CONTENTS

ABSTRACT

Inflow and Outflow record of money can be easily kept with the help of expense tracker. It helps to manage finances. In this project, we will develop an expense tracker that will track our expenses. Let’s start developing the project.

This project develops a user-friendly expense tracker application to facilitate personal financial management. The application enables users to

1. Record
2. Categorize
3. Track expenses
4. Set budgets
5. Generate reports

By providing a comprehensive view of financial transactions, this application aims to simplify expense management, promote financial accountability, and inform data-driven decision-making.

This project presents an expense tracker web application built using Python and Django.

The application enables users to record, categorize, and track their expenses, providing a comprehensive view of their financial transactions. With features such as user authentication, expense categorization, budgeting, and reporting, this application aims to simplify personal financial management.

Leveraging Django's robust framework and Python's versatility, this expense tracker offers an intuitive user interface, automated expense tracking, and data visualization, making it an essential tool for individuals seeking to streamline their financial management.

1. INTRODUCTION

1.1 OVER VIEW OF THE PROJECT

Effective financial management is crucial for individuals, businesses, and organizations to achieve stability, security, and growth. One essential aspect of financial management is tracking expenses, which enables individuals to monitor their spending, identify areas for cost reduction, and make informed financial decisions. However, manual expense tracking can be time-consuming, prone to errors, and often neglected.

To address these challenges, an automated expense tracker can provide a convenient, accurate, and reliable solution. An expense tracker allows users to record, categorize, and analyze their expenses, set budgets, and generate reports. By leveraging technology to streamline expense tracking, individuals can optimize their financial management, reduce financial stress, and achieve their long-term financial goals.

This project aims to design and develop a user-friendly expense tracker web application that enables individuals to efficiently manage their expenses. The application will provide features such as expense categorization, budgeting, reporting, and data visualization to facilitate informed financial decision-making. By developing an intuitive and automated expense tracker, this project seeks to contribute to the financial well-being and stability of individuals.

2. SYSTEM ANALYSIS

Analysis is a detailed study of various operations performed by a system and their relationships within and outside the system. One aspect of analysis is defining the boundaries of the system and determining whether user system should consider other related systems. During analysis data is collected on the available files decision points and transactions handles by the present system.

2.1 EXISTING SYSTEM

The existing system is a manual expense tracking process that relies on spreadsheets, paper receipts, and manual data entry.

If we want to balance a income and expense for each month we have to do it manually but we can’t do this for each and every month those who have a lot of income and expenses, so to reduce the stress for the person and make easy to calculate the income and expense, this python and django web application has been so much helpful for a person to avoid the manual way calculating his income and expenses.

In this application we have features like add expenses categories add income so that we can add what are the income and expenses has been done for a month. But it can used to perform calculation on income and expenses to overcome this problem we propose the new application.

Demerits of existing system

- There's a risk of making errors when tracking manually.
- While free options exist, more feature-rich expense trackers often require subscriptions, which can be a financial burden for some users
- While free options exist, more feature-rich expense trackers often require subscriptions, which can be a financial burden for some users
- Many users may initially use an expense tracker but lose interest or forget to record spending, leading to inaccurate or incomplete data.
- Some expense trackers may lack advanced reporting

2.2 PROPOSED SYSTEM

In this Python and Django application we are going to develop this by adding some extra features. In expenses we have some expense feature like add expenses we can add new expense for a month, add categories , export expenses (it will remain as a specific date how much expense has taken for a month), remove export files (it will remove the remainder for the month), view expenses (we can view what are all expense for a month). In income tracker some of the features like add income, add category, categories, export income, remove export files, and view expenses. In add income (we can add new income for a month), add categories (we can add new categories for a month), export income (it will remind us in a date what we have given i.e. from date and to date), remove export files (it will remove the remainder what we have given to remind). By adding these features it helps the users to work more efficient and in an effective manner.

ADVANTAGES OF PROPOSED SYSTEM

- We can add Expense.
- We can add category.
- We can Record our changes.
- We can Export our income.
- We can remove the exported files.
- We can easily edit our data easily.

3. SYSTEM SPECIFICATION

3.1 HARDWARE SPECIFICATION

• Processor	:	Pentium Core 2 Duo 2.88 GHZ
• System RAM	:	4GB
• Hard disk drive	:	160 GB
• Keyboard	:	Logitech
• Mouse	:	Lenovo
• Monitor	:	15 inch Color Monitor

3.2 SOFTWARE SPECIFICATION

• Operating System	:	Windows 10
• Front End	:	PYTHON
• Back End	:	DJANGO
• DATABASE	:	SQL LITE

4. SYSTEM OVERVIEW

MODULES

- User Module
- Income Module
- Expense Module
- Admin Module

4.1 MODULE DESCRIPTION

USER MODULE:

The User Module is a critical component of the expense tracker system, providing a secure and user-friendly platform for individuals to register, manage their financial transactions, and monitor their financial health. This module ensures that users can efficiently track their income and expenses while maintaining the highest levels of security and data privacy. The integration of secure authentication mechanisms ensures that only authorized users have access to their financial data, preventing unauthorized modifications or breaches. All user activities, such as registration, login, and financial entries, are securely recorded, ensuring transparency and data integrity.

User Registration and Login: Users register by providing their email and a secure password, ensuring unique identification. Once registered, users can log in securely using multi-factor authentication (optional), allowing them to access their financial dashboard. The system ensures that only verified users can manage their transactions, safeguarding data security.

Profile Management: Once logged in, users can update their profile details, including:

- Personal information (name, contact details, etc.).
- Financial goals (monthly savings target, spending limit, etc.).
- Preferred currency and financial preferences.

Dashboard Access: Users gain access to an interactive dashboard, where they can:

- View a summary of income, expenses, and savings balance.
- Analyze their spending trends over time.
- Get personalized financial insights based on their spending habits.

Key Features:

- ➢ Secure user registration and login with email authentication.
- ➢ Profile management with financial goal tracking.
- ➢ Interactive dashboard displaying income, expenses, and savings.
- ➢ Secure database storage ensuring user data integrity.

INCOME MODULE

The Income Module allows users to log and track their earnings efficiently. It ensures that all income records are stored securely and can be accessed for financial planning and reporting. The system provides users with a comprehensive breakdown of their income sources and allows them to analyze income trends over time.

Income Entry: Users can log their earnings with the following details:

- Amount earned
- Income source (Salary, Business, Freelance, etc.)
- Transaction date
- Payment method (Bank Transfer, Cash, etc.)

Income History and Analysis:

- Users can view and filter their income records based on date, source, or payment method.
- The system generates reports summarizing income patterns, allowing users to make informed financial decisions.

Key Features:

- ➢ Easy income entry with source and date tracking.
- ➢ Categorized income records for better financial management.
- ➢ Real-time income tracking with insights and trend analysis.

EXPENSE MODULE

The Expense Module is designed to help users efficiently record and monitor their expenses. It provides categorization options to organize expenses and offers data visualization tools for better financial planning.

Expense Categories: To help users track spending, the module provides predefined and customizable categories, including:

- Essential Expenses (Rent, Bills, Groceries)
- Discretionary Expenses (Dining Out, Shopping, Entertainment)
- Investments & Savings

Expense Analysis and Visualization: Users can analyze their spending with real-time graphical reports, including:

- Pie Charts showing category-wise spending.
- Bar Graphs comparing income vs. expenses.
- Trend Graphs showing spending habits over time.

