

A.Y 2022-23

Employability Skills Development

Project Report **(Expense Tracker App)**

Team Member Details:

Anant Singh – 66 – SE-COMP-C

Divyanshu Singh – 67 – SE-COMP-C

Kinjal Singh – 68 – SE-COMP-C

Arya Tiwari – 69 – SE-COMP-C

Guide Name:

Mrs. Lydia Suganya

INDEX

Sr No.	Name of topic	Page no.
1.	Abstract	3
2.	Introduction	4
3.	Background	5-8
3.	System Description (Block Diagram/Module Diagram)	9
4.	Results and Discussions	10-19
5.	Conclusion	20
6.	Future Scope	20
7.	References	21
8.	Annexure (Code)	22-24

Abstract

Managing personal expenses can be a daunting task, especially when trying to keep track of various expenses across different categories. To address this issue, our project aims to develop a web-based expense management system that allows users to efficiently track their expenses and organize them into different categories. The expense tracker system is an essential tool for managing personal or business expenses efficiently. With the advancement of technology, it has become easier to track expenses, and several expense tracking systems are available in the market. The purpose of this project is to develop a web-based expense tracking system that provides users with an easy-to-use interface to manage their expenses.

The proposed system will allow users to create an account and log in to the system. Users can then add new expenses, update or delete existing ones, and categorize expenses. The system will provide an overview of expenses in the form of graphs and charts to help users identify trends and make informed decisions about their finances. The system will also provide reminders to ensure timely payment of bills and deadlines. Users can set a budget for each category and the system will display the remaining amount they have for that category. The system also provides alerts when a user is nearing their budget limit or has exceeded it. This functionality is designed to help users manage their finances effectively and avoid overspending.

The system will be designed using Flask, a Python-based web application framework. Flask is a lightweight and flexible framework that can be easily integrated with other technologies. The system will use SQLAlchemy, a Python SQL toolkit and Object-Relational Mapping (ORM) library, to manage the database. The database will be implemented using SQLite, a lightweight and efficient database management system. The proposed system will have several features that make it user-friendly and efficient. Users will be able to add new expenses using a simple form that includes fields for the expense name, amount, date, category, and payment method. Users can select from predefined categories or create new ones. The system will allow users to view their expenses in a table format or visualize them using graphs and charts. The proposed system has several advantages over existing systems. The system is web-based, which means users can access it from anywhere with an internet connection. The system is also user-friendly, with a simple and intuitive interface that is easy to use. The system provides users with an overview of their expenses in the form of graphs and charts, making it easier to identify trends and make informed decisions.

Introduction:

In today's fast-paced world, keeping track of one's expenses has become a daunting task. One often struggles to maintain a proper record of their daily expenses, leading to financial mismanagement. In this context, our team has proposed the development of an Expense Tracker App that will help individuals keep track of their expenses in an organized manner. The proposed system will provide a comprehensive view of an individual's expenses, allowing them to manage their finances efficiently.

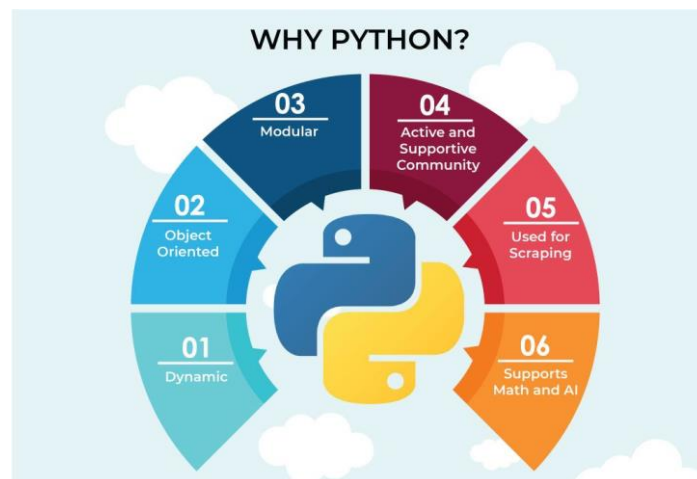
The system will be developed using Flask, a Python-based web application framework. The system will incorporate a database to store expense details, which can be accessed by users through a web interface. The users will be able to view, add, modify, and delete expenses. Moreover, the system will provide users with expense analytics to help them understand their spending habits and identify areas where they can cut down on expenses. The system will also allow users to categorize their expenses, making it easier to track their expenses and manage their finances.

The Expense Tracker App is designed to cater to individuals from various backgrounds, including students, working professionals, and small business owners. The system will be scalable and flexible, allowing it to be customized based on the user's needs. For instance, a student might want to track their monthly expenses, while a small business owner might want to track their expenses based on different departments or projects. One of the primary objectives of the proposed system is to provide users with a user-friendly interface that is easy to navigate. The interface will be intuitive, allowing users to add, modify, and delete expenses without any technical knowledge. The system will help users manage their finances efficiently, allowing them to save money and invest in their future. Moreover, the system will help users understand their spending habits, allowing them to make better financial decisions. For instance, a user might identify that they are spending too much on entertainment and can cut down on their expenses to save money. Additionally, the system will help users keep track of their bills and payments, ensuring they do not miss any payment deadlines.

Background:

1. Python

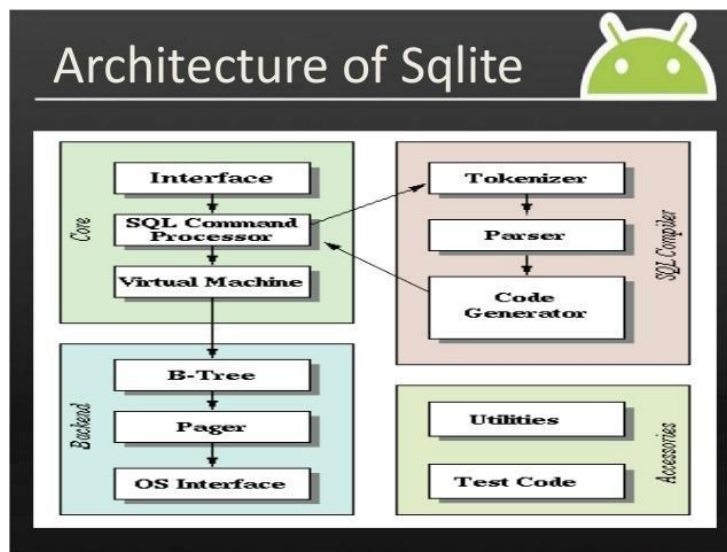
Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2. Python consistently ranks as one of the most popular programming languages.



Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that it is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them. As well as standard desktop integrated development environments, there are Web browser-based IDEs, including SageMath, for developing science- and math-related programs; PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial IDE emphasizing scientific computing. LibreOffice includes Python and intends to replace Java with Python. Its Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013. Python's focus on readability is also a key factor in its popularity among scientific and academic communities, where code needs to be clear and understandable to non-programmers. Python's popularity has also been fueled by the large and growing community of developers and enthusiasts who use and contribute to the language.

