A PROJECT REPORT

CONTENTS

PERSONAL EXPENSE TRACKER

Team ID	PNT2022TMID01701
Project Name	Project – Personal Expense Tracker Application
Team Members	DHANISH D DHINESH KUMAR G ASHWIN S GUGHAN A N P

1. <u>INTRODUCTION</u>

1.1 Project overview

The popularity and usability of mobile applications have surpassed those of web applications in terms of user convenience. Many mobile applications offer ways to handle personal and group expenses, but few of them give a thorough overview of both situations. In this paper, we develop a mobile application for the Android platform that records a user's individual expenses, his or her share of group expenses, top investment options, a view of the stock market at the time, reads authenticated financial news, and allows users to take advantage of the best offers currently available in the market in popular categories. The suggested application would provide the best overview of your expenses while getting rid of messy sticky notes, spreadsheet confusion, and inconsistent data handling issues. With our application can manage their expenses and decide on their budget more effectively.

1.2 Purpose

An expense tracker, also referred to as an expense manager or money manager, is a piece of software or an application that aids in maintaining accurate records of your money coming in and going out. In India, a large number of people rely on fixed incomes and discover that they run out of money toward the end of the month.

2. <u>LITERATURE SURVEY</u>

2.1 Existing problem

The issue with the present generation is that they can't recall where all of the money they earned has gone, so they are forced to make do with the little money they have left over to meet their basic necessities. There is currently no perfect solution that enables a person to quickly and effectively manage their daily expenses and alert them to their financial situation. To achieve this, they must keep extensive ledgers or computer records for the data, and the user must perform the computation manually, which might result in errors and losses. having insufficient tracking

2.2 Reference

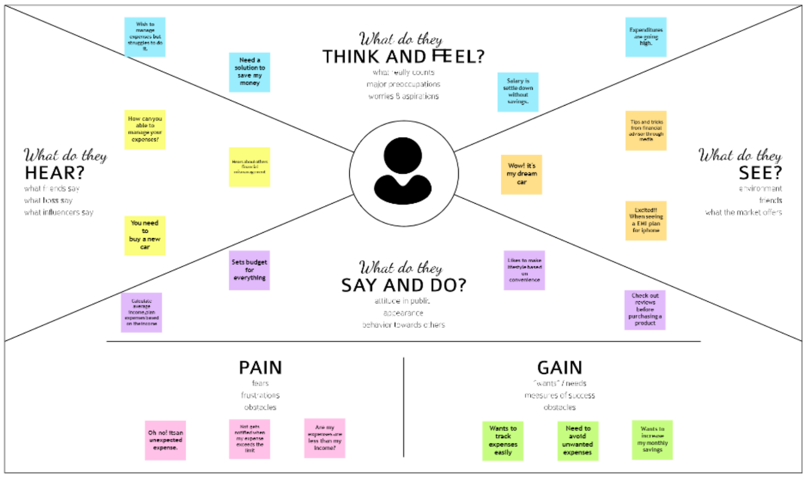
- https://nevonprojects.com/daily-expense-tracker-system/
- https://data-flair.training/blogs/expense-tracker-python/
- https://phpgurukul.com/daily-expense-tracker-using-php-and-mysql/
- https://ijarsct.co.in/Paper391.pdf
- https://kandi.openweaver.com/?landingpage=python_all_projects&utm_source=google&utm_medium=cpc&utm_campaign=promo_kandi_ie&utm_content=kandi_ie_search&utm_term=python_devs&gclid=Cj0KCQiAgribBhDkARIsAASA5bukrZgbI9UZxzpoyf0P-ofB1mZNxzc-okUP-3TchpYMclHTYFYiqP8aAmmwEALw_wcB

2.3 Problem Statement Definition

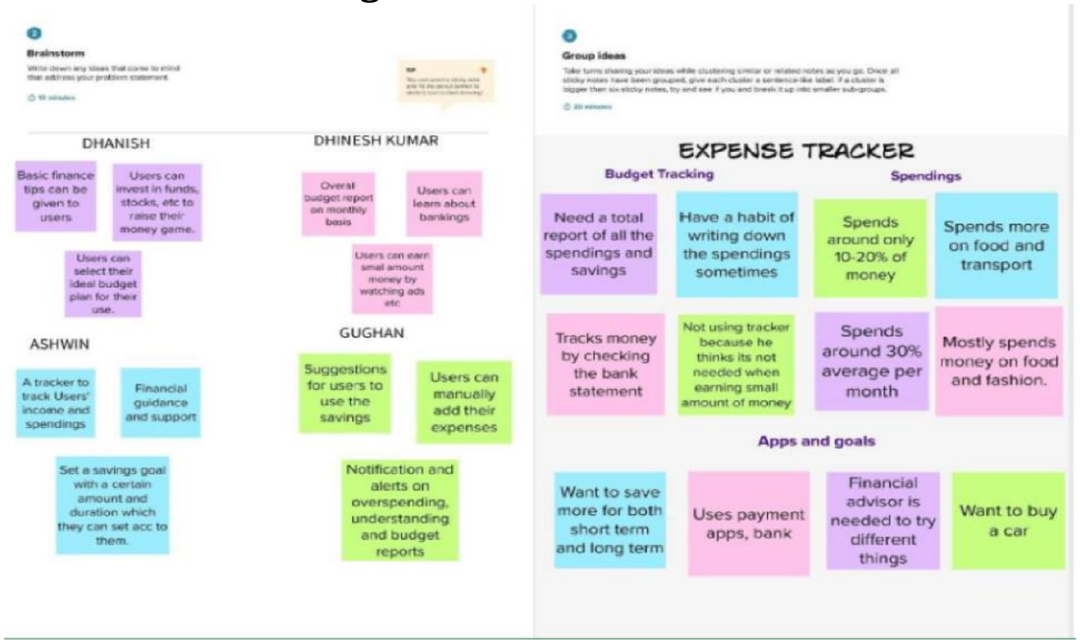
A online programme called "Expense Tracker" enables users to monitor and keep track of both personal and professional costs. The users of this programme can keep digital diaries. It will record a user's earnings and outgoings each day. With the aid of the internet, the user will be able to rapidly input his or her expenses and may check them whenever and wherever. Without putting his or her information at danger and effectively safeguarding his or her privacy, he or she may quickly import transactions from his or her mobile wallets. Files are frequently lost or mistakenly deleted. This spending tracker offers a comprehensive digital fix for this issue. Excel spreadsheets are not particularly helpful for tracking Additionally, they lack the sophisticated capability of automatically creating graphical graphics. Not only would it save individuals time, but it will also guarantee accurate computations. The user only needs to enter their income and expenses; the system will handle the rest. Keywords: Expense Tracker, spending visualisation in graphics, budget, planning, and savings.