Key Features:

- Secure and categorized expense tracking.
- Multiple payment methods support.
- Visual insights via charts and reports.
- Filtering and exporting expense history.

ADMIN MODULE

The Admin Module serves as the backbone of the expense tracker system, overseeing user activities and ensuring data integrity. Admins have access to user transactions, providing oversight and system security.

User and Transaction Management:

Admins can:

- View, verify, and manage user accounts and financial data.
- Monitor income and expense records to ensure compliance with system rules.
- Manage categories and system configurations for accurate tracking.

Financial Data Oversight:

- The admin can analyze overall system transactions to detect any anomalies. Provides a secure audit log of user activities.

Security and System Integrity:

- The system ensures encrypted data storage, protecting user financial records.
- Only authorized admins can make system-wide changes.
- User authentication logs are maintained for transparency.

Key Features:

- ➢ Complete oversight of user registrations and transactions.
- ➢ Audit log for financial security and compliance.
- ➢ Encrypted database to prevent data breaches.
- ➢ Ability to manage system settings for a seamless user experience.

4.2 SOFTWARE DESCPRITION:

HTML:

HTML (Hyper Text Markup Language). HTML is the backbone of every webpage, providing the fundamental structure of web content. It defines different elements such as headings, paragraphs, images, links, buttons, tables, and forms using a system of tags. Without HTML, a webpage would simply be a blank screen, as it is responsible for organizing and presenting textual and multimedia content. HTML follows a hierarchical structure, ensuring that elements are properly nested and displayed in a logical order. Additionally, modern HTML versions, such as HTML5, have introduced new elements like <article>, <section>, and <video>, allowing for better organization and multimedia integration. HTML also supports semantic elements, which improve accessibility and SEO (Search Engine Optimization), making web pages easier to understand for search engines and assistive technologies like screen readers.

CSS:

CSS (Cascading Style Sheets). CSS is a styling language that enhances the appearance of web pages by applying various design elements to HTML content. It controls colors, fonts, spacing, borders, background images, and positioning of elements. Without CSS, web pages would look plain and unformatted, displaying only raw text and images without any styling. CSS enables web designers to create visually appealing and responsive layouts that adapt to different screen sizes, ensuring a seamless experience on desktops, tablets, and mobile devices. With the introduction of CSS3, developers can now use advanced features such as gradients, shadows, animations, and transitions to enhance user experience. CSS frameworks like Bootstrap and Tailwind CSS further simplify web development by providing pre-designed components and layout systems, allowing for faster and more efficient styling.

JAVASCRIPT

JavaScript (JS). JavaScript is a powerful programming language that adds interactivity and dynamic behavior to web pages. While HTML provides structure and CSS enhances visual appeal, JavaScript enables users to interact with a webpage in real time. It is used for tasks such as form validation, user input handling, animations, dynamic content updates, and asynchronous communication with servers via APIs (Application Programming Interfaces). JavaScript plays a crucial role in modern web applications, allowing for the creation of interactive features like drop-down menus, image sliders, live chat functionalities, and single-page applications (SPAs). With the development of frameworks and libraries like React.js, Vue.js, and Angular, JavaScript has become even more powerful, enabling developers to build highly scalable and efficient web applications.

PYTHON

- Python is a popular programming language. Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data.
- Python can be used for rapid prototyping, or for production-ready software development.

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.
- Python can be used for a wide range of applications, including web development (using frameworks like Flask), data analysis (using libraries like NumPy), artificial intelligence, and scientific computing.
- Python has a vast ecosystem of libraries and frameworks that simplify common tasks and enable developers to build complex applications efficiently.

Some popular libraries include:

- **NumPy:** For numerical computing and scientific computing.
- **Scikit-learn:** For machine learning tasks.
- **Tensor Flow:** For deep learning.
- **Keras**: For building and training neural networks.

DJANGO

- Django was invented by Lawrence Journal-World in 2003, to meet the short deadlines in the newspaper and at the same time meeting the demands of experienced web developers.
- Django is a Python framework that makes it easier to create web sites using Python.
- Django takes care of the difficult stuff so that you can concentrate on building your web applications.
- Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).

Model - The data you want to present, usually data from a database.

View - A request handler that returns the relevant template and content - based on the request from the user.

Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

The model provides data from the database:

- In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it easier to work with databases.
- The most common way to extract data from a database is SQL. One problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.
- Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.
- A view is a function or method that takes http requests as arguments, imports the relevant model(s), and finds out what data to send to the template, and returns the final result.

SQL LITE:

SQLite is a lightweight, self-contained, server less, and zero-configuration relational database management system (RDBMS) that stores data in a single file, making it ideal for embedded applications and local storage

5. SYSTEM DESIGN

5.1 DESIGN CONCEPT

The expense tracker design concept is centered around simplicity, ease of use, and visual clarity. The application will have a clean and intuitive interface that allows users to quickly and easily track their expenses.

Layout:

- The application will have a responsive design that adapts to different screen sizes and devices.
- The layout will be divided into sections, each with a clear heading and concise content.
- The navigation menu will be located at the top of the screen and will include links to the dashboard, expenses, budgets, and reports.

Dashboard:

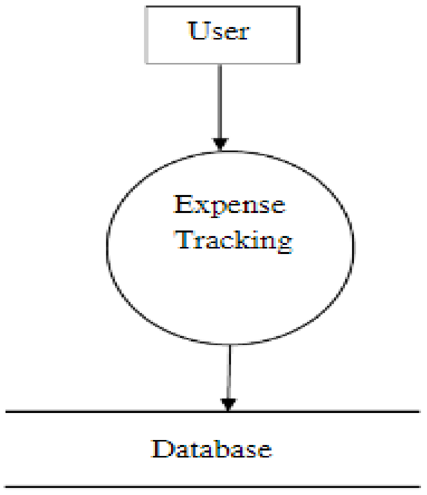
- The dashboard will provide an overview of the user's expenses, including a summary of their spending by category and a list of recent expenses.
- The dashboard will also include a chart or graph that shows the user's spending trends over time.

Expenses:

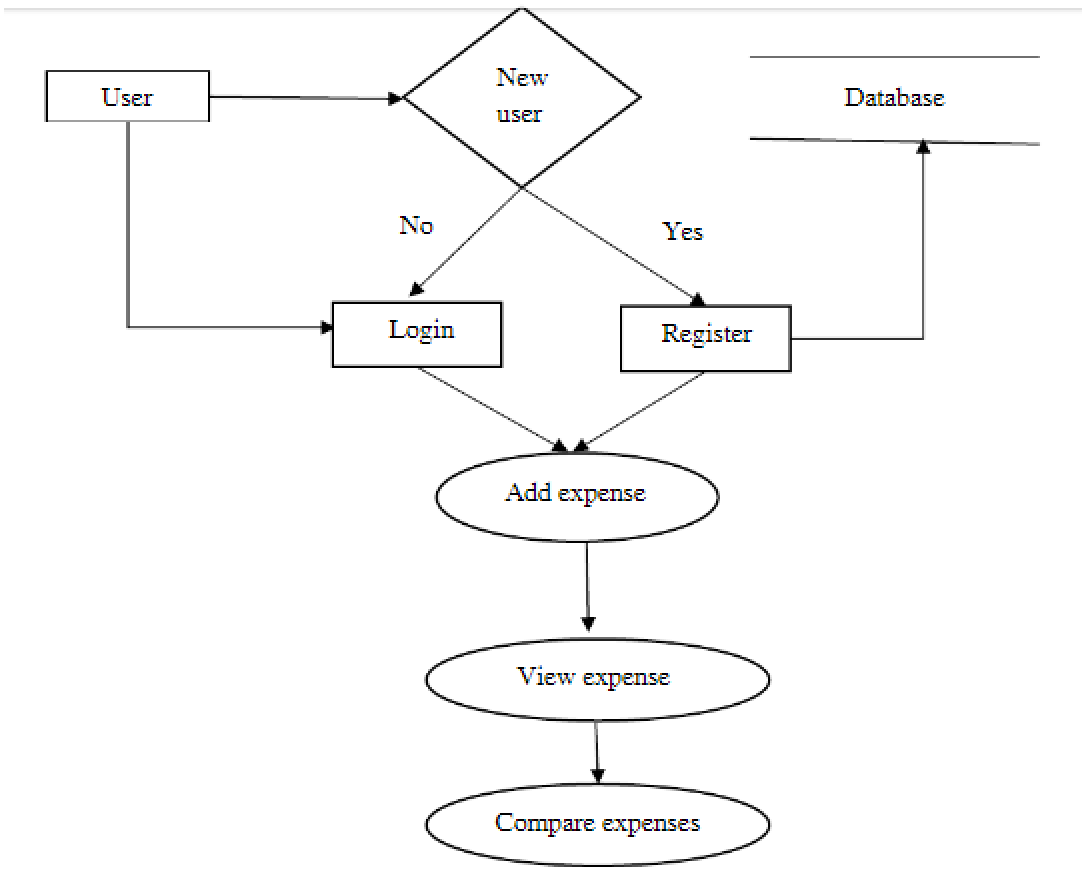
- The expenses section will allow users to add, edit, and delete expenses.
- Each expense will include a date, amount, category, and description.