2. SQLite (For Backend)

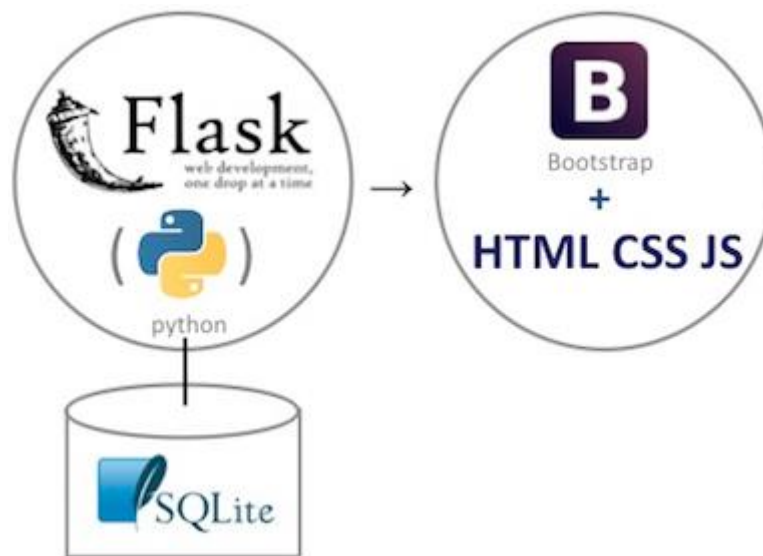
SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a popular choice as an embedded database for local/client storage in application software such as web browsers. It is also used in many other applications that need a lightweight, embedded database. SQLite is ACID-compliant and implements most of the SQL standards, using a dynamically and weakly typed SQL syntax that does not guarantee domain integrity. To use SQLite in a C/C++ program, you can use the sqlite3 API, which provides a lightweight, simple, self-contained, high-reliability, full-featured, and SQL database engine. The API is implemented as a library of C functions that can be called from your program. One of the main benefits of using SQLite is that it is very easy to get started with. To create a new database in SQLite, you simply need to create a new file on your filesystem and connect to it using the sqlite3 API. SQLite was created in the year 2000 by D. Richard Hipp, who continues to lead the development of the software today. SQLite was designed to be a lightweight and simple database engine that could be easily embedded into other applications.



It was created as an alternative to more complex and heavyweight database engines, such as MySQL and PostgreSQL. Over the years, SQLite has gained widespread adoption and is now one of the most widely used database engines in the world. It is used in many applications, including web browsers, mobile phones, and a wide variety of other software. SQLite is a very lightweight database engine, with a small library size (typically less than 1MB). This makes it well-suited for use in applications where the database is embedded directly into the application binary, such as mobile apps. As mentioned earlier, SQLite is a serverless database engine, which means there is no need to set up and maintain a separate database server process. This makes it easy to deploy and manage, as there are no additional dependencies to worry about. It is a self-contained, serverless database engine, which means you can include it in your application without the need for a separate database server.

3. Flask

Flask is a lightweight and popular web framework for Python that helps developers create web applications with ease. It was created by Armin Ronacher in 2010 and has since gained popularity due to its simplicity, flexibility, and ease of use. Flask is a micro-framework, meaning that it does not require any particular tools or libraries and can be easily integrated with other third-party libraries as needed. It provides a simple and intuitive API for building web applications, making it an excellent choice for small to medium-sized projects.



One of the key benefits of Flask is its minimalist approach to web application development. Unlike larger frameworks that come with many pre-built components, Flask provides only the core functionality needed for building web applications. This approach enables developers to quickly create lightweight and efficient web applications that are easy to maintain and scale. Flask provides a powerful routing system, which allows developers to map URL patterns to specific views or functions in the application. This enables the application to respond to specific requests with customized content, making it highly flexible and customizable. Flask also includes built-in support for various data storage systems, including SQL databases, NoSQL databases, and in-memory data stores. Flask provides a wide range of extensions and plugins, which can be easily integrated into the application to add additional functionality. These extensions include support for user authentication, database management, caching, and more. Flask also has a robust community of developers who contribute to its development and maintenance, making it a reliable and continuously evolving framework. Flask can be used to implement an Model-View-Controller (MVC) pattern of web framework: Flask (controller), SQLite (model), Bootstrap (view). Jinja2 is popular for rendering HTML templates and is part of the view. Flask can be used with front-end frameworks such as VueJS but you would have to learn JavaScript. Flask comes with a simple built-in server. This runs on the default port 5000. This is commonly used for development but not recommended for production because it doesn't scale well.

4. HTML, CSS, BOOTSTRAP

HTML (Hypertext Markup Language)

- HTML allows for the creation of structured and organized web content.
- It provides a wide range of tags and attributes to enhance the presentation and functionality of web pages.
- HTML supports multimedia elements such as images, videos, and audio.
- It enables the creation of hyperlinks that connect web pages and facilitate easy navigation.
- Supports multimedia integration for better user experience.
- Platform-independent and compatible with all devices.