3. <u>IDEATION & POPOSED SOLUTION</u>

3.1 Empathy Map canvas


What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Wish to manage expenses but struggles to do it.
Need a solution to save my money
Expenditures are going high.
Salary is settle down without savings.
What do they HEAR?
what friends say
what boss say
what influencers say
How can you able to manage your expenses?
Hears about others financial mismanagement
You need to buy a new car
What do they SEE?
environment
friends
what the market offers
Tips and tricks from financial advisor through media
Wow! it's my dream car
Excited!! When seeing a EMI plan for iphone
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Sets budget for everything
Calculate average income,plan expenses based on the income
Likes to make lifestyle based on convenience
Check out reviews before purchasing a product
PAIN
fears
frustrations
obstacles
Oh no! Its an unexpected expense.
Not gets notified when my expense exceeds the limit
Are my expenses are less than my income?
GAIN
"wants" / needs
measures of success
obstacles
Wants to track expenses easily
Need to avoid unwanted expenses
Wants to increase my monthly savings

3.2 Ideation & Brainstorming


Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
DHANISH
Basic finance tips can be given to users
Users can invest in funds, stocks, etc to raise their money game.
Users can select their ideal budget plan for their use.
DHINESH KUMAR
Overall budget report on monthly basis
Users can learn about bankings
Users can earn small amount money by watching ads etc
ASHWIN
A tracker to track Users' income and spendings
Financial guidance and support
Set a savings goal with a certain amount and duration which they can set acc to them.
GUGHAN
Suggestions for users to use the savings
Users can manually add their expenses
Notification and alerts on overspending, understanding and budget reports
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
EXPENSE TRACKER
Budget Tracking
Spendings
Need a total report of all the spendings and savings
Have a habit of writing down the spendings sometimes
Spends around only 10-20% of money
Spends more on food and transport
Tracks money by checking the bank statement
Not using tracker because he thinks its not needed when earning small amount of money
Spends around 30% average per month
Mostly spends money on food and fashion.
Apps and goals
Want to save more for both short term and long term
Uses payment apps, bank
Financial advisor is needed to try different things
Want to buy a car

3.3 Proposed Solution

To control and monitor their spending patterns, everyone in the earning sector needs a mechanism to manage their financial resources and keep track of their expenses. This helps students comprehend the value of money management and helps them make wiser selections moving forward. They can choose to establish a cap on how much can be used in that month, and if the cap is surpassed, the user will receive an email notice. The people who get monthly payments can watch their payments and avoid unnecessary spending as a solution to this issue. The user will receive an email notice if the limit is exceeded.

3.4 Proposed Solution Fit

The folks who get monthly payments can watch their payments and prevent unforeseen costs as a solution to this issue. The user will receive an email notice if the limit is exceeded.
Originality/Uniqueness Email can be used to get notifications.

• Social Responsibility / Client Satisfaction One may keep track of their own costs and create a monthly or annual budget with this tool. The programme will display a warning notice if your expenses are higher than the allowed amount. This will have an effect on how satisfied customers are with mobile banking.

• Business Plan The application's subscription/premium features can be used by business people to generate income.

• Ability to Scale the Solution Security, the ability of the programme to function even when the network is down, etc. are all factors that affect an application's capacity to scale.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

- FR-1 User Registration ,Registration through Form Registration through Gmail Registration through LinkedIN
- FR-2 User Confirmation ,Confirmation via Email Confirmation via OTP
- FR-3 Tracking Expense Helpful insights about money management
- FR-4 Alert Message Give alert mail if the amount exceeds the budget limit
- FR-5 Category This application shall allow users to add categories of their expenses

4.2 Non Functional requirement

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

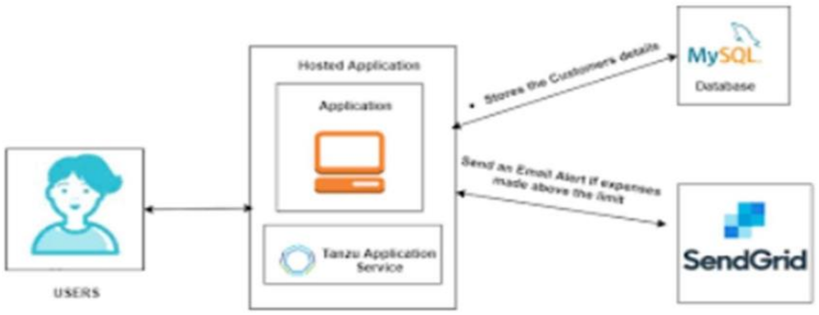
- NFR-1 Usability You will able to allocate money to different priorities and also help you to cut down on unnecessary spending
- NFR-2 Security More security of the customer data and bank account details.
- NFR-3 Reliability Used to manage his/her expense so that the user is the path of financial stability. It is categorized by week, month, and year and also helps to see more expenses made. Helps to define their own categories.
- NFR-4 Performance The types of expense are categories along with an option .Throughput of the system is increased due to light weight database support.
- NFR-5 Availability Able to track business expense and monitor important for maintaining healthy cash flow. NFR-6 Scalability The ability to appropriately handle increasing demands.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

The classic visual depiction of how information moves through a system is a data flow diagram (DFD). A tidy and understandable DFD may visually represent the appropriate quantity of the system demand. It demonstrates how information enters and exits the system, what modifies the information, and where information is kept.

5.2 Solution &Technical Architecture


Hosted Application
Application
Tanzu Application Service
USERS
Stores the Customers details
MySQL
Database
Send an Email Alert if expenses made above the limit
SendGrid

5.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story/ Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password,	I can access my account / dashboard	High	Sprint 1

			and confirming my password.			
		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 by entering the details	Medium	Sprint 1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my dashboard	High	Sprint 1
	Dashboard	USN-6	As a user ,I can log into the dashboard and manage income	I can access my account / dashboard	High	Sprint 1
Customer (Web user)		USN-7	As a user, I can register for the application by Bank account.	I can access my account / dashboard	High	Sprint 1
Customer Care Executive		USN-8	As a user, I can get a report is	I can manage my money by	Medium	Sprint 1

			based on the details	viewing this report		
		USN-9	As a user, I can get an email if the money level is above the limit	I can receive alert email	High	Sprint 1
Administrative	Responsibility	USN-10	As a system administrator ,track the user expenses anytime	I can track expense	High	Sprint 1

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 user ,I can register for the application by entering my email, password ,and confirming my password.	2	High	DHANISH D
Sprint-1		USN-2	As a user ,I will receive confirmation email once I have registered for the application	1	Medium	ASHWIN S
Sprint-	Login	USN-3	As a user ,I can register for the	2	Medium	DHINESH KUMAR G

2			application through Facebook			
Sprint-1	Dashboard	USN-4	As a user, I can register for the application through Gmail	2	High	GUGHAN A N P

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed(as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6Days	24Oct2022	29Oct2022	20	29Oct2022
Sprint-2	20	6Days	31Oct2022	05Nov2022	18	06Nov2022
Sprint-3	20	6Days	07Nov2022	12Nov2022	15	14Nov2022
Sprint-4	20	6Days	14Nov2022	19Nov2022	19	21Nov2022

7. Coding And Solutioning:

7.1. Features

Feature 1: Add Expense

Feature 2: Update expense

Feature 3: Delete Expense

Feature 4: Set Limit

Feature 5: Send Alert Emails to users

7.2. Other Features:

Track your expenses anywhere, anytime. Seamlessly manage your money and budget without any financial paperwork. Just click and submit your invoices and expenditures. Access, submit, and approve invoices irrespective of time and location. Avoid data loss by scanning your tickets and bills and saving in the app. Approval of bills and expenditures in real-time and get notified instantly. Quick settlement of claims and reduced human errors with an automated and streamlined billing process.

