PERSONAL EXPENSE TRACKER APPLICATION

NALAIYA THIRAN PROJECT

BASED-LEARNING ON PROFESSIONAL

READINESS FOR INNOVATION,

EMPLOYMENT AND ENTREPRENEURSHIP

Team ID - PNT20222TMID53089

A PROJECT REPORT BY

Team Leader	-	Pradeep G
Team Member 1	-	Varsini S
Team Member 2	-	Nithin S
Team Member 3	-	Ramya Priya S

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

SRI SIVA SUBRAMANIYA NADAR COLLEGE, CHENNAI -600125

PNT20222TMID53089

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights into money management.

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditures in graphical forms. They have the option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.

1.2 Purpose

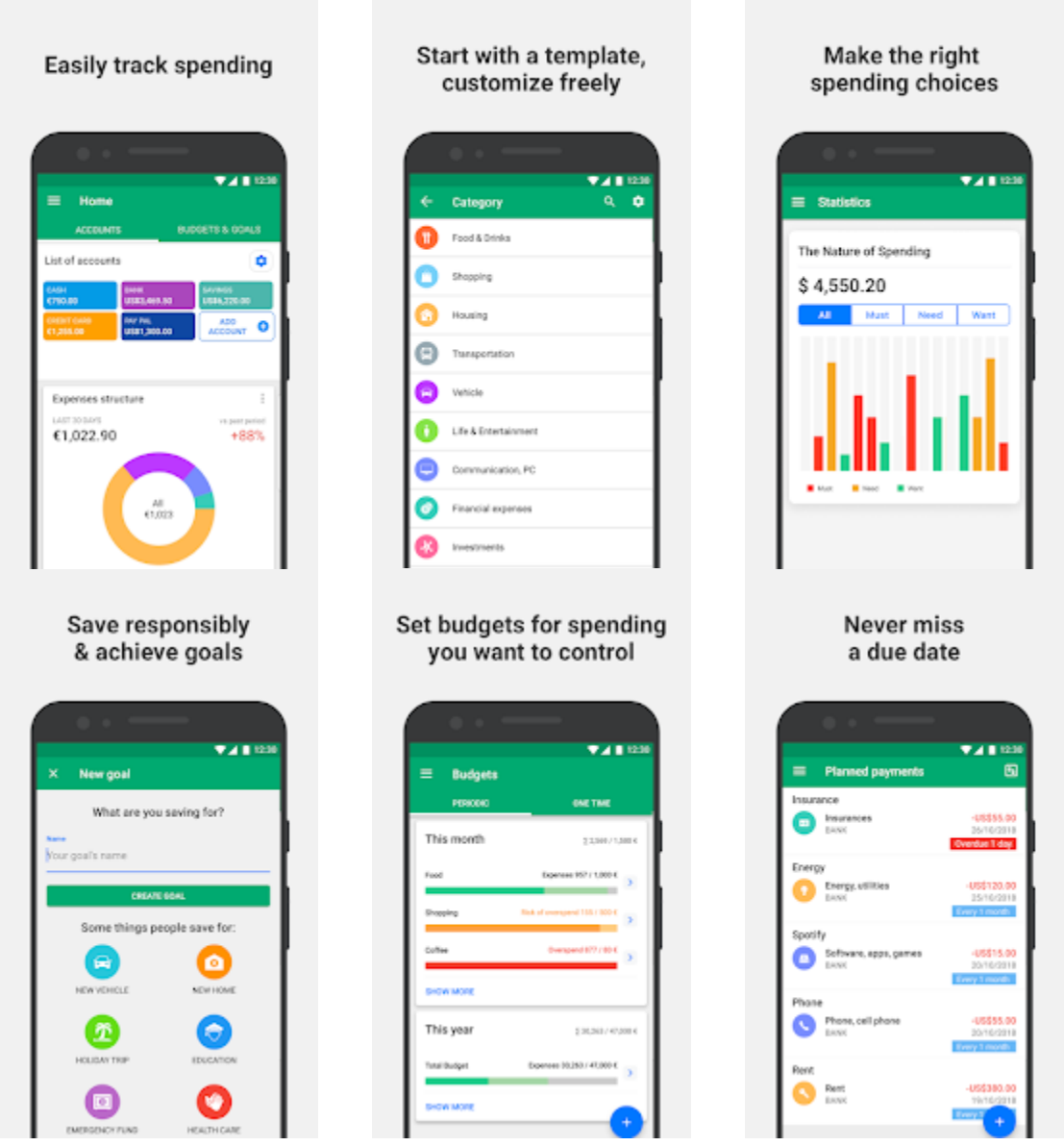
Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses as they find it tedious and time-consuming. Now, you don’t have to worry about managing your expenses, as you can get access to an expense tracker that will help in the active management of your finances. Also known as an expense manager and money manager, an expense tracker is a software or application that helps to keep an accurate record of your money inflow and outflow. Many people in India live on a fixed income, and they find that toward the end of the month they don't have sufficient money to meet their needs. While this problem can arise due to low salary, invariably it is due to poor money management skills. People tend to overspend without realizing it, and this can prove to be disastrous. Using a daily expense manager can help you keep track of how much you spend every day and on what. At the end of the month, you will have a clear picture of where your money is going. This is one of the best ways to get your expenses under control and bring some semblance of order to your finances. Today, there are several

expense manager applications in the market. Some are paid managers while others are free. Even banks like ICICI offer their customers expense trackers to help them out. Before you decide to go in for a money manager, it is important to decide the type you want.

2. LITERATURE SURVEY

2.1 Existing problem

Wallet: Budget Expense Tracker



Wallet is a market-leading personal finance manager, built to help you save money, plan for the future, and see all your finances in one place. With Wallet, you can stay on top of your daily expenses automatically with bank synchronization, dive into weekly reports on

your spending, manage debt, and track bills. Share specific features with loved ones to get on top of your budgets together. The wallet allows you to see your finances your way: anywhere, any time.

WHAT MAKES A WALLET UNIQUE:

- Automatic Bank Updates - Transactions are automatically and securely synced, then smartly categorized, and factored into your budget.. With 3,500 participating banks worldwide, you'll save loads of time not having to track every penny by tracking all your finances in one place.

- Flexible Budgets - Budgets help you plan your spending & to save money for the future thanks to the integrated money manager. Whatever it is you need to accomplish, from paying off debt to buying a car or saving for retirement, this budgeting app offers the flexibility to meet your goals and cleverly react to any changing financial circumstances. With Wallet, budgeting expenses has never been easier.

- Insightful reports - Easy-to-understand graphs and financial overviews give you actionable insights about the state of your finances, across accounts, credit and debit cards, debts and cash. Gain insights on where you should be budgeting more or could save more. And don't forget about your income & expense report!

