Personal Expense Tracker Application

A PROJECT REPORT FROM

Sriram H

Sriram R

Vignesh S

Vishal Surya S K

Team ID : PNT2022TMID05871

Sri Ramakrishna Engineering College

Introduction

1. Project Overview

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money management.Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their 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. Purpose

Following your spending leads to better money management and financial awareness, which helps reach financial goals. One can effectively improve spending habits by properly comprehending expenditures and setting budget boundaries. We provide an application to lessen laborious calculations. Users of this application can keep an automated digital diary. Each user will need to sign up for an account on the system. Application will track expenses, income and daily costs incurred by a user. The greatest businesses have a system for managing and tracking their expenses. This best practise ensures that the expenses are recorded accurately and promptly. A further characteristic of expense and income forecasting improves budget management.

- Prioritize Your Spending
- Become Aware of Poor Spending Habits
- Identify Fraud
- Take Control of Your Finances
- Saving and Investment

3. Problem Statement

Due to the lack of personal finance education in schools and colleges, young adults frequently engage in harmful financial habits and occasionally even incur crippling debt. This is an urgent problem that needs to be resolved with trustworthy tools and quick action. A smart place to start is with a programme designed specifically for young adults who want to begin tracking their spending, splitting their bills, and learning how to set aside money. To take control of their finances, young adults must be urged to regularly assess their spending habits and make any required adjustments. The users of this application would be able to monitor and track their spending effectively with the help of the necessary notifications and alerts.

Literature Survey

1. Existing Problem

For the user's daily and monthly spending, it is necessary to maintain Excel sheets, CSV files, etc. in the current systems. There is no perfect way to easily keep track of daily expenses. To do this, one must maintain a log in a diary or on a computer. Additionally, one must perform all calculations themselves, which can occasionally result in mistakes that result in losses. Since it takes time to produce reports, unravelling the intricacy of financial records by hand could be time-consuming.

Keeping track of expenses is an important aspect of managing personal finances, and expense tracking apps can help you do that easily. While some apps require manual entry of expenses, others automatically take data from linked bank and credit card account statements.

The apps help categorise spending to help understand purchasing habits and generate expense reports.

2. References

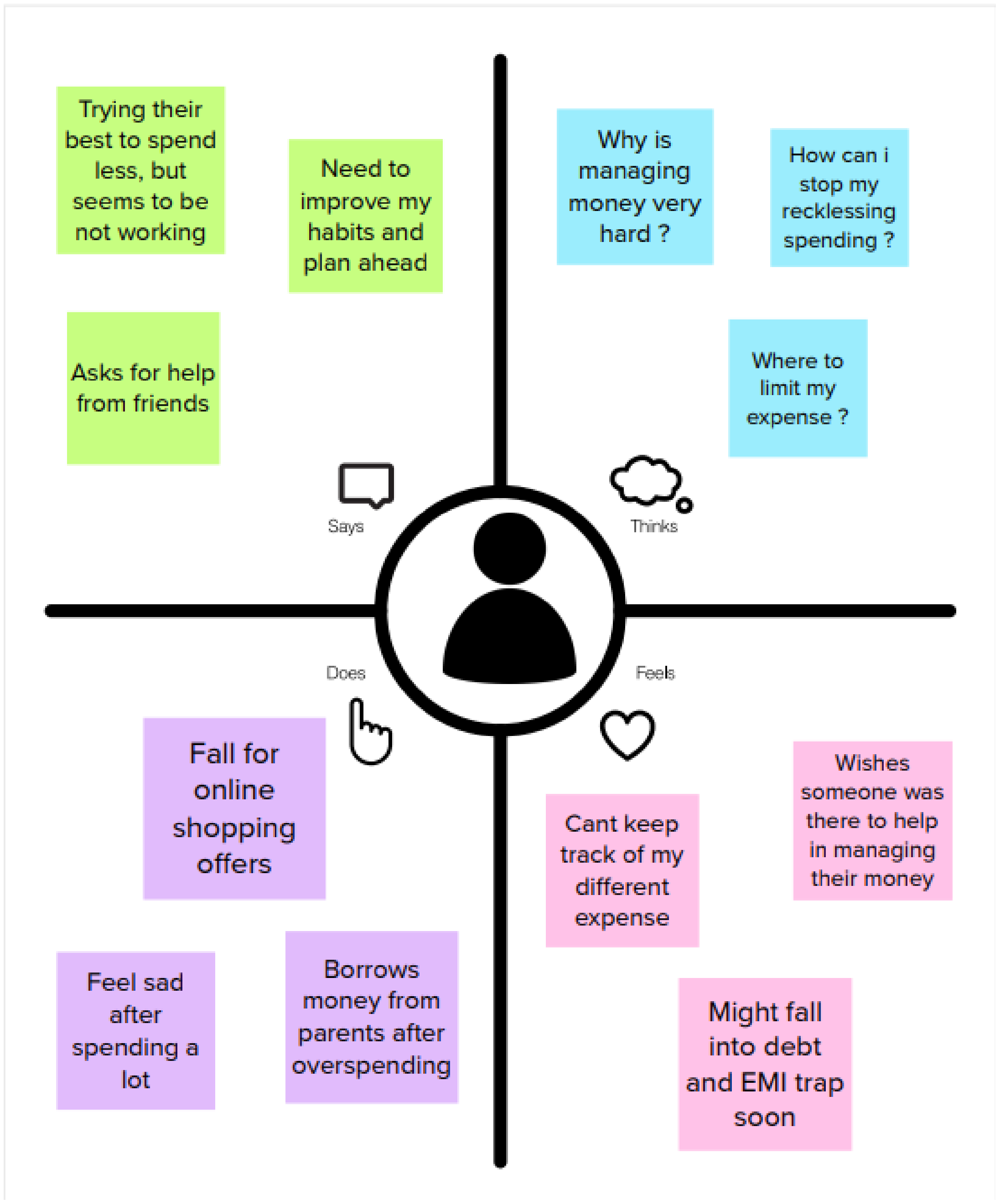
1. https://moneyview.in/insights/best-personal-finance-management-apps-in-india
2. https://www.factmr.com/report/personal-finance-mobile-app-market
3. https://www.moneytap.com/blog/best-money-management-apps/
4. https://relevant.software/blog/personal-finance-app-like-mint/
5. https://www.onmanorama.com/lifestyle/news/2022/01/11/financial-literacy-trend-among- todays-youth-investment.html
6. https://www.livemint.com/money/personal-finance/96-indian-parents-feel-their-children- lack-financial-know-how-survey-11661336110855.html
7. https://economictimes.indiatimes.com/small-biz/money/importance-of-financial-literacy- amongst-youngsters/articleshow/85655134.cms
8. https://www.news18.com/news/education-career/only-27-adults-16-7-of-indian- teenagers-financially-literate-4644893.html

3. Problem Statement Definition

Modern education does not focus on finance management. This is primarily due to lack of resources and the Indian value system on giving money to children. Failing to teach this valuable knowledge had left many Indians to recklessly spend their income and fall into vicious cycles of EMI and debt. Many of them are just a month’s salary away from bankruptcy. This issue is tackled by providing a web application for where people can plan their monthly expenses into categories, set alerts and get visual insights from their spending patterns.

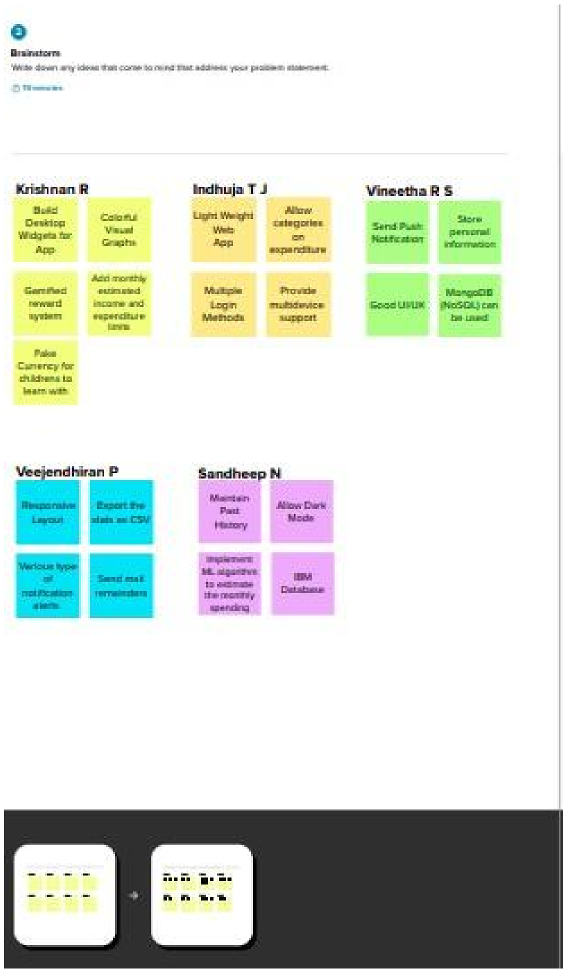
Ideation & Proposed Solution

1. Empathy Map Canvas


Trying their best to spend less, but seems to be not working
Need to improve my habits and plan ahead
Asks for help from friends
Why is managing money very hard ?
How can i stop my recklessing spending ?
Where to limit my expense ?
Says
Thinks
Does
Feels
Fall for online shopping offers
Feel sad after spending a lot
Borrows money from parents after overspending
Cant keep track of my different expense
Wishes someone was there to help in managing their money
Might fall into debt and EMI trap soon

2. Ideation & Brainstorming

The process of developing and expressing new ideas is referred to as "ideation" . It is a creative idea that aims to solve a problem or provide a more efficient way to carry out a task. It involves coming up with new ideas, improving on current ones, and working out how to put new ideas into practise. The most common method of ideation is brainstorming. By interacting with one another, listening to one another, and expanding on one another's ideas, brainstorming aims to maximise the group's collective thinking.



Brainstorm

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

10 minutes

Krishnan R

- Build Desktop Widgets for App
- Colorful Visual Graphs
- Gamified reward system
- Add monthly estimated income and expenditure limits
- Fake Currency for childrens to learn with

Indhuja T J

- Light Weight Web App
- Allow categories on expenditure
- Multiple Login Methods
- Provide multidevice support

Vineetha R S

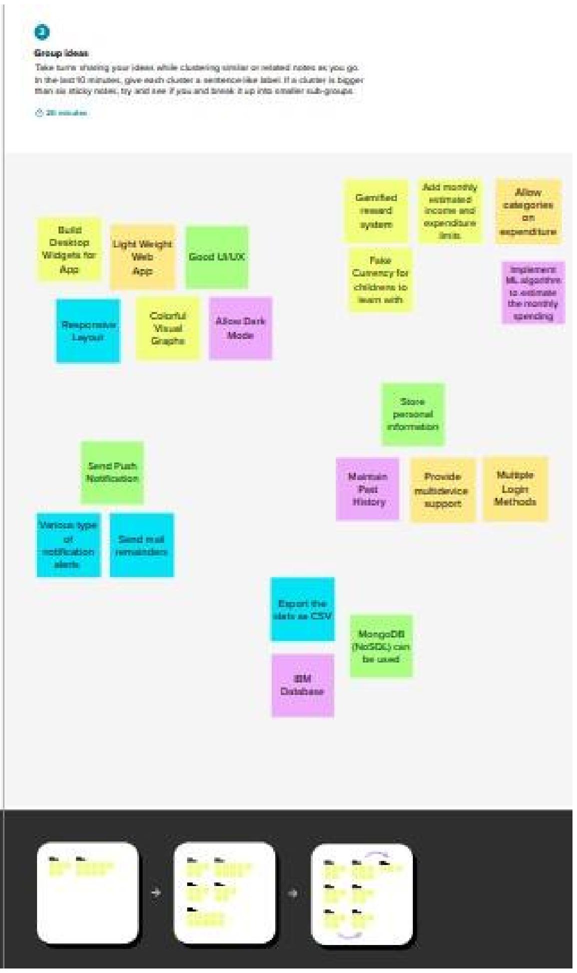
- Send Push Notification
- Store personal information
- Good UI/UX
- MongoDB (NoSQL) can be used

Veejendhiran P

- Responsive Layout
- Export the stats as CSV
- Various type of notification alerts
- Send mail remainders

Sandheep N

- Maintain Past History
- Allow Dark Mode
- Implement ML algorithm to estimate the monthly spending
- IBM Database