Codes:

App. Py:

```
from flask import Flask, render_template, request, flash, redirect, url_for, session
from wtforms import Form, StringField, PasswordField, IntegerField, validators
from wtforms.validators import DataRequired
from passlib.hash import sha256_crypt
from functools import wraps
import timeago
import datetime
from wtforms.fields.html5 import EmailField
from itsdangerous import TimedJSONWebSignatureSerializer as Serializer
from flask_mail import Mail, Message
import plotly.graph_objects as go
import db
from mail import send_mail

app = Flask(__name__, static_url_path='/static')
app.config.from_pyfile('config.py')


@app.route('/')
def index():
    return render_template('index.html')


class SignUpForm(Form):
    first_name = StringField('First Name', [validators.Length(min=1, max=100)])
    last_name = StringField('Last Name', [validators.Length(min=1, max=100)])
    email = EmailField('Email address', [validators.DataRequired(), validators.Email()])
    username = StringField('Username', [validators.Length(min=4, max=100)])
    password = PasswordField('Password', [validators.DataRequired(),
validators.EqualTo('confirm', message='Passwords do not match')])
    confirm = PasswordField('Confirm Password')
    monthly_limit = IntegerField('Monthly Limit', validators=[DataRequired()])


@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if 'logged_in' in session and session['logged_in'] == True:
        flash('You are already logged in', 'info')
        return redirect(url_for('addTransactions'))
    form = SignUpForm(request.form)
    if request.method == 'POST' and form.validate():
        first_name = form.first_name.data
        last_name = form.last_name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.hash(str(form.password.data))
        monthly_limit = int(form.monthly_limit.data) if form.monthly_limit.data else 2000
        result = db.select('SELECT * FROM "users" WHERE "email"=? or "username" = ', [email,
username])
        if (result) and (len(result) > 0):
            flash('The entered email/username address has already been taken. Please try using
or creating another one.', 'info')
            return redirect(url_for('signup'))
        else:
            db.insert('INSERT INTO "users"("first_name", "last_name", "email", "username",
"password", "monthly_limit") VALUES(?, ?, ?, ?, ?, ?)',
                (first_name, last_name, email, username, password, monthly_limit)
            )
            flash('You are now registered and can log in', 'success')
            return redirect(url_for('login'))
    return render_template('signUp.html', form=form)


class LoginForm(Form):
    username = StringField('Username', [validators.Length(min=4, max=100)])
    password = PasswordField('Password', [
        validators.DataRequired(),
    ])


@app.route('/login', methods=['GET', 'POST'])
def login():
    if 'logged_in' in session and session['logged_in'] == True:
        flash('You are already logged in', 'info')
        return redirect(url_for('addTransactions'))
    form = LoginForm(request.form)
    if request.method == 'POST' and form.validate():
        username = form.username.data
        password_input = form.password.data
        result = db.select('SELECT * FROM "users" WHERE "username" = ?', [username])
        if (result) and (len(result) > 0):
            data = result[0]
            userID = data['id']
            password = data['password']
            role = data['role']
            email = data['email']
            monthly_limit = data['monthly_limit'] if data['monthly_limit'] else 2000
            if sha256_crypt.verify(password_input, password):
                session['logged_in'] = True
```

```python
                session['username'] = username
                session['email'] = email
                session['role'] = role
                session['userID'] = userID
                session['monthly_limit'] = monthly_limit
                flash('You are now logged in', 'success')
                return redirect(url_for('transactionHistory'))
            else:
                error = 'Invalid Password'
                return render_template('login.html', form=form, error=error)
        else:
            error = 'Username not found'
            return render_template('login.html', form=form, error=error)
    return render_template('login.html', form=form)


def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args, **kwargs)
        else:
            flash('Please login', 'info')
            return redirect(url_for('login'))
    return wrap


@app.route('/logout')
@is_logged_in
def logout():
    session.clear()
    flash('You are now logged out', 'success')
    return redirect(url_for('login'))

# Add Transactions
@app.route('/addTransactions', methods=['GET', 'POST'])
@is_logged_in
def addTransactions():
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = MONTH(CURRENT DATE) AND YEAR("date") = YEAR(CURRENT DATE) AND "user_id" = ?',
[session['userID']])
    data = result[0]
    totalExpenses = int(data['AMOUNT_SUM']) if data['AMOUNT_SUM'] else 0
    if request.method == 'POST':
        amount = request.form['amount']
        description = request.form['description']
        category = request.form['category']
        db.insert('INSERT INTO "transactions"("user_id", "amount", "description","category")
VALUES(?, ?, ?, ?)', (session['userID'], amount, description, category))
        monthly_limit = int(session['monthly_limit']) if session['monthly_limit'] else 2000
        if totalExpenses > monthly_limit:
            user_email = session['email']
            msg_title = "Monthly Expense limit excedded"
            msg_content = f"Hi,<br>Your expense for this month has excedded {monthly_limit}"
            send_mail(user_email, msg_title, msg_content)
        flash('Transaction Successfully Recorded', 'success')
        return redirect(url_for('addTransactions'))
    else:
        # get the month's transactions made by a particular user
        result = db.select('SELECT * FROM "transactions" WHERE MONTH("date") = MONTH(CURRENT
DATE) AND YEAR("date") = YEAR(CURRENT DATE) AND "user_id" = ? ORDER BY "date" DESC',
[session['userID']])
        if (result) and (len(result) > 0):
            transactions = result
            for transaction in transactions:
                if datetime.datetime.now() - transaction['date'] < datetime.timedelta(days=0.5):
                    transaction['date'] = timeago.format(transaction['date'],
datetime.datetime.now())
                else:
                    transaction['date'] = transaction['date'].strftime('%d %B, %Y')

            return render_template('addTransactions.html', totalExpenses=totalExpenses,
transactions=transactions)
        else:

            return render_template('addTransactions.html', result=result)


@app.route('/transactionHistory', methods=['GET', 'POST'])
@is_logged_in
def transactionHistory():
    if request.method == 'POST':
        month = request.form['month']
        year = request.form['year']
        result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
"user_id" = ?', ([session['userID']]))
        data = result[0]
        totalExpenses = data['AMOUNT_SUM']
        if month == "00":
            result = db.select(f'SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
YEAR("date") = YEAR("{year}-00-00") AND "user_id" = {session["userID"]}')
            data = result[0]
            totalExpenses = data['AMOUNT_SUM']
            result = db.select(f'SELECT * FROM "transactions" WHERE YEAR("date") = YEAR("{year}-
00-00") AND "user_id" = {session["userID"]} ORDER BY "date" DESC')
        else:
            result = db.select(f'SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = MONTH("0000-{month}-00") AND YEAR("date") = YEAR("{year}-00-00") AND "user_id" =
{session["userID"]}')
            data = result[0]
            totalExpenses = data['AMOUNT_SUM']
            result = db.select(f'SELECT * FROM "transactions" WHERE MONTH("date") = MONTH("0000-
{month}-00") AND YEAR("date") = YEAR("{year}-00-00") AND "user_id" = {session["userID"]} ORDER
BY "date" DESC')

        if (result) and (len(result) > 0):
            transactions = result
            for transaction in transactions:
                transaction['date'] = transaction['date'].strftime('%d %B, %Y')
            return render_template('transactionHistory.html', totalExpenses=totalExpenses,
transactions=transactions)
        else:
            result = db.select(f"SELECT MONTHNAME('0000-{month}-00')")
            data = result[0]
            if month != "00":
                monthName = data[f'MONTHNAME(\'0000-{month}-00\')']
                msg = f"No Transactions Found For {monthName}, {year}"
            else:
                msg = f"No Transactions Found For {year}"
            return render_template('transactionHistory.html', result=result, msg=msg)


    else:
        result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
"user_id" = ?', [session['userID']])
        data = result[0]
        totalExpenses = data['AMOUNT_SUM']
        # Get Latest Transactions made by a particular user
        result = db.select('SELECT * FROM "transactions" WHERE "user_id" = ? ORDER BY "date"
DESC', [session['userID']])
        if (result) and (len(result) > 0):
            transactions = result
            for transaction in transactions:
                transaction['date'] = transaction['date'].strftime('%d %B, %Y')
            return render_template('transactionHistory.html', totalExpenses=totalExpenses,
transactions=transactions)
        else:
            flash('No Transactions Found', 'success')
            return redirect(url_for('addTransactions'))



class TransactionForm(Form):
    amount = IntegerField('Amount', validators=[DataRequired()])
    description = StringField('Description', [validators.Length(min=1)])

# Edit transaction
@app.route('/editTransaction/<string:id>', methods=['GET', 'POST'])
@is_logged_in
def editTransaction(id):
    result = db.select('SELECT * FROM "transactions" WHERE "id" = ?', [id])
    transaction = result[0]
    form = TransactionForm(request.form)
    form.amount.data = transaction['amount']
    form.description.data = transaction['description']
    if request.method == 'POST' and form.validate():
        amount = request.form['amount']
        description = request.form['description']
        db.update('UPDATE "transactions" SET "amount"=?, "description"=? WHERE "id" = ?',
(amount, description, id))
        flash('Transaction Updated', 'success')
        return redirect(url_for('transactionHistory'))