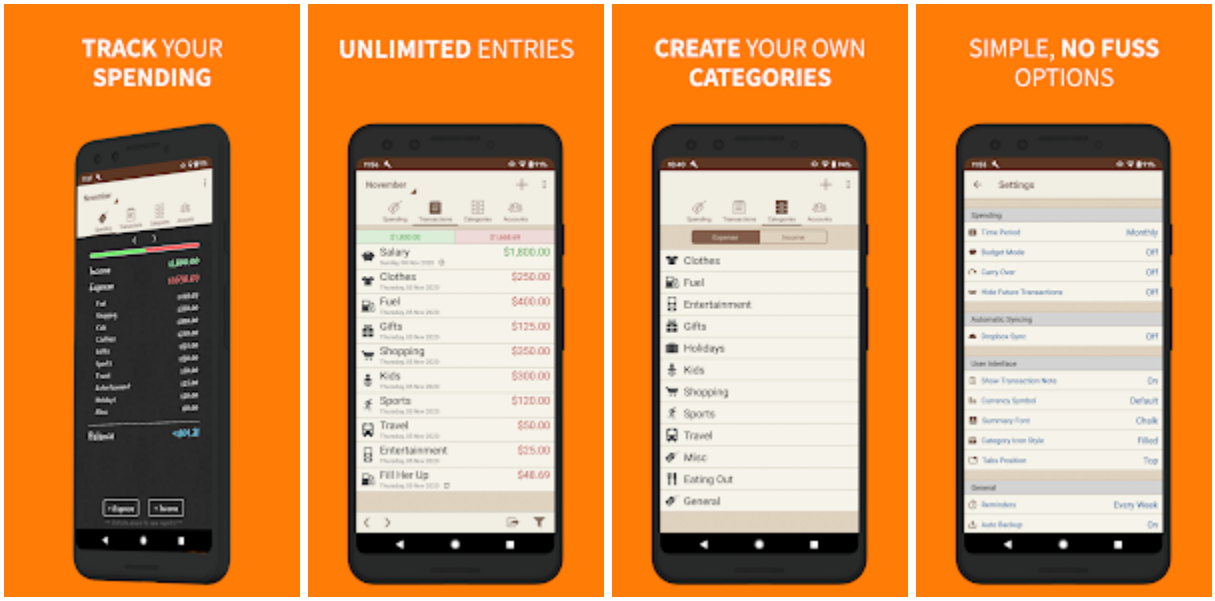
- Planned Payments - Never miss a due date with this bill tracker. Organize bills and keep track of due dates. See upcoming payments and how the payments will impact your cash flow.

- Sharing selected accounts - Selected accounts can be shared with family, friends or colleagues who need to cooperate on a budget. Everyone can contribute from any platform, whether it be Android, iPhone or the Web.

- Imports or Manual Updates - You can now import all your transaction data from sources of your choice so you will get a full report for simple expense tracking. Be it from your bank or your own spreadsheets.

Spending Tracker

Spending Tracker is the easiest and most user-friendly expense manager app in the store. The simple fact is, by tracking your spending you will be able to stick to a budget and therefore SAVE MONEY. So download it for free, enter your expenses and income, and have instant control over your spending!



Features

- Simple and intuitive user interface
  - helps you to easily manage your spending
  - super fast expense entry

- Flexible Time Periods
  - choose to track Daily, Weekly, Monthly or Yearly

- Budget Mode

- optionally set a fixed budget amount to help you meet your spending targets
- carry over any remaining budget to the next month or week

- Summary View
  - overview of your current balance along with expense and income totals
  - see your main areas of spending

- Log Expense and Income
  - repeat your expenses daily, weekly or monthly
  - export to CSV for use in a spreadsheet
  - export to PDF for viewing and printing (Pro Upgrade required)

- Multiple Accounts
  - create separate personal, business and savings accounts for example

- Reports
  - beautiful and interactive charts for easy visualisation of the flow of money
  - view spending grouped by category
  - see your history so you can track your progress

- Categories
  - editable expense and income categories
  - choose a quality icon for each category

- Syncing (Pro Upgrade required)
  - automatically sync your data to other Android devices
  - you can also sync to iOS and Windows versions of Spending Tracker, which will require a separate upgrade on these platforms

- Backups
  - keep your data safe by backing up to Dropbox

- the auto backups feature will take care of it for you

- Widget
  - put the widget on your home screen for easy access
  - quick add buttons

- Local Tips
  - optionally receive tips when you enter participating local venues (requires location and Bluetooth permissions)
  - this could include money-saving offers

- Special design layout for tablets
  - optimal use of larger screen size makes it even better for management

Money Manager Expense & Budget

Money Manager - the #1 financial planning, review, expense tracking, and personal asset management app for Android!

Money Manager makes managing personal finances as easy as pie! Now easily record your personal and business financial transactions, generate spending reports, review your daily, weekly and monthly financial data and manage your assets with Money Manager's spending tracker and budget planner.

FEATURES

* Applying double-entry bookkeeping accounting system

Money Manager facilitates efficient asset management and accounting. It does not just record your money coming in and out of your account but deposits your money into your account as soon as your income is input and draws money from your account as soon as your expense is input.

PNT20222TMID53089

* Budget and expense management function

Money Manager shows your budget and expenses by a graph so you can see the number of your expenses against your budget quickly and make suitable financial inferences

* Credit / Debit Card management function

Entering a settlement date, you can see the payment amount and outstanding payment in the asset tab. You can arrange the automatic debit by simply connecting your debit card with your account.

* Passcode

You can check your passcode so you can safely manage your financial review account book with Money Manager

* Transfer, direct debit and recurrence function

Transfer between assets is possible, which makes your personal and business asset management more efficient. In addition, you can manage your salary, insurance, term deposit and loan more easily by setting automatic transfer and recurrence.

* Instant statistics

Based on the data entered, you can instantly see your expense by category and changes between each month. And you can see the change in your assets and income/expense indicated by a graph as well.

