
4.2 Non-Functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Effective and easy use of the application.
NFR-2	**Security**	Security of personal information provided.
NFR-3	**Reliability**	Highly reliable in a secured environment.
NFR-4	**Performance**	Provide immediate results to user actions.

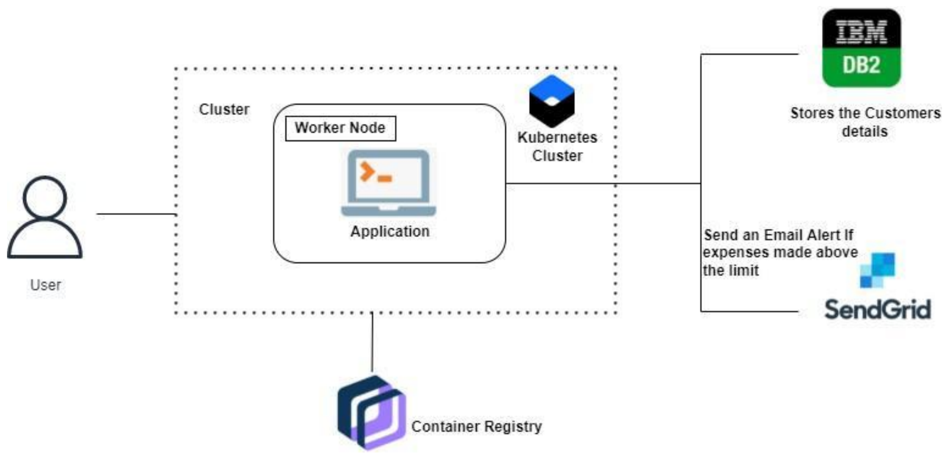
NFR-5	**Availability**	System availability at all times whenever the user logs in.
NFR-6	**Scalability**	Scalable enough to support at least 1000 visits at the same time.

5. Project Design

5.1 Data Flow Diagrams:



5.2 Solution & Technical Architecture:

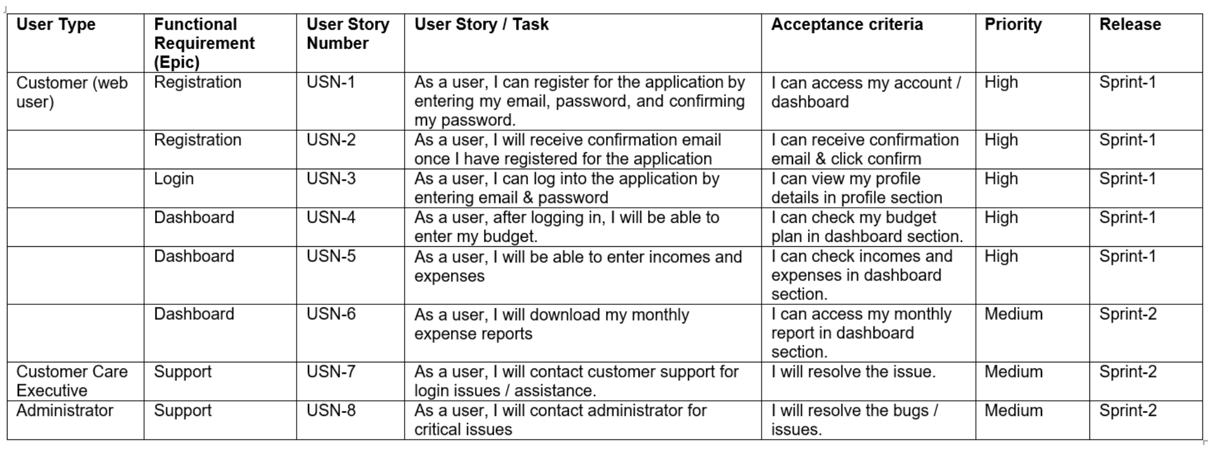


S.No	Component	Description	Technology
1.	User Interface	User interacts with application through Web UI or chatbot	HTML, CSS, JavaScript , IBM Watson Assistant
2.	Login	If the user is a registered user, the user should	Python

		login	
3.	Register	If the user is new, he will be registered and added to database	Python

4.	Dashboard	To add expenses/income, view reports...	Python,IBM Watson Assistant
5.	Database	To store user's daily spendings	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	NA	NA
9.	External API-2	NA	NA
10.	Machine Learning Model	Vizualizing expenses	Matplotlib, other packages
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

5.3 User Stories:



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (web 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
	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password	I can view my profile details in profile section	High	Sprint-1
	Dashboard	USN-4	As a user, after logging in, I will be able to enter my budget.	I can check my budget plan in dashboard section.	High	Sprint-1
	Dashboard	USN-5	As a user, I will be able to enter incomes and expenses	I can check incomes and expenses in dashboard section.	High	Sprint-1
	Dashboard	USN-6	As a user, I will download my monthly expense reports	I can access my monthly report in dashboard section.	Medium	Sprint-2
Customer Care Executive	Support	USN-7	As a user, I will contact customer support for login issues / assistance.	I will resolve the issue.	Medium	Sprint-2
Administrator	Support	USN-8	As a user, I will contact administrator for critical issues	I will resolve the bugs / issues.	Medium	Sprint-2