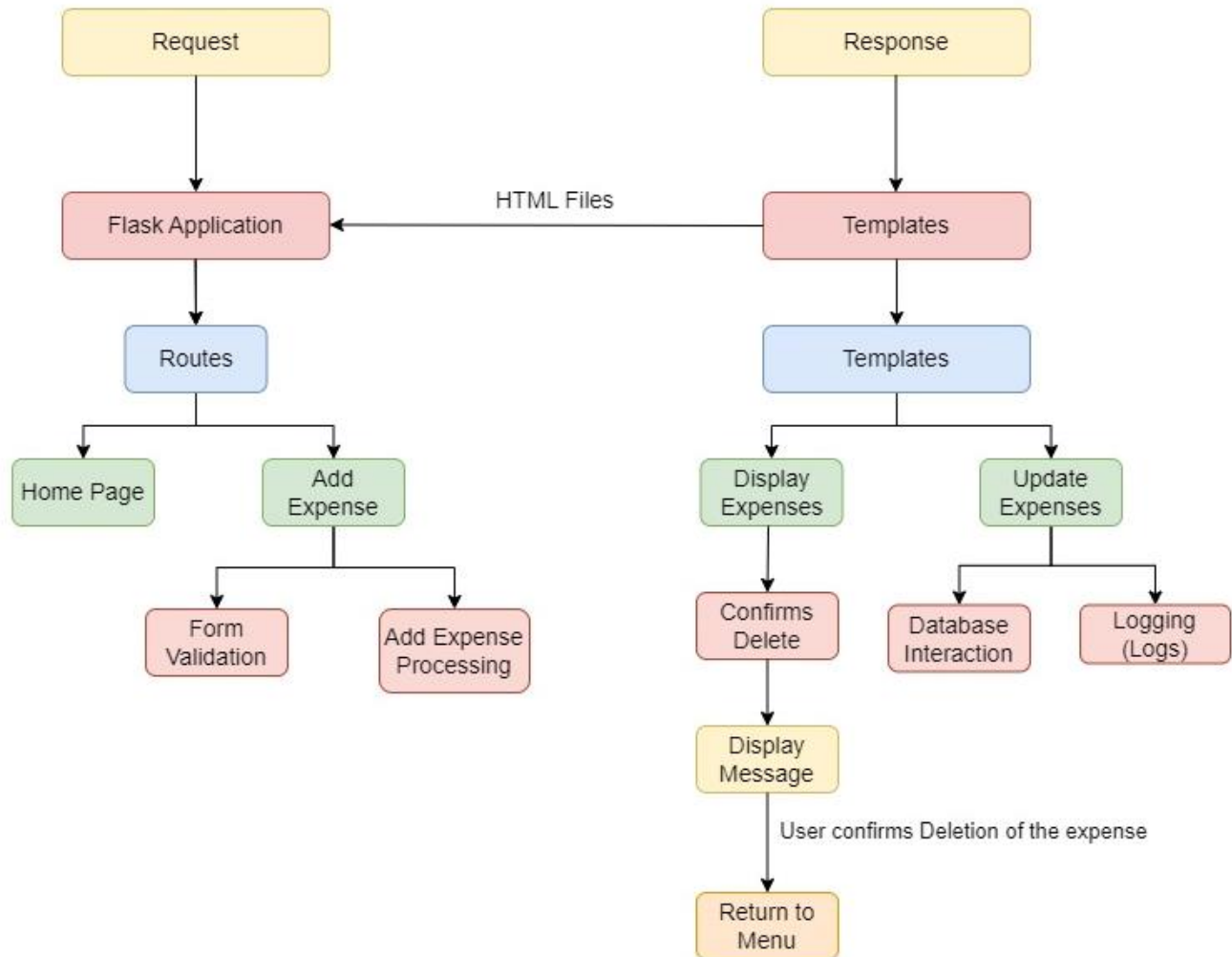
CSS (Cascading Style Sheets)

- **Cascading:** Allows for the cascading effect of styles from parent elements to child elements.
- **Selectors:** Selects HTML elements based on their type, class, or ID.
- **Box Model:** Defines the layout and spacing of HTML elements.
- **Responsive Design:** Enables the creation of websites that adjust to different screen sizes and resolutions.
- **Typography:** Enables the customization of font styles, sizes, and spacing.
- **Color:** Allows for the customization of color schemes and gradients.
- **Animations and Transitions:** Enables the creation of interactive and visually appealing animations and transitions.

BOOTSTRAP

- Bootstrap is designed to create responsive web pages that can adapt to any device screen size.
- Bootstrap provides a set of pre-built components such as navigation bars, forms, buttons, and more to make development faster.
- Bootstrap uses a grid system that allows developers to create responsive layouts for web pages.
- Bootstrap is designed to be compatible with all modern web browsers, ensuring a consistent user experience across different platforms.
- Bootstrap provides a wide range of customizable themes that can be used to change the look and feel of a website.
- Bootstrap includes a set of JavaScript plugins that can be used to add advanced functionality to web pages.

System Description:



Request: The HTTP request from the client, which contains information such as the HTTP method, URL, and headers.

Flask Application: The main application that handles incoming requests and sends back responses.

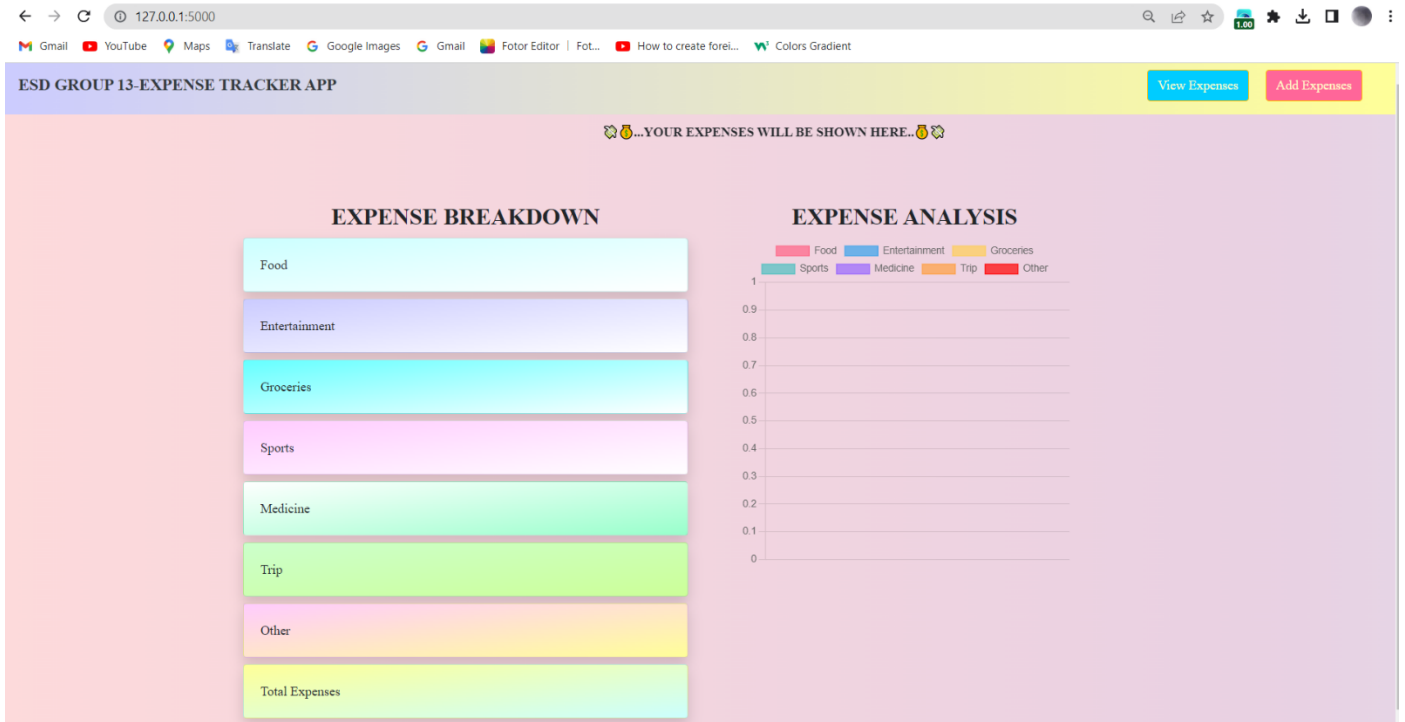
Templates (HTML Files): The HTML files that Flask uses to generate the dynamic content of the web pages.

Routes: The Flask routes that map URLs to specific functions or views.

