<u>PERSONAL EXPENSE TRACKER APPLICATION</u>

<u>NALAIYA THIRAN PROJECT</u>

<u>BASED-LEARNING ON</u>

<u>PROFESSIONAL</u>

<u>READINESS FOR INNOVATION,</u>

<u>EMPLOYMENT AND ENTREPRENEURSHIP</u>

Team ID - PNT2022TMID21350

A PROJECT REPORT BY

Team Leader	-	Deepthi E
Team Member 1	-	Ajai saikumar A K
Team Member 2	-	Kanmani K
Team Member 3	-	Swetha S
Team Member 4	-	Imran Khan I

BACHELOR OF ENGINEERING IN ELECTRONICS AND COMMUNICATION ENGINEERING

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI -625015

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Project Overview

Personal expense tracker tackles all the financial decisions and activities and this Finance app makes your life easier by helping you to manage your finances efficiently and easily. A personal finance app will not only help you with money management.

Personal expense tracker 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

Finance management is an important part of people's lives. However, everyone does not have the time to manage their finances . And, 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 manage your finances. An expense tracker is a software or application that helps to keep an accurate record of your money . Many people have a fixed income, and they find it difficult to keep an account of it.. People tend to overspend, and this can affect their financial stability. Using an expense tracker, can help you keep track of how much you spend and on what. At the end of the month, you will have a clear picture of your expenses. This is an easy way to get your expenses under control.

2. LITERATURE SURVEY

2.1 Existing problem

Expense Manager Application

AUTHOR: Velmurugan, Richard Francis

MONTH AND YEAR PUBLISHED: December 2020

OBJECTIVE OF THE PROJECT:

Mobile applications are top in user convenience and have overpassed the web applications in terms of popularity and usability. There are various mobile applications that provide solutions to manage personal and group expense but not many of them provide a comprehensive view of both cases. In this paper, we develop a mobile application developed for the android platform that keeps record of user personal expenses, his/her contribution in group expenditures, top investment options, view of the current stock market, read authenticated financial news and grab the best ongoing offers in the market in popular categories. The proposed application would eliminate messy sticky notes, spreadsheets confusion and data handling inconsistency problems while offering the best overview of your expenses. With our application we can manage their expenses and decide on their budget more effectively.

Tracking personal finances

AUTHOR: Joseph Jofish , Rebecca

MONTH AND YEAR PUBLISHED: April 2014

OBJECTIVE OF THE PROJECT:

In this paper we present a preliminary scoping study of how 14 individuals in the San Francisco Bay Area earn, save, spend and understand money and their personal and family finances. We describe the practices we developed for exploring the sensitive topic of money, and then discuss three sets of findings. The first is the emotional component of the relationship people have with their finances. Second, we discuss the tools and processes people used to keep track of their financial situation. Finally we discuss how people account for the unknown and unpredictable nature of the future through their financial decisions. We conclude by discussing the future of studies of money and finance in HCI, and reflect on the opportunities for improving tools to aid people in managing and planning their finances.

Student Expense Tracking Application

AUTHOR : Saumya Dubey, Rigved Rishabh Kumar

MONTH AND YEAR PUBLISHED : April 2014

OBJECTIVE OF THE PROJECT :

This project is based on expense and income tracking system. This project aims to create an easier, faster and smooth tracking system between the money spend and the money earned. We are making an android application which is named as "STUDENT EXPENSE TRACKING APPLICATION". This is an android application which is used to track the daily expense of a student. So, for the better expense tracking system, we developed our project that will help the users a lot. Most of the student because of their busy schedule they find it difficult to calculate their expense and income that is the one reason they face money crisis, in this case daily expense tracker can help the student to tracking income-expense day to day and making life tension free and focus easily on their career.

Expense Tracker

AUTHOR : Praphulla S. Kherade, Raj S. Vilankar

MONTH AND YEAR PUBLISHED : September 2021

OBJECTIVE OF THE PROJECT :

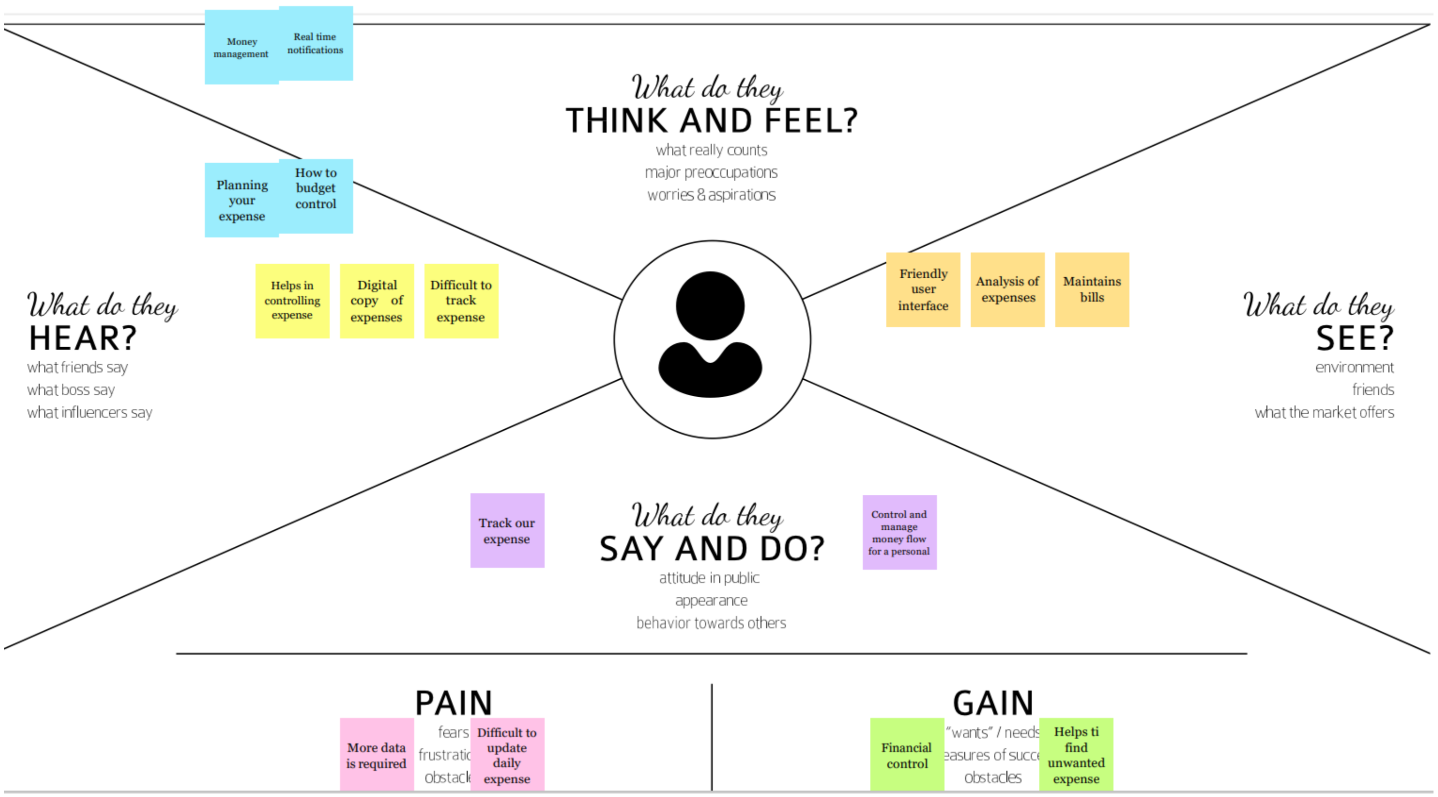
We are building an android application named as "Expense Tracker". As the name suggests, this project is an android app which is used to track the daily expenses of the user. It is like digital record keeping which keeps the records of expenses done by an user. The application keeps the track of the Income and Expenses both of user on a day-to-day basis. This application takes the income of an user and manage its daily expenses so that the user can save money. If you exceed daily expense allowed amount it will give you a warning, so that you don't spend much and that specific day. If you spend less money than the daily expense allowed amount, the money left after spending is added into user's savings. The application generates report of the expenses of each end of the month. The amount saved can be used for celebrating festivals, Birthdays or Anniversary.