6. PROJECT PLANNING & SCHEDULING

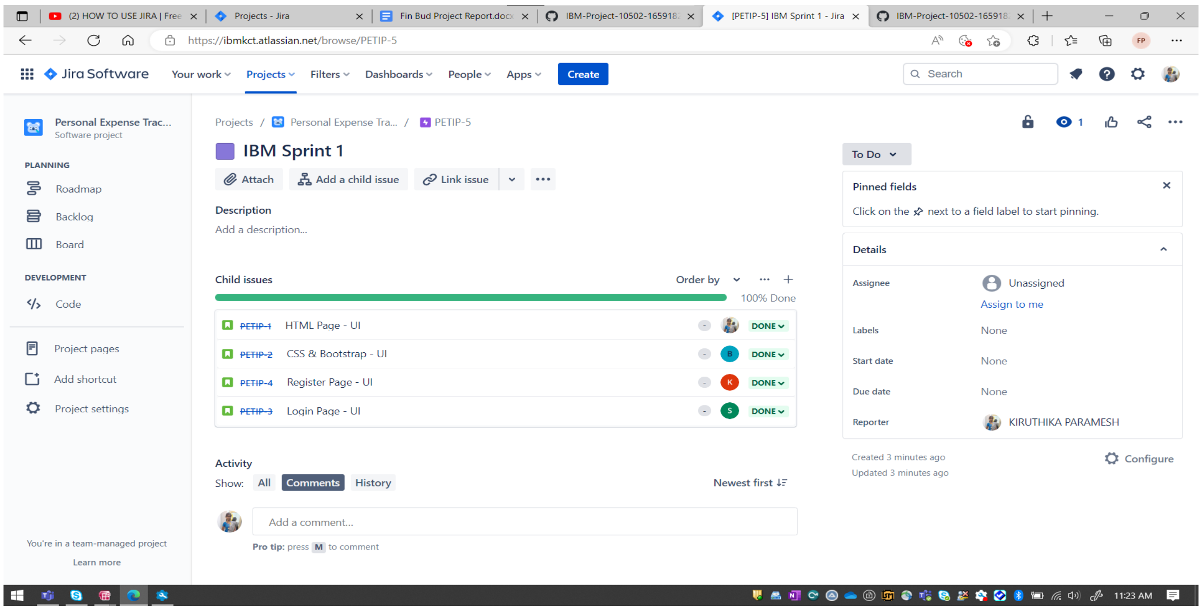
6.1Sprint Planning & Estimation

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.	2	High	Shanmathi
Sprint 2	Login	USN-2	As a user, I can log into the application by entering email & password	2	High	Bhavika
Sprint 2	Dashboard	USN-3	As a user, after logging in, I will be able to enter my budget.	2	High	Kiruthika
Sprint 3	Dashboard	USN-4	As a user, I will be able to enter incomes and Expenses	2	High	Keerthana
Sprint 4	Dashboard	USN-5	As a user, I will download my monthly expense reports	2	Medium	Bhavika
Sprint 1	Support	USN-6	As a user, I will contact customer support for login issues / assistance.	1	Low	Shanmathi
Sprint 1	Support	USN-7	As a user, I will contact administrator for critical issues	1	Low	Keerthana

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	1 Oct 2022	6 Oct 2022	10	6 Oct 2022
Sprint-2	20	6 Days	6 Oct 2022	11 Oct 2022	20	11 Oct 2022
Sprint-3	20	18 Days	12 Oct 2022	30 Oct 2022	20	30 Oct 2022
Sprint-4	20	14 Days	1 Nov 2022	14 Nov 2022	20	14 Nov 2022

6.3 Reports from JIRA



https://ibmkct.atlassian.net/browse/PETIP-5

Projects / Personal Expense Tra... / PETIP-5

IBM Sprint 1

Description
Add a description...

Child issues — 100% Done

~~PETIP-1~~	HTML Page - UI	DONE
~~PETIP-2~~	CSS & Bootstrap - UI	DONE
~~PETIP-4~~	Register Page - UI	DONE
~~PETIP-3~~	Login Page - UI	DONE

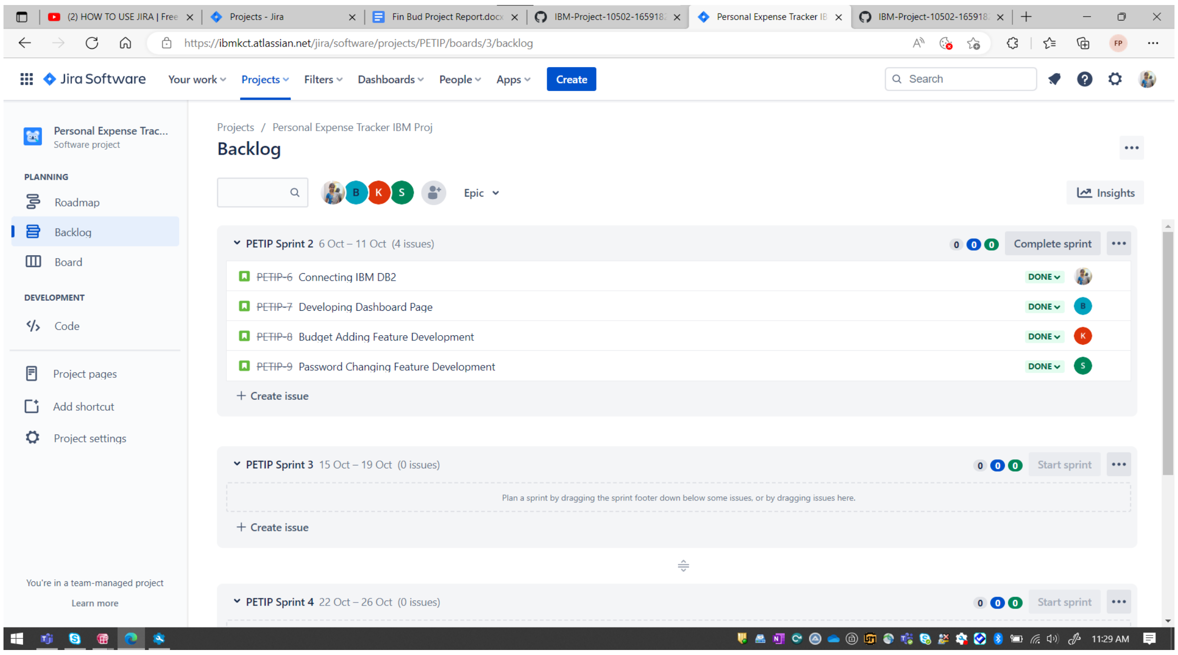
Activity
Show: All Comments History — Newest first
Add a comment...
Pro tip: press M to comment

Pinned fields
Click on the next to a field label to start pinning.