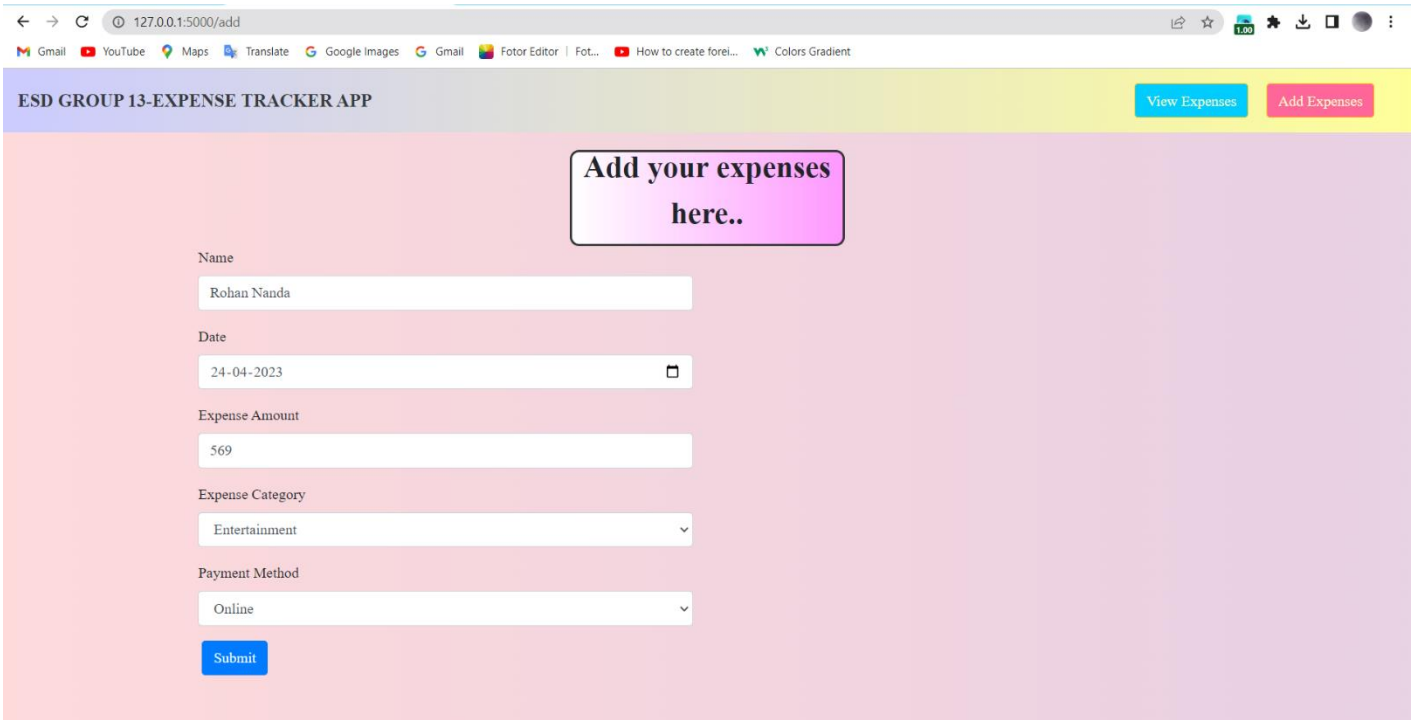
Home Page: The default page that displays when the user first loads the website.

Database: The database where all the expenses are stored.

Results and Discussions:



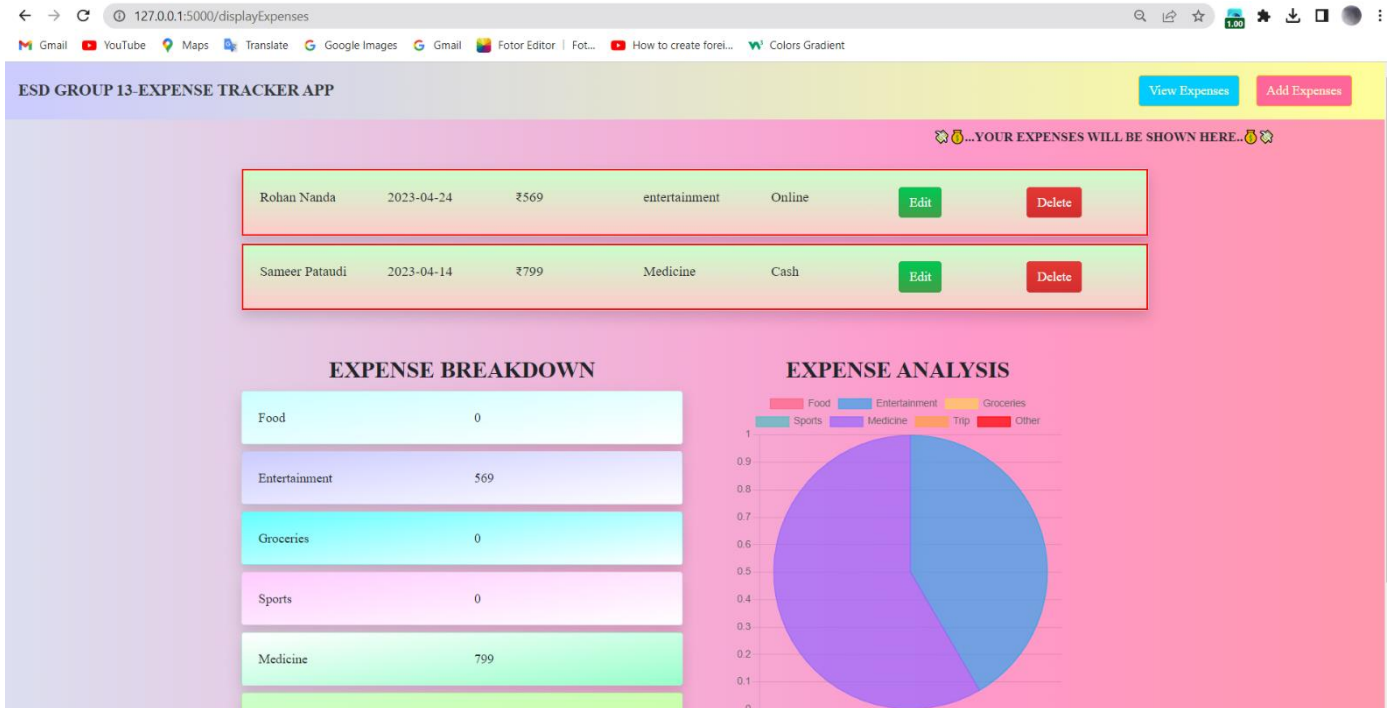
The Welcoming Page



The screenshot shows the 'ESD GROUP 13-EXPENSE TRACKER APP' interface with the 'Add your expenses here..' form. The form includes the following fields:

- Name:** Rohan Nanda
- Date:** 24-04-2023
- Expense Amount:** 569
- Expense Category:** Entertainment (selected from a dropdown menu)
- Payment Method:** Online (selected from a dropdown menu)
- Submit:** A blue button to submit the expense.