2.2 Problem Statement Definition

It is a good habit for a person to record daily expenses and earning but due to unawareness and lack of proper applications to suit their privacy, lacking decision making capacity people are using traditional note keeping methods to do so. Due to lack of a complete tracking system, there is a constant overload to rely on the daily entry of the expenditure and total estimation till the end of the month.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

TOP IDEAS:

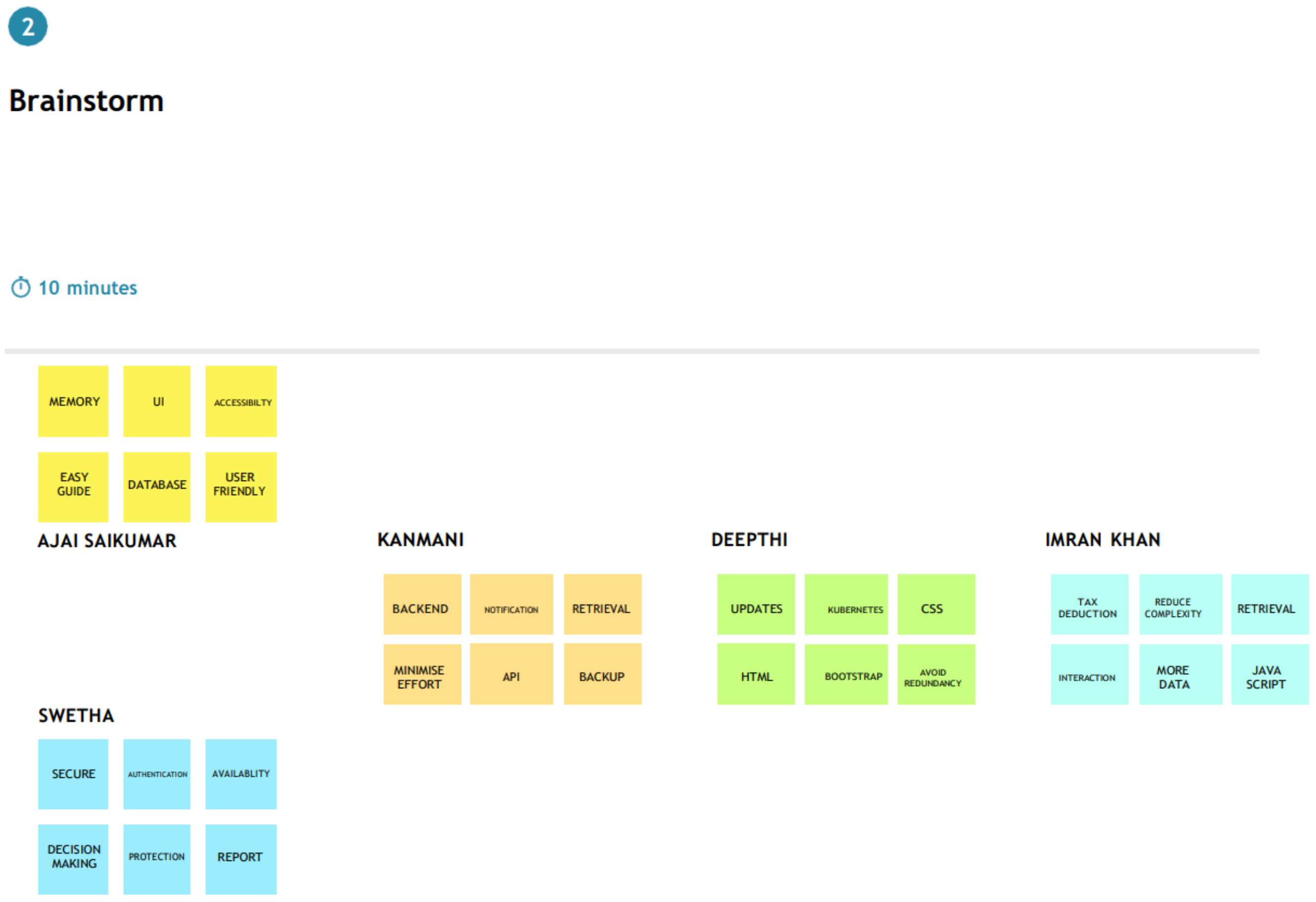
Our objective is to develop an expense-tracking system that would enable users to monitor all financial transactions and analyse past income and expense data.

1. Users get notification daily regarding their expenses. Users can add, update, delete records.

2. Users need to be aware of their finances. They must be reminded when they are running out of money.

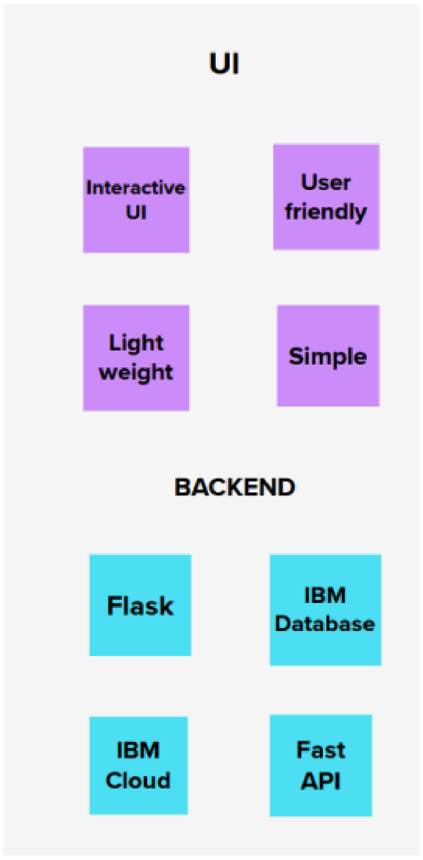
3. A reminderwhen finance crosses a limit must be sent to user.

4. The personal expense tracking app tracks the bills that occurs persistently and also the access to the finance report can be given to group of members in their family.