Details

Assignee	Unassigned (Assign to me)
Labels	None
Start date	None
Due date	None
Reporter	KIRUTHIKA PARAMESH

Created 3 minutes ago
Updated 3 minutes ago



https://ibmkct.atlassian.net/jira/software/projects/PETIP/boards/3/backlog

Projects / Personal Expense Tracker IBM Proj

Backlog

PETIP Sprint 2 6 Oct – 11 Oct (4 issues) 0 0 0 Complete sprint

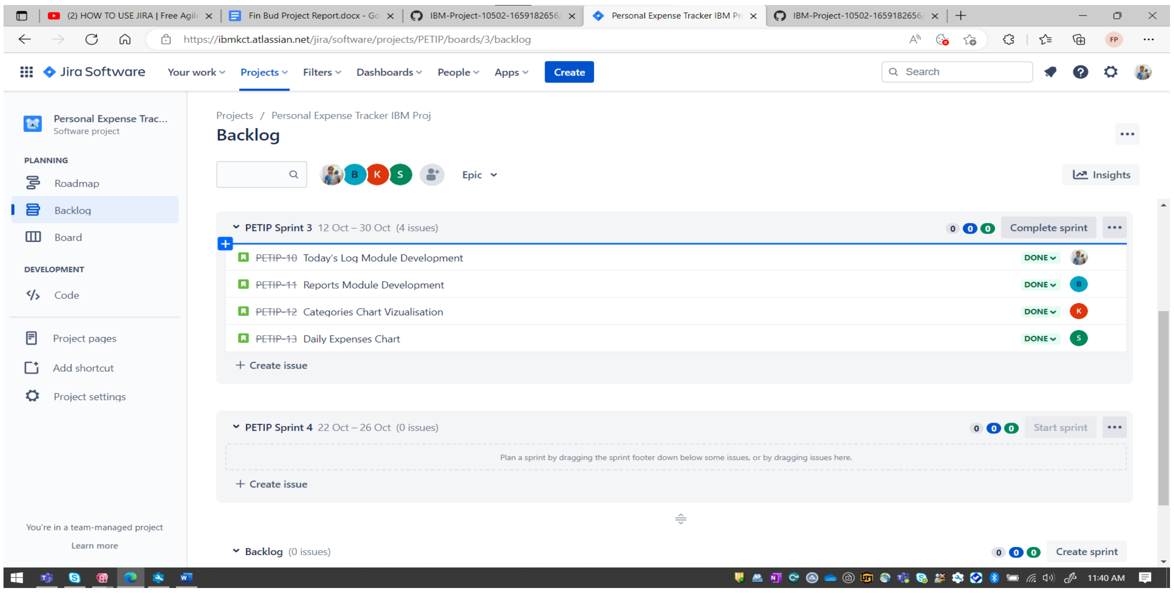
~~PETIP-6~~ Connecting IBM DB2 DONE
~~PETIP-7~~ Developing Dashboard Page DONE
~~PETIP-8~~ Budget Adding Feature Development DONE
~~PETIP-9~~ Password Changing Feature Development DONE
+ Create issue

PETIP Sprint 3 15 Oct – 19 Oct (0 issues) 0 0 0 Start sprint

Plan a sprint by dragging the sprint footer down below some issues, or by dragging issues here.
+ Create issue

PETIP Sprint 4 22 Oct – 26 Oct (0 issues) 0 0 0 Start sprint

11:29 AM



https://ibmkct.atlassian.net/jira/software/projects/PETIP/boards/3/backlog

Projects / Personal Expense Tracker IBM Proj

Backlog

PETIP Sprint 3 12 Oct – 30 Oct (4 issues) 0 0 0 Complete sprint

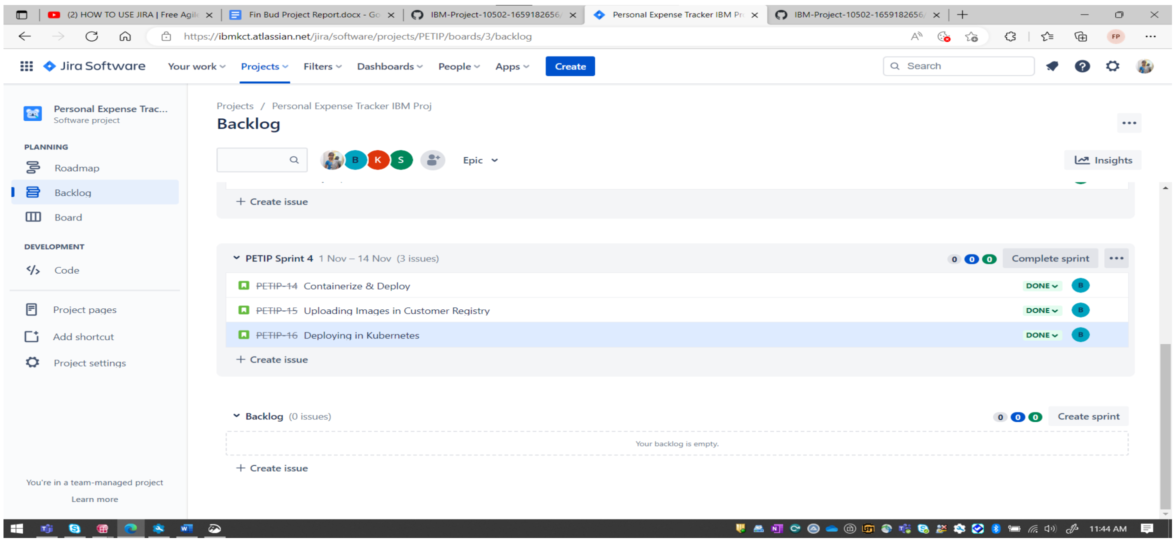
~~PETIP-10~~ Today's Log Module Development DONE
~~PETIP-11~~ Reports Module Development DONE
~~PETIP-12~~ Categories Chart Vizualisation DONE
~~PETIP-13~~ Daily Expenses Chart DONE
+ Create issue