Add Expense Page



View Expense Page with a Pie Chart Analysis

DB Browser for SQLite - C:\Users\ANANT\Documents\ESD_GRP. NO. 13\esd.grp 13.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project

Database Structure Edit Pragmas Browse Data Execute SQL

Table: expense

	id	expenseName	expenseDate	expenseAmount	expenseCategory	expenseMethod
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Rohan Nanda	2023-04-24	569	entertainment	Online
2	2	Sameer Pataudi	2023-04-14	799	Medicine	Cash

Database to store the records of the expenses made

WhatsApp ExpenseManager

127.0.0.1:5000/updateExpense/1

ESD GROUP 13-EXPENSE TRACKER APP

[View Expenses](#) [Add Expenses](#)

EDIT YOUR EXPENSES !!!

Name

Date

Expense Amount

Expense Category

Payment Method

[Update](#)

Edit / Update the existing Expense Page

127.0.0.1:5000/displayExpenses

ESD GROUP 13-EXPENSE TRACKER APP

[View Expenses](#) [Add Expenses](#)

🔔...YOUR EXPENSES WILL BE SHOWN HERE.🔔

Rohan Nanda	2023-04-24	₹652	entertainment	Cash	Edit	Delete
Sameer Pataudi	2023-04-14	₹799	Medicine	Cash	Edit	Delete

Updated Record is being displayed

DB Browser for SQLite - C:\Users\ANANT\Documents\ESD_GRP. NO. 13\esd.grp 13.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project

Database Structure Edit Pragma Browse Data Execute SQL

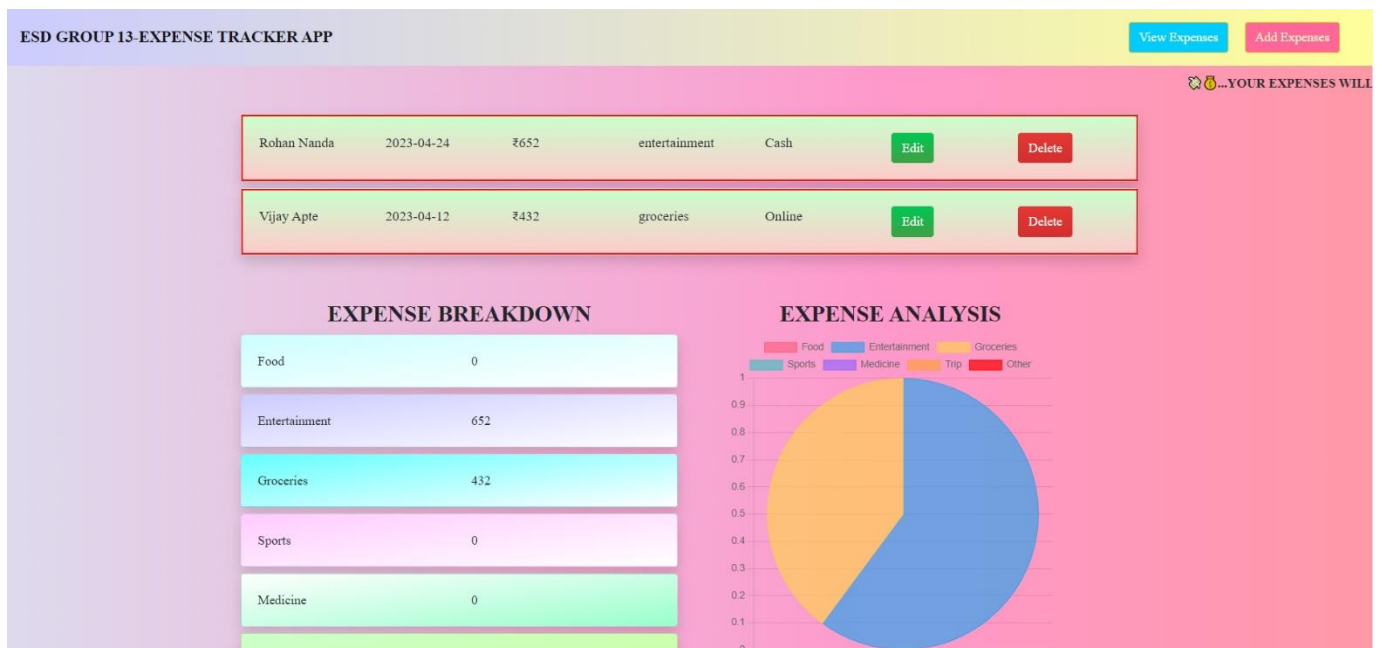
Table: expense

	id	expenseName	expenseDate	expenseAmount	expenseCategory	expenseMethod
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Rohan Nanda	2023-04-24	652	entertainment	Cash
2	2	Sameer Pataudi	2023-04-14	799	Medicine	Cash

Database after updating a record



Pie-Chart Analysis of the Expenses Made



Expense record is deleted from the browser

DB Browser for SQLite - C:\Users\ANANT\Documents\ESD_GRP. NO. 13\esd.grp 13.db

File Edit View Tools Help

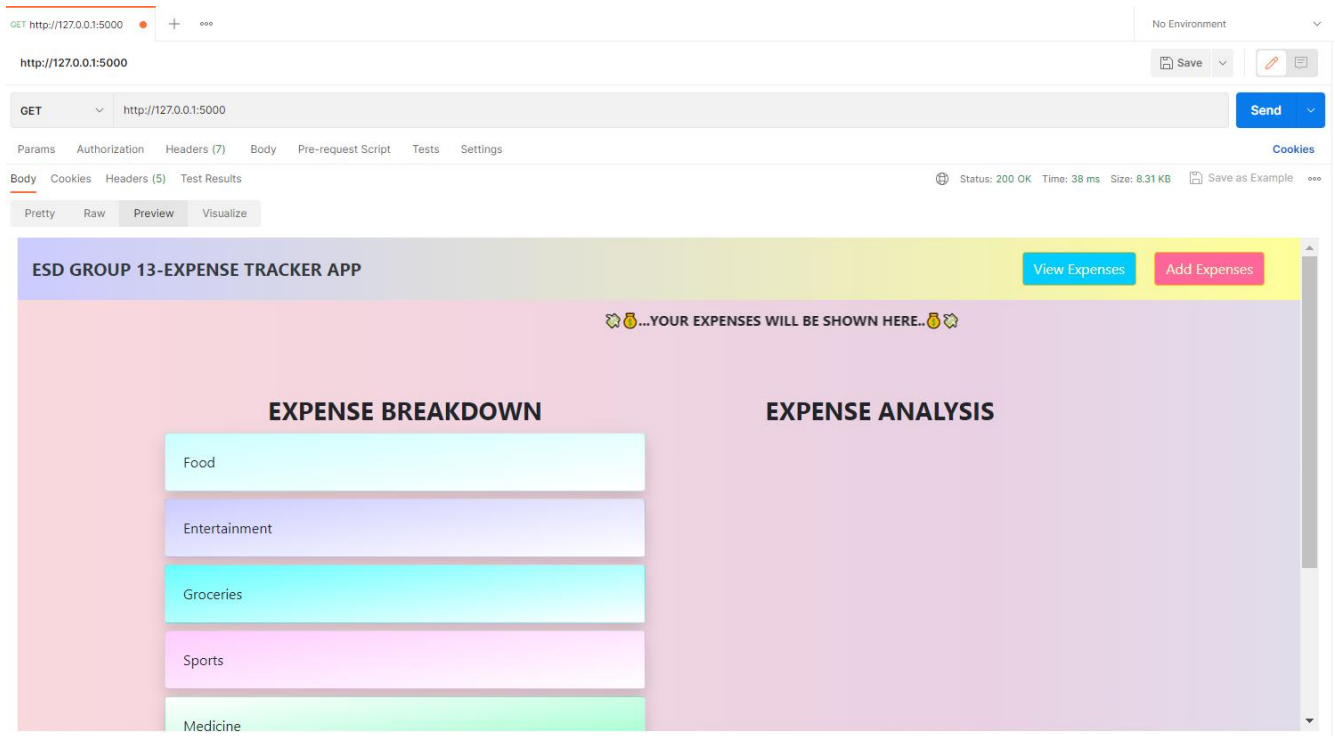
New Database Open Database Write Changes Revert Changes Open Project Save Project