    return render_template('editTransaction.html', form=form)

# Delete transaction
@app.route('/deleteTransaction/<string:id>', methods=['POST'])
@is_logged_in
def deleteTransaction(id):
    db.delete('DELETE FROM "transactions" WHERE "id" = ?', [id])
    flash('Transaction Deleted', 'success')
    return redirect(url_for('transactionHistory'))


@app.route('/editCurrentMonthTransaction/<string:id>', methods=['GET', 'POST'])
@is_logged_in
def editCurrentMonthTransaction(id):
    transaction = db.select('SELECT * FROM "transactions" WHERE "id" = ?', [id])
    form = TransactionForm(request.form)
    form.amount.data = transaction['amount']
    form.description.data = transaction['description']
    if request.method == 'POST' and form.validate():
        amount = request.form['amount']
        description = request.form['description']
        db.update('UPDATE "transactions" SET "amount"=?, "description"=? WHERE "id" = ?',
                  (amount, description, id))

        flash('Transaction Updated', 'success')
        return redirect(url_for('addTransactions'))
    return render_template('editTransaction.html', form=form)

# Delete transaction
@app.route('/deleteCurrentMonthTransaction/<string:id>', methods=['POST'])
@is_logged_in
def deleteCurrentMonthTransaction(id):
    db.delete('DELETE FROM "transactions" WHERE "id" = ?', [id])
    flash('Transaction Deleted', 'success')
    return redirect(url_for('addTransactions'))


class RequestResetForm(Form):
    email = EmailField('Email address', [
                       validators.DataRequired(), validators.Email()])


@app.route("/reset_request", methods=['GET', 'POST'])
def reset_request():
    if 'logged_in' in session and session['logged_in'] == True:
        flash('You are already logged in', 'info')
        return redirect(url_for('index'))
    form = RequestResetForm(request.form)
    if request.method == 'POST' and form.validate():
        email = form.email.data
        result = db.select(
            "SELECT id,username, email FROM users WHERE email = ?", [email])
        if not result or len(result) == 0:
            flash(
                'There is no account with that email. You must register first.', 'warning')
            return redirect(url_for('signup'))
        else:
            data = result[0]
            user_id = data['id']
            user_email = data['email']
            s = Serializer(app.config['SECRET_KEY'], 1800)
            token = s.dumps({'user_id': user_id}).decode('utf-8')
            msg_title = 'Password Reset Request'
            msg_body = f'''To reset your password, visit the following link:
{url_for('reset_token', token=token, _external=True)}
If you did not make password reset request then simply ignore this email and no changes will be
made.
Note:This link is valid only for 30 mins from the time you requested a password change request.
'''
            send_mail(user_email, msg_title, msg_body)
            flash('An email has been sent with instructions to reset your password.', 'info')
            return redirect(url_for('login'))
    return render_template('reset_request.html', form=form)


class ResetPasswordForm(Form):
    password = PasswordField('Password', [
        validators.DataRequired(),
        validators.EqualTo('confirm', message='Passwords do not match')
    ])
    confirm = PasswordField('Confirm Password')


@app.route("/reset_password/<token>", methods=['GET', 'POST'])
def reset_token(token):
    if 'logged_in' in session and session['logged_in'] == True:
        flash('You are already logged in', 'info')
        return redirect(url_for('index'))
    s = Serializer(app.config['SECRET_KEY'])
    try:
        user_id = s.loads(token)['user_id']
    except:
        flash('That is an invalid or expired token', 'warning')
        return redirect(url_for('reset_request'))
    data = db.select('SELECT "id" FROM "users" WHERE "id" = ?', [user_id])
    data = data[0]
    user_id = data['id']
    form = ResetPasswordForm(request.form)
    if request.method == 'POST' and form.validate():
        password = sha256_crypt.hash(str(form.password.data))
        db.update('UPDATE "users" SET "password" = ? WHERE "id" = ?', (password, user_id))
        flash('Your password has been updated! You are now able to log in', 'success')
        return redirect(url_for('login'))
    return render_template('reset_token.html', title='Reset Password', form=form)

# Category Wise Pie Chart For Current Year As Percentage #
@app.route('/category')
def createBarCharts():
    result = db.select(f'SELECT SUM("amount") as AMOUNT_SUM, "category" FROM "transactions"
WHERE YEAR("date") = YEAR(CURRENT DATE) AND "user_id" = {session["userID"]} GROUP BY "category"
ORDER BY "category"')
    if (result) and (len(result) > 0):
        transactions = result
        values = []
        labels = []
        for transaction in transactions:
            values.append(transaction['AMOUNT_SUM'])
            labels.append(transaction['category'])
        fig = go.Figure(data=[go.Pie(labels=labels, values=values)])
        fig.update_traces(textinfo='label+value', hoverinfo='percent')
        fig.update_layout(title_text='Category Wise Pie Chart For Current Year')
        fig.show()
    return redirect(url_for('addTransactions'))

# Comparison Between Current and Previous Year #
@app.route('/yearly_bar')
def yearlyBar():
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('01',
session['userID']))
    if (result) and (len(result) > 0):
        a1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('01',
session['userID']))
    if (result) and (len(result) > 0):
        a2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('02',
session['userID']))
    if (result) and (len(result) > 0):
        b1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('02',
session['userID']))
    if (result) and (len(result) > 0):
        b2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('03',
session['userID']))
    if (result) and (len(result) > 0):
        c1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('03',
session['userID']))
    if (result) and (len(result) > 0):
        c2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('04',
session['userID']))
    if (result) and (len(result) > 0):
        d1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('04',
session['userID']))
    if (result) and (len(result) > 0):
        d2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('05',
session['userID']))
    if (result) and (len(result) > 0):
        e1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('05',
session['userID']))
    if (result) and (len(result) > 0):
        e2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('06',
session['userID']))
    if (result) and (len(result) > 0):
        f1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('06',
session['userID']))
    if (result) and (len(result) > 0):
        f2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('07',
session['userID']))
    if (result) and (len(result) > 0):
        g1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('07',
session['userID']))
    if (result) and (len(result) > 0):
        g2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('08',
session['userID']))
    if (result) and (len(result) > 0):
        h1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('08',
session['userID']))
    if (result) and (len(result) > 0):
        h2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('09',
session['userID']))
    if (result) and (len(result) > 0):
        i1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('09',
session['userID']))
    if (result) and (len(result) > 0):
        i2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('10',
session['userID']))
    if (result) and (len(result) > 0):
        j1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('10',
session['userID']))
    if (result) and (len(result) > 0):
        j2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('11',
session['userID']))
    if (result) and (len(result) > 0):
        k1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('11',
session['userID']))
    if (result) and (len(result) > 0):
        k2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
```

```python
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE 
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ('12', 
session['userID']))
    if (result) and (len(result) > 0):
        l1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE 
MONTH("date") = ?  AND YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ('12', 
session['userID']))
    if (result) and (len(result) > 0):
        l2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" 
WHERE  YEAR("date") = YEAR(CURRENT DATE)  AND "user_id" = ?', ([session['userID']]))
    if (result) and (len(result) > 0):
        m1 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0
    result = db.select('SELECT SUM("amount") as AMOUNT_SUM FROM "transactions" WHERE 
YEAR("date") = YEAR(CURRENT DATE - 1 YEAR)  AND "user_id" = ?', ([session['userID']]))
    if (result) and (len(result) > 0):
        m2 = result[0]['AMOUNT_SUM'] if result[0]['AMOUNT_SUM'] else 0

    year = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June',
            'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec', 'Total']
    fig = go.Figure(data=[
        go.Bar(name='Last Year', x=year, y=[
               a2, b2, c2, d2, e2, f2, g2, h2, i2, j2, k2, l2, m2]),
        go.Bar(name='This Year', x=year, y=[
               a1, b1, c1, d1, e1, f1, g1, h1, i1, j1, k1, l1, m1])
    ])
    fig.update_layout(
        barmode='group', title_text='Comparison Between This Year and Last Year')
    fig.show()
    return redirect(url_for('addTransactions'))

# Current Year Month Wise #
@app.route('/monthly_bar')
def monthlyBar():
    result = db.select(f'SELECT SUM("amount") as "AMOUNT", MONTH("date") as "MONTH" FROM 
"transactions" WHERE YEAR("date") = YEAR(CURRENT DATE) AND "user_id" = ? GROUP BY MONTH("date") 
ORDER BY MONTH("date")', ([session["userID"]]))
    if (result) and (len(result) > 0):
        transactions = result
        year = []
        value = []
        for transaction in transactions:
            year.append(transaction['MONTH'])
            value.append(transaction['AMOUNT'])
        fig = go.Figure([go.Bar(x=year, y=value)])
        fig.update_layout(title_text='Monthly Bar Chart For Current Year')
        fig.show()

    return redirect(url_for('addTransactions'))
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=app.config.get("FLASK_HTTP_PORT"))
```