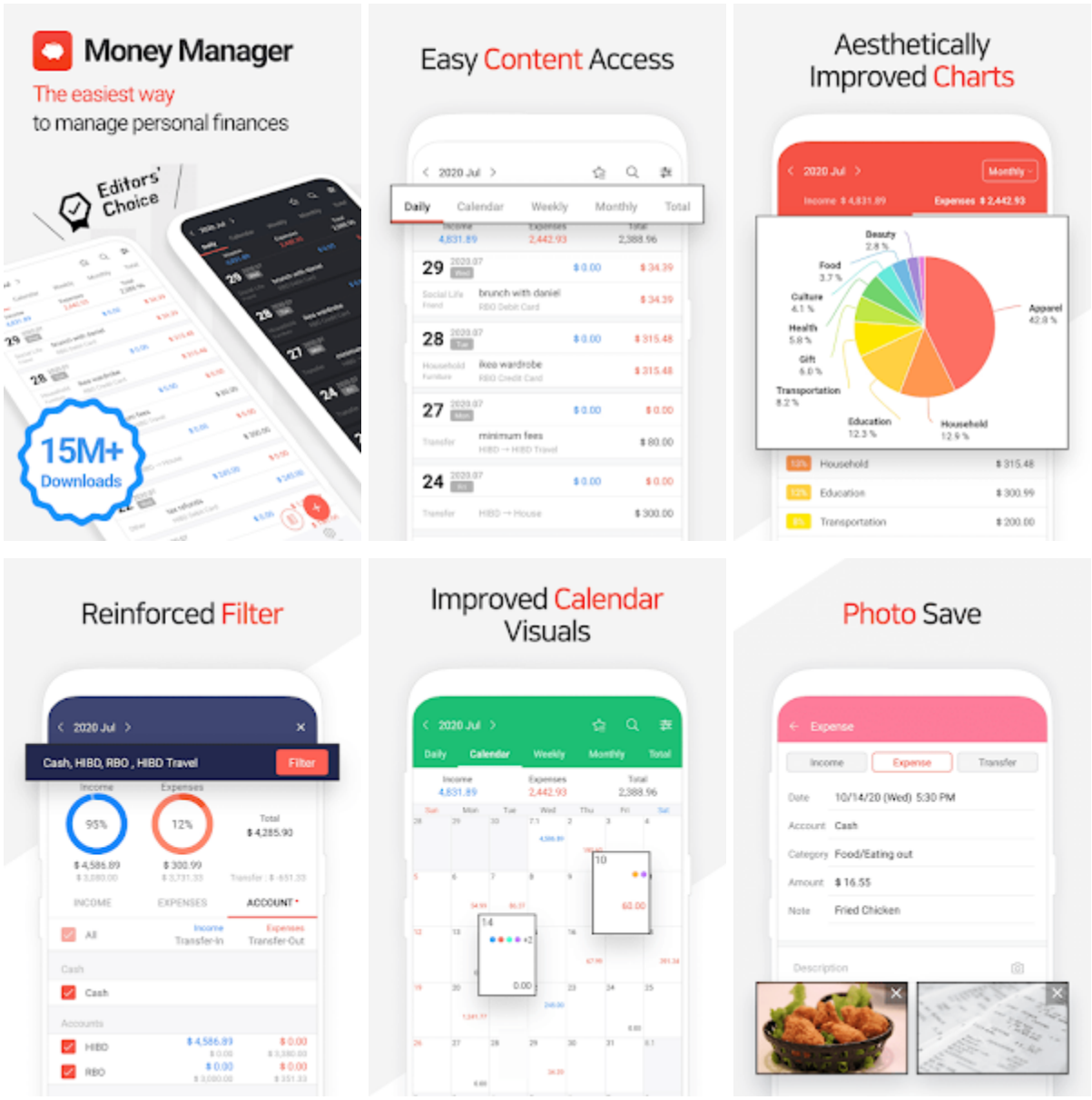
* Bookmark function

You can easily input your frequent expenses all at once by bookmarking them.

* Backup / Restore

You can make and view backup files in Excel file and backup/restore is possible.

PNT20222TMID53089



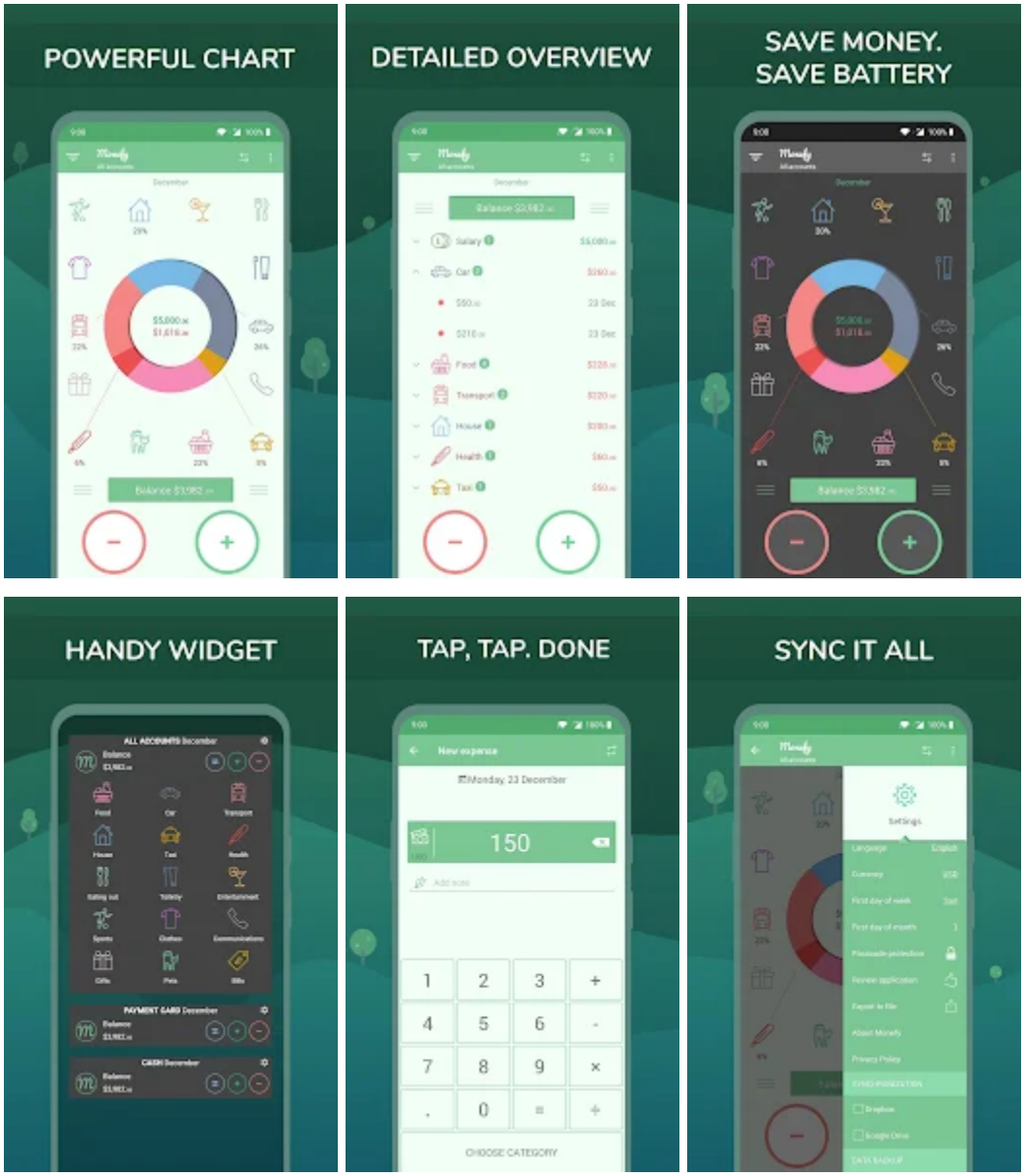
Monefy - Budget & Expenses app

Monefy is more than a money tracker, it's also one of the best savings apps to help you with money management. Keep track of your personal expenses and compare them to your monthly income with the budget planner. Keep your monthly budget in mint condition. Your new budgeting app will help you become a budgeting master and start saving money with Monefy.