Database Structure Edit Pragmas Browse Data Execute SQL

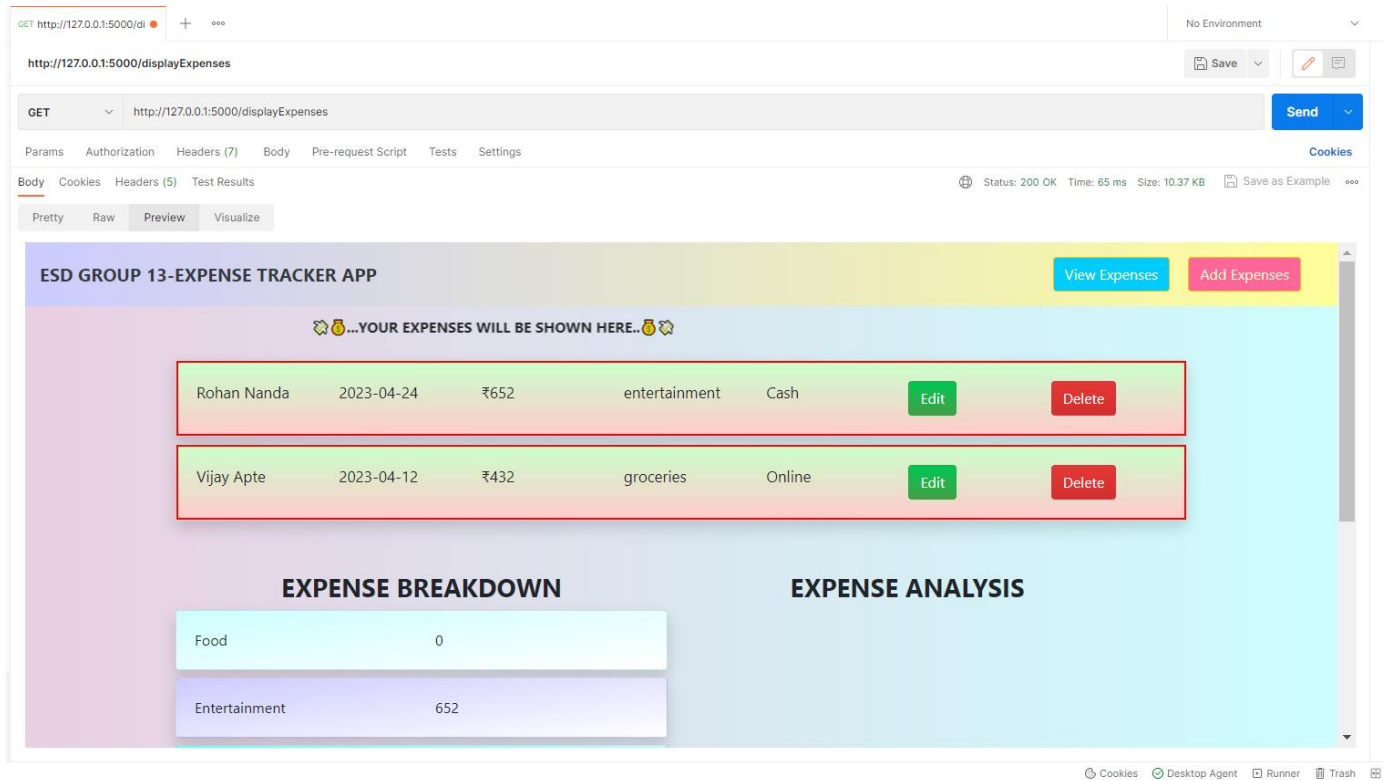
Table: expense

	id	expenseName	expenseDate	expenseAmount	expenseCategory	expenseMethod
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Rohan Nanda	2023-04-24	652	entertainment	Cash
2	3	Vijay Apte	2023-04-12	432	groceries	Online