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

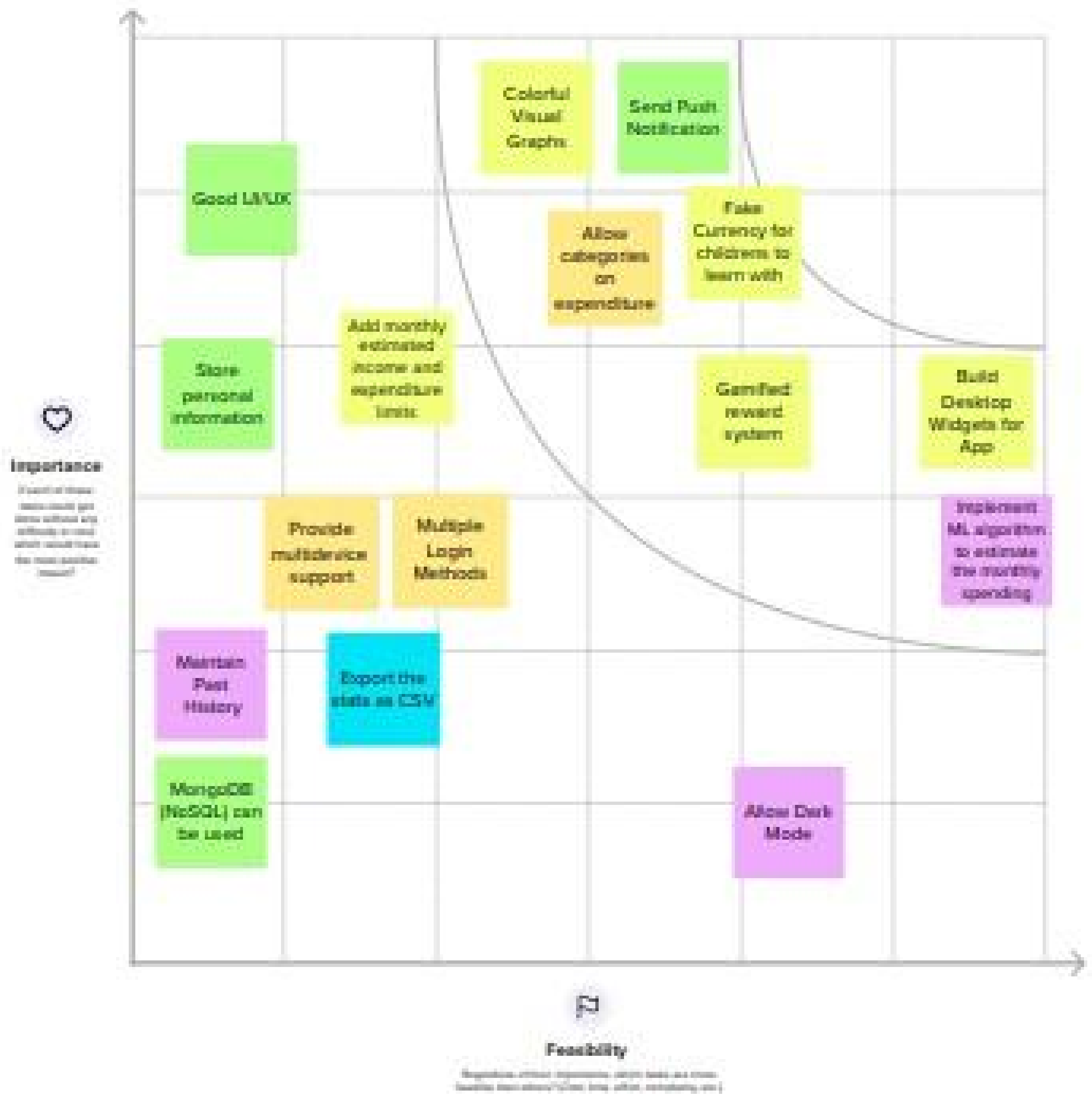
- Build Desktop Widgets for App
- Light Weight Web App
- Good UI/UX
- Responsive Layout
- Colorful Visual Graphs
- Allow Dark Mode
- Gamified reward system
- Add monthly estimated income and expenditure limits
- Allow categories on expenditure
- Fake Currency for childrens to learn with
- Implement ML algorithm to estimate the monthly spending
- Store personal information
- Send Push Notification
- Maintain Past History
- Provide multidevice support
- Multiple Login Methods
- Various type of notification alerts
- Send mail remainders
- Export the stats as CSV
- MongoDB (NoSQL) can be used
- IBM Database

4

Prioritize

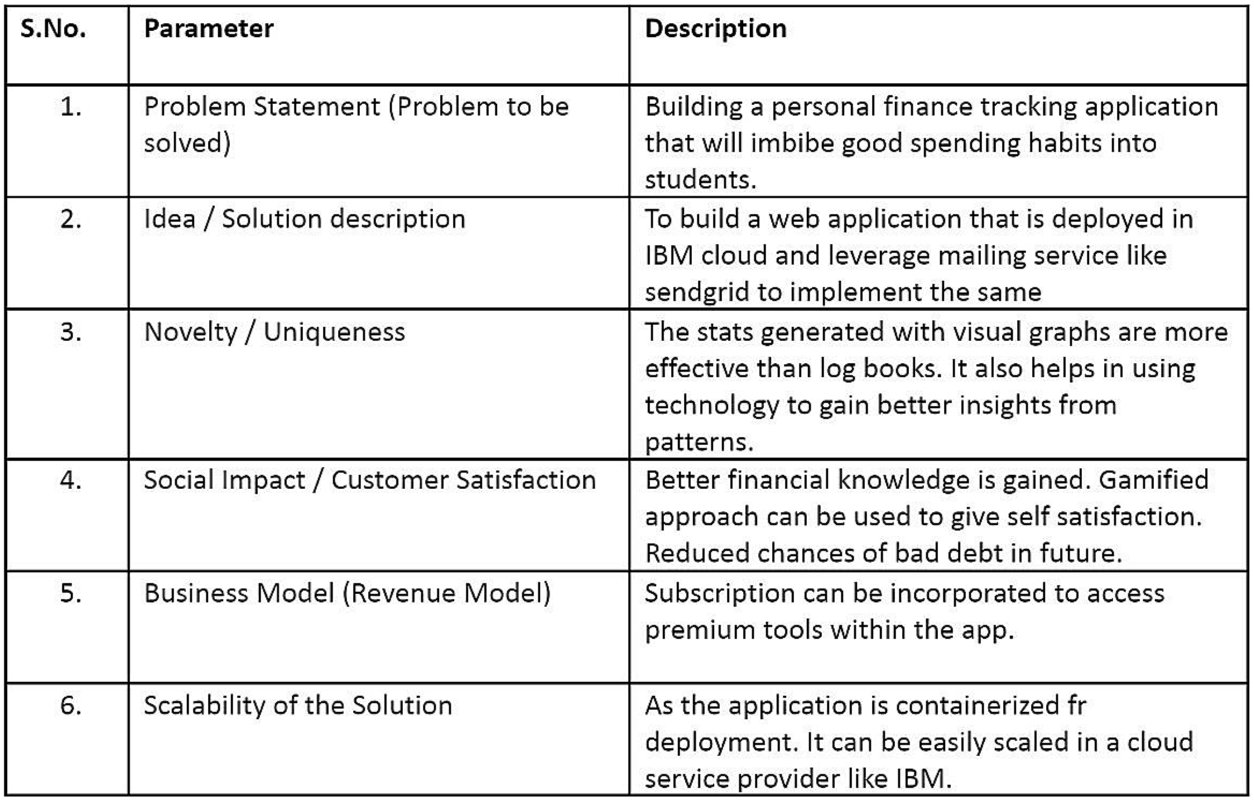
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Colorful Visual Graphs
Send Push Notification
Good UI/UX
Allow categories on expenditure
Fake Currency for childrens to learn with
Add monthly estimated income and expenditure limits
Store personal information
Gamified reward system
Build Desktop Widgets for App
Provide multidevice support
Multiple Login Methods
Implement ML algorithm to estimate the monthly spending
Maintain Past History
Export the stats as CSV
MongoDB (NoSQL) can be used
Allow Dark Mode
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

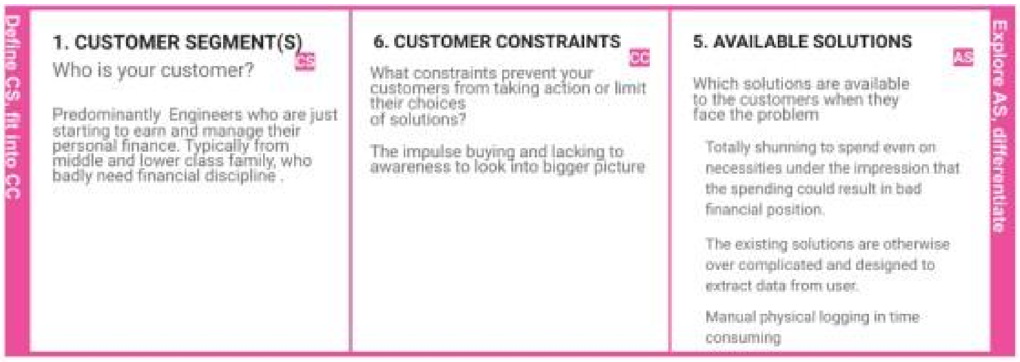
3. Proposed Solution



S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Building a personal finance tracking application that will imbibe good spending habits into students.
2.	Idea / Solution description	To build a web application that is deployed in IBM cloud and leverage mailing service like sendgrid to implement the same
3.	Novelty / Uniqueness	The stats generated with visual graphs are more effective than log books. It also helps in using technology to gain better insights from patterns.
4.	Social Impact / Customer Satisfaction	Better financial knowledge is gained. Gamified approach can be used to give self satisfaction. Reduced chances of bad debt in future.
5.	Business Model (Revenue Model)	Subscription can be incorporated to access premium tools within the app.
6.	Scalability of the Solution	As the application is containerized fr deployment. It can be easily scaled in a cloud service provider like IBM.

- The web application is accessed through use's email. The user can register and login.
- The expenses are added to the application by the user as and when they make payment or get income.
- The user can configure the expenditure limit to get email notification when the expenditure reaches beyond the trigger point.
- The user expense data is also displayed as charts to get better insights into their cash/financial status.

4. Problem Solution Fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

Predominantly Engineers who are just starting to earn and manage their personal finance. Typically from middle and lower class family, who badly need financial discipline .

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?

The impulse buying and lacking to awareness to look into bigger picture

5. AVAILABLE SOLUTIONS AS

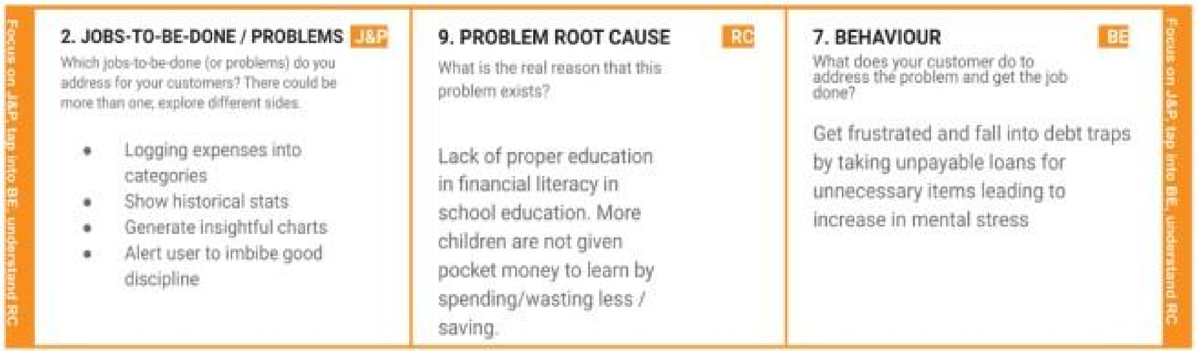
Which solutions are available to the customers when they face the problem

Totally shunning to spend even on necessities under the impression that the spending could result in bad financial position.

The existing solutions are otherwise over complicated and designed to extract data from user.

Manual physical logging in time consuming

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.

- Logging expenses into categories
- Show historical stats
- Generate insightful charts
- Alert user to imbibe good discipline

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?

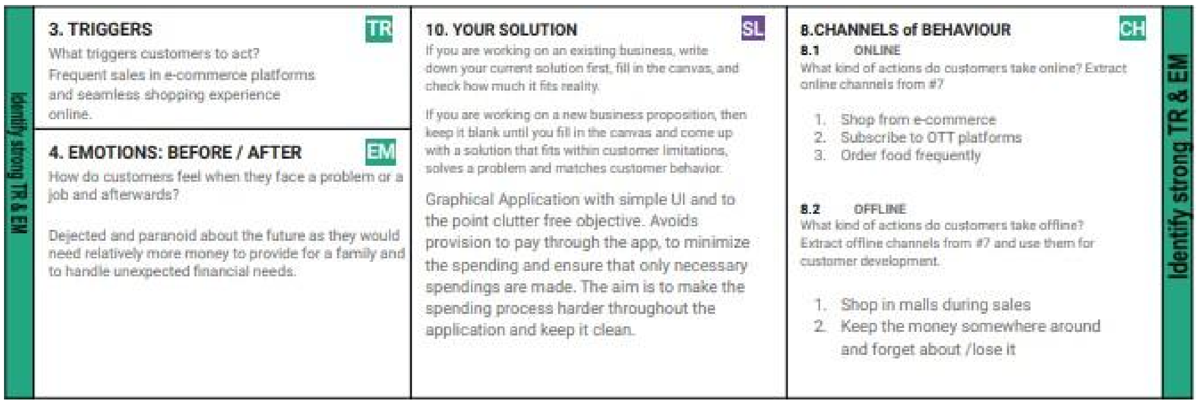
Lack of proper education in financial literacy in school education. More children are not given pocket money to learn by spending/wasting less / saving.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

Get frustrated and fall into debt traps by taking unpayable loans for unnecessary items leading to increase in mental stress

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



Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act?

Frequent sales in e-commerce platforms and seamless shopping experience online.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

Dejected and paranoid about the future as they would need relatively more money to provide for a family and to handle unexpected financial needs.

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 behavior.

Graphical Application with simple UI and to the point clutter free objective. Avoids provision to pay through the app, to minimize the spending and ensure that only necessary spendings are made. The aim is to make the spending process harder throughout the application and keep it clean.

8.CHANNELS of BEHAVIOUR CH

8.1 ONLINE

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

1. Shop from e-commerce
2. Subscribe to OTT platforms
3. Order food frequently

8.2 OFFLINE

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

1. Shop in malls during sales
2. Keep the money somewhere around and forget about /lose it

Identify strong TR & EM

When there is evidence that customers are interested in specific jobs, problems, and rewards, there is a problem-solution fit. At this step, you have determined that a problem exists and have developed a value proposition that considers the tasks, difficulties, and advantages of your clients. When such a solution is found and a company creates a strategy that, from various perspectives,

delivers a game-changer for clients, this is known as a problem-solution-fit. Businesses run the danger of discovering that no one wants their solution, which is regrettable given the time and money invested, if they fail to evaluate the Problem- Solution Fit they produced.

Requirement

Analysis Functional

requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Email/SignUp Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Add expenses	Enter the everyday expenses Split it into categories(example : food, petrol,movies)
FR-4	Reminder mail	Sending reminder mail on target (for ex : if user wants a reminder when his/her balance reaches some amount(5000)) Sending reminder mail to the user if he/she has not filled that day's expenses.
FR-5	Creating Graphs	Graphs showing everyday and weekly expenses. Categorical graphs on expenditure.
FR-6	Add salary	Users must enter the salary at the start of the month.
FR-7	Export CSV	User can export the raw data of their expenditure as CSV

Non-Functional requirements

Usability

Users must be able to access the system using a web application on a computer. The system's user interface is a web application. The system is simple because it is user-friendly.

Availability

The system is used around-the-clock, 365 days a year, and is completely accessible to the user. The system must function seven days a week, twenty-four hours a day.

Scalability

The ability of a system to adapt its performance and cost to changes in application and system processing demands is known as scalability. Security A security requirement is a declaration of essential security functionality that guarantees the fulfilment of one of numerous security properties of software.