Key features which make tracking enjoyable and powerful:

PNT20222TMID53089

- Add new records quickly with the intuitive and easy-to-use interface
- See your spending distribution on an easy-to-read chart, or get detailed information from the records list
- Safely synchronize using your own Google Drive or Dropbox account
- Take control of recurring payments
- Track in multi-currencies
- Access your spending tracker easily with handy widgets
- Manage custom or default categories
- Backup and export personal finance data in one click
- Save money with a budget tracker
- Stay secure with passcode protection
- Use multiple accounts
- Crunch numbers with the built-in calculator

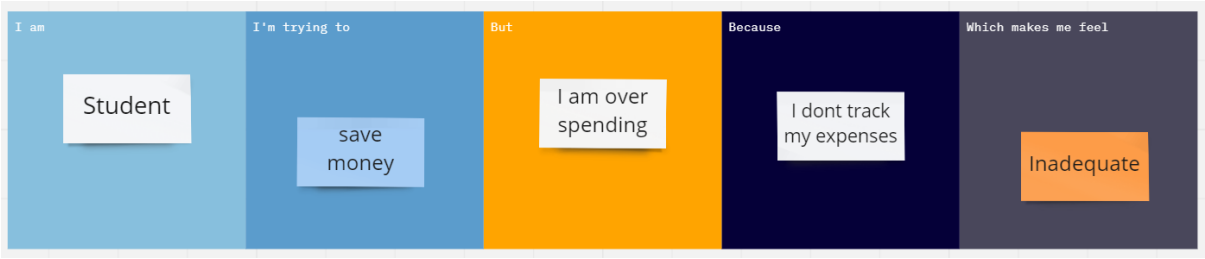


2.2 Problem Statement Definition

In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights into money management.

PNT20222TMID53089

Personal finance applications will ask users to add their expenses and based on their expenses wallet balance will be updated which will be visible to the user. Also, users can get an analysis of their expenditures in graphical forms. They have the option to set a limit for the amount to be used for that particular month if the limit is exceeded the user will be notified with an email alert.



3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

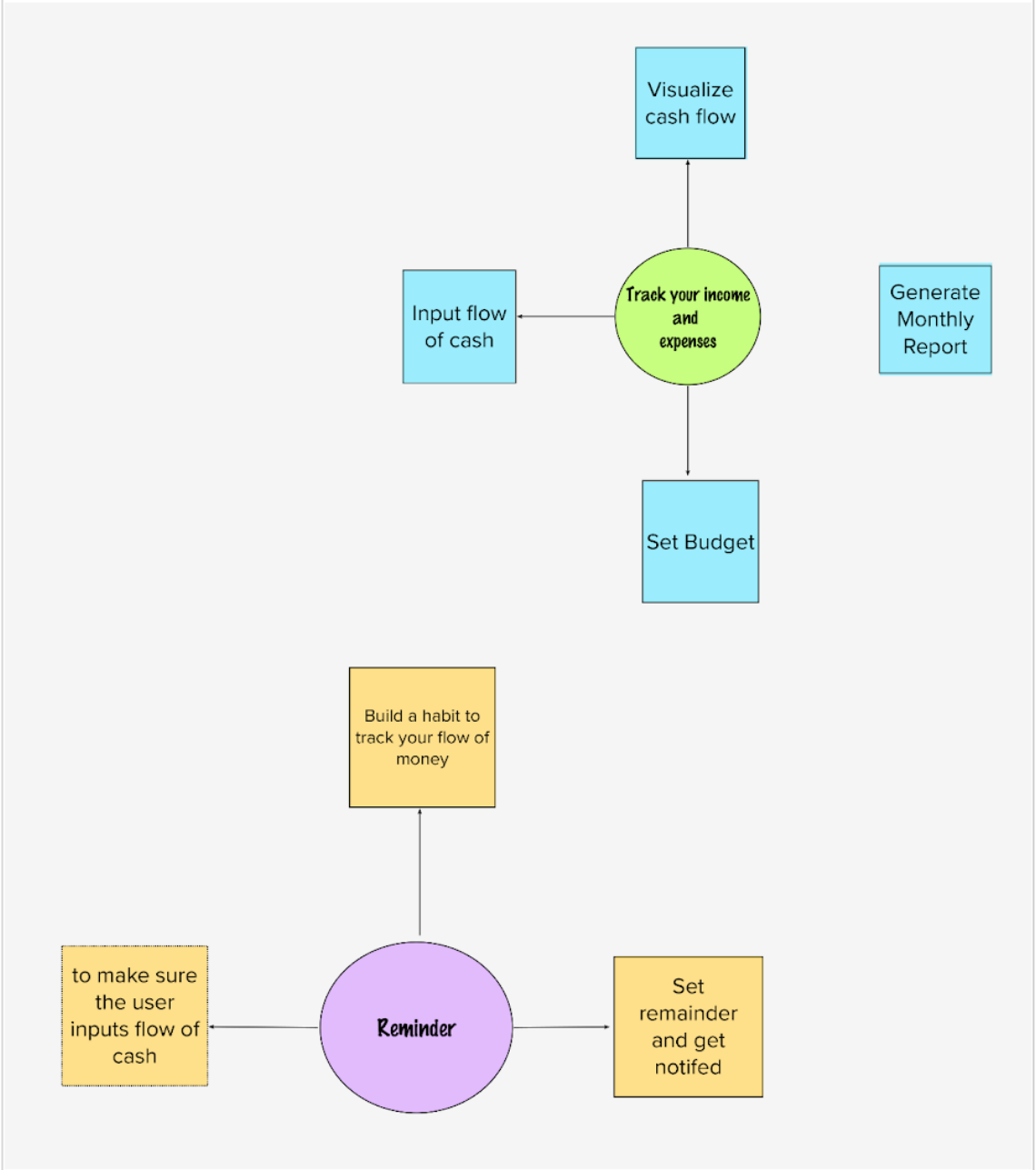
PNT20222TMID53089

3.2 Ideation & Brainstorming

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.