Brainstorm as a group

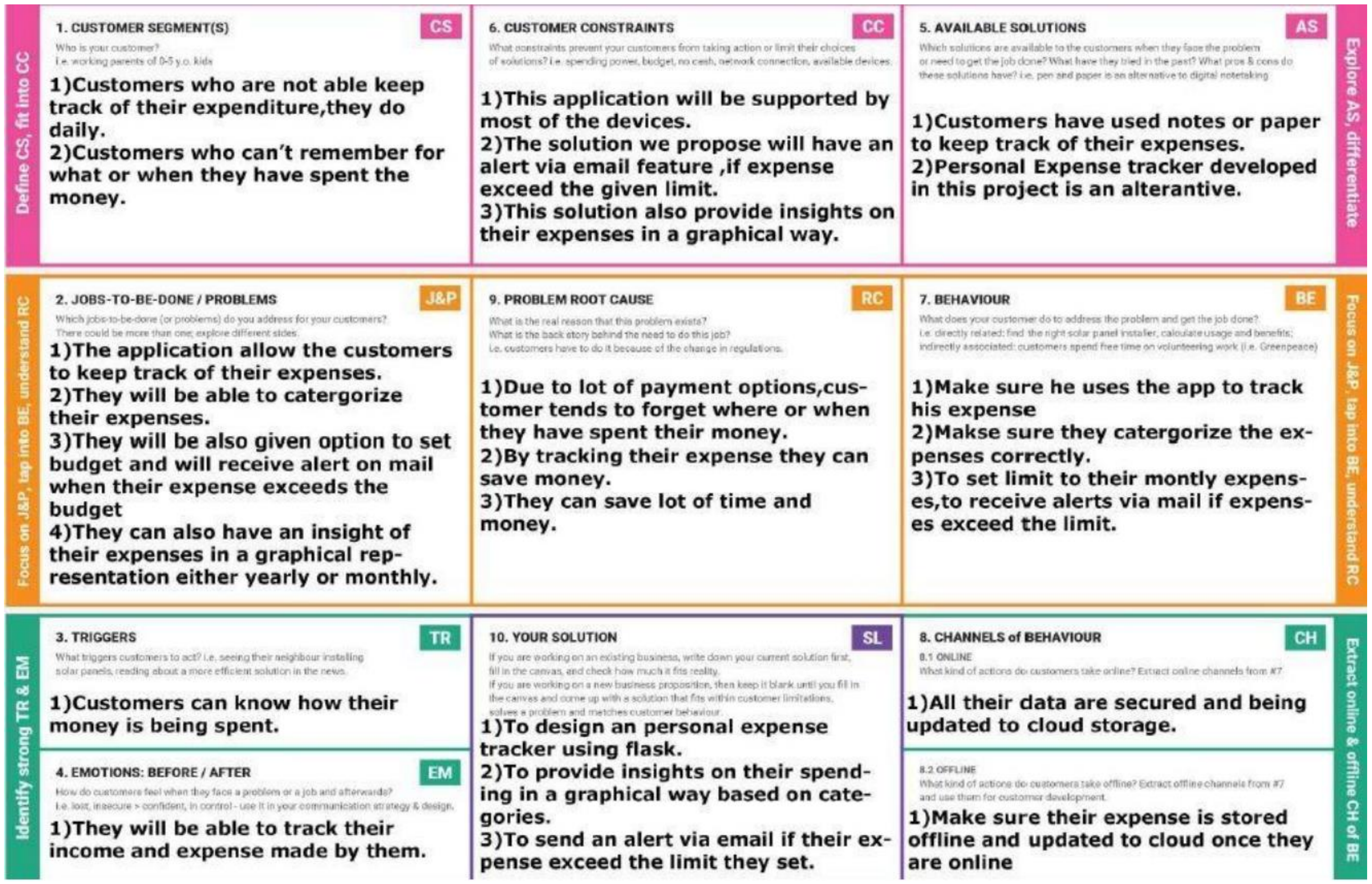
15 minutes



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement	An App to track your entire financial process and manage it.By assisting you in effectively managing your funds, this software makes your financial life easy . Personal Expense Tracker software will assist you with financial management, accounting, and budgeting.
2.	Idea	Personal Expense Trackersoftwarewill not only assist you with tracking the expense, but it will also provide you with valuable advice and control and budgeting.
3.	Novelty	Provide graphical representation on the expense.
4.	Customer Satisfaction	Customers can get to know about expenses and know what to do when they are low on account balance.
5.	Business Model	We can provide personalized advice on framing budgets.
6.	Scalability of the Solution	All Customers data can be stored in ibm cloud and can be retrieved when needed .

3.4 Problem Solution fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

1)Customers who are not able keep track of their expenditure,they do daily.
2)Customers who can't remember for what or when they have spent the money.

6. CUSTOMER CONSTRAINTS CC

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

1)This application will be supported by most of the devices.
2)The solution we propose will have an alert via email feature ,if expense exceed the given limit.
3)This solution also provide insights on their expenses in a graphical way.

5. AVAILABLE SOLUTIONS AS

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

1)Customers have used notes or paper to keep track of their expenses.
2)Personal Expense tracker developed in this project is an alterantive.

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.

1)The application allow the customers to keep track of their expenses.
2)They will be able to catergorize their expenses.
3)They will be also given option to set budget and will receive alert on mail when their expense exceeds the budget
4)They can also have an insight of their expenses in a graphical representation either yearly or monthly.

9. PROBLEM ROOT CAUSE RC

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

1)Due to lot of payment options,customer tends to forget where or when they have spent their money.
2)By tracking their expense they can save money.
3)They can save lot of time and money.

7. BEHAVIOUR BE

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

1)Make sure he uses the app to track his expense
2)Makse sure they catergorize the expenses correctly.
3)To set limit to their montly expenses,to receive alerts via mail if expenses exceed the limit.

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

Identify strong TR & EM

3. TRIGGERS TR

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

1)Customers can know how their money is being spent.

4. EMOTIONS: BEFORE / AFTER EM

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

1)They will be able to track their income and expense made by them.

10. YOUR SOLUTION SL

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

1)To design an personal expense tracker using flask.
2)To provide insights on their spending in a graphical way based on categories.
3)To send an alert via email if their expense exceed the limit they set.

8. CHANNELS of BEHAVIOUR CH

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

1)All their data are secured and being updated to cloud storage.

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

1)Make sure their expense is stored offline and updated to cloud once they are online

Extract online & offline CH of BE

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 is done through email.
FR-2	User Confirmation	Confirmation via Email with Otp.
FR-3	User Login	By entering username and password.
FR-4	Enter your expenses page.	Collects user's expenses with date and time
FR-5	Expenses Report is generated.	Represent all user's data in graphical form for easy understanding of report.
FR-6	Option to add categories and type of expense to the data	The app can organize expenses based on different categories.