mail.py:

```
from config import MAIL_DEFAULT_SENDER, MAIL_API_KEY
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

sg = SendGridAPIClient(api_key = MAIL_API_KEY)

def send_mail(to_email, msg_title, msg_content):
    message = Mail(
        from_email = MAIL_DEFAULT_SENDER,
        to_emails = to_email,
        subject = msg_title,
        html_content = msg_content
    )
    sg.send(message)
```

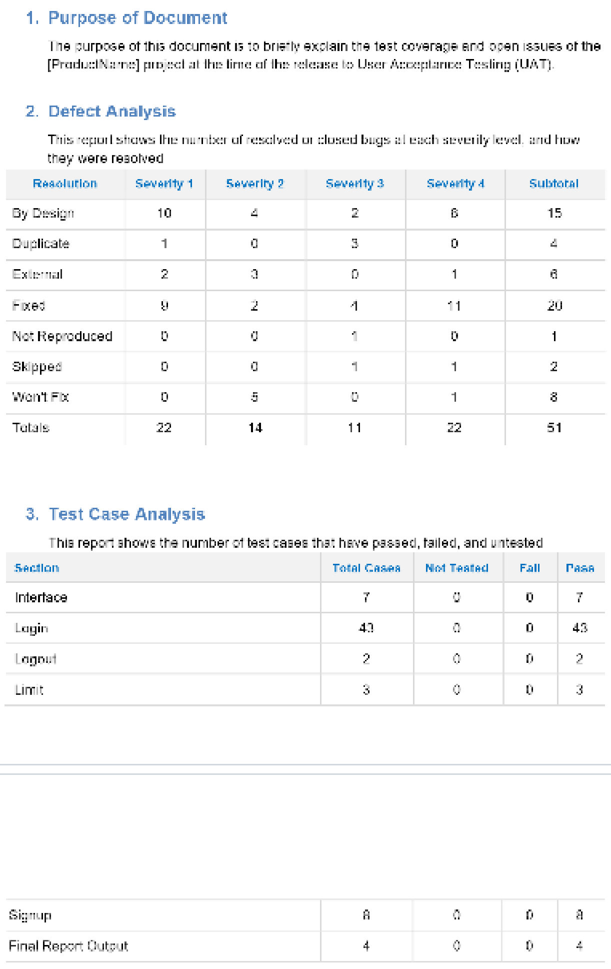
The other code features are submitted in github.