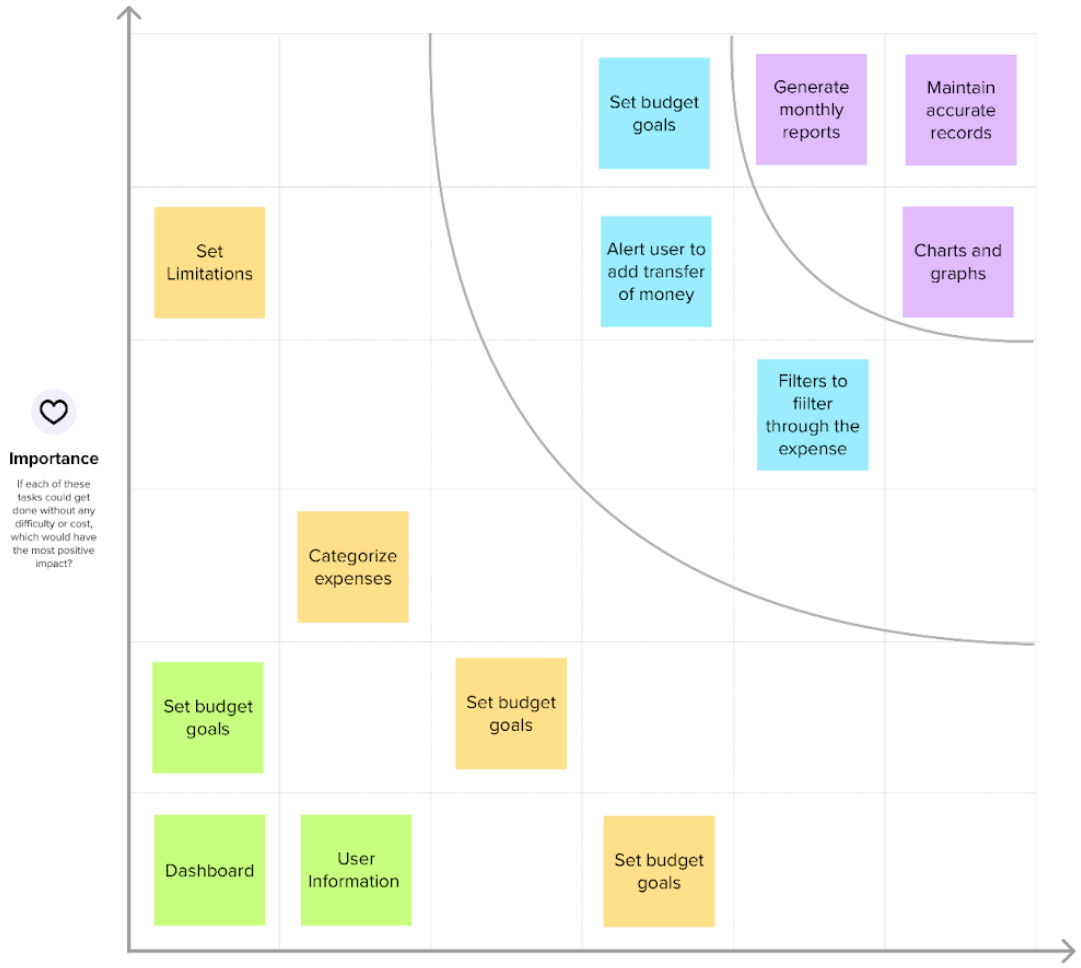


PNT20222TMID53089

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.

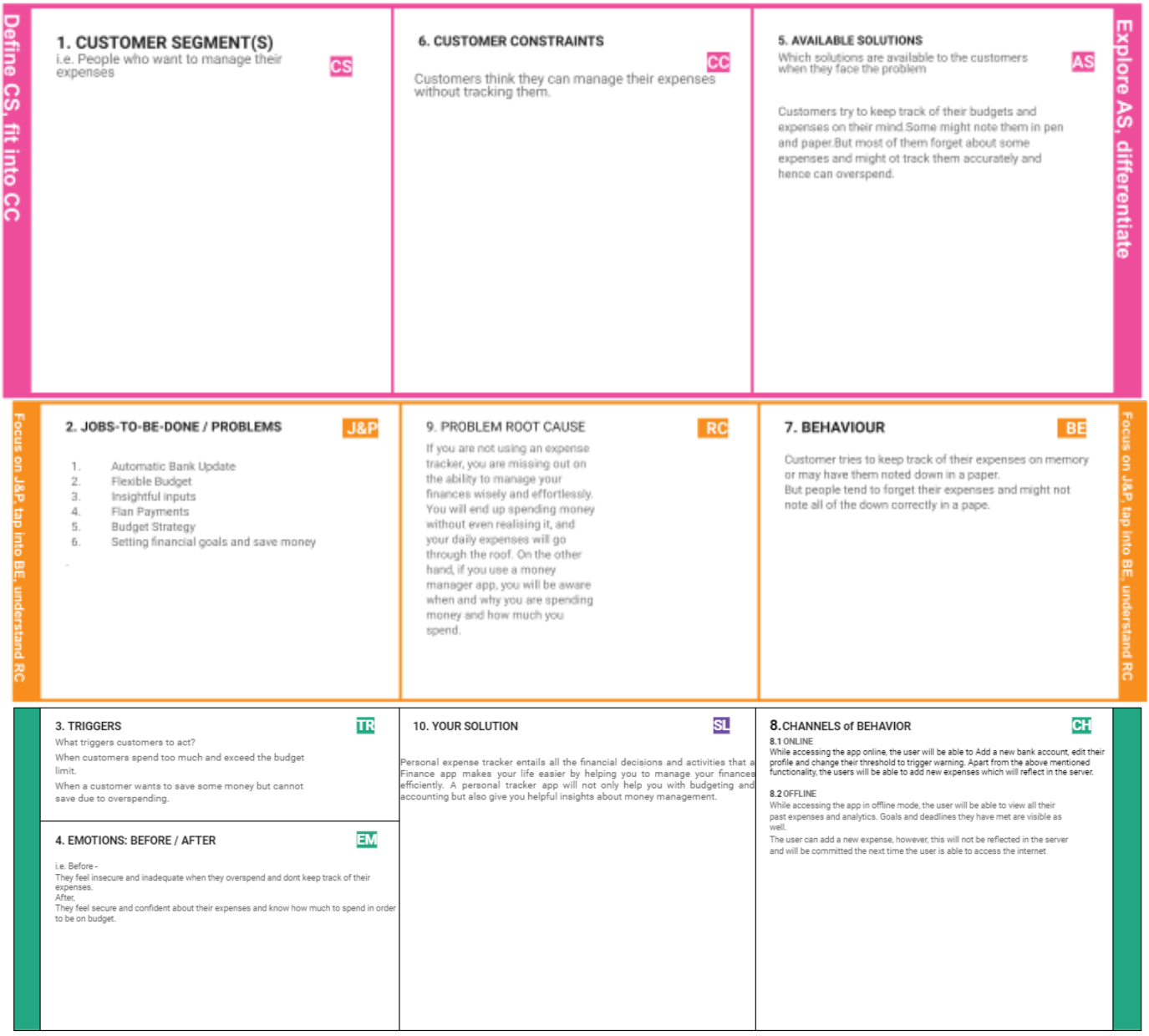


3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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 help you with budgeting and accounting and give you helpful insights into money management.
2.	Idea / Solution description	The proposed personal expense tracking app will be developed using flask and will be hosted on IBM cloud and Sendgrid. The app will provide access to customers upon a one-time payment and will be a one-stop solution for all expense-tracking-related issues.
3.	Novelty / Uniqueness	Common for all applications. One time payment for lifetime usage. Better UI design and efficient usage that meets all the customer needs. Keep track of your money lent or borrowed.
4.	Social Impact / Customer Satisfaction	Customers are able to manage their expenses and save money which they were previously unable to do. Users become mindful of their spending and inculcate the habit of spending only on what is necessary.
5.	Business Model (Revenue Model)	Customers initially make a one-time payment to get complete access to all the features of the app. Future updates are included as well.
6.	Scalability of the Solution	1. Since the application is hosted on the cloud, it can handle any number of requests from any number of clients. 2. This Application can be scaled to manage the warranty of electronic goods by notifying the user a set time before the warranty ends so that they can extend the appliance's warranty or insurance.