PETIP Sprint 4 22 Oct – 26 Oct (0 issues) 0 0 0 Start sprint

Plan a sprint by dragging the sprint footer down below some issues, or by dragging issues here.
+ Create issue

Backlog (0 issues) 0 0 0 Create sprint

11:40 AM



https://ibmkct.atlassian.net/jira/software/projects/PETIP/boards/3/backlog

Projects / Personal Expense Tracker IBM Proj

Backlog

+ Create issue

PETIP Sprint 4 1 Nov – 14 Nov (3 issues) 0 0 0 Complete sprint

~~PETIP-14~~ Containerize & Deploy DONE
~~PETIP-15~~ Uploading Images in Customer Registry DONE
~~PETIP-16~~ Deploying in Kubernetes DONE
+ Create issue

Backlog (0 issues) 0 0 0 Create sprint

Your backlog is empty.
+ Create issue

11:44 AM

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

7.1 Login

Login to your account to view reports or add any transactions

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

def login():

    msg = ""

    if request.method == "POST":

        username = request.form["username"]

        password = request.form["password"]

        sql = "SELECT clients.*,budgets.MAXBUDGET FROM clients LEFT
JOIN BUDGETS ON CLIENTs.ID=BUDGETS.ID 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"]

            session["email"] = account["EMAIL"]

            session["username"] = account["USERNAME"]

            session["budget"] = account["MAXBUDGET"]

            print(session["Loggedin"])

            return redirect("/dashboard")

        else:
            msg = "Incorrect login credentials"

    flash(msg)

    return render_template("login.html", title="Login")
```

7.2 Register

Register functionality to use the app

```
@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"]

        password1 = request.form["password1"]

        sql = "SELECT * FROM CLIENTS WHERE username =? or email=? "

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, username)

        ibm_db.bind_param(stmt, 2, email)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

        if account:

            msg = "Account already exists"

        elif password1 != password:

            msg = "re-entered password doesnt match"

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

            msg = "Username should be only alphabets and numbers"

        else:
            sql = "INSERT INTO clients(EMAIL,USERNAME,PASSWORD) VALUES
(?,?,?)"

            stmt = ibm_db.prepare(conn, sql)

            ibm_db.bind_param(stmt, 1, email)

            ibm_db.bind_param(stmt, 2, username)

            ibm_db.bind_param(stmt, 3, password)

            ibm_db.execute(stmt)

            return redirect("/dashboard")

    flash(msg)

    return render_template("register.html", title="Register")
```

7.3 Logout

```
@app.route("/logout")

def logout():

    session.clear()

    return redirect("/")
```

7.4 Add Expense

Add Expenses that were incurred today

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

def addExpense():

    msg = ""

    if request.method == "POST":

        amount = request.form["Amount"]

        need = request.form["Need/Want"]

        category = request.form["category"]

        sql = "INSERT INTO
TRANSACTIONS(USER_ID,AMOUNT,NEED,CATEGORY,DATEADDED)
VALUES(?,?,?,?,CURRENT_DATE)"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, session["id"])
        ibm_db.bind_param(stmt, 2, amount)

        ibm_db.bind_param(stmt, 3, need)

        ibm_db.bind_param(stmt, 4, category)

        if ibm_db.execute(stmt):

            msg = "Successfully Added Expense!!!!"

        else:

            msg = "Expense not added"


    flash(msg)

    return redirect(url_for("logToday"))
```

7.5 Add Budget

Add a monthly budget to your account

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

def addBudget():

    msg = "Enter the budget"

    if request.method == "POST":

        budgetAmount = request.form["budgetAmountToAdd"]

        sql = "INSERT INTO BUDGETS(id,maxbudget) VALUES(?,?)"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, session["id"])

        ibm_db.bind_param(stmt, 2, budgetAmount)

        if ibm_db.execute(stmt):

            session["budget"] = budgetAmount

            msg = "Successfully Set The Budget!!!!"

        else:

            msg = "Budget not set yet"

    flash(msg)

    return redirect(url_for("dashboard"))
```

7.6 Add Income

Any income received during the day

```
@app.route("/addIncome/", methods=["POST", "GET"])
def addIncome():
    msg = ""
    if request.method == "POST":
        amount = request.form["AmountIncome"]
        sql = "INSERT INTO INCOME(ID,AMOUNT,DATEADDED) 
VALUES(?,?,CURRENT_DATE)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, session["id"])
        ibm_db.bind_param(stmt, 2, amount)
        if ibm_db.execute(stmt):
            msg = "Successfully Added Income!!!!"
        else:
            msg = "Income not added"

    flash(msg)
    return redirect(url_for("logToday"))
```

7.7 Change Password

Change password if old one is forgotten or want a better password

```
@app.route("/changePassword/", methods=["POST", "GET"])
def changePassword():
    msg = "Enter the new password"
    if request.method == "POST":
        pass1 = request.form["pass1"]
        pass2 = request.form["pass2"]
        if pass1 == pass2:
            sql = "UPDATE CLIENTS SET password=? where id=?"

            stmt = ibm_db.prepare(conn, sql)

            ibm_db.bind_param(stmt, 1, pass1)

            ibm_db.bind_param(stmt, 2, session["id"])

            if ibm_db.execute(stmt):

                msg = "Successfully Changed Password!!!!"


        else:

            msg = "Passwords not equal"

    flash(msg)

    return redirect(url_for("dashboard"))
```

7.8 Change budget

You can change your monthly budget

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

def changeBudget():

    msg = "Enter the new budget"

    if request.method == "POST":

        budgetAmount = request.form["budgetAmount"]

        sql = "UPDATE BUDGETS SET maxBudget=? where id=?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, budgetAmount)

        ibm_db.bind_param(stmt, 2, session["id"])

        if ibm_db.execute(stmt):

            session["budget"] = budgetAmount

            msg = "Successfully Changed Budget!!!!"

        else:

            msg = "Budget not changed"

    flash(msg)
    return redirect(url_for("dashboard"))
```