8. TESTING:

8.1. TESTING:

- Login Page (Funcional)
- Login Page (UI)
- Add Expense Page (Functional)

8.2. User Acceptance Testing:



1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2. 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	10	4	2	6	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	11	20
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	0	1	8
Totals	22	14	11	22	51

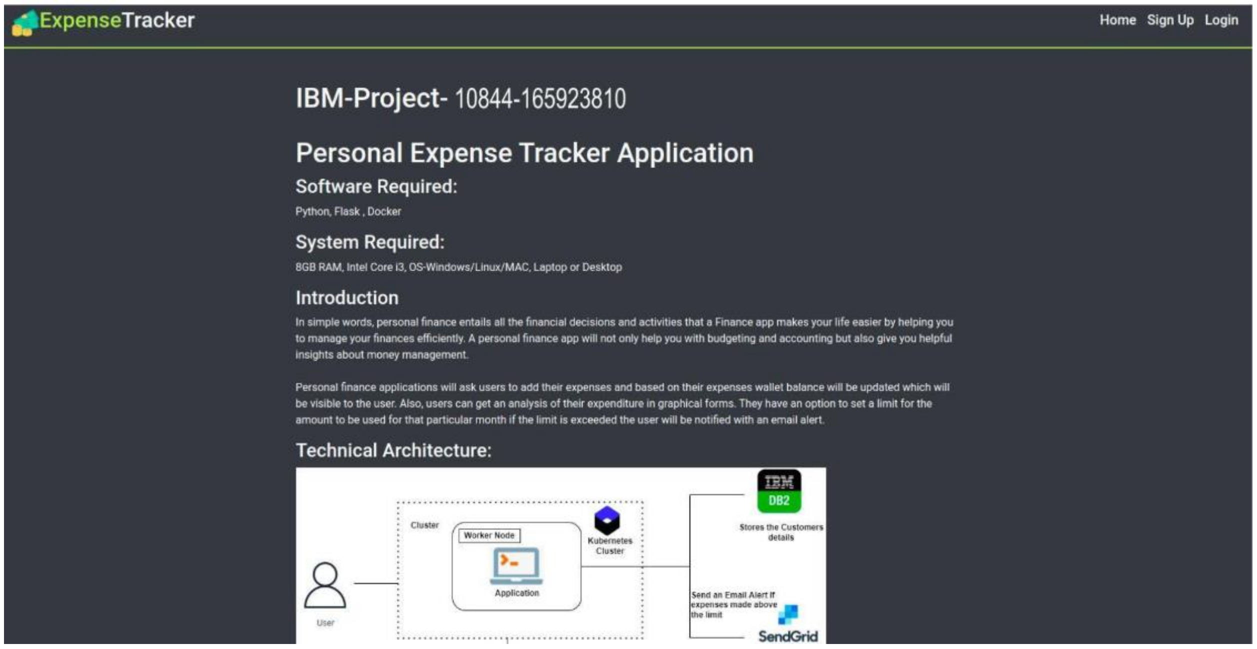
3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Interface	7	0	0	7
Login	43	0	0	43
Logout	2	0	0	2
Limit	3	0	0	3
Signup	8	0	0	8
Final Report Output	4	0	0	4

9. RESULTS

9.1 HomePage


ExpenseTracker Home Sign Up Login

IBM-Project- 10844-165923810

Personal Expense Tracker Application

Software Required:

Python, Flask , Docker

System Required:

8GB RAM, Intel Core i3, OS-Windows/Linux/MAC, Laptop or Desktop

Introduction

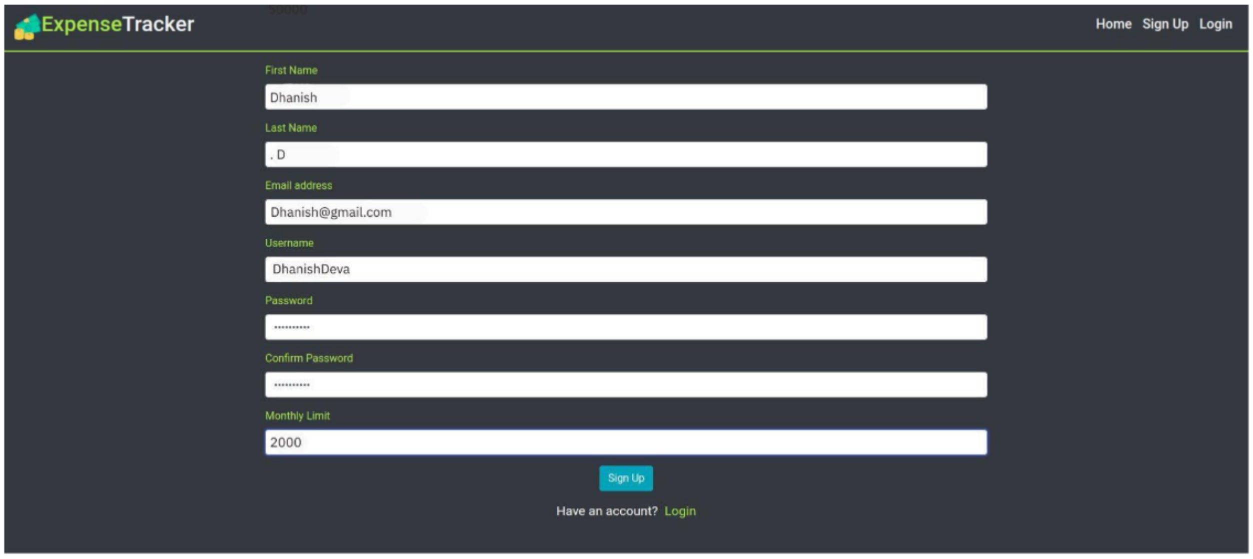
In simple words, personal finance entails all the financial decisions and activities that a Finance app makes your life easier by helping you to manage your finances efficiently. A personal finance app will not only help you with budgeting and accounting but also give you helpful insights about money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

Technical Architecture:

User
Cluster
Worker Node
Application
Kubernetes Cluster
IBM DB2
Stores the Customers details
Send an Email Alert if expenses made above the limit
SendGrid