		3. The app can be used to share data with 3rd party vendors on getting permission from the customers. Data such as which of the vendor's products is fastest moving vs which products fail often leading to customers claiming the warranty.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User login	Login through OTP
FR-4	View expenses	Current expenses can be viewed through expenses dashboard
FR-5	Add new expense	New expenses can be added in offline mode but will be reflected the server only when online. Provision to add Tags, Amount, Date
FR-6	Delete expense	Expense can be easily deleted and the new balance will be refreshed immediately
FR-7	Sorting	The app allows users to sort their expenses in many views namely, Daily, weekly, monthly and yearly
FR-8	Visualisation of expenses	Using visualisation tools to display the expense split

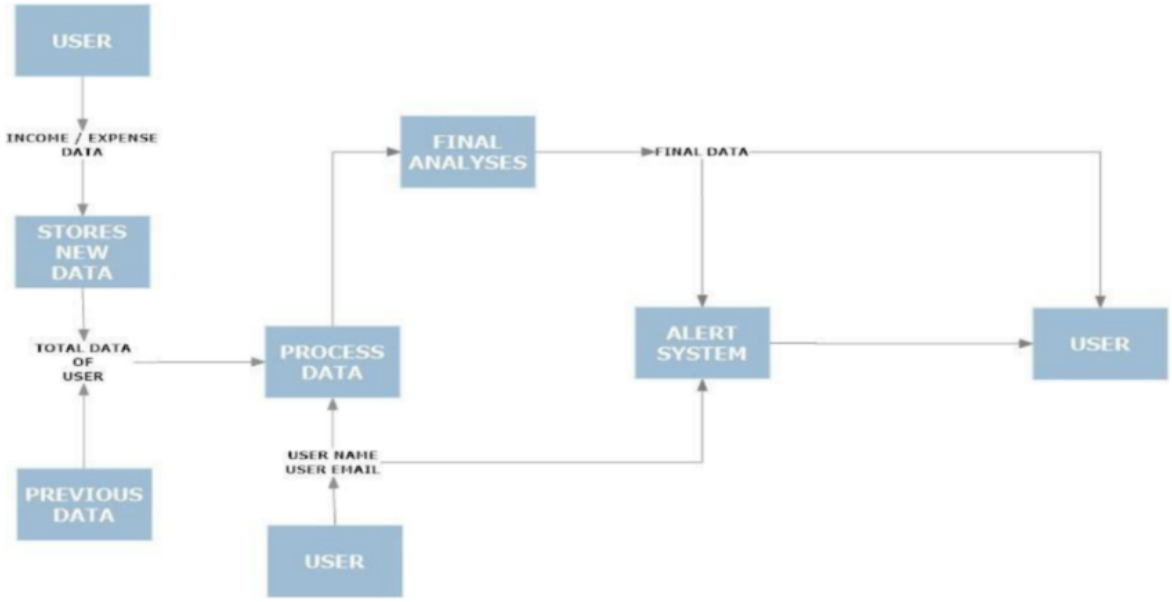
4.2 Non-Functional requirements

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

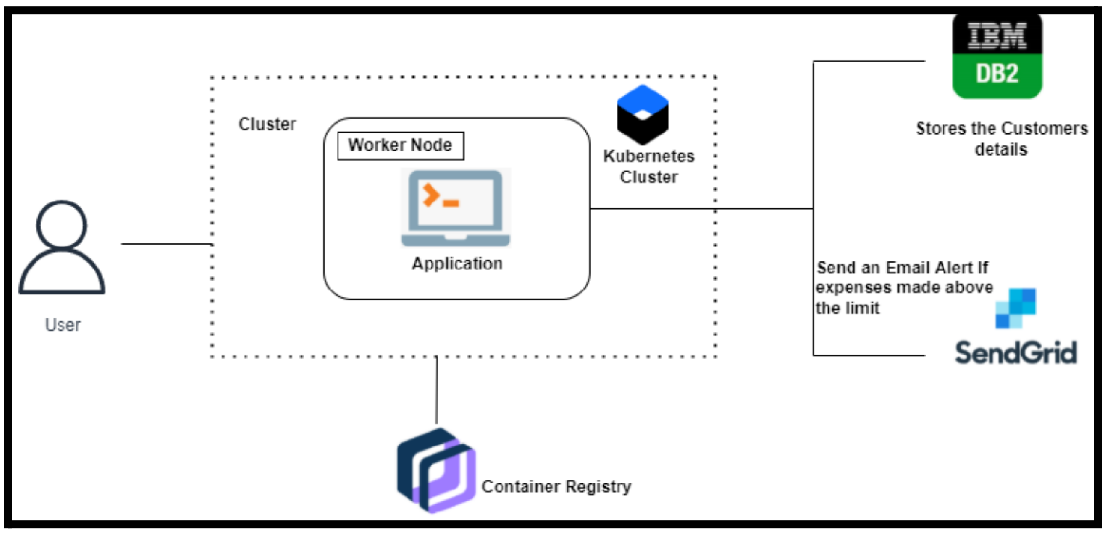
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application will be user-friendly with a good UI and smooth operation without any glitches
NFR-2	**Security**	Access to the application will be through OTP on the registered mobile phone. Data stored in the IBM cloud will be protected by IBM encryption standards
NFR-3	**Reliability**	The system shall be in operational mode for at least 320 days in a year. In case of failure, the system shall be able to recover in under 5 seconds.
NFR-4	**Performance**	The performance of the application will not degrade even upon high-stress loads
NFR-5	**Availability**	The application will not experience downtime and will strive to be online on all days
NFR-6	**Scalability**	The project will be scalable to include more functionality in the future

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



PNT20222TMID53089

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		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
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard with Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
Customer Care Executive	Dashboard	USN-6	As a user i can enter my income and expenditure details	I can view my daily,monthly and yearly expenses	High	Sprint-1
		USN-7	As a customer care executive I can solve the log in issues and other issues of the application	I can provide support or solution at any time 24*7	Medium	

6. PROJECT PLANNING & SCHEDULING

6.1 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, password, and confirming my password.	2	High	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	2	High	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-2		USN-3	As a user, I can register for the application through social media accounts	2	Low	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	2	High	Pradeep G Ramya priya S Varsini S Nithin S
	Dashboard	USN-5	As a user i can enter my income and expenditure details	4	High	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-2	Financial account	USN-6	As a user, I can add and remove any financial accounts	2	High	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-3	Notifications	USN-7	As a user, I can receive alerting notifications on untracked expenses	2	High	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-3		USN-8	As a user, I can receive suggesting notifications for saving and earning money	2	Medium	Pradeep G Ramya priya S Varsini S Nithin S
Sprint-4	Security	USN-9	As a user, I am assured for linking my financial accounts securely	4	High	Pradeep G Ramya priya S Varsini S Nithin S

PNT20222TMID53089

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-4	Customer care	USN-10	As a user, I can access the customer care for any queries and issues regarding the applications	2	Medium	Pradeep G Ramya priya S Varsini S Nithin S

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	3	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	2	
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	3	
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	2	

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

7. CODING & SOLUTIONING

7.1 Source Code

```
from flask import Flask, render_template, request, redirect, session
# from flask_mysqldb import MySQL
# import MySQLdb.cursors
import re
import ibm_db
import datetime
import math

app = Flask(__name__)


app.secret_key = 'a'

print('Connecting')
conn                                                                    =
   ibm_db.connect("DATABASE=bludb;HOSTNAME=1bbf73c5-d84a-4bb0-85b9-ab1a4348f4
   a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=32286;SECURITY=SSL;
   SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=qtj16063;PWD=bc2ajnvAWdN
   QEqCA",'','')
print('Connected')


#HOME--PAGE
@app.route("/home")
def home():
    return render_template("homepage.html")

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

```
#SIGN--UP--OR--REGISTER


@app.route("/signup")
def signup():
    return render_template("login_new.html")



@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = username
        password = request.form['password']