5.2 DESIGN PROCESS

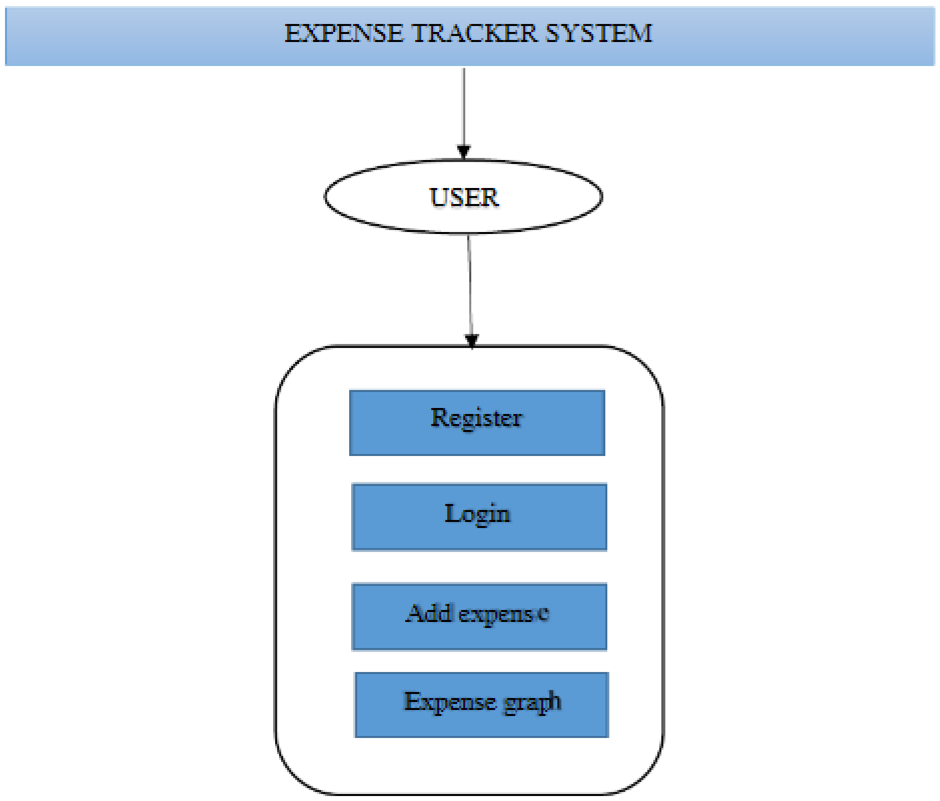
LEVEL 0



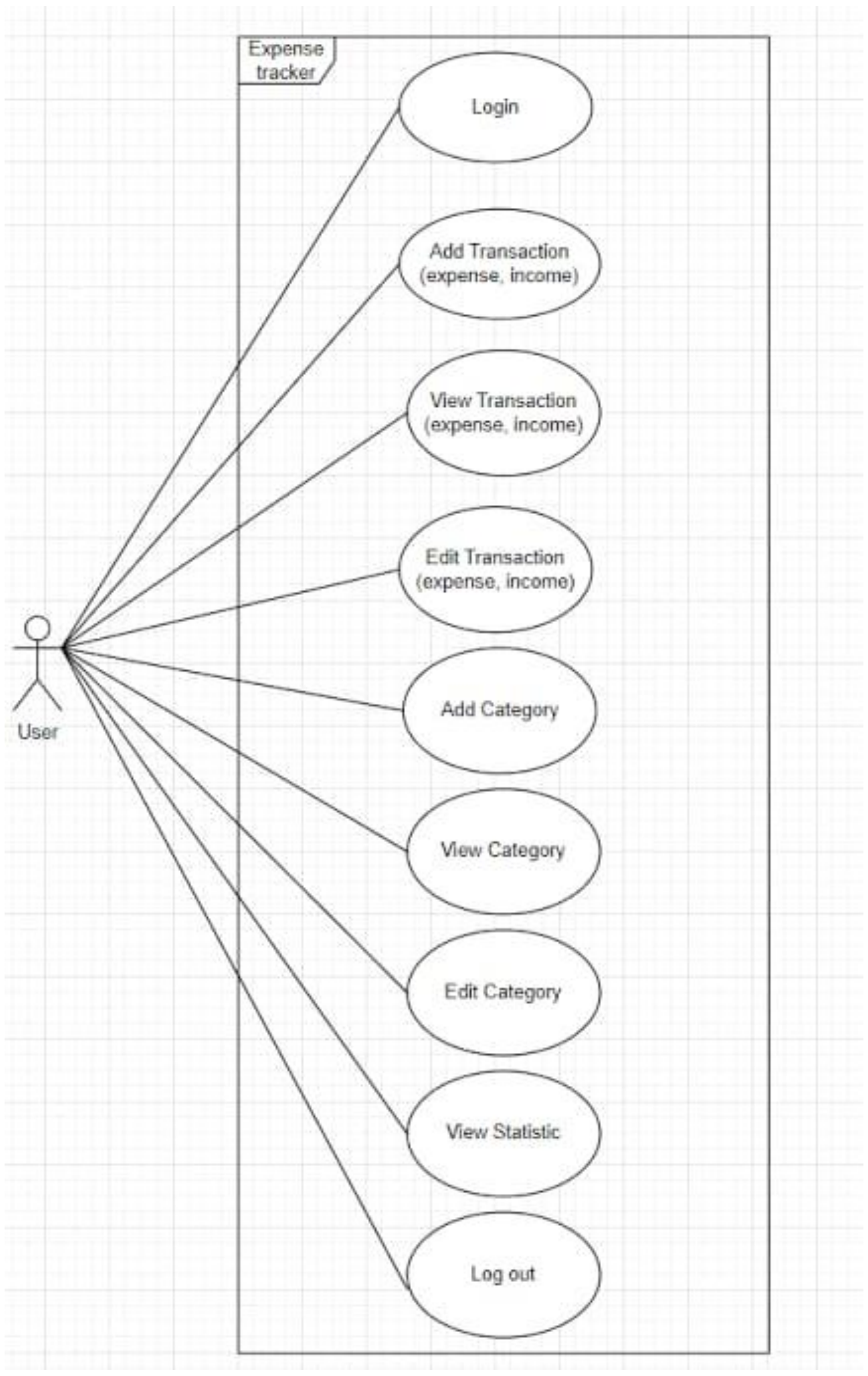
LEVEL 1

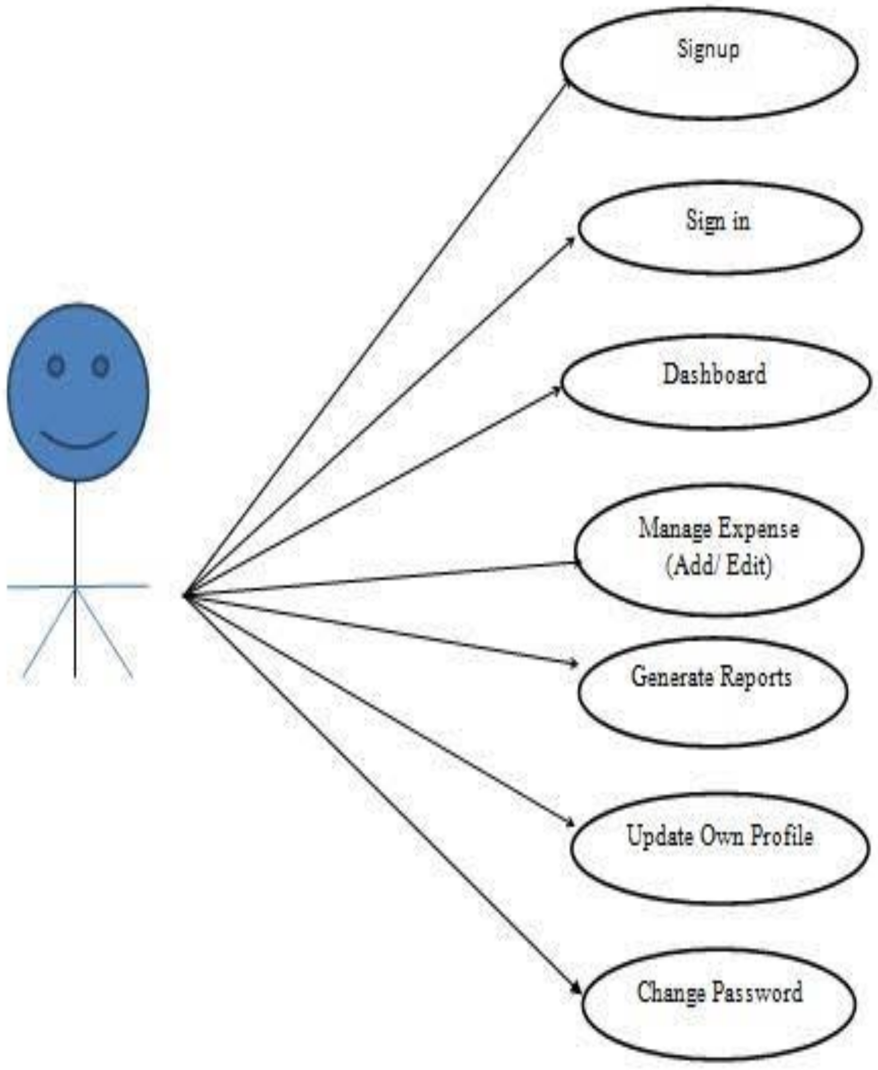


LEVEL 2


EXPENSE TRACKER SYSTEM
USER
Register
Login
Add expense
Expense graph

5.3 ER DIAGRAM


Expense tracker
Login
Add Transaction (expense, income)
View Transaction (expense, income)
Edit Transaction (expense, income)
Add Category
View Category
Edit Category
View Statistic
Log out
User


Signup
Sign in
Dashboard
Manage Expense (Add/ Edit)
Generate Reports
Update Own Profile
Change Password

6. SYSTEM TESTING AND IMPLEMENTATION

6.1 SOFTWARE TESTING METHODOLOGIES

The testing process ensures that the Expense Tracker System meets all functional requirements, is free of defects, and operates efficiently across different environments. A structured quality control strategy is implemented to validate the system against the predefined requirements. The testing framework is developed based on industry best practices to ensure accuracy, reliability, and usability.

Types of Tests

Unit Testing:

Unit testing is performed on individual components of the system to ensure that each function operates as expected. This involves testing functions such as expense addition, deletion, modification, and report generation. Every method and logical condition within the application is validated to ensure that correct outputs are produced for given inputs. Unit testing is conducted before integrating components into the full system.

Functional Testing:

Functional testing verifies that the system functions correctly as per business and technical requirements. This includes testing various functionalities like user registration, login, adding expenses, updating expenses, and generating reports.

The following key aspects are tested:

- **Valid Input:** Ensuring that correctly formatted data is accepted (e.g., numeric values for expense amounts).
- **Invalid Input:** Identifying and rejecting incorrect data entries (e.g., negative amounts, empty fields).
- **Functionality Testing:** Validating that all features, such as filtering expenses, exporting reports, and setting budget limits, work as intended.
- **Output Verification:** Ensuring correct calculations and display of expenses and financial summaries.

Integration Testing:

Integration testing ensures that different modules, such as the database, frontend, and backend services, communicate effectively. It verifies that data flows correctly between the user interface and the database, confirming that expense records are stored and retrieved accurately.

Performance Testing:

Performance testing is conducted to assess the application's responsiveness and stability under different loads. The system is tested with varying numbers of users and transactions to ensure it can handle concurrent operations without delays or crashes.

Security Testing:

Security testing is performed to identify vulnerabilities, ensuring that user data is protected. Authentication mechanisms are tested to prevent unauthorized access, and encryption methods are evaluated to secure financial data.

6.2 SYSTEM IMPLEMENTATION

The Expense Tracker System has been implemented as a secure, interactive, and user-friendly web application using Python and Flask for the backend and HTML, CSS, and JavaScript for the frontend. The database is managed using SQLite, ensuring efficient storage and retrieval of expense records. The frontend provides an intuitive user interface with smooth navigation and interactive elements. Users can register, log in, add, edit, and delete expenses effortlessly. The system incorporates form validation and error handling to prevent invalid entries. The expense dashboard provides a graphical representation of spending trends, improving financial awareness. On the backend, the django framework manages API requests, authentication, and data processing. The RESTful API ensures seamless communication between the frontend and the database. Secure session management and encryption mechanisms are used to protect sensitive user data. The implementation also includes automated data backups to prevent data loss.