9.2 Sign Up Page

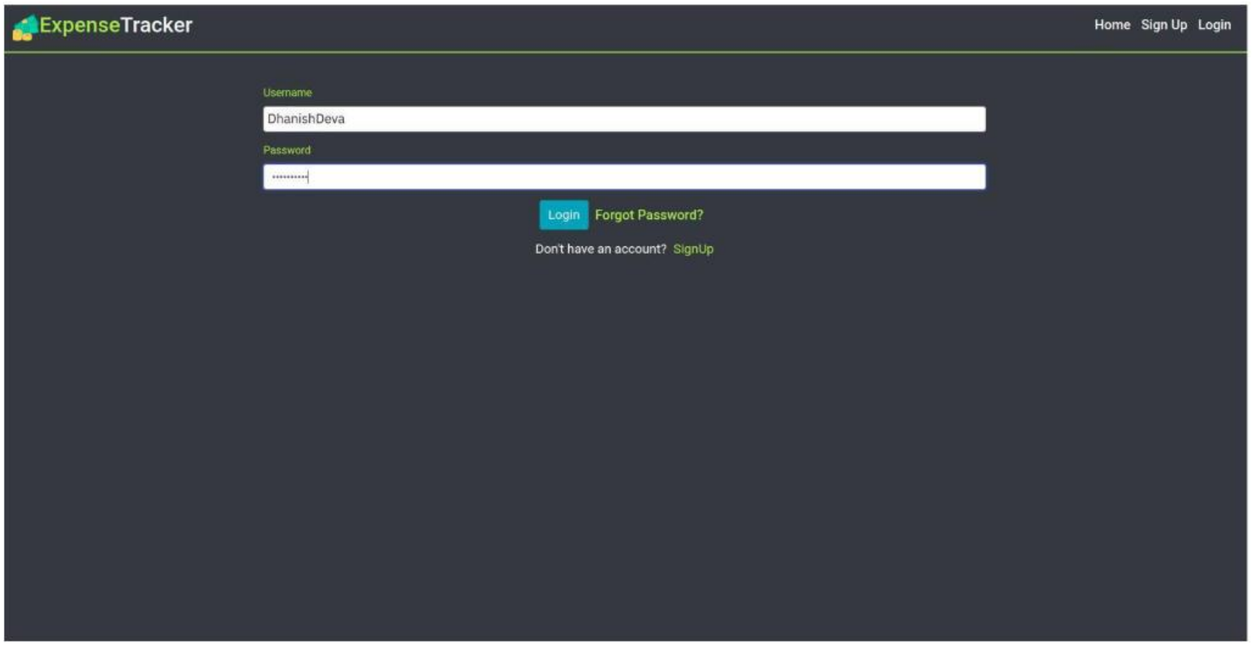

ExpenseTracker Home Sign Up Login

First Name	Dhanish
Last Name	. D
Email address	Dhanish@gmail.com
Username	DhanishDeva
Password	••••••••••
Confirm Password	••••••••••
Monthly Limit	2000

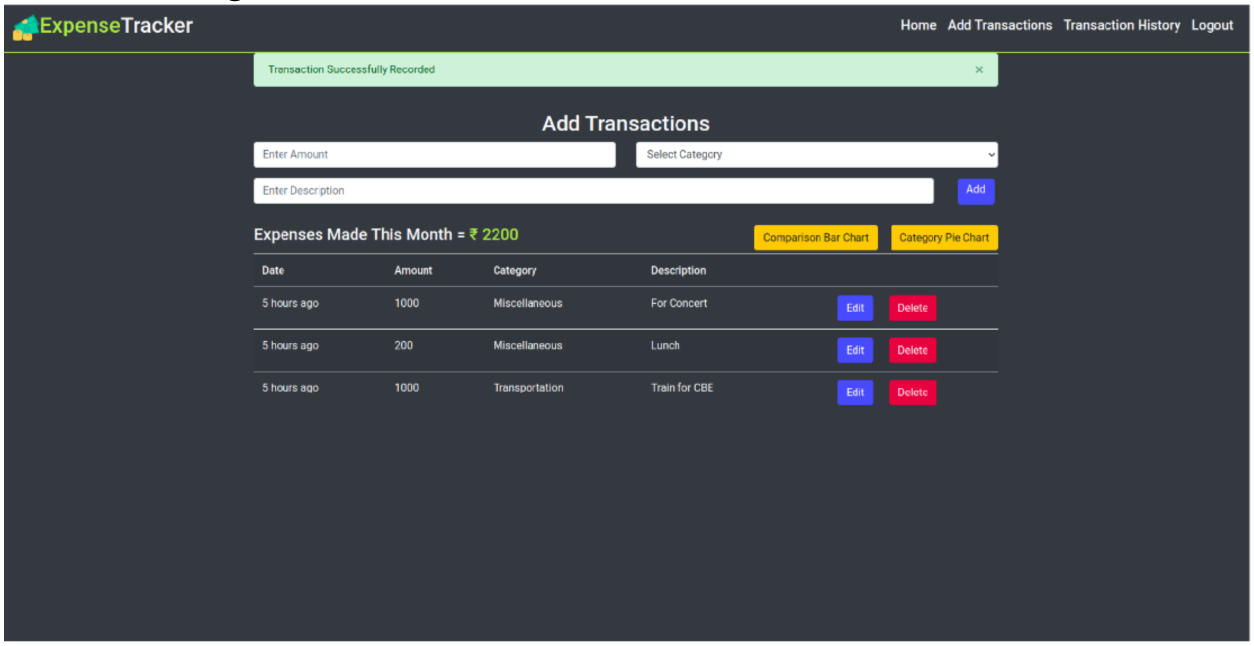
Sign Up

Have an account? Login

9.3 Login Page



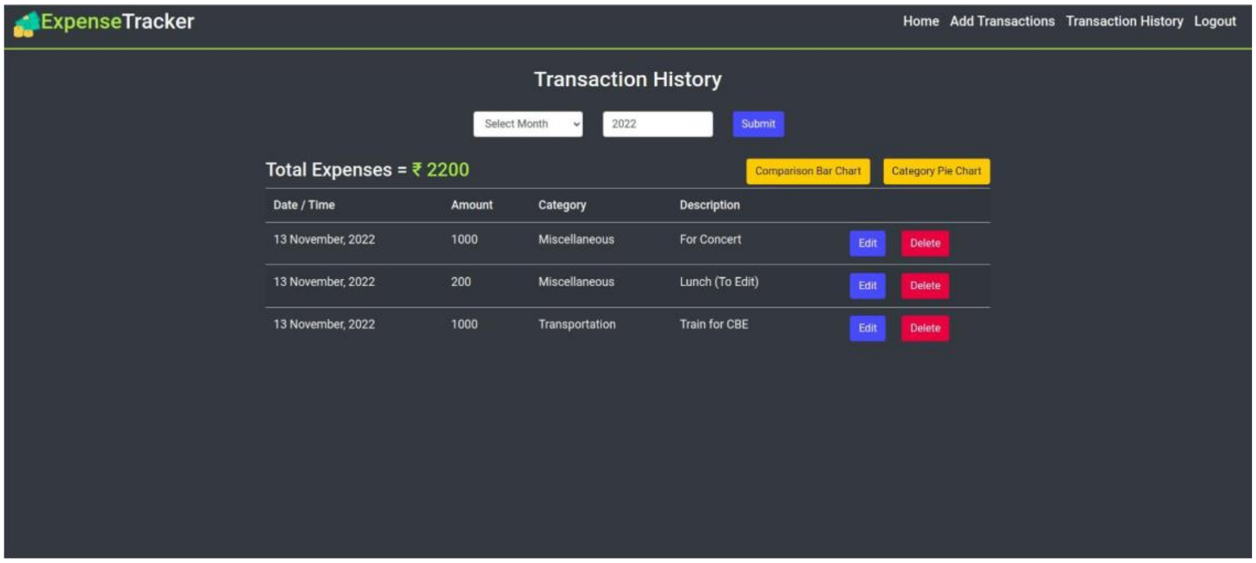
9.4 Transaction Page



Expenses Made This Month = ₹ 2200

Date	Amount	Category	Description	
5 hours ago	1000	Miscellaneous	For Concert	Edit Delete
5 hours ago	200	Miscellaneous	Lunch	Edit Delete
5 hours ago	1000	Transportation	Train for CBE	Edit Delete