4.2 Non - Functional requirements

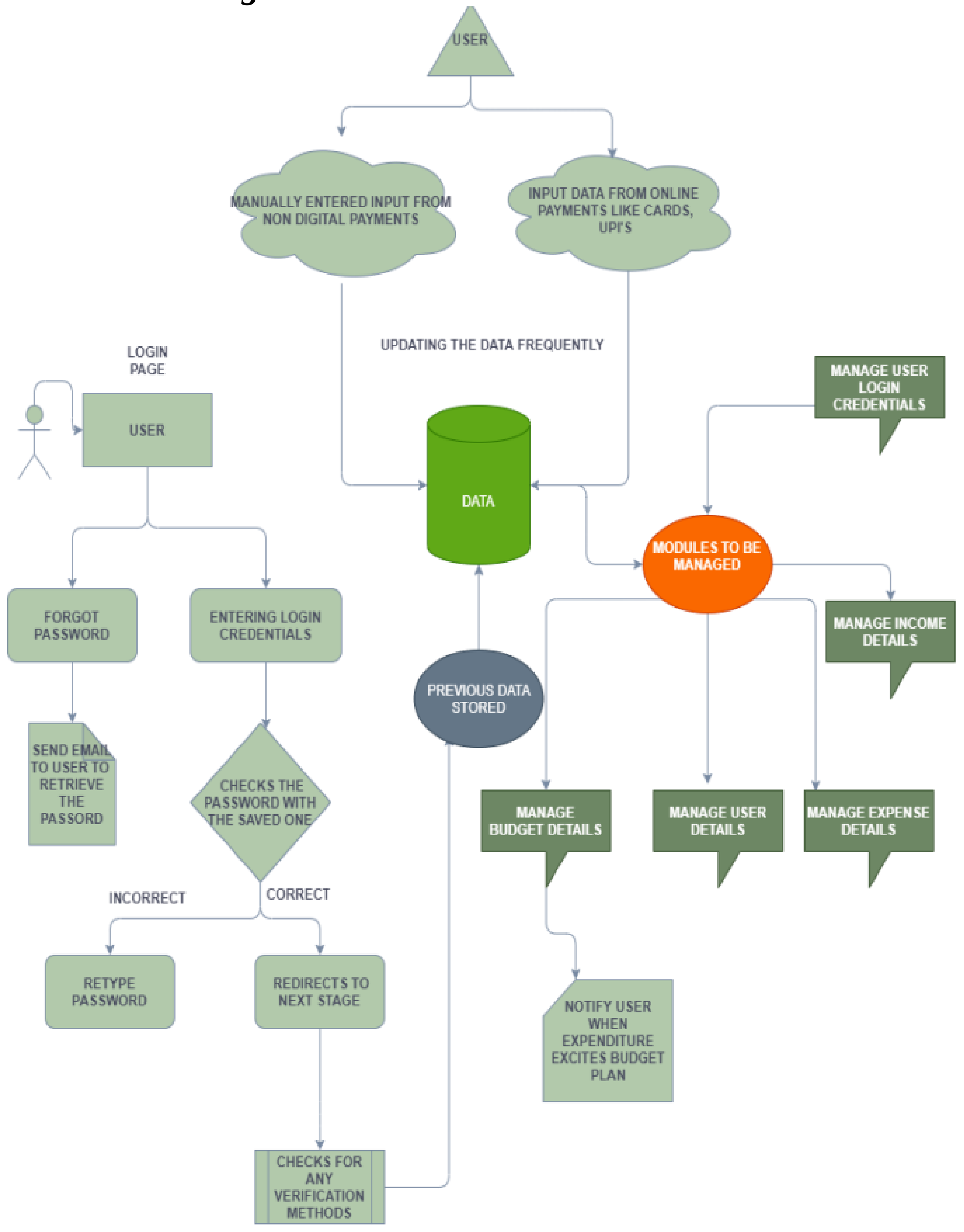
Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Provides an effective and user-friendly way to keep track of all the users expenses.
NFR-2	**Security**	Data is protected by giving a unique login ID and password.
NFR-3	**Reliability**	Since the app is hosted on the web it can be accessed anytime and from anywhere on all devices if you have a device with internet connectivity.
NFR-4	**Performance**	User data is stored in a very data efficient way using cloud which reduces load time of the application.

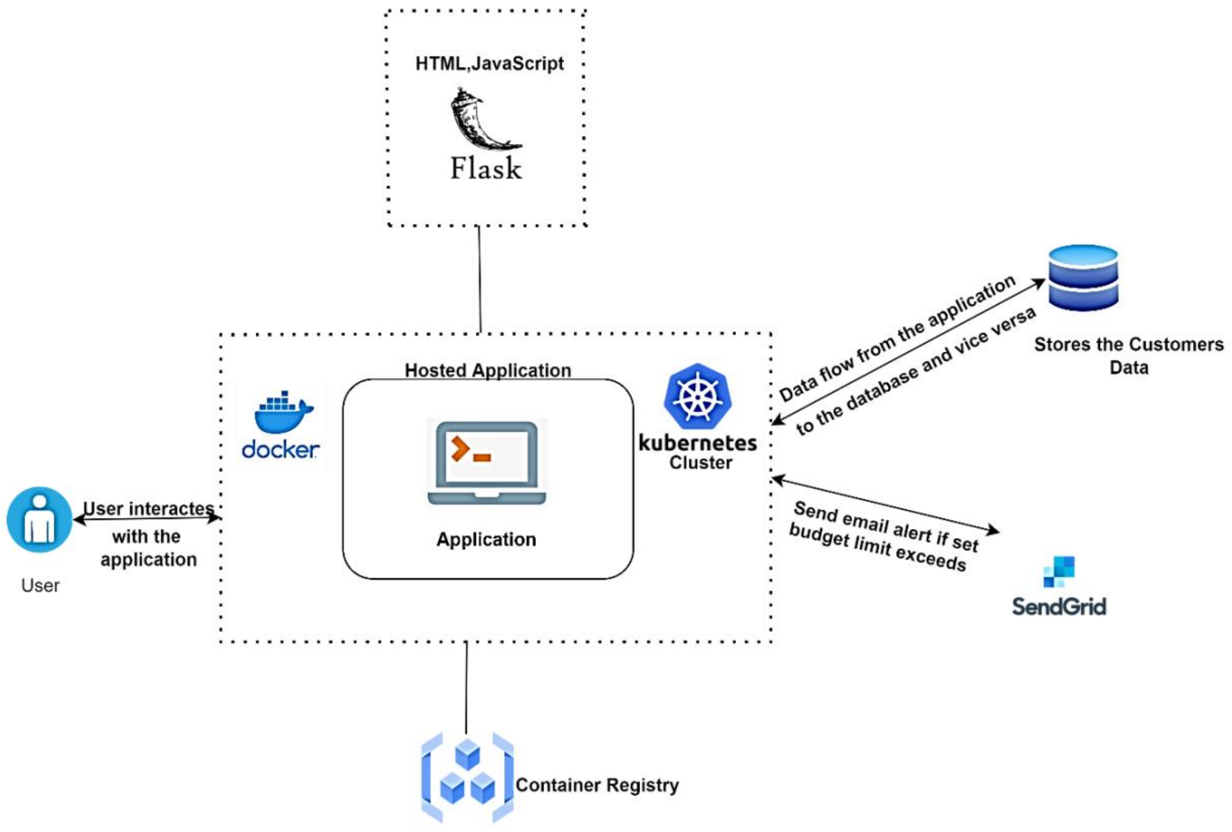
NFR-5	**Availability**	Application is hosted on the web and should be available 24/7 for the user.
NFR-6	**Scalability**	Can be scaled by increasing database size and better UI design to suit a larger audience as we are using cloud.