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

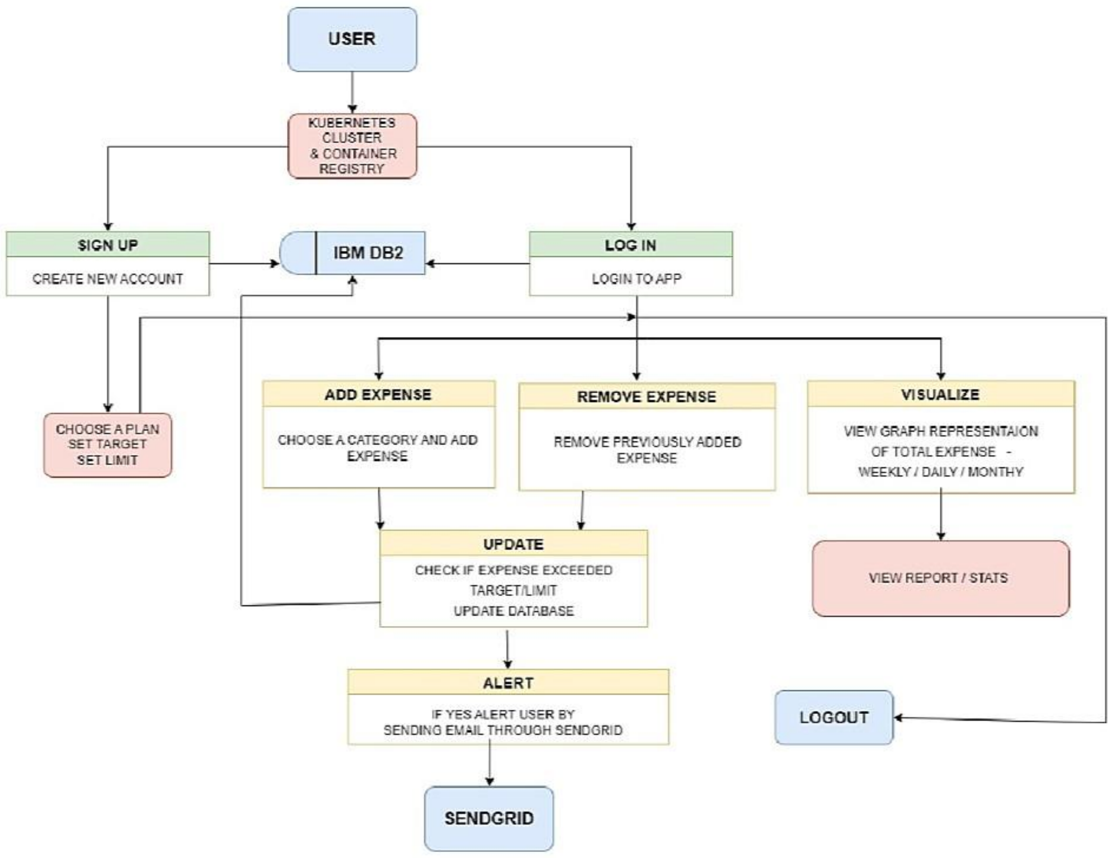
Reliability

Due to the value of data and the harm that inaccurate or incomplete data can do, the system must be completely reliable. It will function every day of the week. Every single day.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	A simple web application which is accessible across devices
NFR-2	**Security**	The OAuth Google sign in and email login are secure with hashed and salted secure storage of credentials.
NFR-3	**Reliability**	Containerized service ensures that new instance can kick up when there is a failure
NFR-4	**Performance**	The load is managed through the load balancer used with docker. Thus ensuring good performance
NFR-5	**Availability**	With load balancing and multiple container instances, the service is always available.
NFR-6	**Scalability**	Docker and Kubernetes are designed to accommodate scaling based on need

Project Design

1. Data Flow Diagrams


USER
KUBERNETES CLUSTER & CONTAINER REGISTRY
SIGN UP
CREATE NEW ACCOUNT
IBM DB2
LOG IN
LOGIN TO APP
CHOOSE A PLAN
SET TARGET
SET LIMIT
ADD EXPENSE
CHOOSE A CATEGORY AND ADD EXPENSE
REMOVE EXPENSE
REMOVE PREVIOUSLY ADDED EXPENSE
VISUALIZE
VIEW GRAPH REPRESENTAION OF TOTAL EXPENSE - WEEKLY / DAILY / MONTHY
UPDATE
CHECK IF EXPENSE EXCEEDED TARGET/LIMIT
UPDATE DATABASE
VIEW REPORT / STATS
ALERT
IF YES ALERT USER BY SENDING EMAIL THROUGH SENDGRID
LOGOUT
SENDGRID

2.Solution & Technical Architecture

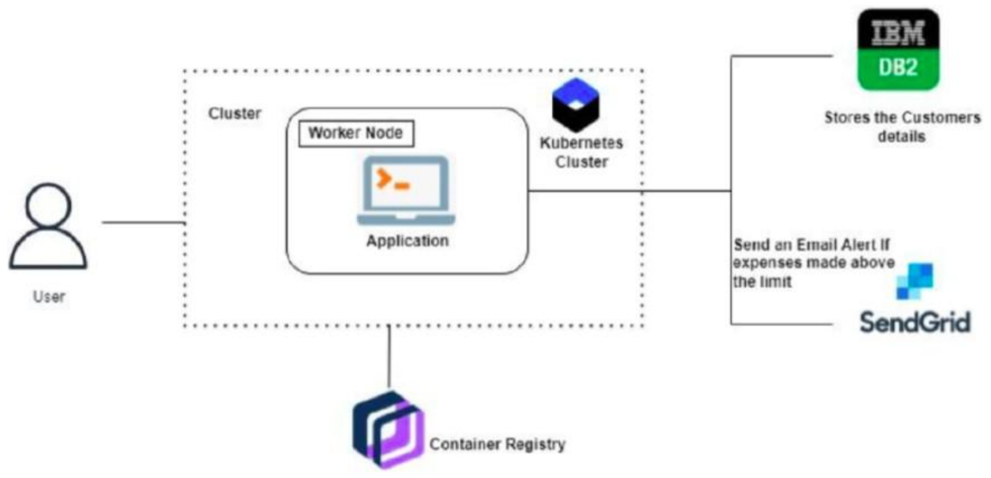


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript in Python Flask
2.	User Login	The user can login either through their gmail account or an account in the app server	Google Oauth for Google Signin. Hashed password in DB
3.	Graph Visualisation	Rendering plots and graphs based on the user spending data	Seaborn, Mathplotlib

4.	Database	Data Type, Configurations etc.	NoSQL database can be used as it promotes flexible structuring of data
5.	Cloud Database	Database Service on Cloud	IBM DB2 is used to store the user details and the data entries

6.	SendGrid	a cloud-based SMTP provider that allows you to send email without having to maintain email servers	SendGrid is used to trigger mail to user emails when a particular condition is met
7.	Google OAuth	OAuth 2.0 allows users to share specific data with an application while keeping their usernames, passwords, and other information private.	Enables login through gmail account, thus making the application accessible
8.	Cloud Deployment	Application Deployment onCloud Server	Docker and Kubernetes is used for deployment as it promises scalability and high availability

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology

1.	Open-Source Frameworks	Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.	Python Flask Framework
2.	Security Implementations	Passwords cant be stored as plaintext so it is hashed and salted	BCrypt
3.	Scalable Architecture	Containerized application is deployed to rapidly increase scale on demand	Docker
4.	Availability, Performan ce	Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Availability and Performance enhances user experience	Kubernetes

5.4 User Stories

User Type	**Functional Requirement**	**User Story**	**User Story / Task**

	(Epic)	**Number**	
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
	Login	USN-2	As a user, I can log into the application by entering email & password
	Add	USN -3	As a user , I can add in new expenses.
	Remove	USN – 4	As a user , I can remove previously added expenses.
	View	USN - 5	As a user , I can view my expenses in the form of graphs and get insights.
	Get alert message	USN - 6	As a user , I will get alert messages if I exceed my target amount.
Administrator	Add / remove user	USN – 7	As admin , I can add or remove user details on db2 manually.
		USN - 8	As admin , I can add or remove user details on sendgrid.

Project Planning & Scheduling

Sprint 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 , new password and confirming the same password.	10	High	Sriram Sriram Vignesh

	Login	USN -2	As a user , I can log into the application by entering email and password.	7	High	Vishal
	Dashboard	USN -3	Logging in takes the user to their dashboard.	3	Low	
Sprint - 2		USN -4	As a user ,I will update my salary at the start of each month.	4	Medium	Sriram

Sprint - 3		USN -5	As a user , I will set a target/limit to keep track of my expenditur e.	4	Medium	Vishal

Workspace	USN -6	Workplace for personal expense tracking	2	Medium	Sriram
Charts	USN -7	Graphs to show weekly and everyday expenditure	6	High	Vignesh
Database	USN - 8	Working with database	4	High	Vignesh
IBM DB2	USN -9	Linking database with dashboard	6	High	Sriram
	USN -10	Making dashboard interactive with JS	6	High	Vishal

Sprint - 4	Watson Assistant	USN -11	Embedding Chatbot to clarify user's queries.	2	Low	Vignesh

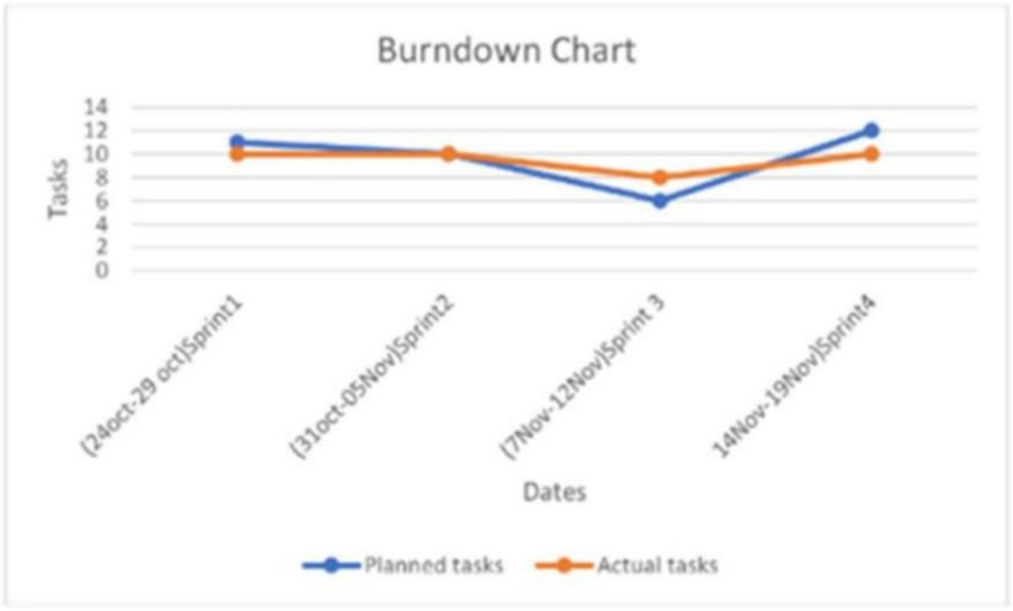
BCrypt	USN -12	Using BCrypt to store passwords securely.	2	Medium	Sriram
SendGrid	USN -13	Using SendGrid to send mail to the user. (To alert or remind)	4	Medium	Vignesh
Integration	USN -14	Integrating frontend and backend.	6	High	Sriram
Docker	USN -15	Creating Docker image of web app.	3	High	Sriram
Cloud Registry	USN -16	Uploading docker image to IBM cloud registry.	3	High	Vignesh

	Kubernetes	USN -17	Creating container using docker and hosting the webapp.	5	High	Vishal
	Exposing Deployment	USN -18	Exposing IP/Ports for the site.	3	Medium	vignesh

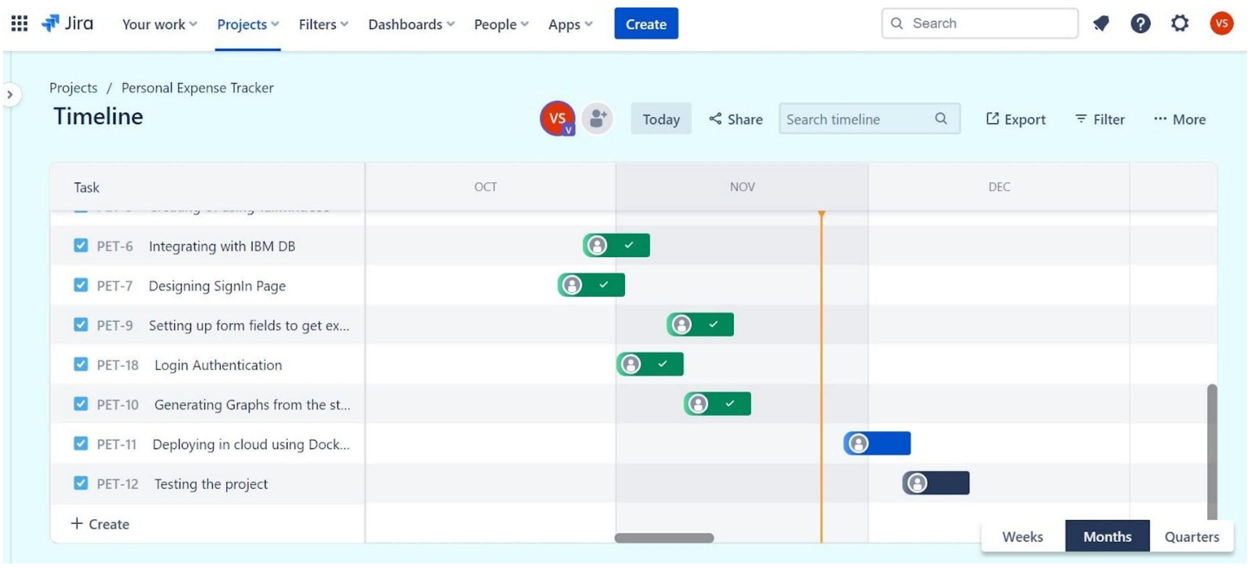
Sprint Delivery Schedule

Sprint	Total Story Points	Sprint Start Date	Sprint End Date	Sprint Release Date	Sprint Duration
Sprint 1	20	24 Oct 2022	31 Oct 2022	29 Oct 2022	6 days
Sprint 2	20	31 Oct 2022	05 Nov 2022	6 Nov 2022	6 days
Sprint 3	20	07	12 Nov	14 Nov 2022	6 days