9.5 Transaction History



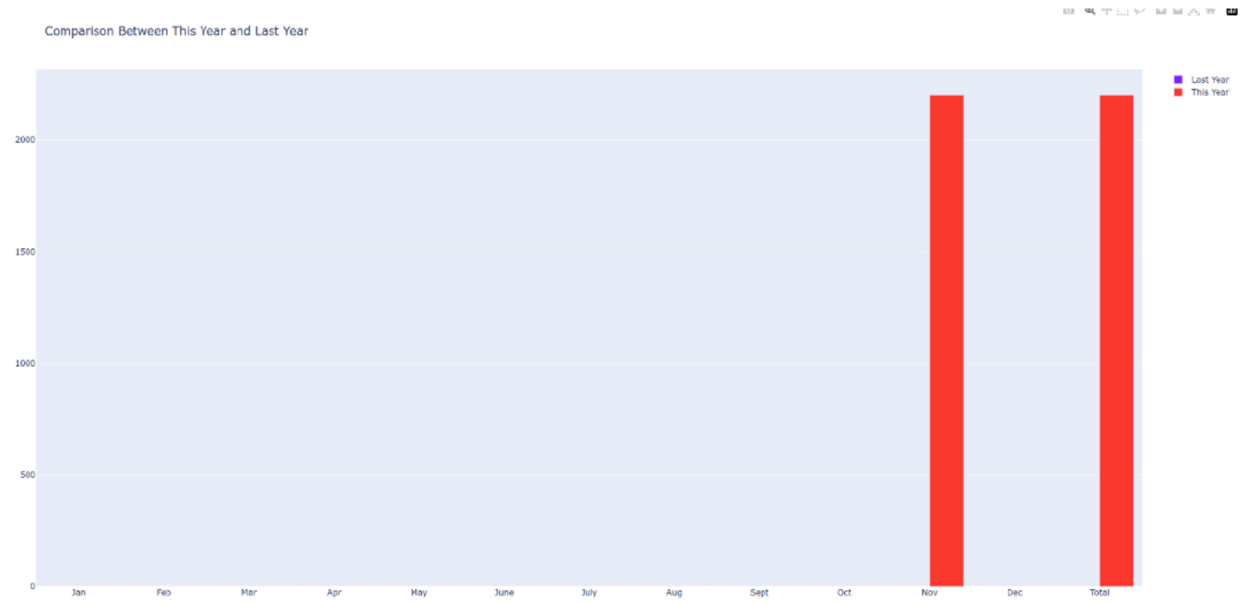
Transaction History

2022

Total Expenses = ₹ 2200

Date / Time	Amount	Category	Description
13 November, 2022	1000	Miscellaneous	For Concert
13 November, 2022	200	Miscellaneous	Lunch (To Edit)
13 November, 2022	1000	Transportation	Train for CBE

9.6 Comparison page



10. ADVANTAGES AND DISADVANTAGES

10.1. ADVANTAGES:

Being constantly aware of one's personal financial situation is one of the main benefits of keeping track of expenditures. You can stay within your budget by keeping track of your spending in each category, such as housing, food, transportation, and gifts, rather than just overall. Cons: It can be tedious and time-consuming to manually track every dollar spent; pros: It can be quick and easy to do this automatically.

Another benefit is the abundance of free automated expenditure tracking software tools. Being able to verify the programmer before making a purchase to confirm the budget is accessible might be a major benefit of having the programmer on a hand-held device. Another benefit is that there are solutions available for people who would want to continue keeping track of their expenditure manually with paper and a pen or by inputting information into an electronic spreadsheet. Some people like to keep a file folder or box where they may keep track of their daily cash expenditures and receipts. A pro of this simple daily tracking system is that it can make one more aware of where the money is going way before the end of a pay period or month.

10.2. DISADVANTAGES:

Any system for tracking spending has the drawback that it may be started, then slacked off until it is completely forgotten. This risk exists for any new objective, including attempting to lose weight or stop smoking. The tracking objective can be helpful if a person first creates a budget plan, then puts money in savings before spending any each new pay period or month. This reduces the amount of time needed to track spending and make sure all receipts are accounted for to once or twice per month.

There is no assurance that one will achieve their financial goals, even with ongoing surveillance of their spending patterns. Although this can be viewed as a drawback of spending tracking, it might be turned into a benefit if one decides to maintain attempting to handle all funds well. Errors can also be a con when spending is being tracked, but if tracking is done regularly, this con might be turned into a plus. Frequent monitoring of cash expenditures can help one identify and fix mistakes, enabling the budget plan to still be followed in spite of the error.

11. CONCLUSION:

A thorough money management plan necessitates decision-making that is clear and firmly held. To understand your business and personal finances, you will need a specific objective and a distinct vision. An expenditure monitoring app enters the scene at that point. A specialised package of services called an expenditure monitoring software is available to those who want to effectively manage their income and budget their spending and savings. It enables you to keep track of all transactions on a daily, weekly, and monthly basis, including bills, refunds, wages, receipts, taxes, etc.

12. FUTURE SCOPE:

- Achieve your business goals with a tailored mobile app that perfectly fits your business.
- Scale-up at the pace your business is growing.
- Deliver an outstanding customer experience through additional control over the app.
- Control the security of your business and customer data.
- Open direct marketing channels with no extra costs with methods such as push notifications.
- Boost the productivity of all the processes within the organization.
- Increase efficiency and customer satisfaction with an app aligned to their needs.
- Seamlessly integrate with existing infrastructure.

- Ability to provide valuable insights.
- Optimize sales processes to generate more revenue through enhanced data collection.
- Robo Advisors: Get expert investment advice and solutions with the Robo-advisors feature. This feature will analyze, monitor, optimize, and improve diversification in investments by turning data into actionable insights in real-time.
- Chats: Equip your expense tracking app with a bot that can understand and answer all user queries and address their needs such as account balance, credit score, etc.
- Prediction: With the help of AI, your mobile app can predict your next purchase, according to your spending behavior. Moreover, it can recommend products and provide unique insights on saving money. It brings out the factors causing fluctuations in your expenses.
- Employee Travel Budgeting: Most businesses save money with a travel budgeting app as it helps prepare a budget for an employee's entire business trip. The feature will predict the expenses and allocate resources according to the prediction.

13. APPENDIX:

SOURCE CODE

The source code has been uploaded in github. To refer the final sourse code click ‘

GITHUB LINK:

The github link : IBM-EPBL/IBM-Project-17517-1659672810: Personal Expense Tracker Application (github.com)