        # cursor = mysql.connection.cursor()
            # cursor.execute('SELECT * FROM register WHERE username = % s',
   (username, ))
        # account = cursor.fetchone()
        # print(account)

        sql = "SELECT * FROM REGISTER WHERE email =?"
        print(username)
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        # print(account)

        if account:
            msg = 'Account already exists !'
            print(msg)
            return render_template('login_new.html', msg = msg)

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
            print(msg)
```

```
            return render_template('login_new.html', msg = msg)

        else:

            sql = "insert into register values (?, ?)"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt,1,username)
            ibm_db.bind_param(stmt,2,password)
            ibm_db.execute(stmt)
            # print(res)
            msg = 'You have successfully registered !'
            print(msg)
            return render_template('login_new.html', msg = msg)




 #LOGIN--PAGE

@app.route("/signin")
def signin():
    return render_template("login_new.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''


    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        print(username)
        print(password)

        sql = "SELECT * FROM REGISTER WHERE email =? AND pass=?"
        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['EMAIL']
            userid =  account['EMAIL']
            session['username'] = account['EMAIL']

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

#ADDING----DATA


@app.route("/add")
def adding():
    return render_template('add.html')


@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():

    date = request.form['date'] + ''
    expensename = request.form['expensename'] + ''
    amount = request.form['amount'] + ''
    paymode = request.form['paymode'] + ''
    category = request.form['category'] + ''

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

    today = str(datetime.datetime.today())
    date = date.split('T')[0]
    today = today[:11]
```

```
    sql = "select limit from limits where email = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    lim = ibm_db.fetch_tuple(stmt)
    if not lim:
        lim = math.inf
    else:
        lim = int(lim[0])

    if today <= date:
        msg = "Date can't be in future"
        return render_template('add.html', msg = msg)
    else:
        month = today[:7]
        if date[:7] == month:
            sql = "select * from expenses where email = ? and date like '" +
month + "%'"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt,1,session['id'])
            ibm_db.execute(stmt)
            expense = ibm_db.fetch_tuple(stmt)
            month_total = 0
            while expense:
                month_total += int(expense[4])
                expense = ibm_db.fetch_tuple(stmt)

            if lim >= month_total + int(amount):
                    sql = "insert into expenses (email, date, expensename,
amount, paymode, category) values (?, ?, ?, ?, ?, ?)"
                stmt = ibm_db.prepare(conn, sql)
                ibm_db.bind_param(stmt,1,session['id'])
                ibm_db.bind_param(stmt,2, date)
                ibm_db.bind_param(stmt,3, expensename)
                ibm_db.bind_param(stmt,4, amount)
                ibm_db.bind_param(stmt,5, paymode)
                ibm_db.bind_param(stmt,6, category)
                ibm_db.execute(stmt)
                return redirect("/display")
```

```
            else:
                msg = "Monthly limit exceeded"
                  return render_template('add.html', msg="Can't add expense as
    monthly limit exceeded")

        else:
               sql = "insert into expenses (email, date, expensename, amount,
    paymode, category) values (?, ?, ?, ?, ?, ?)"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt,1,session['id'])
            ibm_db.bind_param(stmt,2, date)
            ibm_db.bind_param(stmt,3, expensename)
            ibm_db.bind_param(stmt,4, amount)
            ibm_db.bind_param(stmt,5, paymode)
            ibm_db.bind_param(stmt,6, category)
            ibm_db.execute(stmt)
            return redirect("/display")




#DISPLAY---graph

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

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

      sql = "SELECT * FROM expenses WHERE email = ? order by expenses.date
    desc"
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    expense_list = []
    expense = ibm_db.fetch_tuple(stmt)

    if expense:
        expense = list(expense)
        expense[2] = expense[2].replace('T', '  ')
        expense_list.append(expense)

    while expense != False:
        expense = ibm_db.fetch_tuple(stmt)
        if expense:
            expense = list(expense)
            expense[2] = expense[2].replace('T', '  ')
            expense_list.append(expense)

    for x in expense_list:
        total += int(x[4])
        if x[6] == "food":
            t_food += int(x[4])

        elif x[6] == "entertainment":
            t_entertainment += int(x[4])

        elif x[6] == "business":
            t_business += int(x[4])

        elif x[6] == "rent":
            t_rent += int(x[4])

        elif x[6] == "EMI":
            t_EMI += int(x[4])

        elif x[6] == "other":
            t_other += int(x[4])
```

```
        return  render_template('display.html'  ,texpense  =  expense_list,
   expense=expense_list,  total=total,
                           t_food=t_food, t_entertainment=t_entertainment,
                           t_business=t_business,  t_rent=t_rent,
                           t_EMI=t_EMI,  t_other=t_other)



# delete---the--data

@app.route('/delete/<string:id>', methods = ['POST', 'GET' ])
def delete(id):
    #  cursor = mysql.connection.cursor()
    #  cursor.execute('DELETE FROM expenses WHERE  id = {0}'.format(id))
    #  mysql.connection.commit()
    #  print('deleted successfully')

    sql = "delete from expenses where id = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,id)
    ibm_db.execute(stmt)

    return redirect("/display")



#  #limit
@app.route("/limit" )
def limit():
    sql = "select limit from limits where email = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    lim = ibm_db.fetch_tuple(stmt)
    print(lim)
    msg = ""
    if lim:
        msg = "Currently your MONTHLY limit is ₹ " + lim[0]
```

```
    else:
        msg = "Currently no monthly limit is set"

    return render_template('limit.html', y=msg)

@app.route("/limitnum" , methods = ['POST', 'GET' ])
def limitnum():
    sql = "select limit from limits where email = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    msg = "Currently your MONTHLY limit is ₹ "
    lim = ibm_db.fetch_tuple(stmt)
    if not lim:
        number = request.form['number'] + ''
        sql = "insert into limits values (?, ?)"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,session['id'])
        ibm_db.bind_param(stmt,2,number)
        ibm_db.execute(stmt)
        msg += number

    if limit:
        number = request.form['number'] + ''
        sql = "update limits set limit = ? where email = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,number)
        ibm_db.bind_param(stmt,2,session['id'])
        ibm_db.execute(stmt)
        msg += number
    return redirect('/limit')

# #REPORT

@app.route("/today")
def today():
    ttotal = 0
    import datetime
```

```
    x = datetime.datetime.now()
    y = str(x).split(' ')[0]

    sql = "SELECT * FROM expenses  WHERE email = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    expense_list = []
    texpense_list = []
    expense = ibm_db.fetch_tuple(stmt)
    if expense:
        print(expense[2])
        date = expense[2].split('T')[0]
        ttotal += int(expense[4])
        texpense_list.append(expense)
        print(date)
        print(y)
        if date == y:
            expense_list.append(expense)
            print(expense)

        while expense != False:
            expense = ibm_db.fetch_tuple(stmt)

            if expense != False:
                ttotal += int(expense[4])
                texpense_list.append(expense)
                date = expense[2].split('T')[0]
                if date == y:
                    expense_list.append(expense)
                    print(expense)
    total = 0
    t_food = 0
    t_entertainment = 0
    t_business = 0
    t_rent = 0
    t_EMI = 0
    t_other = 0
```