7. CONCLUSION

The Expense Tracker System is a robust, efficient, and secure financial management tool designed to help users track their expenses with ease. The implementation of modern web technologies ensures a seamless user experience while maintaining high levels of security and data integrity. The system provides key functionalities such as expense tracking, categorization, report generation, and real-time graphical analysis, enabling users to monitor and manage their spending effectively. Through rigorous testing methodologies, the system has been validated for performance, security, and accuracy, ensuring its reliability in real-world scenarios. Overall, the Expense Tracker System offers a structured approach to financial planning, helping users gain better insights into their spending habits. With its intuitive interface, automated data handling, and cloud-based deployment, the system stands as an innovative solution for personal and small-business financial management. The integration of secure authentication mechanisms and data encryption further strengthens the system's credibility, making it a reliable and scalable tool for users looking to maintain financial discipline and achieve better financial stability.

8. FUTURE ENHANCEMENTS

The Expense Tracker System can be further enhanced by integrating artificial intelligence (AI) and machine learning (ML) to provide users with smarter financial management tools. These technologies can help predict future spending patterns, automate expense categorization, and offer personalized budgeting suggestions. AI-driven insights would allow the system to detect unusual spending behavior, warn users about potential overspending, and help them plan their finances more effectively. This would make the platform even more intuitive, providing users with actionable data to make informed financial decisions.

Additionally, expanding the system’s accessibility across multiple platforms is a key area for future development. A dedicated mobile application would allow users to track their finances on the go, while the addition of voice assistants and chatbots could further streamline the user experience. The system could also benefit from integration with financial institutions, allowing for seamless synchronization of transactions from bank accounts and digital wallets. These enhancements would not only improve the user experience but also increase the system's efficiency and security, making it an even more reliable tool for managing personal and business finances.

9. BIBLIOGRAPHY

REFERENCES:

- Gupta, R., & Sharma, N. (2020). "Design and Development of an Automated Expense Tracker Application." International Journal of Computer Applications, 182(11), 1-6.
- Anderson, C., & Taylor, S. (2019). "The Impact of Digital Financial Tools on Personal Finance Management." Journal of Financial Technologies, 12(2), 45-60.
- Lee, J. K., & Choi, H. S. (2021). "AI-Driven Financial Planning: A Case Study in Personal Expense Management." Journal of Artificial Intelligence in Finance, 7(3), 23-39.
- Kumar, P., & Singh, A. (2020). "Blockchain for Financial Data Security in Expense Tracking Systems." International Journal of Blockchain and Cryptocurrency, 8(4), 102-115.
- Patel, R., & Mishra, S. (2021). "Cloud-Based Financial Management Solutions: Opportunities and Challenges." International Journal of Cloud Computing and Services, 15(1), 75-90.
- Sato, T., & Watanabe, K. (2018). "Financial Literacy and Mobile Application Tools: An Evaluation of Financial Tracking Systems." Journal of Financial Education, 43(4), 60-72.

WEBSITES:

- **www.w3school python.com**
- **www.w3schooldjango.com**
- **Django Forms Documentation**
- **Django Models Documentation**
- **Django Documentation**
- **Django for Beginners**
- **Django Full Tutorial on YouTube**

PAPER PUBLISHED:

CUSTOMER STATISFACTION AND RESPONSE TIMES : ANALYZING ON-ROAD VECHICLE BREAKING DOWN SERVICES USING LIVE MECHANIC SUPPORT