5.PROJECT DESIGN

4.3 Data Flow Diagrams


USER
MANUALLY ENTERED INPUT FROM NON DIGITAL PAYMENTS
INPUT DATA FROM ONLINE PAYMENTS LIKE CARDS, UPI'S
UPDATING THE DATA FREQUENTLY
LOGIN PAGE
USER
DATA
MANAGE USER LOGIN CREDENTIALS
MODULES TO BE MANAGED
FORGOT PASSWORD
ENTERING LOGIN CREDENTIALS
MANAGE INCOME DETAILS
PREVIOUS DATA STORED
SEND EMAIL TO USER TO RETRIEVE THE PASSORD
CHECKS THE PASSWORD WITH THE SAVED ONE
MANAGE BUDGET DETAILS
MANAGE USER DETAILS
MANAGE EXPENSE DETAILS
INCORRECT
CORRECT
RETYPE PASSWORD
REDIRECTS TO NEXT STAGE
NOTIFY USER WHEN EXPENDITURE EXCITES BUDGET PLAN
CHECKS FOR ANY VERIFICATION METHODS

4.4Solution & Technical Architecture


HTML,JavaScript
Flask
Hosted Application
docker
Application
kubernetes
Cluster
User interactes with the application
User
Data flow from the application to the database and vice versa
Stores the Customers Data
Send email alert if set budget limit exceeds
SendGrid
Container Registry

The above diagram represents the technology architecture of personal expense tracker application which is designed using html, css, javascript/Python along with flask framework since it is web based application. After designing the web application it being deployed to the docker and managed with kubenetes. The application interacts with the database for storing and retriving customer details and also stores cookies within the browser for improved application performance. The deployed application also has been integrated with the sendgrid services for sending mails to the user regrading their budget limit exceed.

5. PROJECT PLANNING & SCHEDULING

5.1 Sprint Planning & Estimation

Sprint	**Functional Requirement (Epic)**	**User Story Number** **User Story / Task**	**Story Points**	**Priority**	**Team Members**
	Registration	USN-1 As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Deepthi
		As a user, I can register for the application through the Gmail. USN-2	3	High	Kanmani
Sprint 1	Confirmation	USN-3 As a user, I will receive confirmation email once I have registered for the application.	4	High	Deepthi
	Login	As a user, I can log into the application by entering email & password. USN-4	3	High	Swetha
	Connecting to IBM DB2	USN-3 Linking database to store and verify login credentials of users.	3	High	Ajai Saikumar
	Dashboard Workspace	USN-5 Logging in takes to the dashboard. Dashboard is used to access all the services provided to the user. USN-1 Workspace for personal expense tracking.	`4 6	High High	Ajai saikumar Imran Khan
Sprint 2	Connecting to IBM DB2	USN-3 Linking database with dashboard.	5	High	Ajai Saikumar
		USN-4 Making dashboard interactive with JS.	5	High	Kanmani
Sprint-3		USN-1 Wrapping up the server side works of frontend.	5	Medium	Ajai Saikumar

	Watson Assistant	Creating Chatbot for expense tracking and for clarifying user's query. USN-2	4	Medium	Deepthi
	SendGrid	USN-3 Using SendGrid to send mail to the user about their expenses.	5	High	Kanmani
		USN-4 Integrating both frontend and backend.	6	High	Swetha
Sprint-4	Docker	USN-1 Creating image of website using docker.	5	High	Imran Khan
	Cloud Registry	USN-2 Uploading docker image to IBM Cloud registry.	5	High	Deepthi
	Kubernetes	USN-3 Create container using the docker image and hosting the site.	5	High	Swetha
	Exposing	USN-4 Exposing IP/Ports for the site.	5	High	Imran Khan

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

Sprint	**Total Story Points**	**Sprint End Date Duration Sprint Start Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	6 Days 26 Oct 2022 29 Oct 2022	20	1 Nov 2022
Sprint-2	20	6 Days 02 Nov 2022 05 Nov 2022	20	12 Nov 2022
Sprint-3	20	6 Days 09 Nov 2022 12 Nov 2022	20	14 Nov 2022
Sprint-4	20	6 Days 16 Nov 2022 19 Nov 2022	20	19 Nov 2022

Velocity:

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint). Calculating the team's average velocity (AV).

AV = **SprintDuration / Velocity = 20 / 6= 3.3**

6. CODING & SOLUTIONING

6.1 Source Code

initialiasing

```
import os

from flask import Flask, g, redirect, render_template, url_for
from expense_tracker.auth import login_required
from expense_tracker.db import get_db


def create_app(test_config=None):
    # create and configure the app
    app = Flask(__name__, instance_relative_config=True)
    app.config.from_mapping(
        SECRET_KEY='dev')

    if test_config is None:
        # load the instance config, if it exists, when not testing
        app.config.from_pyfile('config.py', silent=True)
    else:
        # load the test config if passed in
        app.config.from_mapping(test_config)

    UPLOAD_FOLDER = 'receipts'
    ALLOWED_EXTENSIONS = {'pdf', 'png', 'jpg', 'jpeg'}
    app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
    app.config['ALLOWED_EXTENSIONS'] = ALLOWED_EXTENSIONS
    app.config['MAX_CONTENT_LENGTH'] = 5 * 1024 * 1024  # 5mb

    # ensure the instance folder exists
    try:
        os.makedirs(app.instance_path)
        os.makedirs(UPLOAD_FOLDER)
    except OSError:
        pass

    from . import db
    db.init_app(app)
    from . import auth
    app.register_blueprint(auth.bp)
    from . import dashboard
    app.register_blueprint(dashboard.bp)
    app.add_url_rule('/', endpoint='index')
    from . import transaction
    app.register_blueprint(transaction.bp)

    # Index page
    @app.route('/')
    def index():
        if g.user:
            return redirect(url_for('dashboard.index'))
        return render_template('index.html')
    return app
import functools

from flask import (
    Blueprint, flash, g, redirect, render_template, request, session, url_for
)
from werkzeug.security import check_password_hash, generate_password_hash
import ibm_db
from expense_tracker.db import get_db

bp = Blueprint('auth', __name__, url_prefix='/auth')


@bp.route('/register', methods=('GET', 'POST'))
def register():
    if g.user:
        print(g.user)
        return redirect(url_for('dashboard.index'))
    if request.method == 'POST':
        name = request.form['name']
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        db = get_db()
        error = None

        if not name:
            error = 'Name is required.'
        if not username:
            error = 'Username is required.'
        if not email:
            error = 'Email is required.'
        elif not password:
            error = 'Password is required.'

        if error is None:
            insert = "INSERT INTO user (name, username, email,  password) VALUES (?,
?, ?, ?)"
            stmt = ibm_db.prepare(db, insert)
            ibm_db.bind_param(stmt, 1, name)
            ibm_db.bind_param(stmt, 2, username)
            ibm_db.bind_param(stmt, 3, email)
            ibm_db.bind_param(stmt, 4, generate_password_hash(password))
            try:
                ibm_db.execute(stmt)
            except:
                error = "Account with username or email already exist"
            else:
                return redirect(url_for("auth.login"))

        flash(error)

    return render_template('auth/register.html')


@bp.route('/login', methods=('GET', 'POST'))
def login():
    if g.user:
        return redirect(url_for('dashboard.index'))
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        db = get_db()
        error = None
        sql = "SELECT * FROM user WHERE username=?"
        stmt = ibm_db.prepare(db, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        user = ibm_db.fetch_assoc(stmt)

        if not user:
            error = 'Incorrect username or password'
        elif not check_password_hash(user['PASSWORD'], password):
            error = 'Incorrect username or password.'

        if error is None:
            session.clear()
            session['user_id'] = user['ID']
            return redirect(url_for('dashboard.index'))

        flash(error)

    return render_template('auth/login.html')


@bp.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('index'))


@bp.before_app_request
def load_logged_in_user():
    user_id = session.get('user_id')

    if user_id is None:
        g.user = None
    else:
        db = get_db()
        sql = "SELECT * FROM user WHERE id=?"
        stmt = ibm_db.prepare(db, sql)
        ibm_db.bind_param(stmt, 1, user_id)
        ibm_db.execute(stmt)
        g.user = ibm_db.fetch_assoc(stmt)


def login_required(view):
    @functools.wraps(view)
    def wrapped_view(**kwargs):
        if g.user is None:
            return redirect(url_for('auth.login'))

        return view(**kwargs)
    return wrapped_view
```