7.9 Reports

Reports of Need v Want, Categories and Transactions-Budget-Time

```
@app.route("/reports")
def reports():
    return render_template("reports.html", title="Reports")



@app.route("/needVwant/")
def needVwant():
    sql = "SELECT Sum(amount) AS amount, need FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND  user_id = ? GROUP BY NEED ORDER BY need"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    values = []
    labels = []
    print(transactions)
    for transaction in transactions:
        values.append(transaction["AMOUNT"])
        labels.append(transaction["NEED"])
    fig = plt.figure(figsize=(10, 7))
    plt.pie(values)
    plt.title("Need v Want")
    plt.legend(["WANT", "NEED"])
    canvas = FigureCanvas(fig)
    img = BytesIO()
    fig.savefig(img)
    img.seek(0)
    return send_file(img, mimetype="image/png")
```

```
@app.route("/categoriesChart/")
def categoriesChart():
    sql = "SELECT Sum(amount) AS amount, category FROM transactions
WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND  user_id = ? GROUP BY
category ORDER BY category"
    stmt = ibm_db.prepare(conn, sql)
    transactions = fetchall(stmt)
    values = []
    labels = []
    print(transactions)
    for transaction in transactions:
        values.append(transaction["AMOUNT"])
        labels.append(transaction["CATEGORY"])
    fig = plt.figure(figsize=(10, 7))
    plt.pie(values, labels=labels)
    plt.title("Categories")
    plt.legend()
    canvas = FigureCanvas(fig)
    img = BytesIO()
    fig.savefig(img)
    img.seek(0)
    return send_file(img, mimetype="image/png")


##edit the legend... all visualizations workkkkkk!!!!!!!
@app.route("/dailyLineChart/")
def dailyLineChart():
    sql = "SELECT Sum(amount) AS amount, DAY(dateadded) as dateadded
FROM transactions WHERE DAYS(CURRENT_DATE)-DAYS(DATEADDED)<29 AND
user_id = ? GROUP BY dateadded ORDER BY dateadded"

    stmt = ibm_db.prepare(conn, sql)

    transactions = fetchall(stmt)

    x = []

    y = []

    print(transactions)

    for transaction in transactions:

        y.append(transaction["AMOUNT"])

        x.append(transaction["DATEADDED"])

        ##get budget

    sql = "SELECT MAXBUDGET FROM budgets WHERE id = ?"

    stmt = ibm_db.prepare(conn, sql)

    ibm_db.bind_param(stmt, 1, session["id"])

    ibm_db.execute(stmt)

    budget = ibm_db.fetch_assoc(stmt)

    print(budget)

    fig = plt.figure(figsize=(10, 7))

    plt.scatter(x, y)

    plt.plot(x, y, "-")

    if budget:

        plt.axhline(y=budget["MAXBUDGET"], color="r", linestyle="-")

    plt.xlabel("Day")

    plt.ylabel("Transaction")

    plt.title("Daily")

    plt.legend()

    canvas = FigureCanvas(fig)

    img = BytesIO()

    fig.savefig(img)
    img.seek(0)

    return send_file(img, mimetype="image/png")
```

7.10 Database Schema (if Applicable)

Tables:

- Clients
- Budgets
- Income
- Transactions

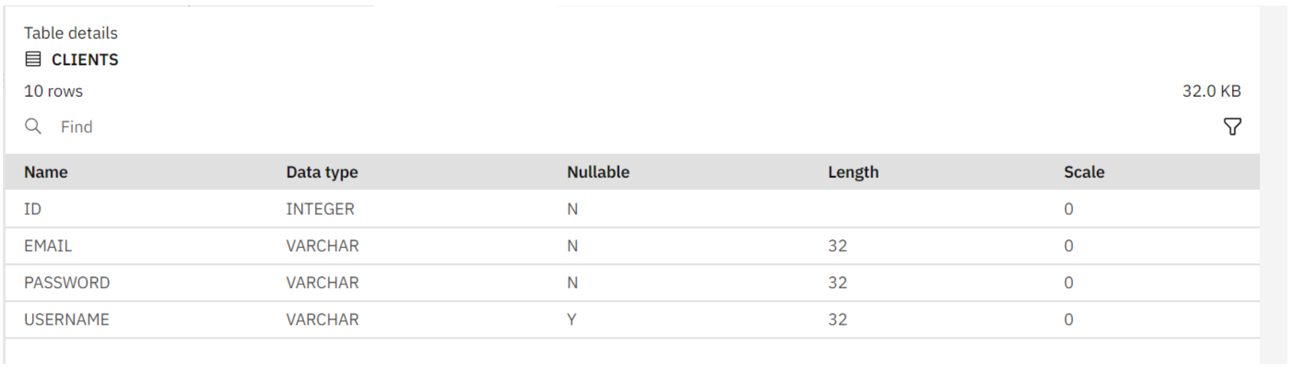


Table details
CLIENTS
10 rows
32.0 KB

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
EMAIL	VARCHAR	N	32	0
PASSWORD	VARCHAR	N	32	0
USERNAME	VARCHAR	Y	32	0