10.APPENDIX

10.1 SCREENSHOTS

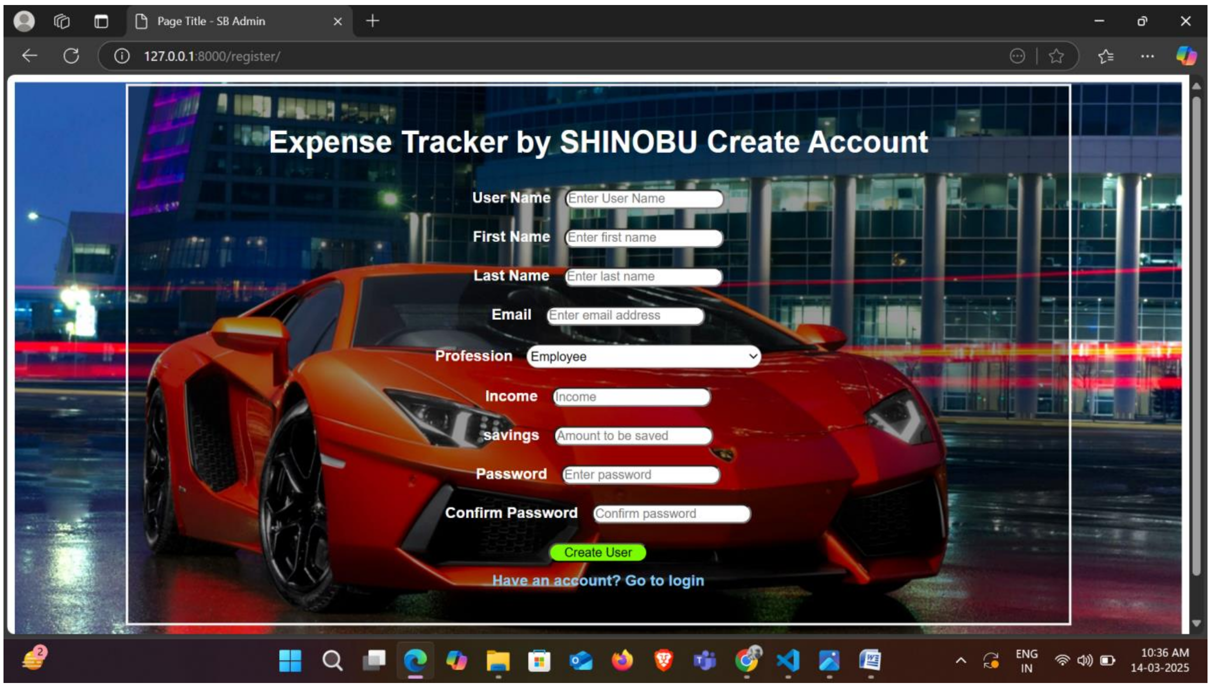


Figure 1. SIGN UP PAGE

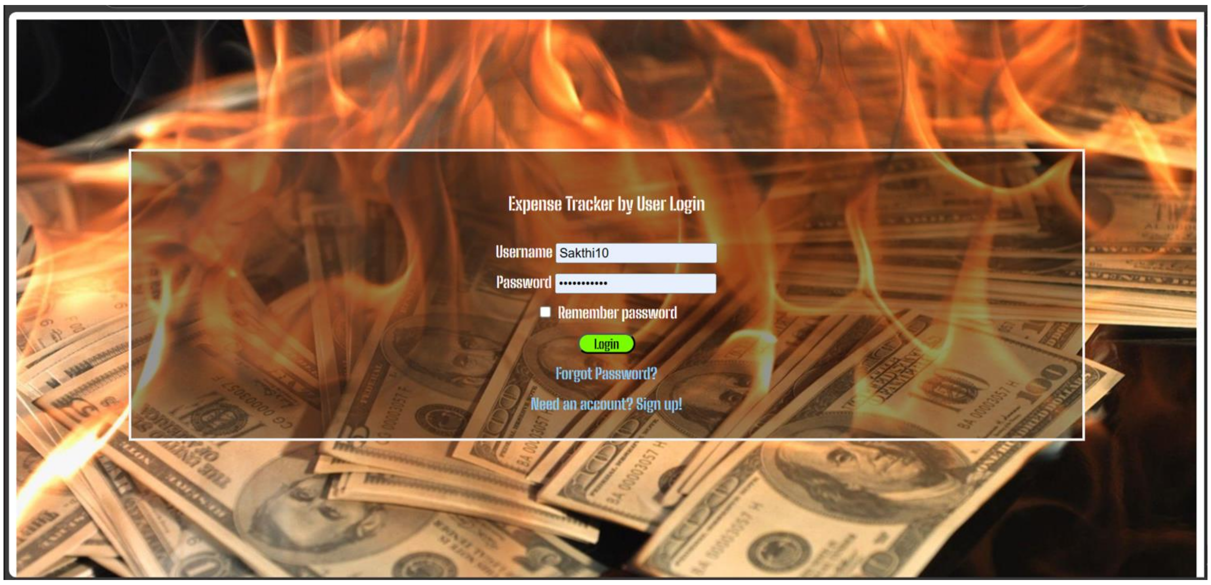
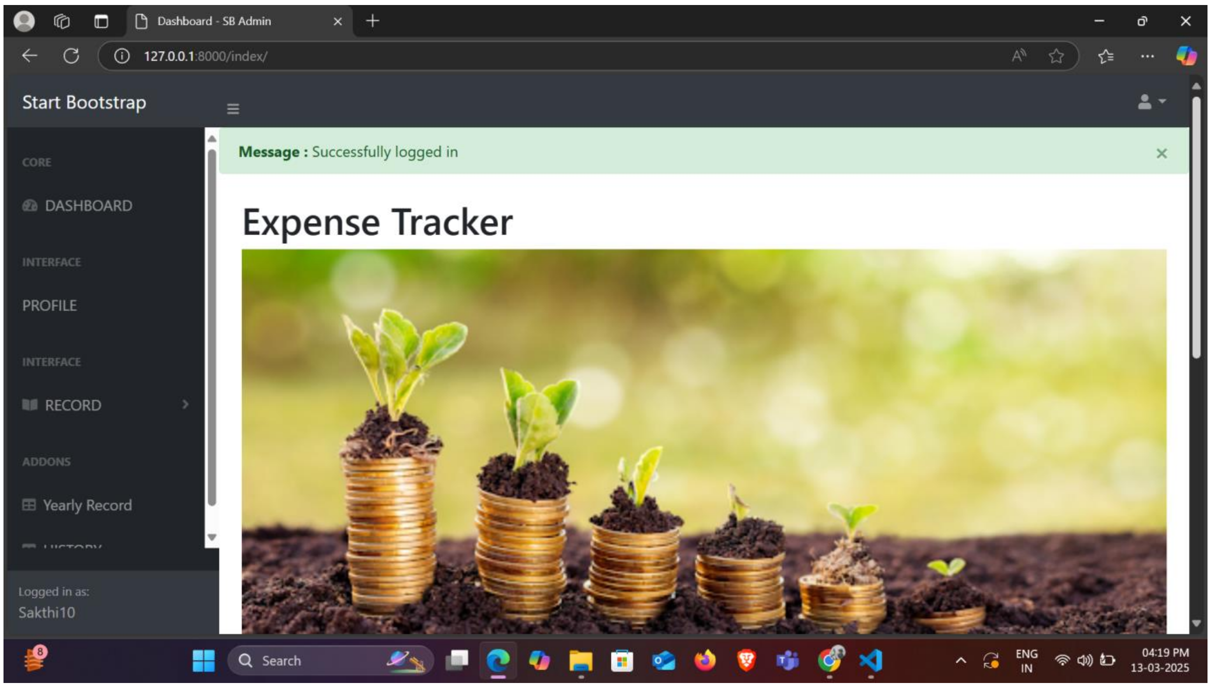


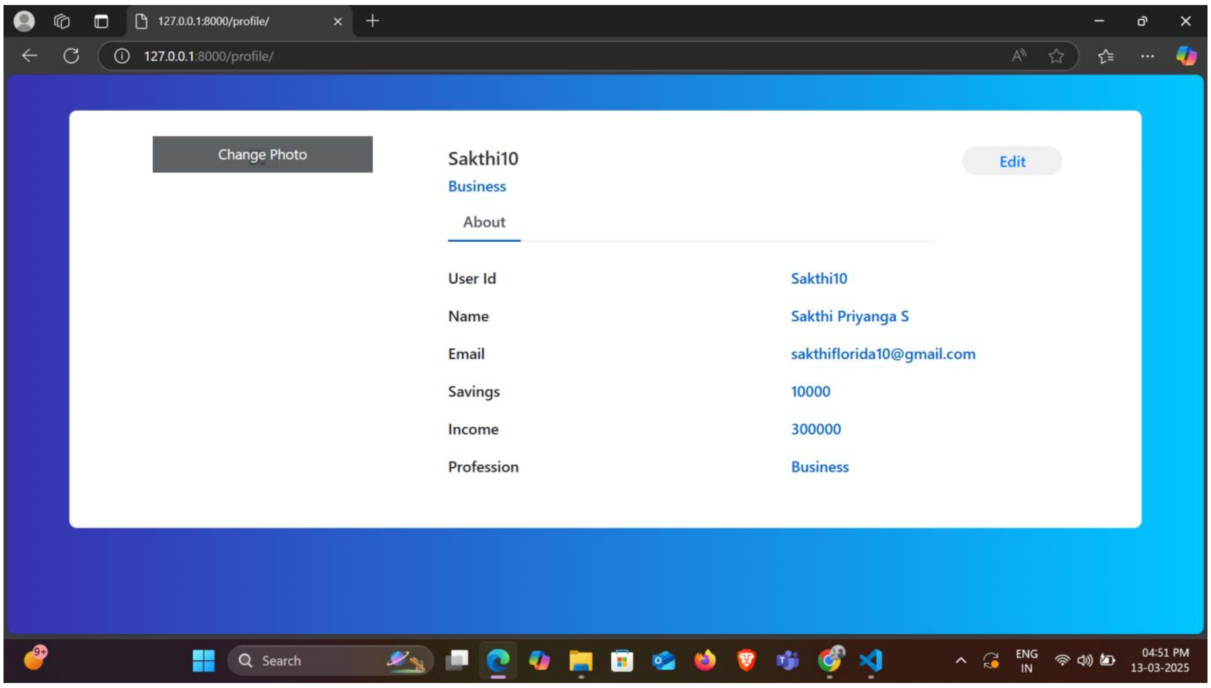
Figure 2. LOGIN PAGE



Message : Successfully logged in

Expense Tracker

Figure 3. HOME PAGE



Sakthi10
Business

User Id	Sakthi10
Name	Sakthi Priyanga S
Email	sakthiflorida10@gmail.com
Savings	10000
Income	300000
Profession	Business