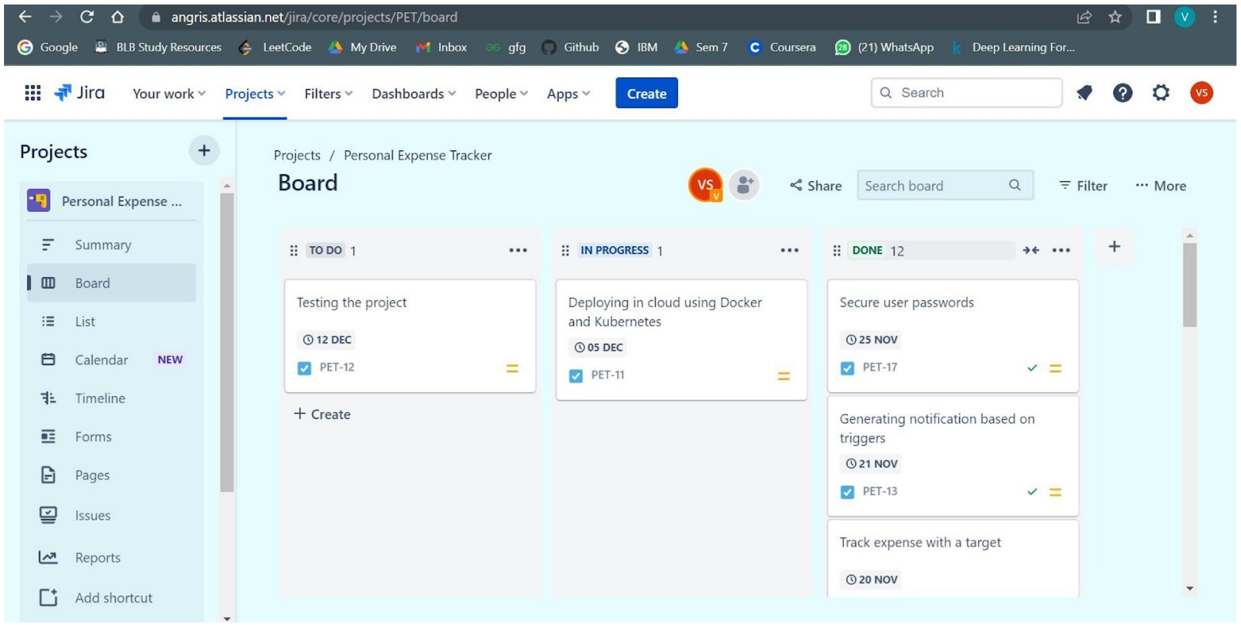
		Nov2022	2022		
Sprint 4	20	13 Nov 2022	19 Nov 2022	21 Nov 2022	6 days



Reports From Jira



PET-6 Integrating with IBM DB
PET-7 Designing SignIn Page
PET-9 Setting up form fields to get ex...
PET-18 Login Authentication
PET-10 Generating Graphs from the st...
PET-11 Deploying in cloud using Dock...
PET-12 Testing the project
+ Create



Coding & Solutioning

Feature 1 : Sign-up and Login

TailwindCSS is used to build the form. It ensures that the design is consistent and minimalistic. The no distraction UI also ensures that the user is not distracted while using the application.

A post request is made to the IBM DB to store and retrieve the credentials.

The API keys for the IBM DB2 is stored as .env variables in the python application.

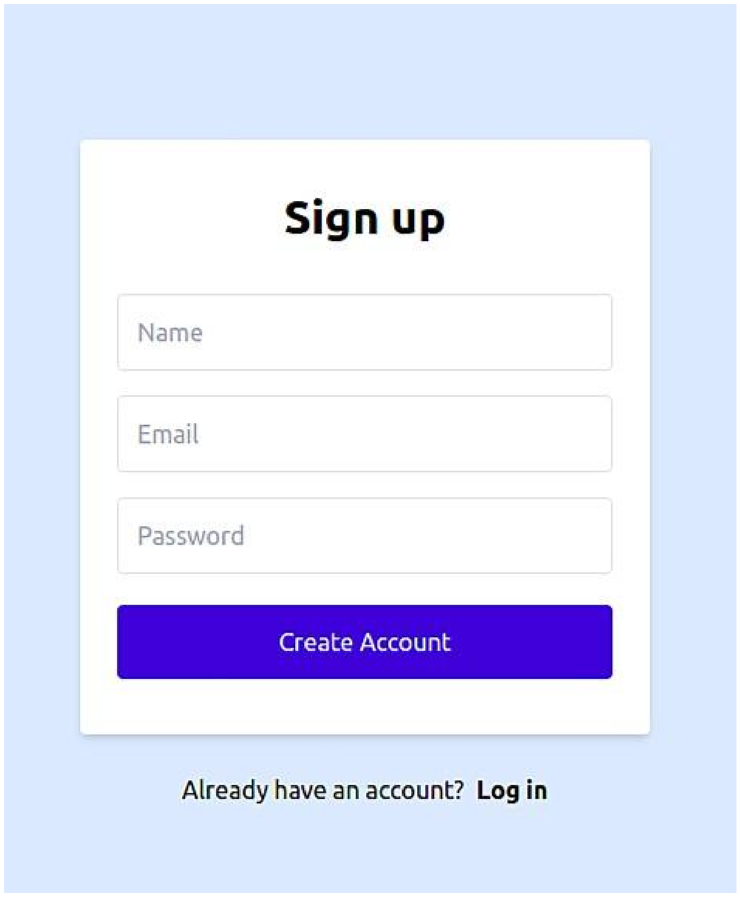
UI Code

```
    <div class="bg-gray-lighter min-h-screen flex flex-col">
        <div class="container max-w-sm mx-auto flex-1 flex flex-
col items- center justify-center px-2">
            {% if error %}
```

```
              <div class="bg-red-100 border-red-200 border-2
   rounded-md p-2 mt-2 mb-2 w-full text-center "
   >{{error}}</div>
              {% endif %}
              <form class="bg-white px-6 py-8 rounded shadow-
   md text-black w-full" method="POST">
                  <h1 class="mb-8 text-3xl text-center font-bold">Sign
   up</h1>



   rounded mb-4"






   rounded mb-4"
<input
    type="text"
    class="block border border-gray-light w-full p-3

    name="name"
    placeholder="Name" />

<input
    type="text"
    class="block border border-gray-light w-full p-3
                        name="ema
```

```
                    il"
                    placehold
                    er="Emai
                    l" />




   rounded mb-4"
<input
    type="password"
    class="block border border-gray-light w-full p-3

    name="password"
    placeholder="Password" />



                <button
                    type="submit"
                    class="w-full text-center py-3
   rounded bg-blue-700 text-white hover:bg-blue-800
   focus:outline-none my-1"
                >Create Account</button>

            </form>

        <div class="text-gray-dark
            mt-6 flex"> Already have
            an account?
```

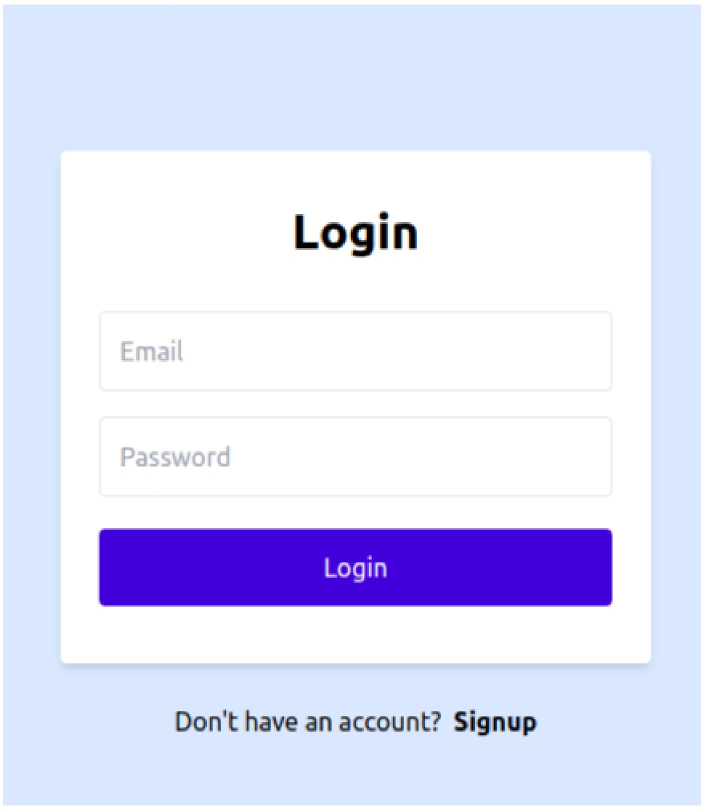
```
                    <a class="ml-2 no-underline border-b border-blue
                    text-
        blue" href="/">
        <p class="font-bold">Log in</p>
    </a>
                  </div>
              </div>
          </div>
```



UI Code

```
      <div class="bg-gray-lighter min-h-screen flex flex-col">
        <div class="container max-w-sm mx-auto flex-1 flex flex-
col items- center justify-center px-2">
            {% if error %}
            <div class="bg-red-100 border-red-200 border-2
rounded-md p-2 mt-2 mb-2 w-full text-center "
>{{error}}</div>
            {% endif %}
            <form class="bg-white px-6 py-8 rounded shadow-
md text-black w-full" method="POST" >
                <h1 class="mb-8 text-3xl text-center font-
                bold">Login</h1>




rounded mb-4"








rounded mb-4"
<input
   type="text"
   class="block border border-gray-light w-full p-3

   name="email"
   placeholder="Email" />
```

```
<input
    type="password"
    class="block border border-gray-light w-full p-3

    name="password"
    placeholder="Password" />



                   <button
                       type="submit"
                       class="w-full text-center py-3
   rounded bg-blue-700 text-white hover:bg-blue-800
   focus:outline-none my-1"
                   >Login</button>

               </form>

           <div class="text-gray-dark
               mt-6 flex"> Don't have an
               account?
                   <a class="ml-2 no-underline border-b border-
   blue text- blue" href="/signup">
                       <p class="font-bold">Signup</p>
                   </a>
               </div>
           </div>
       </div>
```



Flask server code to serve the login/signup pages along with the form handling code to interact withcollect the valus and pass to DB

Server Code

```
def check_credentials(e, p):
if utils.getPassword(
    e) == p:
    session['logged_in
    '] = True
    session['email'] =
    e print("Valid
          User")
              return redirect(url_for('dashboard'))
          return render_template('login.html', error="Invalid
          Credentials")



  def
      register(
      u, p, e):
      print(u,
      p, e)
                                                                           try:
r = utils.addUser(u, e, p)
if (r == "Username Exists"):
    return render_template('signup.html', error="Username Exists")
return render_template('login.html')
          except:
             return render_template('signup.html', error="Error in
                                    inserting
      user")




      @app.route('/',
      methods=['GET', 'POST'])
      def login():
      if request.method ==
          'POST':
          print("Checking
          Credentials")
```

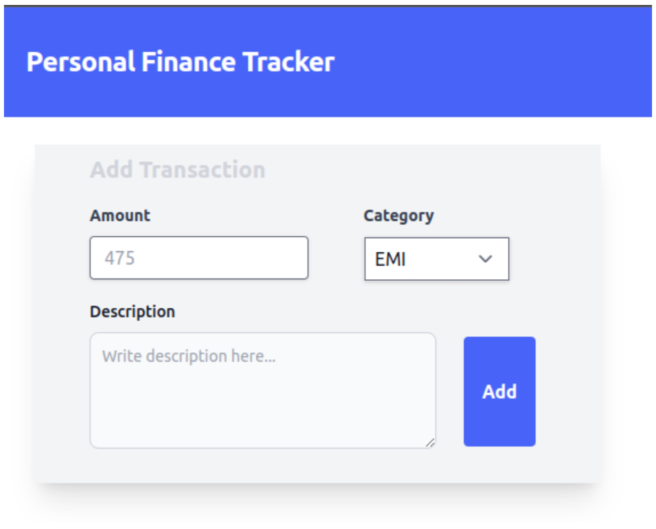
```
        return
check_credentials(request.form['email'],
request.form['password'])
    else:
        if session.get('logged_in'):
            return
        redirect(url_for('dashboard'
        )) return
        render_template('login.html
        ')
```

Feature 2 : Add expense records

The user can add their personal epense throught the provided form. The data is added in the database against their email account. The amount, description and category (from predefined list) is set and added as entry.