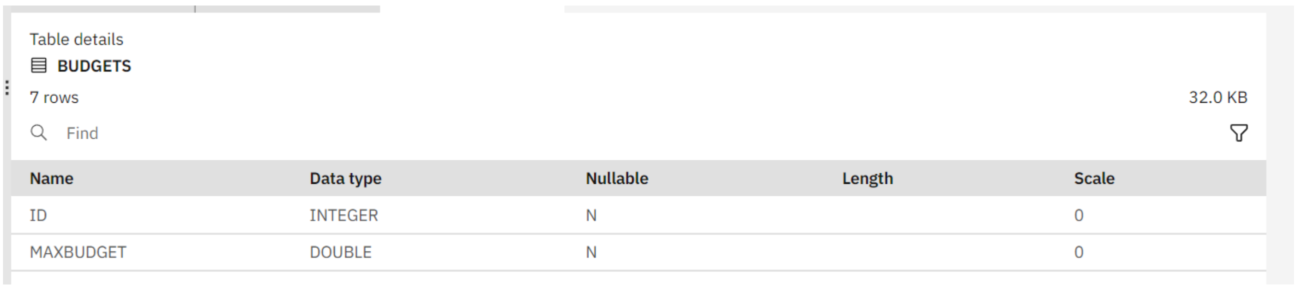


Table details
BUDGETS
7 rows
32.0 KB

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
MAXBUDGET	DOUBLE	N		0

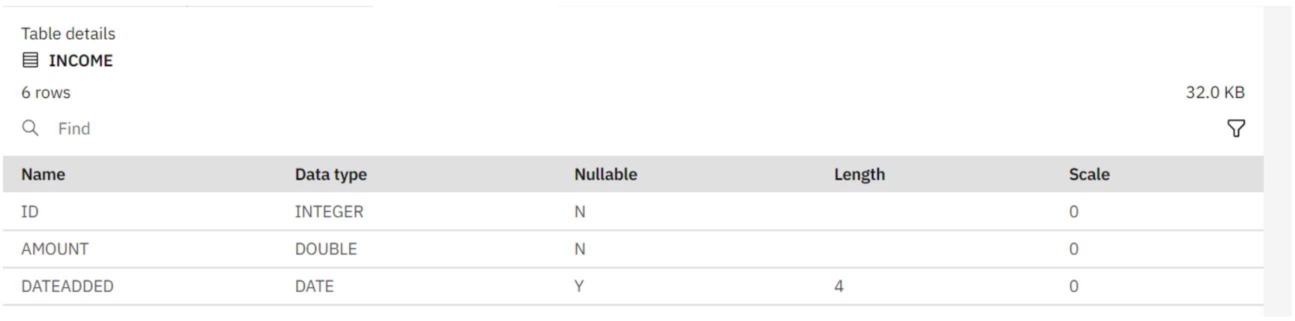


Table details
INCOME
6 rows
32.0 KB

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
AMOUNT	DOUBLE	N		0
DATEADDED	DATE	Y	4	0

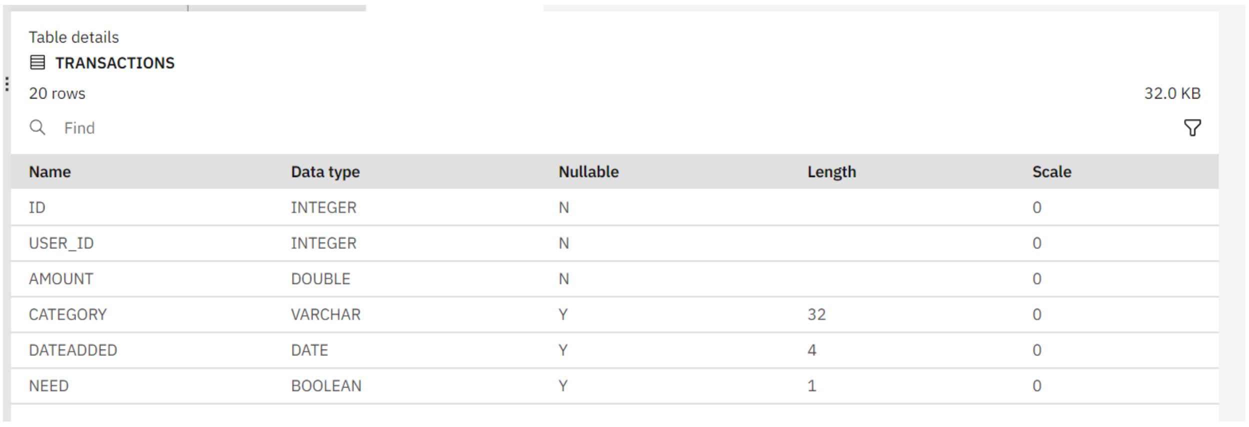


Table details

TRANSACTIONS

20 rows | 32.0 KB

Find

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
USER_ID	INTEGER	N		0
AMOUNT	DOUBLE	N		0
CATEGORY	VARCHAR	Y	32	0
DATEADDED	DATE	Y	4	0
NEED	BOOLEAN	Y	1	0