Expense record is deleted from the database as well



Display of our webpage using Postman



GET http://127.0.0.1:5000/displayExpenses

Status: 200 OK Time: 65 ms Size: 10.37 KB

ESD GROUP 13-EXPENSE TRACKER APP

👛...YOUR EXPENSES WILL BE SHOWN HERE...👛

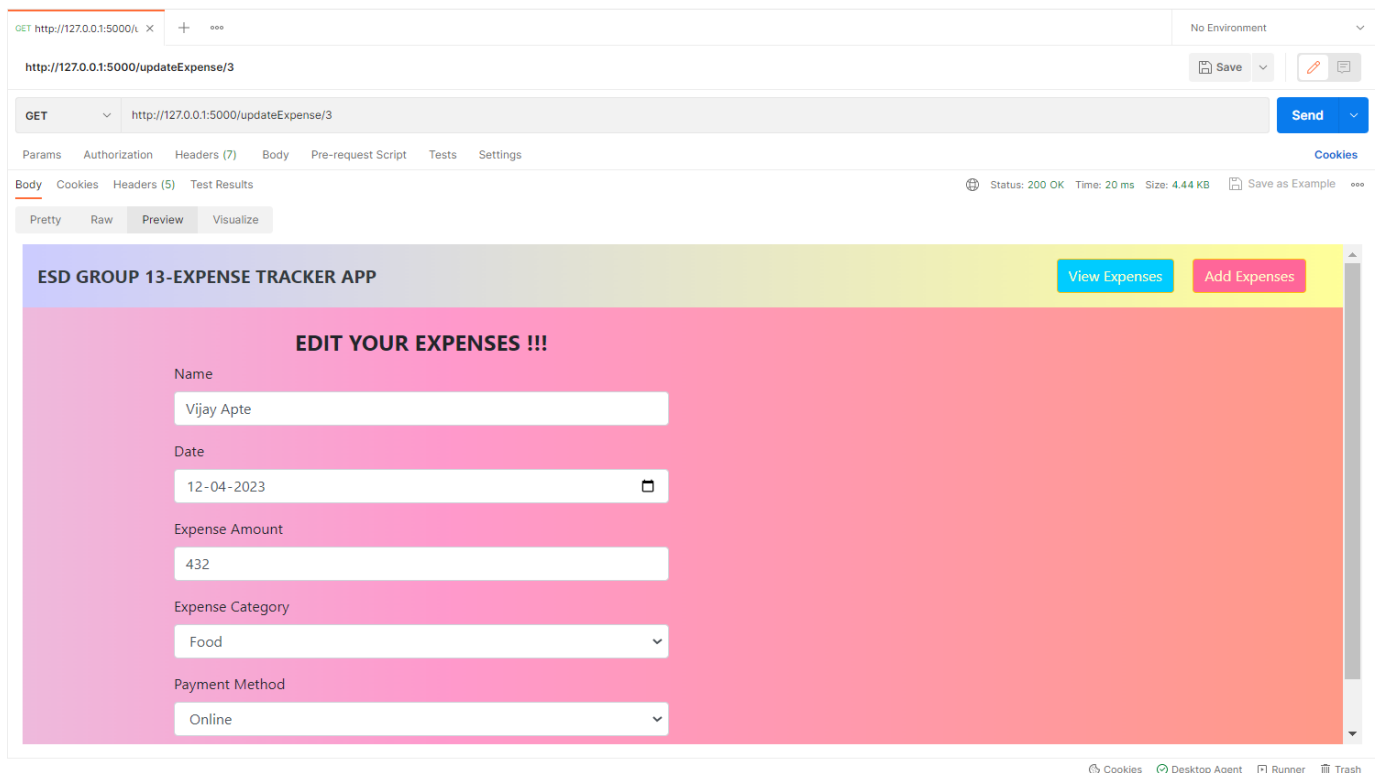
Rohan Nanda	2023-04-24	₹652	entertainment	Cash	Edit	Delete
Vijay Apte	2023-04-12	₹432	groceries	Online	Edit	Delete

EXPENSE BREAKDOWN

Food	0
Entertainment	652

EXPENSE ANALYSIS

Display of expenses using Postman



GET http://127.0.0.1:5000/updateExpense/3

Status: 200 OK Time: 20 ms Size: 4.44 KB

ESD GROUP 13-EXPENSE TRACKER APP

EDIT YOUR EXPENSES !!!

Name

Date

Expense Amount

Expense Category

Payment Method

Update Expense using Postman

POST http://127.0.0.1:5000/ x + ... No Environment

http://127.0.0.1:5000/editExpense Save Edit

POST http://127.0.0.1:5000/editExpense Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> id	3		
<input checked="" type="checkbox"/> name	Samar Apte		
<input checked="" type="checkbox"/> date	10-03-2023		
<input checked="" type="checkbox"/> amount	1500		
<input checked="" type="checkbox"/> category	Trip		
<input checked="" type="checkbox"/> method	Online		

Updating an existing Expense record using Postman

Body Cookies Headers (5) Test Results Status: 200 OK Time: 34 ms Size: 10.36 KB Save as Example

Pretty Raw Preview Visualize

...YOUR EXPENSES WILL BE SHOWN HERE...

Rohan Nanda	2023-04-24	₹652	entertainment	Cash	Edit	Delete
Samar Apte	10-03-2023	₹1500	Trip	Online	Edit	Delete

Record Updated using Postman

DB Browser for SQLite - C:\Users\ANANT\Documents\ESD_GRP. NO. 13\esd.grp 13.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project

Database Structure Edit Pragmas Browse Data Execute SQL

Table: expense Filter in any column

	id	expenseName	expenseDate	expenseAmount	expenseCategory	expenseMethod
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Rohan Nanda	2023-04-24	652	entertainment	Cash
2	3	Samar Apte	10-03-2023	1500	Trip	Online

Updated Record is being displayed in the database which is made by Postman Request

POST http://127.0.0.1:5000/ add Expense

Save Send

POST http://127.0.0.1:5000/addExpense

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> name	Ranbir Rai		
<input checked="" type="checkbox"/> date	2023-12-04		
<input checked="" type="checkbox"/> amount	576		
<input checked="" type="checkbox"/> category	sports		
<input checked="" type="checkbox"/> method	Cash		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 23 ms Size: 12.39 KB Save as Example

Pretty Raw Preview Visualize

...YOUR EXPENSES WILL BE SHOWN HERE...

Rohan Nanda	2023-04-24	₹652	entertainment	Cash	Edit	Delete
Samar Apte	10-03-2023	₹1500	Trip	Online	Edit	Delete
Ranbir Rai	2023-12-04	₹576	sports	Cash	Edit	Delete

Adding record using Postman

DB Browser for SQLite - C:\Users\ANANT\Documents\ESD_GRP. NO. 13\esd.grp 13.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project

Database Structure Edit Pragmas Browse Data Execute SQL

Table: expense

	id	expenseName	expenseDate	expenseAmount	expenseCategory	expenseMethod
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Rohan Nanda	2023-04-24	652	entertainment	Cash
2	3	Samar Apte	10-03-2023	1500	Trip	Online
3	5	Ranbir Rai	2023-12-04	576	sports	Cash

Added record is being stored in the database

DEL http://127.0.0.1:5000/c x + ... No Environment

http://127.0.0.1:5000/deleteExpense/3 Save

DELETE http://127.0.0.1:5000/deleteExpense/3 Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 28 ms Size: 10.36 KB Save as Example

Pretty Raw Preview Visualize

ESD GROUP 13-EXPENSE TRACKER APP View Expenses Add Expenses

...YOUR EXPENSES WILL BE SHOWN HERE..

Rohan Nanda	2023-04-24	₹652	entertainment	Cash	Edit	Delete
Ranbir Rai	2023-04-12	₹576	sports	Cash	Edit	Delete

Data is deleted by calling DELETE API

127.0.0.1:5000/displayExpenses

ESD GROUP 13-EXPENSE TRACKER APP View Expenses Add Expenses

...YOUR EXPENSES WILL BE SHOWN HERE..

Rohan Nanda	2023-04-24	₹652	entertainment	Cash	Edit	Delete
Ranbir Rai	2023-04-12	₹576	sports	Cash	Edit	Delete

EXPENSE BREAKDOWN

Food	0
Entertainment	652
Groceries	0
Sports	576
Medicine	0

EXPENSE ANALYSIS

Food Entertainment Groceries Sports Medicine Tip Other

Data is deleted from the browser as well

DB Browser for SQLite - C:\Users\ANANT\Documents\ESD_GRP. NO. 13\esd.grp 13.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project

Database Structure Edit Pragmas Browse Data Execute SQL

Table: expense

	id	expenseName	expenseDate	expenseAmount	expenseCategory	expenseMethod
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Rohan Nanda	2023-04-24	652	entertainment	Cash
2	4	Ranbir Rai	2023-04-12	576	sports	Cash

Data is deleted from the database as well

GET http://127.0.0.1:5000/add

http://127.0.0.1:5000/add

GET http://127.0.0.1:5000/add

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 17 ms Size: 4.69 KB Save as Example

Pretty Raw Preview Visualize

Add your expenses here..