Available Categories

```
positive_money = ['Salary Credited', 'Festival Bonus']
negative_money = ['EMI', 'Food', 'Transportation',
'Groceries','Clothing', 'Electronic', 'Entertainment', 'Rent',
'Vacations']
```



UI Code

```
<form method="POST" class="bg-gray-100 shadow-xl rounded-xg
w-[31rem] px- 12 pt-2 pb-4 m-8 mt-6">
    <span class="text-xl text-gray-300 font-bold">Add
    Transaction</span>
    <div class="flex mt-4">
      <input type="hidden" id="t_type"
name="t_type" value="add_transaction"/>
      <div class="mb-4">
        <label class="block text-gray-700 text-sm font-bold
mb-2" for="amount">
          Amount
        </label>
        <input required class="shadow appearance-none border
rounded w-48 py-2 px-3 text-gray-900 leading-tight
focus:outline-none focus:shadow- outline" name="amount"
id="amount" type="number" placeholder="475">
      </div>
      <div class="mb-4 ml-12">
        <label class="block text-gray-700 text-sm font-bold
mb-2" for="amount">
          Category
        </label>
        <div class="form-group bg-white shadow appearance-none
        border
rounded h-10 w-50 text-gray-900 leading-tight focus:outline-none">
          <select required class="bg-white focus:ring-0 px-2
py-2" name="category" id="category">
            <option value="Salary Credited">Salary</option>
            <option value="Festival Bonus">Festival Bonus</option>
            <option selected="selected" value="EMI">EMI</option>
            <option value="Food">Food</option>
            <option value="Transportation">Transportation</option>
            <option value="Groceries">Groceries</option>
            <option value="Clothing">Clothing</option>
            <option value="Electronic">Electronic</option>
            <option value="Entertainment">Entertainment</option>
            <option value="Rent">Rent</option>
            <option value="Vacations">Vacations</option>
          </select>
        </div>
      </div>


    </div>
    <div class="flex ">
      <div>
        <label for="description" class="block mb-2 text-
sm font-medium text-gray-900 dark:text-
```

```
white">Description</label>
        <textarea id="description" required
name="description" rows="4" class="block p-2.5 w-[19rem]
text-sm text-gray-900 bg-gray-50 rounded-lg border border-
gray-300 focus:ring-blue-500 focus:border-blue-500 dark:bg-
gray-700 dark:border-gray-600 dark:placeholder-gray-400
dark:text-white dark:focus:ring-blue-500 dark:focus:border-
blue-500" placeholder="Write description
here..."></textarea>
      </div>

      <div>
        <div>
        <button type="submit" class="bg-blue-500 hover:bg-
blue-800 text- white font-bold py-2 px-4 mx-6 my-4 mt-8 h-
24 rounded inline-flex items- center">
          <span>Add</span>
        </button>
        </div>
      </div>
    </div>
</form>
```

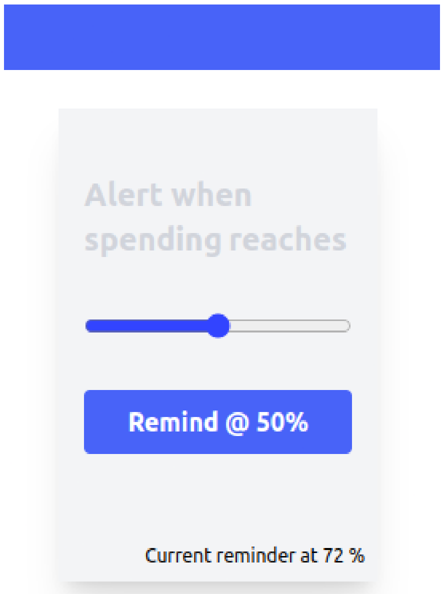
Server Code

```
@app.route('/dashboard',
methods=['GET', 'POST']) def
dashboard():
    if request.method == 'POST':
        if not session['logged_in']:
            return
        render_template('login.html
        ') email = session['email']
        print(request.form)
        # if True:
    if request.form['t_type'] ==
        'add_transaction': now =
        datetime.now()
            dt_string =
            now.strftime("%Y/%m/%d
            %H:%M") date = dt_string
            print(
            date)
            add_fin
            ance_r
            ecord(
                email, request.form['category'],
request.form['amount'], request.form['description'], date)
    elif request.form['t_type'] ==
        'set_trigger': limit =
        int(request.form['trigger'])
        utils.setReminder(email, limit)
        else:
            print("Lol error bro")
        return
    redirect(url_for('dashboard'
    )) else:
```

```
    if
        session.get('lo
        gged_in'):
        email =
        session['email
        ']
            rows = utils.fetchFinanceRecord(email)
            spending =
            utils.getIncomeExpend(emai
            l) limit =
            utils.getReminder(email)
            graph =
            utils.getGraphDetails(e
            mail) # limit = "100"
            percent = 0
            if spending["income"] != 0:
                percent =
                (spending['expend']*100)/spending['income']

            percent =
            min(100,
            percent) l =
            len(rows)
            left = "Rs "+str(spending['expend']) + \
                " spent out of Rs "+str(spending['income'])
            return render_template('dashboard.html',
rows=rows, len=l, left=left, percent=str(percent)+"%",
limit=limit, graph=graph)
        return render_template('login.html')
```

Feature 3 : Set trigger limit

The user can set limit to configure when they need to be reminded for. When the expense crosses that threshhold. The user will recieve a mail, warning them.



UI Code

```
<form method="POST" class="bg-gray-100 shadow-xl rounded-
xg relative w- [12.5rem] px-4 pt-2 pb-4 m-8 mt-6">
    <div class="mt-8">
        <input type="hidden" id="t_type"
name="t_type" value="set_trigger"/>
        <span class="text-xl text-gray-300 font-bold">Alert
        when</span>
        <span class="text-xl text-gray-300 font-
        bold">spending</span>
        <span class="text-xl text-gray-300 font-bold">reaches</span>
        <div class="flex mt-4">
            <input name="trigger" id="trigger"
type="range" min="0" max="100" value="50" class="w-
full mt-4" />
        </div>
        <button type="submit" id="remind-button" class="bg-
blue-500 hover:bg-blue-800 text-white font-bold py-2 px-4
my-4 mt-8 w-full rounded ">
            <span>Remind @ 10%</span>
        </button>
    </div>
    <span class="text-xs absolute bottom-0 right-0 mb-
2 mr-2 float- right">Current reminder at {{limit}}
%</span>
</form>

<script>
    const selectElement =
    document.querySelector('#trigger'); const
    result = document.querySelector('#remind-
    button'); let x= selectElement.value;
    result.textContent = "Remind @ "+x+"%";
    selectElement.addEventListener('change',
    (event) => {
        x= selectElement.value;
        result.textContent =
        "Remind @ "+x+"%";
    });
```

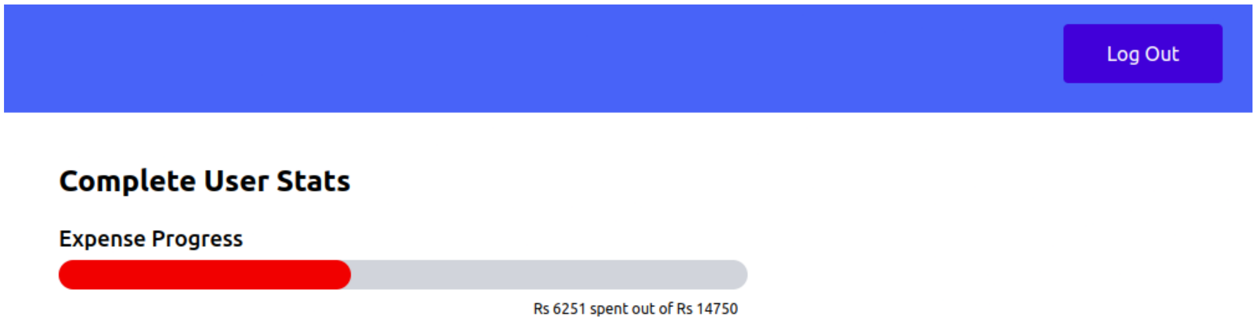
```
</script>
```

Server Code

```
@app.route('/dashboard',

methods=['GET', 'POST']) def

dashboard():

    if request.method == 'POST':

        if not session['logged_in']:

            return

        render_template('login.html

        ') email = session['email']

        print(request.form)

        # if True:

    if request.form['t_type'] ==

        'add_transaction': now =

        datetime.now()

            dt_string =

            now.strftime("%Y/%m/%d

            %H:%M") date = dt_string

            print(

            date)

            add_fin

            ance_r

            ecord(

                email, request.form['category'],

request.form['amount'], request.form['description'], date)

    elif request.form['t_type'] ==

        'set_trigger': limit =
        int(request.form['trigger'])
        utils.setReminder(email, limit)
        else:
            print("Lol error bro")
        return redirect(url_for('dashboard'))
```

Feature 4 : Progress Bar

A progress bar on the dashboard shows the total amount left and amount spent but the user. This gives a better indication for the user about their condition.



Rs 6251 spent out of Rs 14750

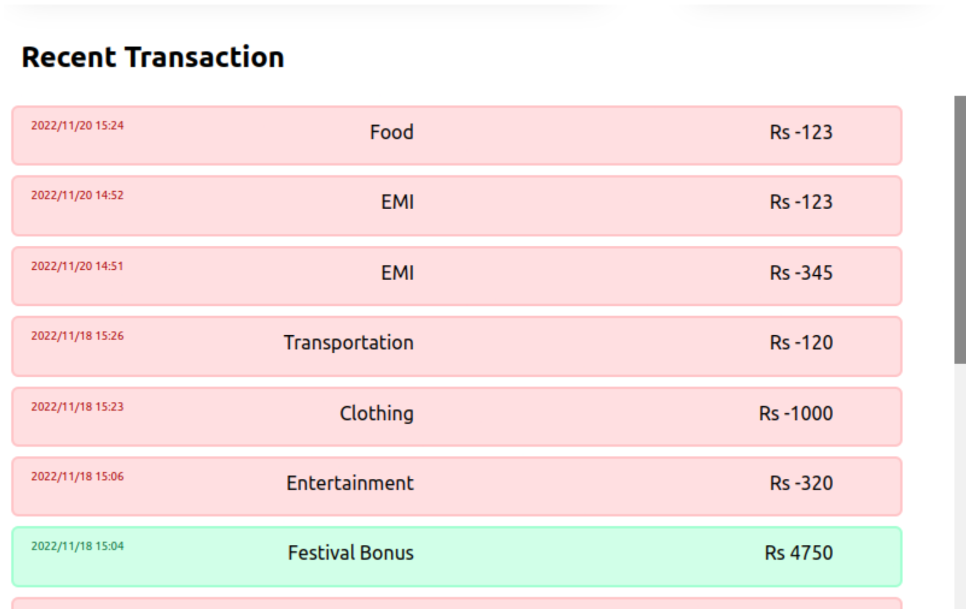
UI Code

```
  <div class="w-[30vw] mt-4">
    <div>
      <div class="mb-1 text-lg font-medium dark:text-
white">Expense Progress</div>
      <div class="w-full h-6 bg-gray-300 rounded-full dark:bg-gray-
      700">
        <div class="h-6 bg-red-600 rounded-full dark:bg-red-
500" style="width: {{percent}}">
        </div>
        <span class="mb-1 text-xs float-right pr-2 pt-2">{{left}}
        </span>
      </div>
    </div>
 </div>
```

The data for this progress bar is sent by the /dashboard component of the server.

Feature 6 : Spending List

The expenditure is fetched from the server and displayed as a list. Each row of the list shows the date of expense, category and the amount involved. Based on the type of transaction(credit/debit) , the rows is show in different colour for better visual cue.



Recent Transaction

2022/11/20 15:24	Food	Rs -123
2022/11/20 14:52	EMI	Rs -123
2022/11/20 14:51	EMI	Rs -345
2022/11/18 15:26	Transportation	Rs -120
2022/11/18 15:23	Clothing	Rs -1000
2022/11/18 15:06	Entertainment	Rs -320
2022/11/18 15:04	Festival Bonus	Rs 4750

UI Code

```
<div>
```

```
    <div class="font-bold text-2xl px-8 py-2 m-8 mt-2
mb-2 pt-2 pl- 2">Recent Transaction</div>
    <div style="overflow-y: scroll; height:45vh">
    {%for i in range(0, len)%}
        {% if rows[i]['AMOUNT']>0 %}
          <div class="bg-green-100 flex w-[40vw] text-right border-
          green-
200 border-2 rounded-md px-8 py-2 m-8 mt-2 mb-2 pt-2 pl-2" >
          <span class="w-[12%] mb-4 text-[0.6rem] text-green-800 mr-
8">{{rows[i]['DATE'].strip()}}</span>
        {% else %}
          <div class="bg-red-100 flex w-[40vw] text-right
border-red-200 border-2 rounded-md px-8 py-2 m-8 mt-2 mb-2
pt-2 pl-2" >
          <span class="w-[12%] mb-4 text-[0.6rem] text-red-800 mr-
8">{{rows[i]['DATE'].strip()}}</span>
        {% endif %}
          <span class="w-
          [30%]">{{rows[i]['CATEGORY'].strip()}}</span>
          <span class="w-[30%]"></span>
          <span class="w-[20%] float-right">Rs
{{rows[i]['AMOUNT']}}</span>
        </div>
        {%endfor%}
        <div class="bg-red-100 flex w-[40vw] text-right
border-red-200 border-2 rounded-md px-8 py-2 m-8 mt-2 mb-
2 pt-2 pl-2 invisible"
>Placeholder</div>
    </div>
  </div>
```

Feature 7 : Graph Visualiser

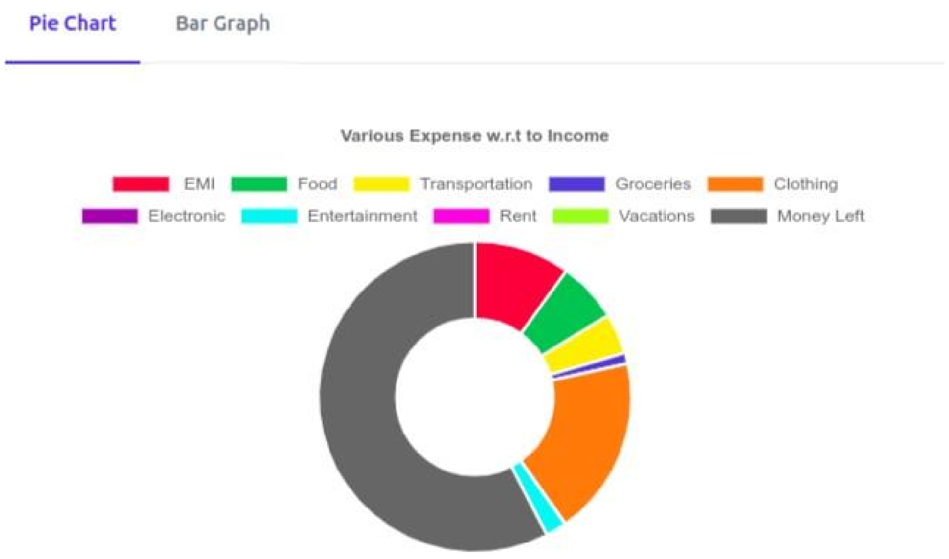
To chartJS is used to render colourful visualisations from the expenses made by the user. A pie chart and bar graph is generated for the existing categories of the income. On hovering, the chart element shows the individual spending of the same.