8. TESTING

8.1Test Cases

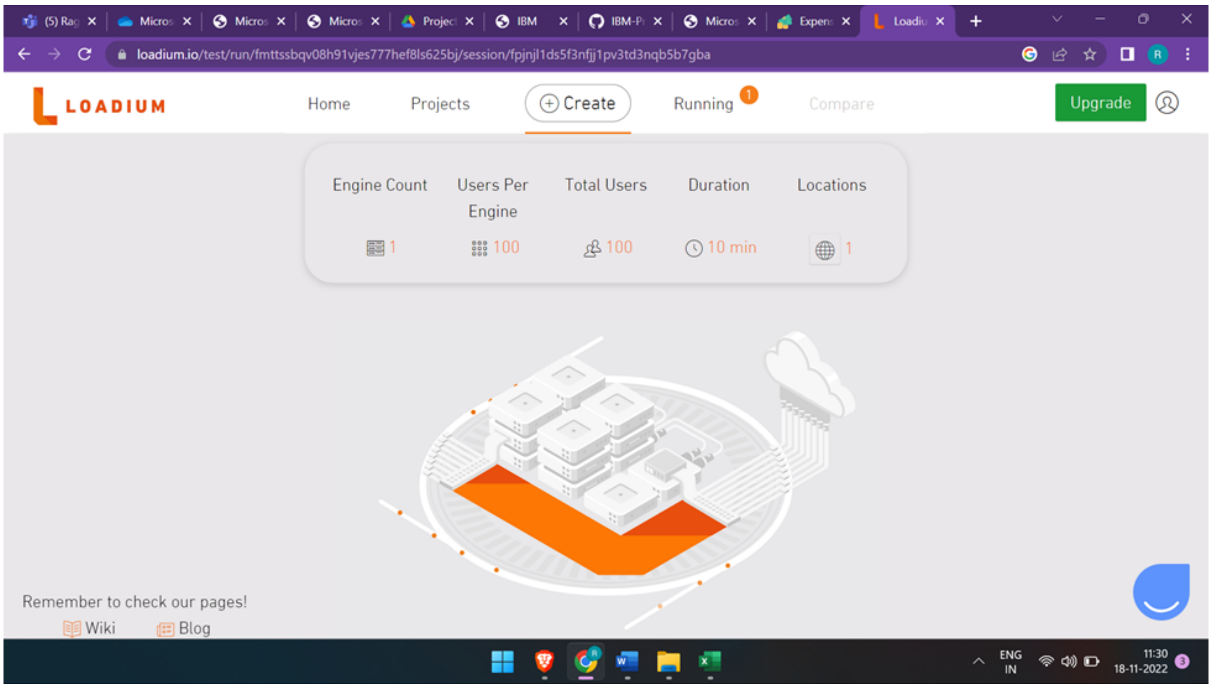
Test case ID	Test Scenario	Expected Result	Actual Result
HomePage_TC_OO1	Verify user is able to land on home page and see the Login & Signup buttons when user enters the url	Login/Signup button should display	Working as expected
HomePage_TC_OO2	Verify the UI elements in Login/Signup button	Application should show below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link	Working as expected
RegistrationPage_TC_OO3	Verify the user is able to register and login with the signed up credentials	Application should show below UI elements: a.username text box b.email text box c.password text box c.confirm password text box d.register button e.Already have account? Login instead link	Working as expected
LoginPage_TC_OO4	Verify user is able to log into application with the registered credentials	User should navigate to dashboard page	Working as expected
LoginPage_TC_OO5	Verify user is able to log into application with InValid credentials	Application should show 'Incorrect login credentials ' validation message.	Working as expected
Dashboard_TC_OO6	Verify setting a budget	Application should set the budget and display on the dashboard page.	Working as expected
Dashboard_TC_OO7	Verify if password can be changed	Application should update the password and display 'Successfully changed password!' validation message.	Working as expected
LogPage_TC_OO8	Verify adding expenses	Application should update the expenses and display 'Successfully added expense!' validation message.	Working as expected
LogPage_TC_O09	Verify adding income	Application should update the income and display 'Successfully added income!' validation message.	Working as expected
ReportPage_TC_O10	Verify whether chart is properly displayed or not	The need vs. want comparision is shown in pie chart. Categories of expense are shown in pie chart. Daily expenses are shown as timeline chart.	Working as expected

8.2 User Acceptance Testing

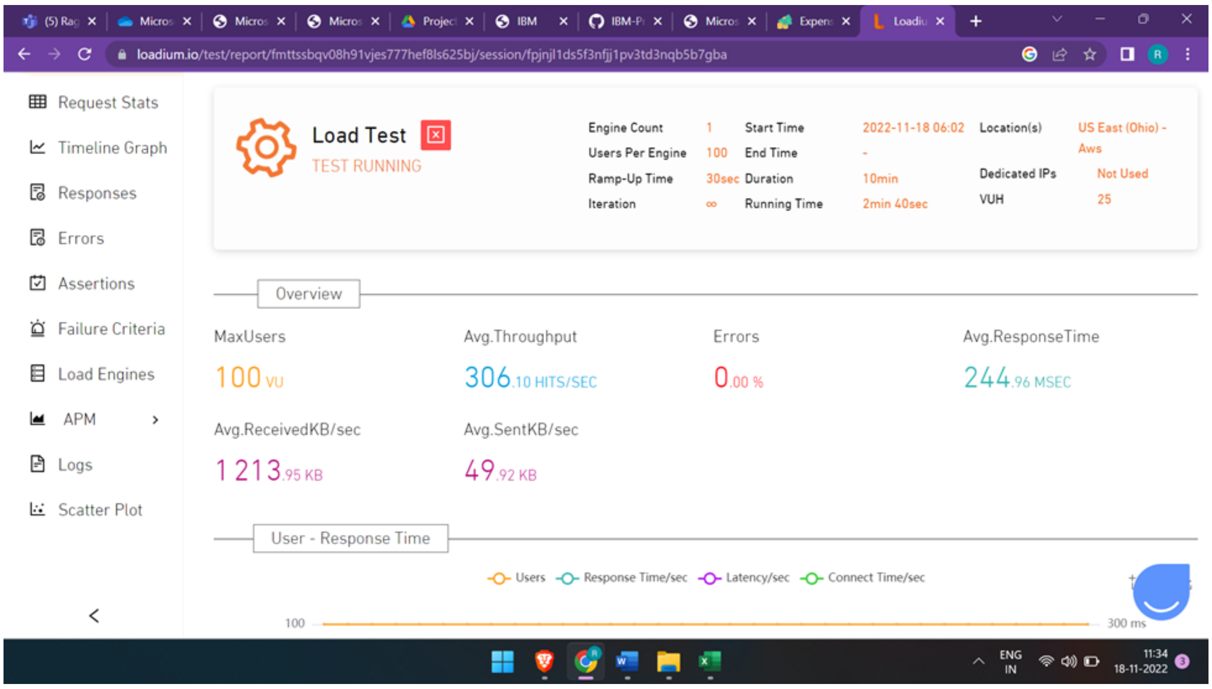
Test case ID	Test Scenario	Test Type	Responsibility	Priority	Test Result	Test Date
HomePage_TC_OO1	Verify user is able to land on home page and see the Login & Signup buttons when user enters the url	Functional	Bhavika	High	Pass	01-11-2022
HomePage_TC_OO2	Verify the UI elements in Login/Signup button	Functional	Kiruthika	High	Pass	02-11-2022
RegistrationPage_TC_OO3	Verify the user is able to register and login with the signed up credentials	Functional	Shanmathi	High	Pass	03-11-2022
LoginPage_TC_OO4	Verify user is able to log into application with the registered credentials	Functional	Keerthana	High	Pass	04-11-2022
LoginPage_TC_OO5	Verify user is able to log into application with InValid credentials	Security	Bhavika	High	Pass	05-11-2022
Dashboard_TC_OO6	Verify setting a budget	Functional	Kiruthika	High	Pass	06-11-2022
Dashboard_TC_OO7	Verify if password can be changed	Security	Shanmathi	High	Pass	07-11-2022
LogPage_TC_OO8	Verify adding expenses	Functional	Keerthana	Medium	Pass	08-11-2022
LogPage_TC_OO9	Verify adding income	Functional	Bhavika	Medium	Pass	09-11-2022
ReportPage_TC_O10	Verify whether chart is properly disp	Functional	Kiruthika	Medium	Pass	10-11-2022