```
    for x in expense_list:
        total += int(x[4])
        if x[6] == "food":
            t_food += int(x[4])

        elif x[6] == "entertainment":
            t_entertainment += int(x[4])

        elif x[6] == "business":
            t_business += int(x[4])

        elif x[6] == "rent":
            t_rent += int(x[4])

        elif x[6] == "EMI":
            t_EMI += int(x[4])

        elif x[6] == "other":
            t_other += int(x[4])

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)

          return  render_template("today.html",  texpense  =  texpense_list,
   expense=expense,  total=total,
                          t_food=t_food, t_entertainment=t_entertainment,
                          t_business=t_business,  t_rent=t_rent,
                          t_EMI=t_EMI,  t_other=t_other)


@app.route("/month")
def month():
    ttotal = 0
```

```
import datetime
x = datetime.datetime.now()
month = str(x).split()[0].split('-')
y = month[0] + '-' + month[1]

sql = "SELECT * FROM expenses  WHERE email = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,session['id'])
ibm_db.execute(stmt)
expense_list = []
texpense_list = []
expense = ibm_db.fetch_tuple(stmt)
if expense:
    date = expense[2].split('T')[0].split('-')
    date = date[0] + '-' + date[1]
    ttotal += int(expense[4])
    texpense_list.append(expense)
    print(date)
    print(y)
    if date == y:
        expense_list.append(expense)
        print(expense)

    while expense != False:
        expense = ibm_db.fetch_tuple(stmt)

        if expense != False:
            ttotal += int(expense[4])
            texpense_list.append(expense)
            date = expense[2].split('T')[0].split('-')
            date = date[0] + '-' + date[1]
            if date == y:
                expense_list.append(expense)
                print(expense)
total = 0
t_food = 0
t_entertainment = 0
t_business = 0
t_rent = 0
```

```
    t_EMI = 0
    t_other = 0

    for x in expense_list:
        total += int(x[4])
        if x[6] == "food":
            t_food += int(x[4])

        elif x[6] == "entertainment":
            t_entertainment += int(x[4])

        elif x[6] == "business":
            t_business += int(x[4])

        elif x[6] == "rent":
            t_rent += int(x[4])

        elif x[6] == "EMI":
            t_EMI += int(x[4])

        elif x[6] == "other":
            t_other += int(x[4])

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)

          return  render_template("month.html",  texpense  =  texpense_list,
   expense=expense,  total=total,
                          t_food=t_food, t_entertainment=t_entertainment,
                          t_business=t_business,  t_rent=t_rent,
                          t_EMI=t_EMI,  t_other=t_other)

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

```
def year():
    ttotal = 0
    import datetime
    x = datetime.datetime.now()
    month = str(x).split()[0].split('-')
    y = month[0]

    sql = "SELECT * FROM expenses  WHERE email = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    expense_list = []
    texpense_list = []
    expense = ibm_db.fetch_tuple(stmt)
    if expense:
        date = expense[2].split('T')[0].split('-')
        date = date[0]
        ttotal += int(expense[4])
        texpense_list.append(expense)
        print(date)
        print(y)
        if date == y:
            expense_list.append(expense)
            print(expense)

        while expense != False:
            expense = ibm_db.fetch_tuple(stmt)

            if expense != False:
                ttotal += int(expense[4])
                texpense_list.append(expense)
                date = expense[2].split('T')[0].split('-')
                date = date[0]
                if date == y:
                    expense_list.append(expense)
                    print(expense)
    total = 0
    t_food = 0
    t_entertainment = 0
```

```
    t_business = 0
    t_rent = 0
    t_EMI = 0
    t_other = 0

    for x in expense_list:
        total += int(x[4])
        if x[6] == "food":
            t_food += int(x[4])

        elif x[6] == "entertainment":
            t_entertainment += int(x[4])

        elif x[6] == "business":
            t_business += int(x[4])

        elif x[6] == "rent":
            t_rent += int(x[4])

        elif x[6] == "EMI":
            t_EMI += int(x[4])

        elif x[6] == "other":
            t_other += int(x[4])

    print(total)

    print(t_food)
    print(t_entertainment)
    print(t_business)
    print(t_rent)
    print(t_EMI)
    print(t_other)

           return   render_template("year.html",   texpense   =   texpense_list,
    expense=expense,  total=total,
                            t_food=t_food, t_entertainment=t_entertainment,
                            t_business=t_business,  t_rent=t_rent,
                            t_EMI=t_EMI,  t_other=t_other)
```

```
#
# #log-out

@app.route('/logout')

def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    msg = 'Successfully logged out'
    return render_template('home.html', msg = msg)

if __name__ == "__main__":
    app.run(host="0.0.0.0",debug=True)
    # app.run()
```

7.2 Database Schema

Table definition

EXPENSES

No statistics available.

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
EMAIL	VARCHAR	Y	25	0
DATE	VARCHAR	Y	25	0
EXPENSENAME	VARCHAR	Y	25	0
AMOUNT	VARCHAR	Y	25	0
PAYMODE	VARCHAR	Y	25	0
CATEGORY	VARCHAR	Y	25	0

PNT20222TMID53089

Table definition

REGISTER

No statistics available.

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	Y	25	0
PASS	VARCHAR	Y	25	0

Table definition

LIMITS

No statistics available.

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	Y	25	0
LIMIT	VARCHAR	Y	25	0

8. TESTING

8.1 Test Cases

https://github.com/IBM-EPBL/IBM-Project-12977-1659504253/blob/main/Final%20Deliverables/Testcases%20Report.xlsx

8.2 User Acceptance Testing

https://github.com/IBM-EPBL/IBM-Project-12977-1659504253/blob/main/Final%20Deliverables/UAT%20Report.docx.pdf

9. ADVANTAGES & DISADVANTAGES

1. Achieve your business goals with a tailored mobile app that perfectly fits your business.
2. Scale up at the pace your business is growing.
3. Deliver an outstanding customer experience through additional control over the app.
4. Control the security of your business and customer data
5. Open direct marketing channels with no extra costs with methods such as push notifications.
6. Boost the productivity of all the processes within the organization.
7. Increase efficiency and customer satisfaction with an app aligned to their needs.
8. Seamlessly integrate with existing infrastructure.
9. Ability to provide valuable insights.
10. Optimize sales processes to generate more revenue through enhanced data collection.

10. CONCLUSION

From this project, we are able to manage and keep tracking the daily expenses as well as income. While making this project, we gained a lot of experience working as a team. We discovered various predicted and unpredictable problems and we enjoyed a lot solving them as a team. We adopted things like video tutorials, text tutorials, the internet and learning materials to make our project complete.

11. FUTURE SCOPE

The project assists well to record the income and expenses in general.

However, this project has some limitations:

1. The application is unable to maintain the backup of data once it is uninstalled.
2. This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following features to be incorporated into the system:

1. Multiple language interfaces.
2. Provide backup and recovery of data.
3. Mobile app advantage.

12. APPENDIX

Source Code GitHub Link:
IBM-EPBL/IBM-Project-12977-1659504253: Personal Expense Tracker Application (github.com)

Project Demo Link:

https://drive.google.com/file/d/1RMgBRLSegcULzcnHflGdjA63k3xnoSow/view?usp=sharing