dashboard:

```
from flask import Blueprint, render_template, g
import ibm_db
import ibm_db_dbi
from datetime import datetime
from expense_tracker.auth import login_required
from expense_tracker.db import get_db
from expense_tracker.expense import get_current_month_expense

bp = Blueprint('dashboard', __name__, url_prefix='/dashboard')


@bp.route('/')
@login_required
def index():
    current_month_expense = get_current_month_expense()
    return render_template('dashboard/index.html',
current_month_expense=current_month_expense)
```

database:

```
import click
import ibm_db
from flask import current_app, g


def get_db():
    if 'db' not in g:
        g.db = ibm_db.connect(current_app.config["CONN_STR"], "", "")

    return g.db


def close_db(e=None):
    db = g.pop('db', None)

    if db is not None:
        ibm_db.close(db)
```

```
def init_db():
    db = get_db()
    ibm_db.exec_immediate(db, 'DROP TABLE IF EXISTS user')
    ibm_db.exec_immediate(db, 'DROP TABLE IF EXISTS transaction')
    ibm_db.exec_immediate(
        db, """CREATE TABLE user (id INT NOT NULL GENERATED ALWAYS AS IDENTITY, name
varchar(255) NOT NULL, username varchar(255) NOT NULL UNIQUE, email varchar(255) NOT
NULL UNIQUE, password varchar(255) NOT NULL, limit DECIMAL NOT NULL DEFAULT 0, PRIMARY
KEY (id))""")
    ibm_db.exec_immediate(
        db, "CREATE TABLE transaction (id INT NOT NULL GENERATED ALWAYS AS IDENTITY,
user_id INT NOT NULL, category varchar(255) NOT NULL, description varchar(1000) NOT
NULL, amount DECIMAL NOT NULL, date date NOT NULL, receipt varchar(255) NOT NULL,
PRIMARY KEY (id))")
    ibm_db.exec_immediate(
        db, "ALTER TABLE transaction ADD CONSTRAINT Expense_fk0 FOREIGN KEY (user_id)
REFERENCES User(id)")


@click.command('init-db')
def init_db_command():
    """Clear the existing data and create new tables."""
    init_db()
    click.echo('Initialized the database.')


def init_app(app):
    app.teardown_appcontext(close_db)
    app.cli.add_command(init_db_command)
```

expense:

```
import ibm_db
from flask import g
from datetime import datetime
from expense_tracker.db import get_db


def get_current_month_expense():
    id = g.user['ID']
    db = get_db()
    month = datetime.today().month
    year = datetime.today().year
    query = f'SELECT SUM(transaction.amount) FROM transaction INNER JOIN user on 
transaction.user_id=user.id WHERE MONTH(transaction.date)={month} and 
YEAR(transaction.date)={year} and user.id={id}'
    stmt = ibm_db.prepare(db, query)
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)['1']
    return 0 if result is None else int(result)
```

mail service:

```
from mailjet_rest import Client
from flask import current_app, g


def send_email_alert():

    mail_api_key = current_app.config['MAIL_API_KEY']
    mail_api_secret = current_app.config['MAIL_API_SECRET']
    mail_admin = current_app.config['ADMIN_MAIL']
    mailjet = Client(auth=(mail_api_key, mail_api_secret), version='v3.1')
    name = g.user["NAME"]
    email = g.user["EMAIL"]
    data = {
        'Messages': [
            {
                "From": {
                    "Email": mail_admin,
                    "Name": "Admin"
                },
                "To": [
                    {
                        "Email": email,
                        "Name": name
                    }
                ],
                "Subject": "Expense Alert",
                "HTMLPart": f"<h3>Dear {name},<br/> your expenses have crossed the 
monthly expense limit set by you.",
            }
        ]
    }
    result = mailjet.send.create(data=data)
    print(result)
```

object storage:

```
import ibm_boto3
from ibm_botocore.client import Config, ClientError
import uuid
from flask import current_app


def get_uuid():
    return str(uuid.uuid4().hex)


def multi_part_upload(file_path):
    COS_ENDPOINT = current_app.config['COS_ENDPOINT']
    COS_API_KEY_ID = current_app.config['COS_API_KEY_ID']
    COS_INSTANCE_CRN = current_app.config['COS_INSTANCE_CRN']
    COS_BUCKET_NAME = current_app.config['COS_BUCKET_NAME']
    # Create resource
    cos = ibm_boto3.resource("s3",
                             ibm_api_key_id=COS_API_KEY_ID,
                             ibm_service_instance_id=COS_INSTANCE_CRN,
                             config=Config(signature_version="oauth"),
                             endpoint_url=COS_ENDPOINT
                             )
    item_name = get_uuid() + '.' + get_extension(file_path)
    try:
        print("Starting file transfer for {0} to bucket: {1}\n".format(
            item_name, COS_BUCKET_NAME))
        # set 5 MB chunks
        part_size = 1024 * 1024 * 5

        # set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15

        # set the transfer threshold and chunk size
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
        )

        # the upload_fileobj method will automatically execute a multi-part upload
        # in 5 MB chunks for all files over 15 MB
        with open(file_path, "rb") as file_data:
            cos.Object(COS_BUCKET_NAME, item_name).upload_fileobj(
                Fileobj=file_data,
                Config=transfer_config
            )

        print("Transfer for {0} Complete!\n".format(item_name))
        return item_name  # send object name
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
        return 'none'  # no file
    except Exception as e:
        print("Unable to complete multi-part upload: {0}".format(e))
        return 'none'  # no file


def get_extension(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1]


def get_signed_url(receipt):

    bucket_name = current_app.config['COS_BUCKET_NAME']
    key_name = receipt
    http_method = 'get_object'
    expiration = 60  # time in seconds, default:600

    access_key = current_app.config['COS_HMAC_ACCESS_KEY']
    secret_key = current_app.config['COS_HMAC_SECRET_KEY']
    # Current list avaiable at https://control.cloud-object-
storage.cloud.ibm.com/v2/endpoints
    cos_service_endpoint = current_app.config['COS_ENDPOINT']

    cos = ibm_boto3.client("s3",
                           aws_access_key_id=access_key,
                           aws_secret_access_key=secret_key,
                           endpoint_url=cos_service_endpoint
                           )

    signedUrl = cos.generate_presigned_url(http_method, Params={
        'Bucket': bucket_name, 'Key': key_name}, ExpiresIn=expiration)
    return signedUrl
```