9. RESULTS

9.1 Performance Metrics


loadium.io/test/run/fmttssbqv08h91vjes777hef8ls625bj/session/fpjnjl1ds5f3nfjj1pv3td3nqb5b7gba

Engine Count	Users Per Engine	Total Users	Duration	Locations
1	100	100	10 min	1



Load Test
TEST RUNNING

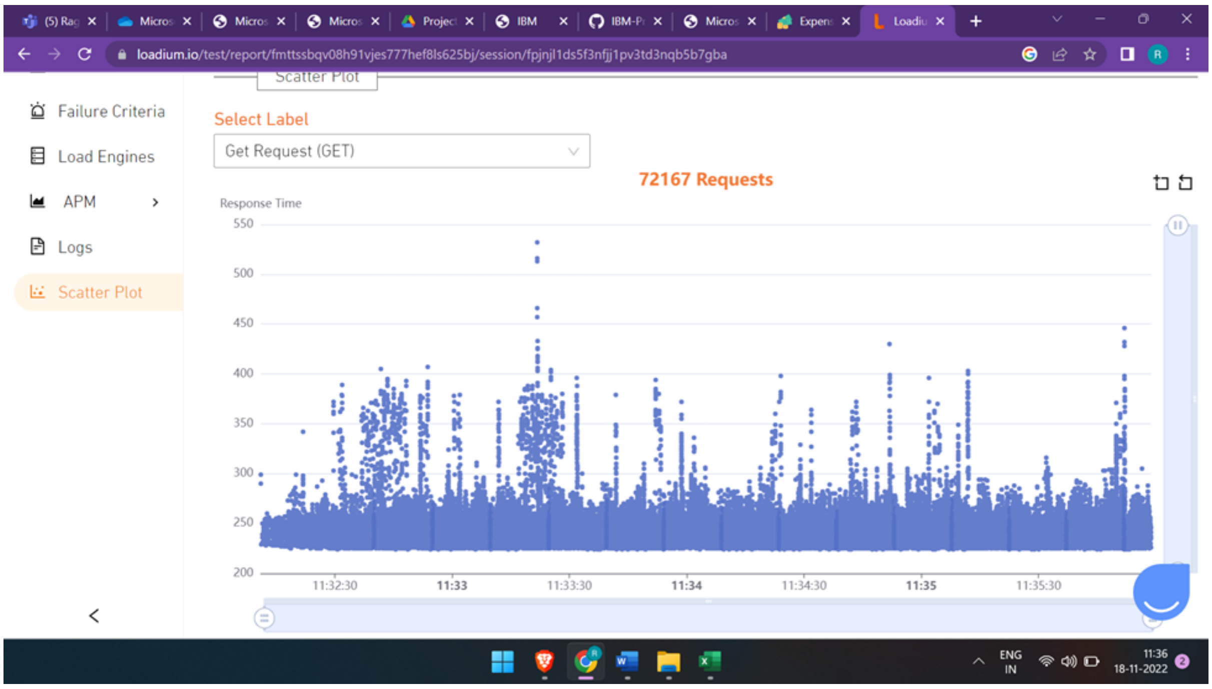
Engine Count	1	Start Time	2022-11-18 06:02	Location(s)	US East (Ohio) - Aws
Users Per Engine	100	End Time	-		
Ramp-Up Time	30sec	Duration	10min	Dedicated IPs	Not Used
Iteration	∞	Running Time	2min 40sec	VUH	25

Overview

MaxUsers	Avg.Throughput	Errors	Avg.ResponseTime
100 VU	306.10 HITS/SEC	0.00 %	244.96 MSEC

Avg.ReceivedKB/sec	Avg.SentKB/sec
1 213.95 KB	49.92 KB

User - Response Time



Select Label: Get Request (GET)

72167 Requests

10. ADVANTAGES & DISADVANTAGES

Advantages:

- Being aware of the state of one's personal finances.
- Having the program on a hand-held device can be a main pro since it can be checked before spending occurs in order to be sure of the available budget.
- Make one more aware of where the money is going way before the end of a pay period or month.

Disadvantages:

Even with constant tracking of one's spending habits, there is no guarantee that financial goals will be met. Although this can be considered to be a con of tracking spending, it could be changed into a pro if one makes up his or her mind to keep trying to properly manage all finances.

11. CONCLUSION

Fin Bud will allow the users to set a budget limit & users will be aware of their spending. This will help them to make better financial decisions before buying a product.

12. FUTURE SCOPE

Voice Based Expense / Income addition

Alerts to friends when spent more than planned budget

13. APPENDIX

Source Code : https://github.com/IBM-EPBL/IBM-Project-10502-1659182656

GitHub & Project Demo Link : https://clipchamp.com/watch/AXJt1GQodTT