Name

Date

Expense Amount

Expense Category

Payment Method

Cookies Desktop Agent Runner Trash

Add expenses view

Conclusion:

The application allows users to add new expenses, display a list of all expenses, update existing expenses, and view expense breakdowns by category. The application uses Flask to handle HTTP requests and SQLAlchemy to manage the database. The code follows the Model-View-Controller (MVC) architectural pattern, where the model represents the data in the database, the views display the data to the user, and the controller handles user actions and updates the model. The application allows users to add new expenses by filling out a form that includes the expense name, date, amount, category, and payment method. Users can view all expenses in a table, with the option to edit or delete individual expenses. The application also displays a breakdown of expenses by category, showing the total expenses in each category and the percentage of total expenses. Overall, the application provides a simple and easy-to-use interface for managing personal expenses, allowing users to quickly add, view, and analyse their spending habits. In terms of scalability, the code can be extended by adding more features such as generating reports, sending email notifications, and integrating with third-party services like payment gateways. Also, the application can be enhanced by implementing user authentication and authorization to restrict access to certain functionalities. It provides a clear structure for building more complex applications, and the use of popular Python libraries makes it easy to maintain and extend.

Future Scope:

1. Add user authentication and authorization for system security.
2. Allow users to create profiles and add personal information for system customization.
3. Enhance the system with data analysis features such as charts and graphs for better insights.
4. Integrate with third-party APIs.
5. Add email notifications for users to stay informed about expenses and updates.
6. Develop a mobile application for access to the system from mobile devices.
7. Provide multi-language support for users who speak different languages.
8. The system can be enhanced by adding gamification features to make budgeting more fun and engaging for users.
9. Integration with social media platforms.
10. The system can be enhanced by implementing a machine learning algorithm.

References:

1. <https://flask.palletsprojects.com/en/2.0.x/>
2. <https://flask-sqlalchemy.palletsprojects.com/>
3. <https://www.sqlalchemy.org/>
4. <https://flask-migrate.readthedocs.io/en/latest/>
5. <https://dev.mysql.com/doc/connector-python/en/>
6. <https://jinja.palletsprojects.com/en/3.0.x/>
7. <https://getbootstrap.com/>
8. <https://docs.python.org/3/>
9. <https://www.sqlite.org/index.html>
10. <https://dev.mysql.com/doc/connector-python/en/>

Annexure (Code) :

```
from flask import Flask, render_template,
request,
redirect, flash, session, url_for, jsonify, flash
from flask_sqlalchemy import
SQLAlchemy
from sqlalchemy import create_engine
import os
from flask_session import Session
from flask_migrate import Migrate
import mysql.connector
project_dir =
os.path.dirname(os.path.abspath(__file__))
database_file =
"sqlite:///{}".format(os.path.join(project_d
ir, "esd.grp 13.db"))
```

```
app = Flask(__name__)
app.config['SESSION_TYPE'] =
'filesystem'
app.config['SECRET_KEY'] =
'redsfsfsfis'

app.config["SQLALCHEMY_DATABASE_
E_URI"] = database_file
db = SQLAlchemy(app)
migrate = Migrate()
```

```
class Expense(db.Model):
    id = db.Column(db.Integer,
primary_key=True)
    expenseName =
db.Column(db.String(40), nullable=False)
    expenseDate =
db.Column(db.String(40), nullable=False)
    expenseAmount =
db.Column(db.Integer, nullable=False)
    expenseCategory =
db.Column(db.String(40), nullable=False)
    expenseMethod =
db.Column(db.String(40), nullable=False)
```

Creation of the database tables within the application context.

```
with app.app_context():
    db.create_all()
```

```
@app.route('/')
def homePage():
    return
render_template('displayExpenses.html')
```

Add expenses view

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

Display expenses view

```
@app.route('/displayExpenses')
def displayExpenses():
    expenses = Expense.query.all()
    sum = 0
    sum_food = 0
    sum_entertainment = 0
    sum_groceries = 0
    sum_sports = 0
    sum_medicine = 0
    sum_trip = 0
    sum_other = 0
```

```
for expense in expenses:
    sum += expense.expenseAmount
```

```
if expense.expenseCategory == 'food':
    sum_food +=
expense.expenseAmount
elif expense.expenseCategory ==
'entertainment':
    sum_entertainment +=
expense.expenseAmount
```

```

    elif expense.expenseCategory ==
'groceries':
        sum_groceries +=
expense.expenseAmount
    elif expense.expenseCategory ==
'sports':
        sum_sports +=
expense.expenseAmount
    elif expense.expenseCategory ==
'medicine':
        sum_medicine +=
expense.expenseAmount
    elif expense.expenseCategory ==
'trip':
        sum_trip +=
expense.expenseAmount
    else:
        sum_other +=
expense.expenseAmount

    return
render_template('displayExpenses.html',
                expenses=expenses,
                sum=sum,
                sum_food=sum_food,
                sum_entertainment=sum_
entertainment,
                sum_groceries=sum_groc
eries,
                sum_sports=sum_sports,
                sum_medicine=sum_medi
cine,
                sum_trip=sum_trip,
                sum_other=sum_other)

@app.route('/deleteExpense/<int:id>',meth
ods=['DELETE', 'GET'])
def deleteExpense(id):
    expense =
Expense.query.filter_by(id=id).first()
    db.session.delete(expense)
    db.session.commit()

```

```

    return redirect('/displayExpenses')

@app.route('/updateExpense/<int:id>')
def updateExpense(id):
    expense =
Expense.query.filter_by(id=id).first()
    return
render_template('updateExpense.html',
expense = expense)

@app.route('/editExpense',
methods=['POST'])
def editExpense():
    id = request.form['id']
    name = request.form['name']
    date = request.form['date']
    amount = request.form['amount']
    category = request.form['category']
    method=request.form['method']
    db.session.query(Expense).filter_by(id=
id).update({"expenseName":name,
"expenseDate":date,
"expenseAmount":amount,
"expenseCategory":category,"expenseMet
hod":method})

    db.session.commit()
    return redirect('/displayExpenses')

@app.route('/addExpense',
methods=['GET', 'POST'])
def addExpense():
    if request.method == 'POST':
        name = request.form['name']
        date = request.form['date']
        amount = request.form['amount']
        category = request.form['category']
        method=request.form['method']
        expense =
Expense(expenseName=name,expenseDat
e=date, expenseAmount=amount,

```

```
expenseCategory=category,expenseMetho      return
d=method)                                render_template('addExpense.html')

db.session.add(expense)

db.session.commit()                        if __name__ == '__main__':

return redirect('/displayExpenses')        app.run(debug=True)

else:
```