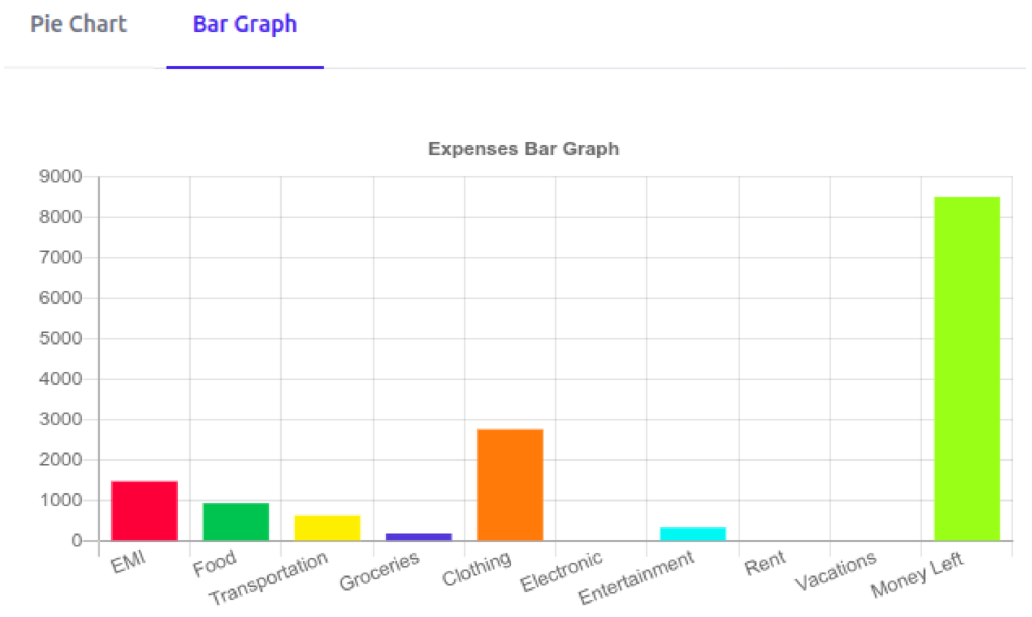
UI Code

```
<canvas id="barGraph" class="w-full"></canvas>
<div class="hidden" id="x-val"> {{graph['x']}} </div>
<div class="hidden" id="y-val"> {{graph['y']}} </div>
<script>
    x=document.querySelector("#x-
    val").innerHTML;
    y=document.querySelector("#y-
    val").innerHTML; x=x.split(",")
    y
    =
    y
    .
    s
    p
    l
    i
    t
    (
    "
    ,
    "
    )
    x
    V
    a
    l
    u
    e
    s
    =
    x
    y
    V
    a
    l
    u
    e
    s
    =
    y
    var barColors = ['#e6194b', '#3cb44b', '#ffe119',
'#4363d8', '#f58231', '#911eb4', '#46f0f0', '#f032e6',
'#bcf60c', '#00000'];

    new
    Chart("bar
    Graph", {
    type:
    "bar",
    data: {
        lab
        el
        s:
        xVa
        lue
        s,
        dat
        ase
        ts:
        [{
        backgroundColor: barColors,
        data: yValues
        }]
    },
    options: {
        legend:
        {display:
        false},
        title: {
            display: true,
            text: "Expenses Bar Graph"
        }
    }
    });
</script>



<canvas id="pieGraph" ></canvas>
<div class="hidden" id="x-val"> {{graph['x']}} </div>
<div class="hidden" id="y-val"> {{graph['y']}} </div>
<script>
    x=document.querySelector("#x-
    val").innerHTML;
    y=document.querySelector("#y-
    val").innerHTML;

    x=x.split(",")

    y=y.split(",")
```

```
          xValues=x

          yValues=y

          var barColors = ['#e6194b', '#3cb44b', '#ffe119',

        '#4363d8', '#f58231', '#911eb4', '#46f0f0', '#f032e6',

        '#bcf60c', '#00000'];


        new

          Chart("pie

          Graph", {

          type:

          "doughnut

          ", data: {

labels: xValues,

datasets: [{

                backgroundColo

                r: barColors,

                data: yValues

              }]

            },

            options: {

            titl

              e:

              {

            displ

              ay:

              tru

              e,

                text: "Various Expense w.r.t to Income"
```

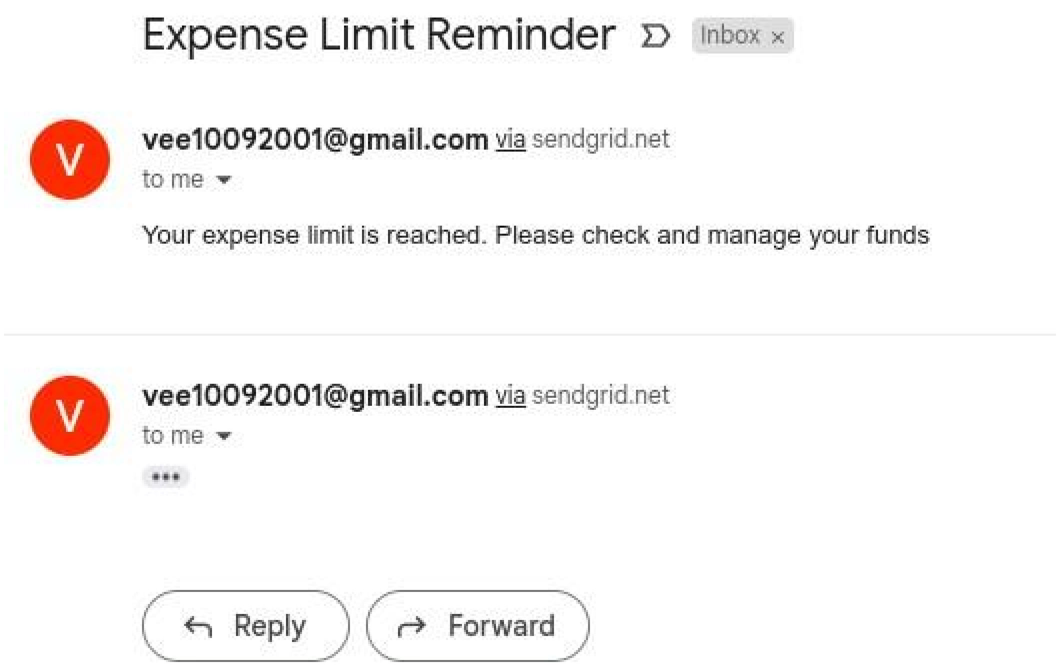
```
                    }
  }
  }
);
        </script>
```


Pie Chart
Bar Graph
Various Expense w.r.t to Income
EMI
Food
Transportation
Groceries
Clothing
Electronic
Entertainment
Rent
Vacations
Money Left


Pie Chart
Bar Graph
Expenses Bar Graph
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
EMI
Food
Transportation
Groceries
Clothing
Electronic
Entertainment
Rent
Vacations
Money Left

The data for this graph component is sent by the /dashboard component of the server.

Feature 8 : Email Notification


Expense Limit Reminder

vee10092001@gmail.com via sendgrid.net
to me

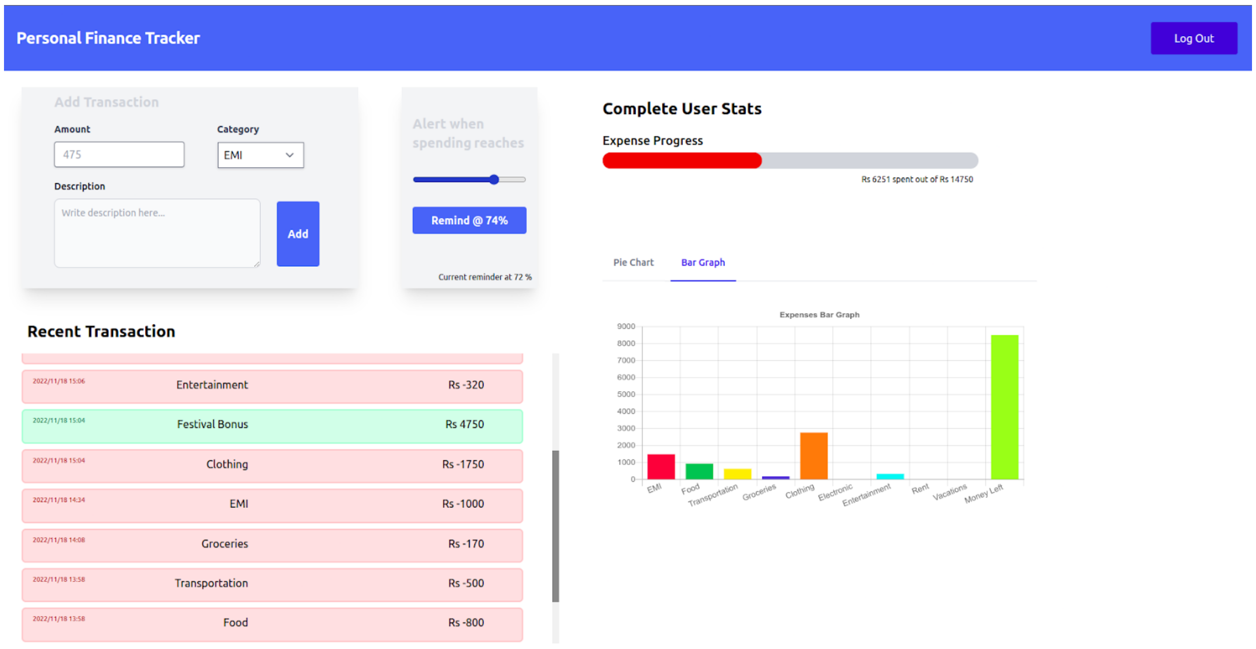
Your expense limit is reached. Please check and manage your funds

vee10092001@gmail.com via sendgrid.net
to me

Server Code

```
def triggerMail(email):
    sg =
    sendgrid.SendGridAPIClient(sendG
    ripdAPI) # Change to your
    verified sender
    from_email = Email(sendgridSender)
    to_email = To(email) # Change to your
```

```
        recipient subject = "Expense Limit
        Reminder"
        content = Content(
            "text/plain", "Your expense limit is reached.
    Please check and manage your funds")
        mail = Mail(from_email, to_email, subject, content)
        # Get a JSON-ready representation of the
        Mail object mail_json = mail.get()
        # Send an HTTP POST request to /mail/send
        response =
        sg.client.mail.send.post(request_body=mail_jso
        n) # print("mail triggered with send grid")
```



Personal Finance Tracker

Log Out

Add Transaction

Amount: 475

Category: EMI

Description: Write description here...

Add

Alert when spending reaches

Remind @ 74%

Current reminder at 72 %

Recent Transaction

2022/11/18 15:06	Entertainment	Rs -320
2022/11/18 15:04	Festival Bonus	Rs 4750
2022/11/18 15:04	Clothing	Rs -1750
2022/11/18 14:34	EMI	Rs -1000
2022/11/18 14:08	Groceries	Rs -170
2022/11/18 13:58	Transportation	Rs -500
2022/11/18 13:58	Food	Rs -800

Complete User Stats

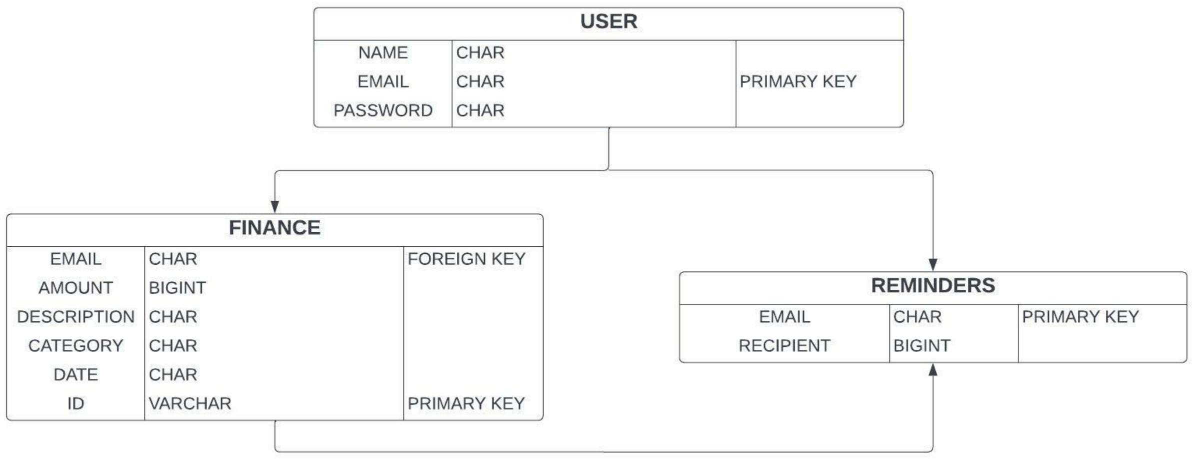
Expense Progress

Rs 6251 spent out of Rs 14750

Pie Chart | Bar Graph

Complete Screen of the Dashboard

Database Schema



Server Code

```
import os import uuid
from connect import execDB, execReturn


from dotenv import find_dotenv, load_dotenv


positive_money = ['Salary Credited', 'Festival

Bonus'] negative_money = ['EMI', 'Food',

'Transportation', 'Groceries',

                    'Clothing', 'Electronic', 'Entertainment', 'Rent',

'Vacations']




def getGraphDetails(email):
    sql_fd = f"SELECT * FROM finance WHERE email='{email}'
order by date desc"
    r =
    execRetur
    n(sql_fd)
    d =
    dict()
    s = 0
    n = 0
for i in
    negative_mo
    ney: d[i] =
    0

    for i in r:
    if i['CATEGORY'].strip() in negative_money:
        d[i['CATEGORY'].strip()] +=
        abs(int(i['AMOUNT']))
            n = n +
        abs(int(i['AMOUNT']
        )) else:
            s = s +
    int(i['AMOUNT']) if
    (s > n):
        d["Money
    Left"] = s-n
    else:
        d["Mon
    ey Left"]
```

```
          = 0 k = ""
      for i in
          list(d.keys
          ()): k =
          k+i+","

          v = ""
      for i in
          list(d.values
          ()): v =
          v+str(i)+","
          return {"x": k[:-1],
      "y": v[:-1]} def
      getReminder(email):
          sql_fd = f"SELECT percent FROM reminders WHERE
          email='{email}'" r = execReturn(sql_fd)
          return r[0]['PERCENT']


  def
      setReminder(emai
      l, limit): limit
      = int(limit)
          s = f"UPDATE reminders SET percent={limit} WHERE
          email='{email}'" r = execDB(s)


      def isLimitReached(email):
          sql_fd1 = f"SELECT SUM(AMOUNT) FROM finance WHERE AMOUNT>0 AND
      email='{email}'"
```