Figure 4. USER PROFILE PAGE

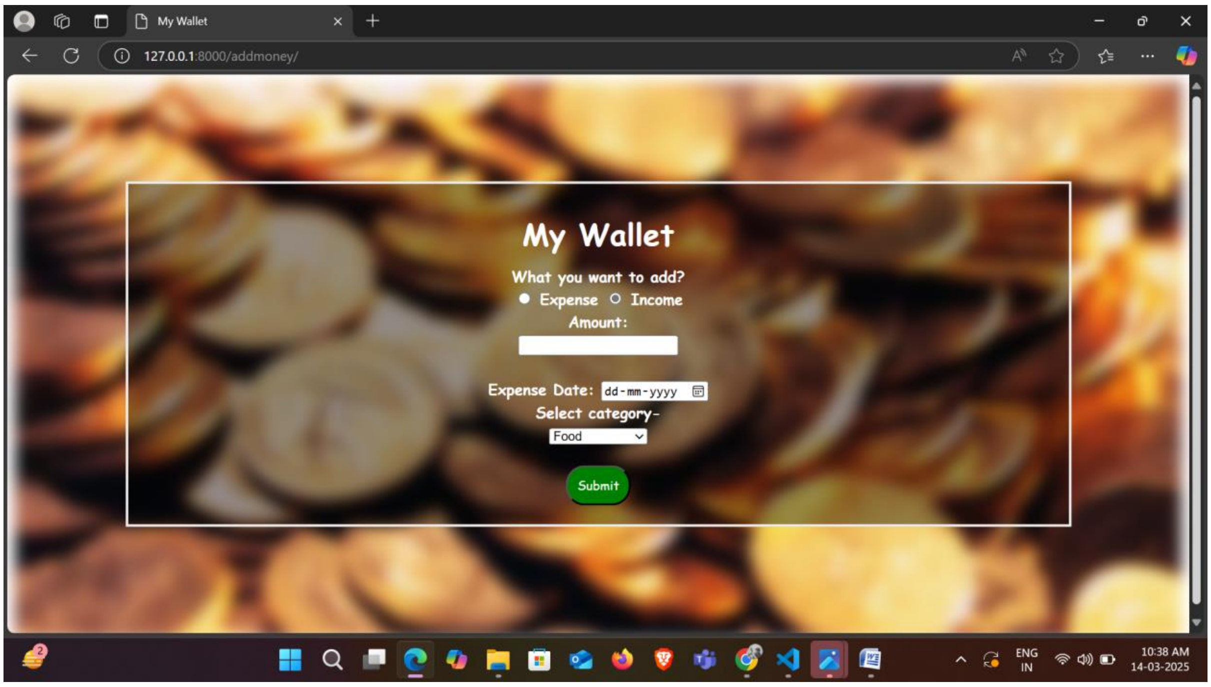
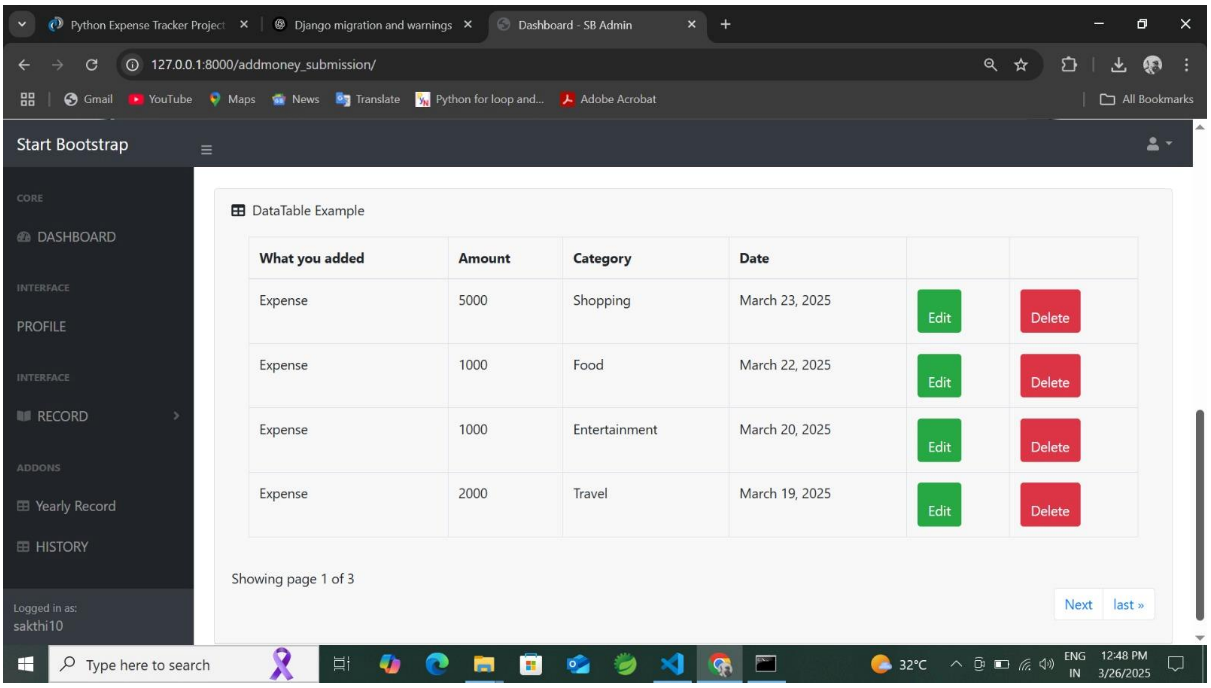