transaction:

```
from flask import Blueprint, current_app, flash, g, render_template, request, 
redirect, url_for
from werkzeug.utils import secure_filename
import ibm_db
import ibm_db_dbi
import random
import os
from expense_tracker.auth import login_required
from expense_tracker.db import get_db
from expense_tracker.storage import multi_part_upload, get_signed_url
from expense_tracker.expense import get_current_month_expense
from expense_tracker.mailjet import send_email_alert
from datetime import datetime

bp = Blueprint('transaction', __name__, url_prefix='/transaction')


@bp.route('/')
@login_required
def index():
    db = get_db()
    user_id = g.user["ID"]
    query = f"SELECT transaction.amount, transaction.description, 
transaction.category, transaction.date, transaction.receipt FROM transaction INNER 
JOIN user ON transaction.user_id=user.id where user.id={user_id} ORDER BY 
transaction.id DESC"
    conn = ibm_db_dbi.Connection(db)
    curr = conn.cursor()
    curr.execute(query)
    transactions = curr.fetchall()
    return render_template('transaction/index.html', transactions=transactions)


@bp.route('/receipt/<receipt>')
def get_receipt(receipt):
    db = get_db()
    user_id = g.user["ID"]
    query = "SELECT * FROM transaction WHERE user_id=? AND receipt=?"
    stmt = ibm_db.prepare(db, query)
    ibm_db.bind_param(stmt, 1, user_id)
    ibm_db.bind_param(stmt, 2, receipt)
    try:
        ibm_db.execute(stmt)
    except:
        flash("Some error occurred")
    else:
        transaction = ibm_db.fetch_assoc(stmt)
        if not transaction:
            flash("No such receipt exists")
        else:
            return redirect(get_signed_url(receipt))
    return redirect(url_for('transaction.index'))


@bp.route('/all')
@login_required
def all():
    db = get_db()
    user_id = g.user["ID"]
    month = datetime.today().month
    year = datetime.today().year
    query = f"SELECT transaction.amount, transaction.description, transaction.category
FROM transaction INNER JOIN user ON transaction.user_id=user.id WHERE
MONTH(transaction.date)={month} AND YEAR(transaction.date)={year} AND
user.id={user_id}"
    conn = ibm_db_dbi.Connection(db)
    curr = conn.cursor()
    curr.execute(query)
    transactions = curr.fetchall()
    return transactions


@bp.route('/add', methods=["GET", "POST"])
@login_required
def add():
    if request.method == "POST":
        db = get_db()
        user_id = g.user["ID"]
        category = request.form["category"]
        description = request.form["description"]
        amount = request.form["amount"]
        date = request.form["date"]
        file = request.files['receipt']
        uuid = 'none'
        if not file.filename == '' and file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file_path = os.path.join(
                current_app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)
            uuid = multi_part_upload(file_path)
        current_expense = get_current_month_expense()
        insert = "INSERT INTO transaction (user_id, category, amount, description,
date, receipt) VALUES (?, ?, ?, ?, ?, ?)"
        stmt = ibm_db.prepare(db, insert)
        ibm_db.bind_param(stmt, 1, user_id)
        ibm_db.bind_param(stmt, 2, category)
        ibm_db.bind_param(stmt, 3, amount)
        ibm_db.bind_param(stmt, 4, description)
        ibm_db.bind_param(stmt, 5, date)
        ibm_db.bind_param(stmt, 6, uuid)
        error = None
        try:
            ibm_db.execute(stmt)
        except:
            error = "Error occurred while adding transaction"
        else:
            if current_expense < int(g.user["LIMIT"]) and current_expense +
int(amount) > int(g.user["LIMIT"]):
                flash('Alert expense exceeds set limit')
                send_email_alert()
            return redirect(url_for('transaction.index'))
        flash(error)
    return render_template('transaction/add.html')


@bp.route('/limit', methods=["GET", "POST"])
@login_required
def limit():
    if request.method == "POST":
        db = get_db()
        id = g.user["ID"]
        limit = request.form["limit"]
        insert = f"UPDATE USER SET limit=? WHERE ID={id}"
        stmt = ibm_db.prepare(db, insert)
        ibm_db.bind_param(stmt, 1, limit)
        error = None
        try:
            ibm_db.execute(stmt)
        except:
            error = "Error occurred while setting limit"
        else:
            flash("Limit updated successfully")
            return redirect(url_for('dashboard.index'))
        flash(error)

    return render_template('transaction/limit.html')


def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower(
           ) in current_app.config['ALLOWED_EXTENSIONS']
```

6.2 Database Schema

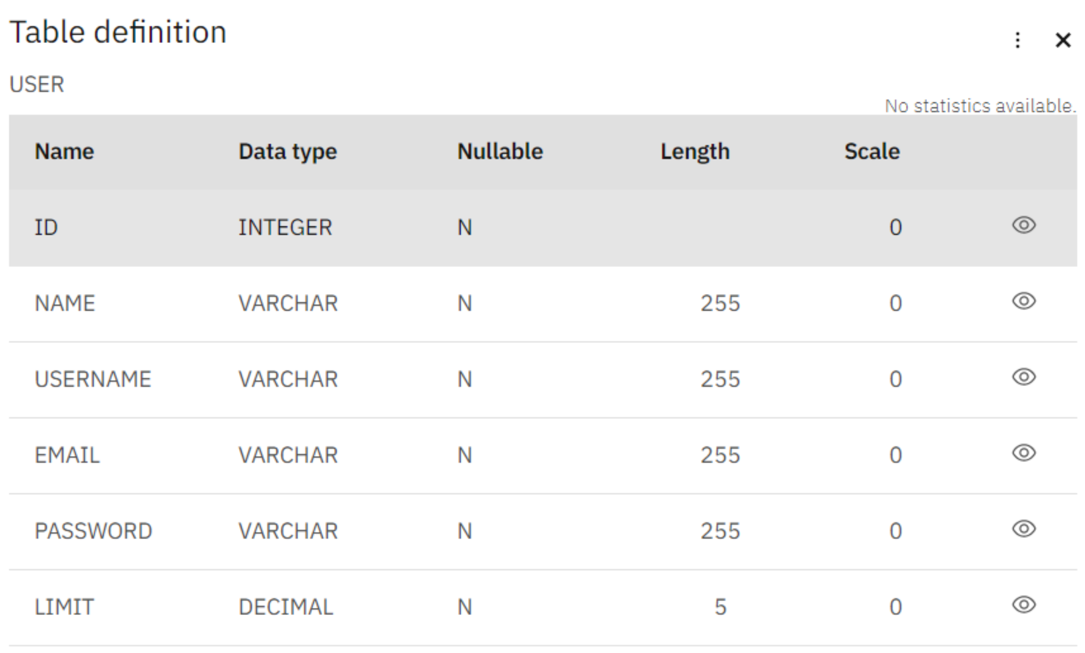


Table definition

USER

No statistics available.