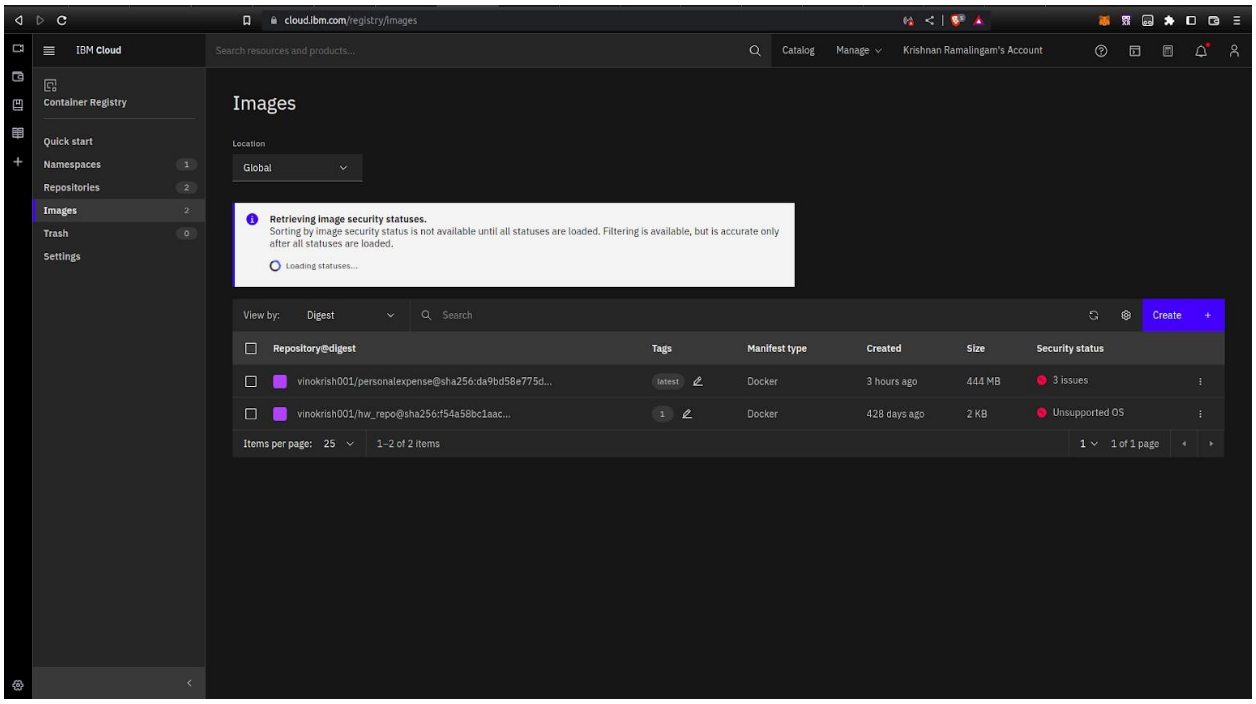
```
        sql_fd2 = f"SELECT SUM(AMOUNT) FROM finance WHERE AMOUNT<0 AND
    email='{email}'"
        r1 = execReturn(sql_fd1)
        r2 =

        execReturn(

        sql_fd2)

        income =

        r1[0]['1']

        expense = -

        r2[0]['1']

        percent =

        expense/inc

        ome percent

        =

        percent*100

        sql_fd = f"SELECT percent FROM reminders WHERE

        email='{email}'" r = execReturn(sql_fd)

        limit =

        int(r[0]['PERCE

        NT']) if limit

        < percent:

            triggerMail()



def addUser(name, email,

    password): print(name,

    email, password)

        sql_fd = f"SELECT * FROM user WHERE

        email='{email}'" r = execReturn(sql_fd)
        if r != []:
            return "Email Exists"

        sql_st = f"INSERT INTO user(name , email , password )
    values ( '{name}' , '{email}' , '{password}' )"
        r = execDB(sql_st)
        sql_st = f"INSERT INTO reminders(email , percent ) values (
        '{email}'
    , 90 )"
        # 90 is the default
        reminder percent r =
        execDB(sql_st)
        return "User registered successfully"



    def getPassword(email):
        sql_fd = f"SELECT password FROM user WHERE
        email='{email}'" r = execReturn(sql_fd)

#print(r[
0]) try:
            return r[0]['PASSWORD'].strip()
        except:
            return ""



    def fetchFinanceRecord(email):
        sql_fd = f"SELECT * FROM finance WHERE email='{email}'
    order by date desc"
```

```
        r =
        execRetur
        n(sql_fd)
        return r


    def getIncomeExpend(email):
        sql_fd1 = f"SELECT SUM(AMOUNT) FROM finance WHERE AMOUNT>0 AND
    email='{email}'"
        sql_fd2 = f"SELECT SUM(AMOUNT) FROM finance WHERE AMOUNT<0 AND
    email='{email}'"
        r1        =
        execReturn(
        sql_fd1)  r2
        =
        execReturn(
        sql_fd2)
        print(r1,
        r2)
        if not r1[0]['1']:
            r1[0]['1'] = 0
        if not r2[0]['1']:
            r2[0]['1'] = 0
        return {"income": r1[0]['1'], "expend": -r2[0]['1']}


def createFinanceRecord(email, category, amount,
    description, date): amount = int(amount)
    if category in
        negative_money:

        amount = -amount

        print("FINANCE", email, amount, category,

        description, date) sql_st = f"INSERT INTO

        finance(id,email , amount , category ,

    description , date ) values ( '{uuid.uuid1()}','{email}' , {amount}
    ,
'{category}' , '{description}' ,

    '{date}' )" r = execDB(sql_st)

        print(r)
        return "Record created successfully"
```

Deployment

```
FROM python:3.10
ENV sendGripdAPI _
ENV sendgridSender _
LABEL maintainer="Krishnan R,
vinokrish001@gmail.com" COPY
./requirements.txt /app/requirements.txt
WORK
DIR
/app
COPY
.
/app
RUN pip install -r
requirements.txt
```

```
EXPOSE 5000

CMD ["python" , "app.py" ]


# docker build - < Dockerfile
```



Images

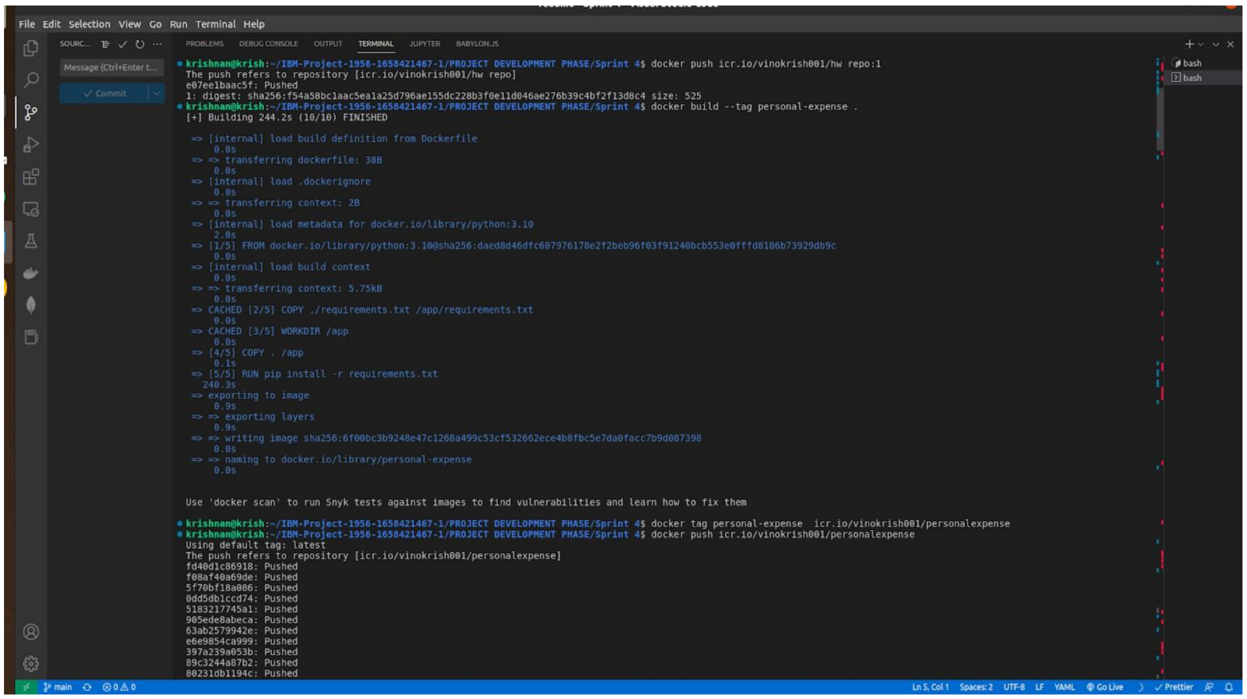
Location: Global

Retrieving image security statuses.
Sorting by image security status is not available until all statuses are loaded. Filtering is available, but is accurate only after all statuses are loaded.
Loading statuses...

View by: Digest

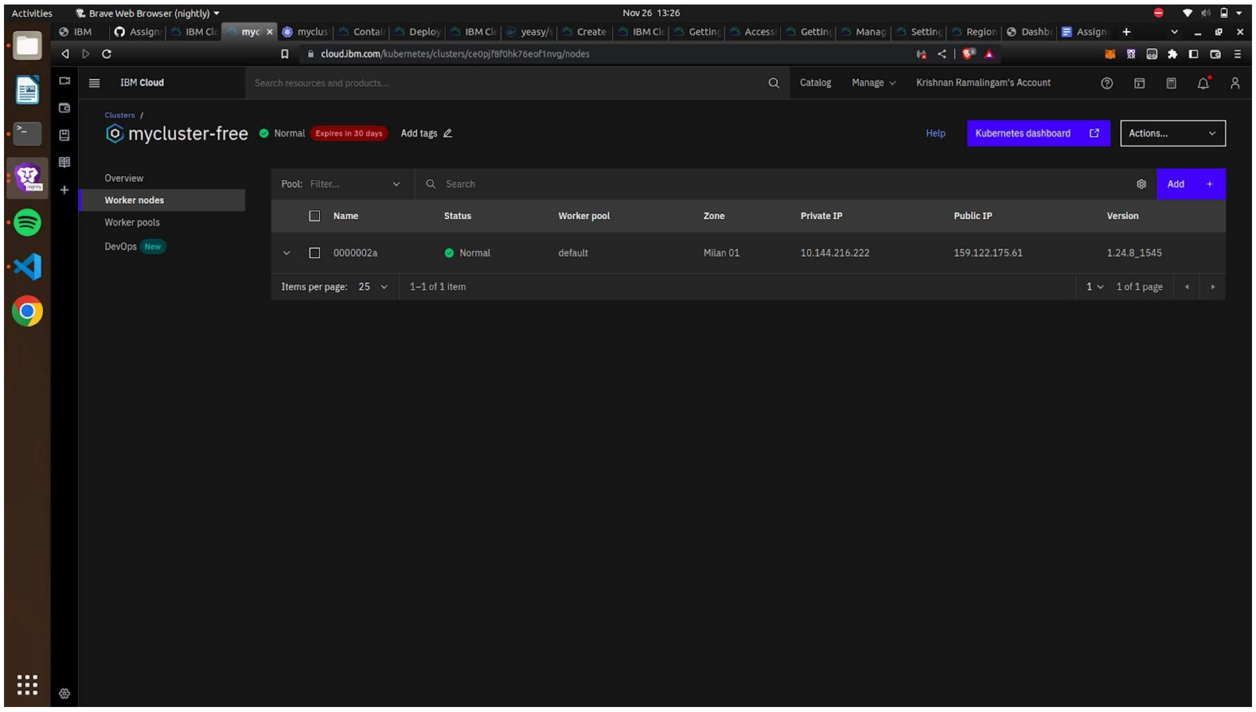
Repository@digest	Tags	Manifest type	Created	Size	Security status
vinokrish001/personalexpense@sha256:da9bd58e775d...	latest	Docker	3 hours ago	444 MB	3 issues
vinokrish001/hw_repo@sha256:f54a58bc1aac...	1	Docker	428 days ago	2 KB	Unsupported OS

Items per page: 25 | 1–2 of 2 items | 1 of 1 page



```
● krishnan@krish:~/IBM-Project-1956-1658421467-1/PROJECT DEVELOPMENT PHASE/Sprint 4$ docker push icr.io/vinokrish001/hw repo:1
The push refers to repository [icr.io/vinokrish001/hw repo]
e07ee1baac5f: Pushed
1: digest: sha256:f54a58bc1aac5ea1a25d796ae155dc228b3f0e11d046ae276b39c4bf2f13d8c4 size: 525
● krishnan@krish:~/IBM-Project-1956-1658421467-1/PROJECT DEVELOPMENT PHASE/Sprint 4$ docker build --tag personal-expense .
[+] Building 244.2s (10/10) FINISHED

 => [internal] load build definition from Dockerfile
     0.0s
 => => transferring dockerfile: 38B
     0.0s
 => [internal] load .dockerignore
     0.0s
 => => transferring context: 2B
     0.0s
 => [internal] load metadata for docker.io/library/python:3.10
     2.8s
 => [1/5] FROM docker.io/library/python:3.10@sha256:daed8d46dfc607976178e2f2beb96f03f91240bcb553e0fffd8186b73929db9c
     0.0s
 => [internal] load build context
     0.0s
 => => transferring context: 5.75kB
     0.0s
 => CACHED [2/5] COPY ./requirements.txt /app/requirements.txt
     0.0s
 => CACHED [3/5] WORKDIR /app
     0.0s
 => [4/5] COPY . /app
     0.1s
 => [5/5] RUN pip install -r requirements.txt
   240.3s
 => exporting to image
     0.9s
 => => exporting layers
     0.9s
 => => writing image sha256:6f00bc3b9248e47c1268a499c53cf532662ece4b8fbc5e7da0facc7b9d087398
     0.0s
 => => naming to docker.io/library/personal-expense
     0.0s


Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

● krishnan@krish:~/IBM-Project-1956-1658421467-1/PROJECT DEVELOPMENT PHASE/Sprint 4$ docker tag personal-expense  icr.io/vinokrish001/personalexpense
● krishnan@krish:~/IBM-Project-1956-1658421467-1/PROJECT DEVELOPMENT PHASE/Sprint 4$ docker push icr.io/vinokrish001/personalexpense
Using default tag: latest
The push refers to repository [icr.io/vinokrish001/personalexpense]
fd40d1c86918: Pushed
f08af40a69de: Pushed
5f70bf18a086: Pushed
0dd5db1ccd74: Pushed
5183217745a1: Pushed
905ede8abeca: Pushed
63ab2579942e: Pushed
e6e9854ca999: Pushed
397a239a053b: Pushed
89c3244a87b2: Pushed
80231db1194c: Pushed
```

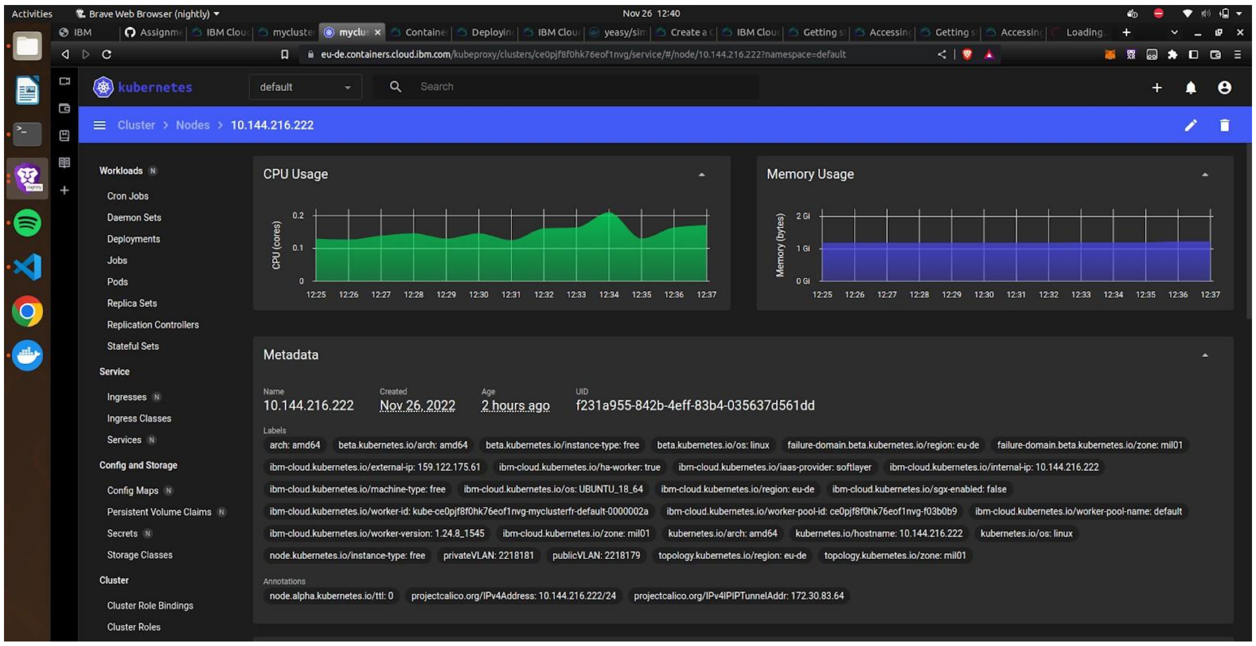
```
        spec:
repli
cas:
1
selec
tor:
```



mycluster-free Normal Expires in 30 days Add tags

Name	Status	Worker pool	Zone	Private IP	Public IP	Version
0000002a	Normal	default	Milan 01	10.144.216.222	159.122.175.61	1.24.8_1545

Items per page: 25 1–1 of 1 item 1 1 of 1 page



Metadata

Name	Created	Age	UID
10.144.216.222	Nov 26, 2022	2 hours ago	f231a955-842b-4eff-83b4-035637d561dd

Labels

```
arch: amd64  beta.kubernetes.io/arch: amd64  beta.kubernetes.io/instance-type: free  beta.kubernetes.io/os: linux  failure-domain.beta.kubernetes.io/region: eu-de  failure-domain.beta.kubernetes.io/zone: mil01
ibm-cloud.kubernetes.io/external-ip: 159.122.175.61  ibm-cloud.kubernetes.io/ha-worker: true  ibm-cloud.kubernetes.io/iaas-provider: softlayer  ibm-cloud.kubernetes.io/internal-ip: 10.144.216.222
ibm-cloud.kubernetes.io/machine-type: free  ibm-cloud.kubernetes.io/os: UBUNTU_18_64  ibm-cloud.kubernetes.io/region: eu-de  ibm-cloud.kubernetes.io/sgx-enabled: false
ibm-cloud.kubernetes.io/worker-id: kube-ce0pjf8f0hk76eof1nvg-myclusterfr-default-0000002a  ibm-cloud.kubernetes.io/worker-pool-id: ce0pjf8f0hk76eof1nvg-f03b0b9  ibm-cloud.kubernetes.io/worker-pool-name: default
ibm-cloud.kubernetes.io/worker-version: 1.24.8_1545  ibm-cloud.kubernetes.io/zone: mil01  kubernetes.io/arch: amd64  kubernetes.io/hostname: 10.144.216.222  kubernetes.io/os: linux
node.kubernetes.io/instance-type: free  privateVLAN: 2218181  publicVLAN: 2218179  topology.kubernetes.io/region: eu-de  topology.kubernetes.io/zone: mil01
```

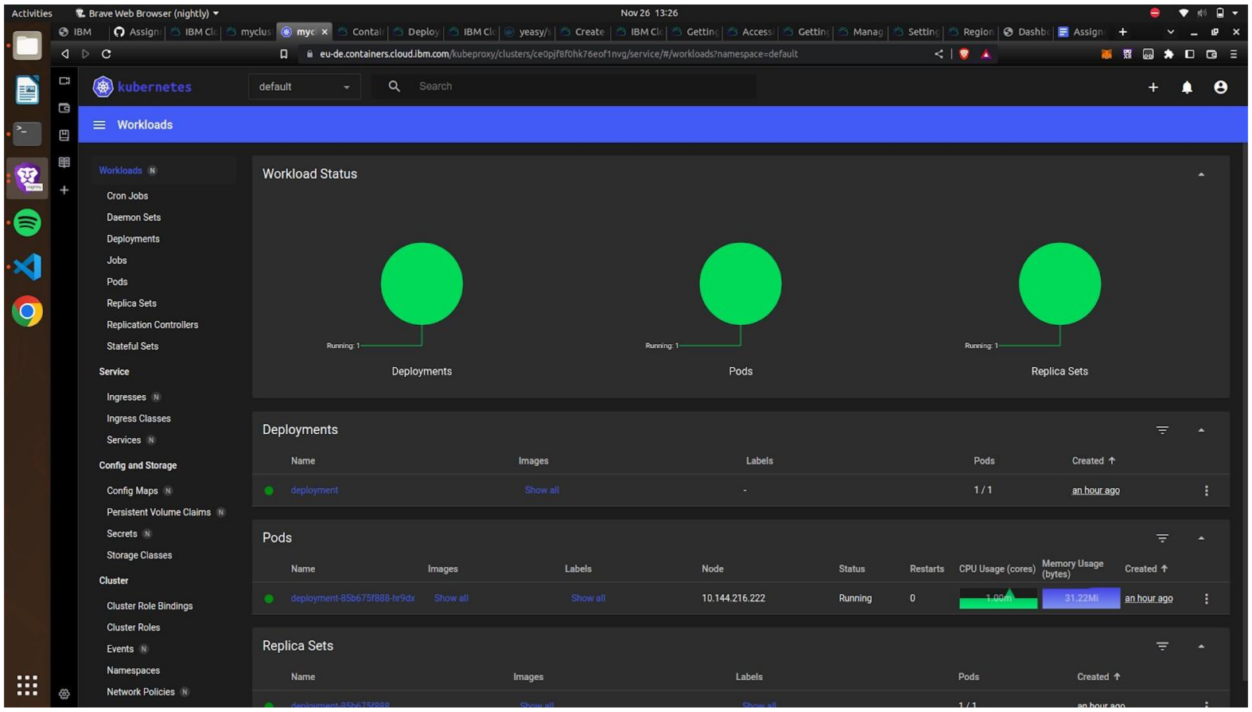
Annotations

```
node.alpha.kubernetes.io/ttl: 0  projectcalico.org/IPv4Address: 10.144.216.222/24  projectcalico.org/IPv4IPIPTunnelAddr: 172.30.83.64
```

```
  matchLabels:
    app:
vinokris
h001
templat
e:
  me
  t
  a
  d
  a
  t
  a
  :

  l
  a
```

```
          b

          e

          l

          s

          :


              app:

          vinokris

          h001

          spec:

            containers:

              - name: vinokrish001

                image:

                icr.io/vinokrish001/personalexpe

                nse ports:

                  - containerPort: 5000

      ---apiVersion: v1 kind: Service metadata:
name:

service

spec:

        selector:

          app:

        vinokr

        ish001

        type:

        NodePo

        rt

        ports:

          - port: 5000
```

```
nodePort: 31514
```



Deployments

	Name	Images	Labels	Pods	Created ↑
●	deployment	Show all	-	1 / 1	an hour ago

Pods

	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
●	deployment-85b675f888-hr9dx	Show all	Show all	10.144.216.222	Running	0	1.00m	31.22Mi	an hour ago

Replica Sets

	Name	Images	Labels	Pods	Created ↑
●	deployment-85b675f888	Show all	Show all	1 / 1	an hour ago

Testing

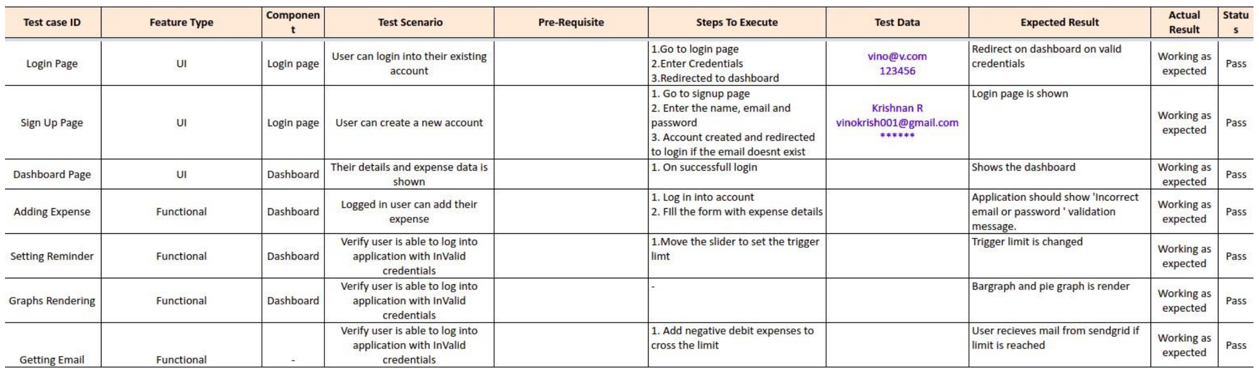
Test Cases

A test case has elements that explain input, action, and an anticipated result in order to assess if an application feature is functioning properly. In order to validate a specific test objective or target, a test case is a series of instructions that, when followed,will indicate whether or not the system behaves as planned.

Qualities of an effective test case:

Accurate: Meets the goal precisely.

Economical: No extraneous actions
or words.
Traceable: Able to be tracked back to
specifications. Repeatable: Allows for repeated
administration of the test. Reusable: May be
used again as required.



Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
Login Page	UI	Login page	User can login into their existing account		1.Go to login page 2.Enter Credentials 3.Redirected to dashboard	vino@v.com 123456	Redirect on dashboard on valid credentials	Working as expected	Pass
Sign Up Page	UI	Login page	User can create a new account		1. Go to signup page 2. Enter the name, email and password 3. Account created and redirected to login if the email doesnt exist	Krishnan R vinokrish001@gmail.com ******	Login page is shown	Working as expected	Pass
Dashboard Page	UI	Dashboard	Their details and expense data is shown		1. On successfull login		Shows the dashboard	Working as expected	Pass
Adding Expense	Functional	Dashboard	Logged in user can add their expense		1. Log in into account 2. FIll the form with expense details		Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Setting Reminder	Functional	Dashboard	Verify user is able to log into application with InValid credentials		1.Move the slider to set the trigger limt		Trigger limit is changed	Working as expected	Pass
Graphs Rendering	Functional	Dashboard	Verify user is able to log into application with InValid credentials		-		Bargraph and pie graph is render	Working as expected	Pass
Getting Email	Functional	-	Verify user is able to log into application with InValid credentials		1. Add negative debit expenses to cross the limit		User recieves mail from sendgrid if limit is reached	Working as expected	Pass

User Acceptance Testing

Users, clients, or other authorised organisations conduct this type of testing to determine the specifications and operational practises of an application or piece of software. Acceptance testing is the most important testing phase since it establishes whether or not the customer will accept the application or software. It could involve the user interface, functionality, usability, and usefulness of the application. It is also known as operational acceptance testing, user acceptability testing, and end-user testing (UAT).

Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	4	2	2	1	9
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	7	2	4	6	19
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	0	0	0	0
Totals	14	7	11	9	41

ADVANTAGES & DISADVANTAGES

Advantages

- Better spending awareness
- Alerts via Email notifications
- Graphical representation of expenses to provide better insights
- No more data loss
- Anytime and anywhere access
- On the go expense submission

Disadvantages

- Because of incorrect details, the application might provide inaccurate results.
- Too many categories of expenses might confuse the user.

CONCLUSION

In this project, we have come up with a personal expense tracker web application to keep track of the expenses spent on each category customised by the user .The user will have to login ,incase of a new user

,a registering of their details is mandatory inorder to access the features of the application.The user after a successful login can enter his details of expenses into various categories and get the dymanic visualisation of graphs and charts.There is an option to limit the expenses by setting a target ,above which no further transactions can be made providing an alert to the user via email that has been registered.

FUTURE SCOPE

In future,the application can be extended by integrating it to multiple cross culture platforms through which transactions are made.The details can be fetched automatically from the applications without requiring the user intervention to manually enter the expenses.This can be extended in a way that the application is accessed concurrently in multiple devices by the same user

Appendix

Github

Github link

Video

Demo Video