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
NAME	VARCHAR	N	255	0
USERNAME	VARCHAR	N	255	0
EMAIL	VARCHAR	N	255	0
PASSWORD	VARCHAR	N	255	0
LIMIT	DECIMAL	N	5	0

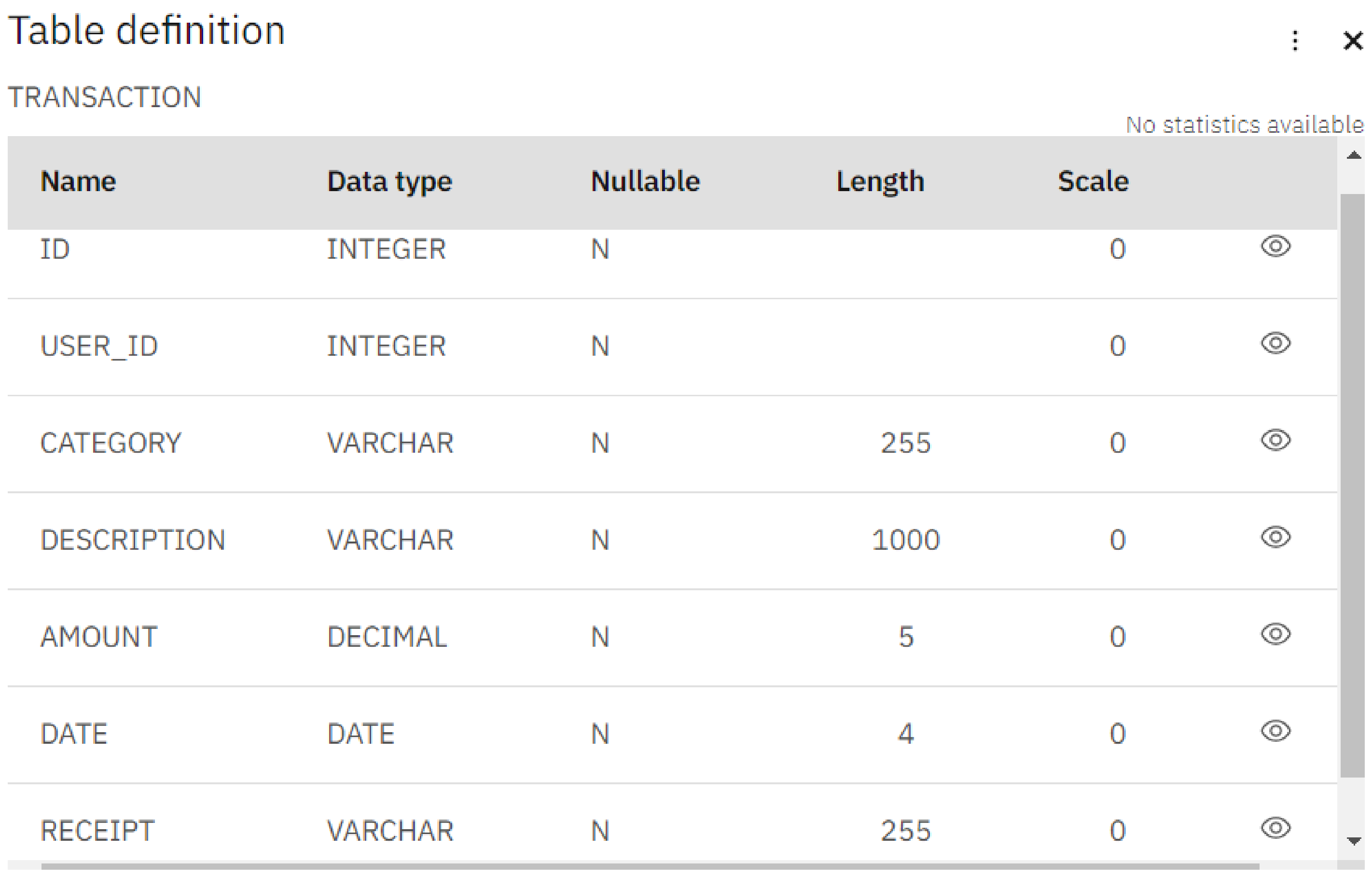


Table definition

TRANSACTION

No statistics available

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
USER_ID	INTEGER	N		0
CATEGORY	VARCHAR	N	255	0
DESCRIPTION	VARCHAR	N	1000	0
AMOUNT	DECIMAL	N	5	0
DATE	DATE	N	4	0
RECEIPT	VARCHAR	N	255	0

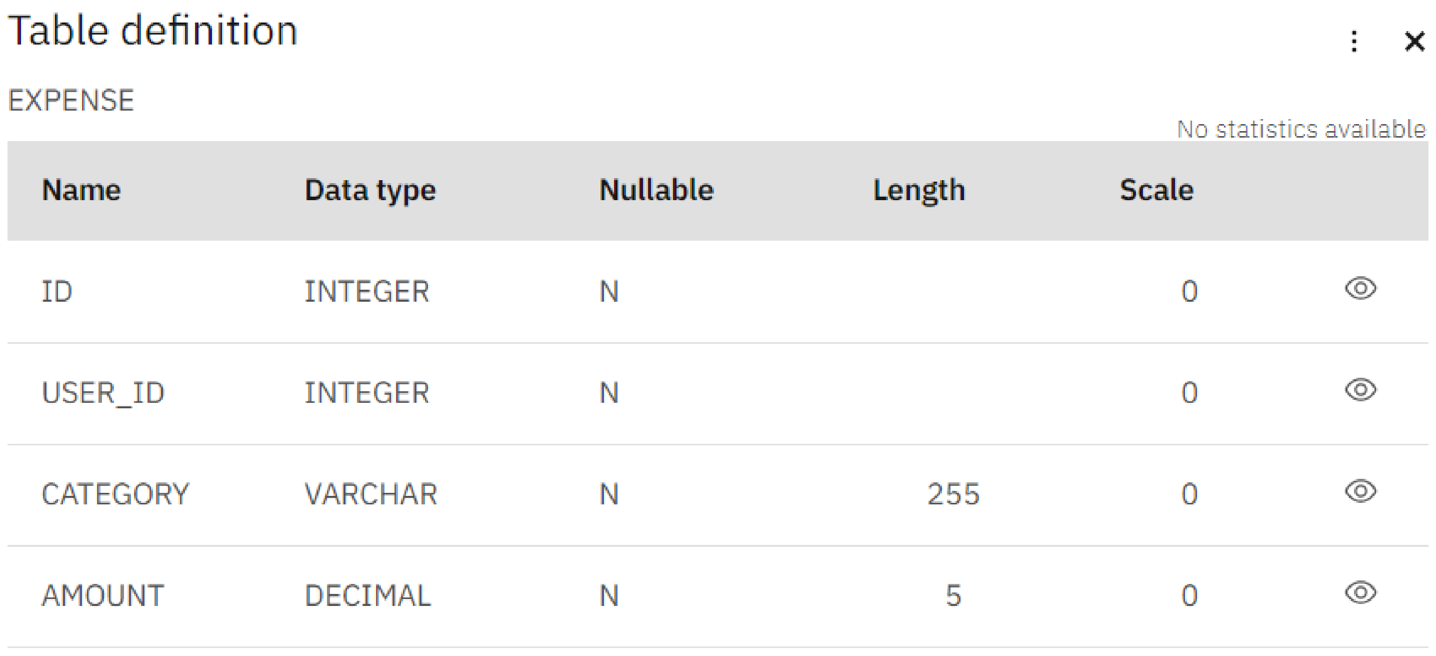


Table definition

EXPENSE

No statistics available

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
USER_ID	INTEGER	N		0
CATEGORY	VARCHAR	N	255	0
AMOUNT	DECIMAL	N	5	0

7. TESTING

7.1 Test Cases

https://github.com/IBM-EPBL/IBM-Project-34866-1660278554/blob/main/Final%20Deliverables/Testcases%20Report.xlsx

7.2 User Acceptance Testing

https://github.com/IBM-EPBL/IBM-Project-34866-1660278554/blob/main/Final%20Deliverables/UAT%20Report-2.pdf

8. 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:

https://github.com/IBM-EPBL/IBM-Project-34866-1660278554

Project Demo Link:

https://youtu.be/2IVWPGGMC6U