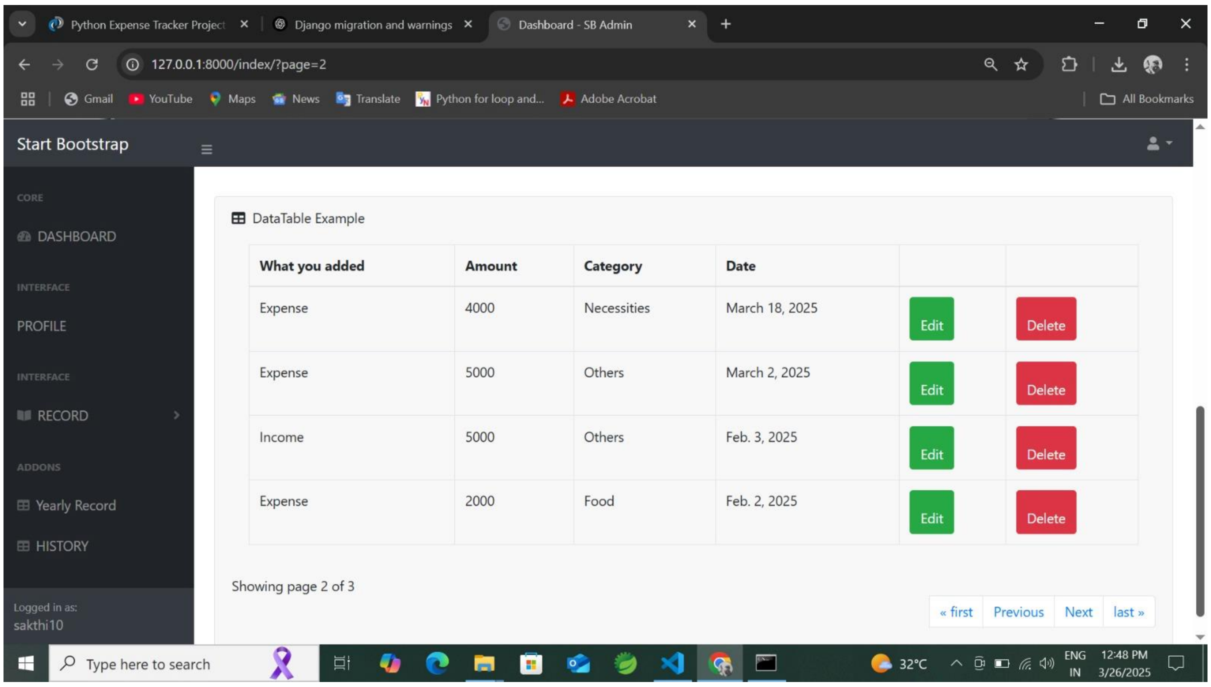
Figure 5. MY WALLET



What you added	Amount	Category	Date		
Expense	5000	Shopping	March 23, 2025	Edit	Delete
Expense	1000	Food	March 22, 2025	Edit	Delete
Expense	1000	Entertainment	March 20, 2025	Edit	Delete
Expense	2000	Travel	March 19, 2025	Edit	Delete

Showing page 1 of 3

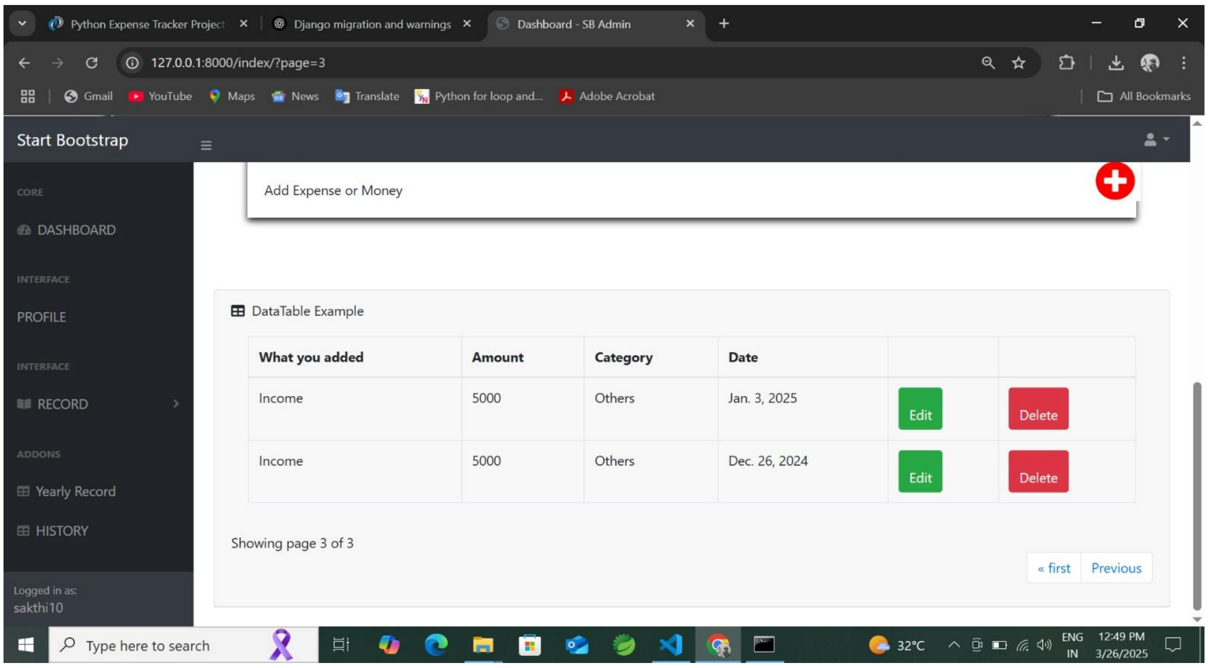
Figure 6. ADDING EXPENSE 1



What you added	Amount	Category	Date		
Expense	4000	Necessities	March 18, 2025	Edit	Delete
Expense	5000	Others	March 2, 2025	Edit	Delete
Income	5000	Others	Feb. 3, 2025	Edit	Delete
Expense	2000	Food	Feb. 2, 2025	Edit	Delete

Showing page 2 of 3

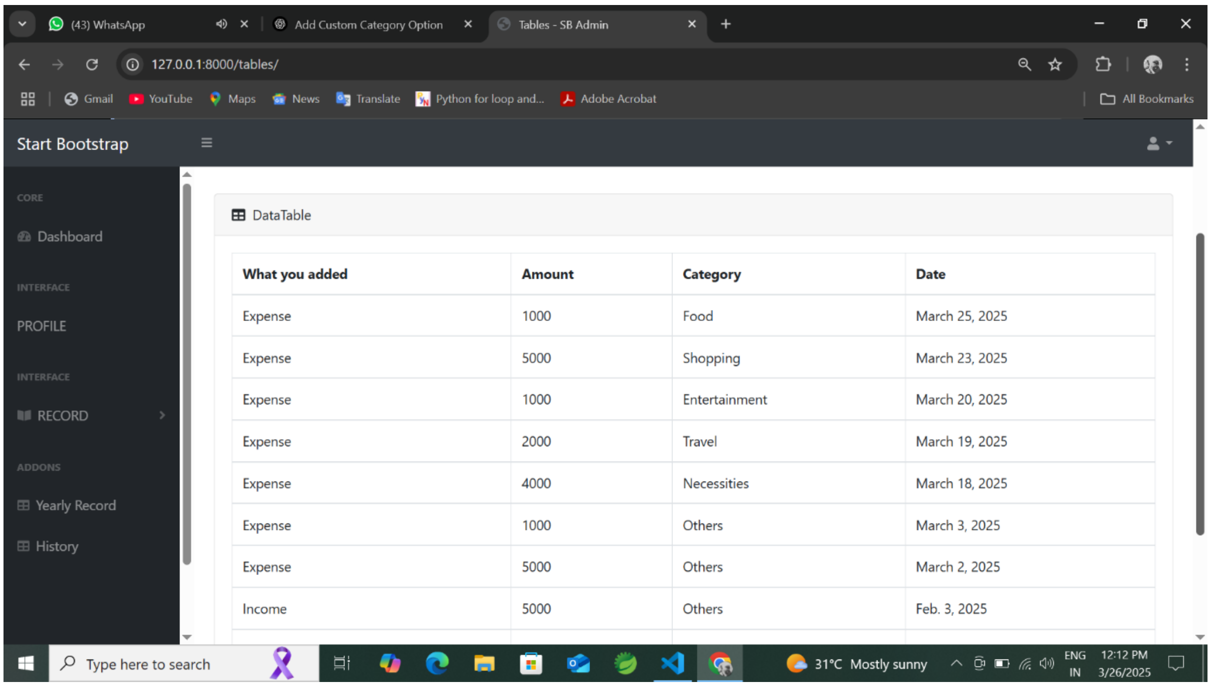
Figure 7. ADDING EXPENSE 2



What you added	Amount	Category	Date		
Income	5000	Others	Jan. 3, 2025	Edit	Delete
Income	5000	Others	Dec. 26, 2024	Edit	Delete

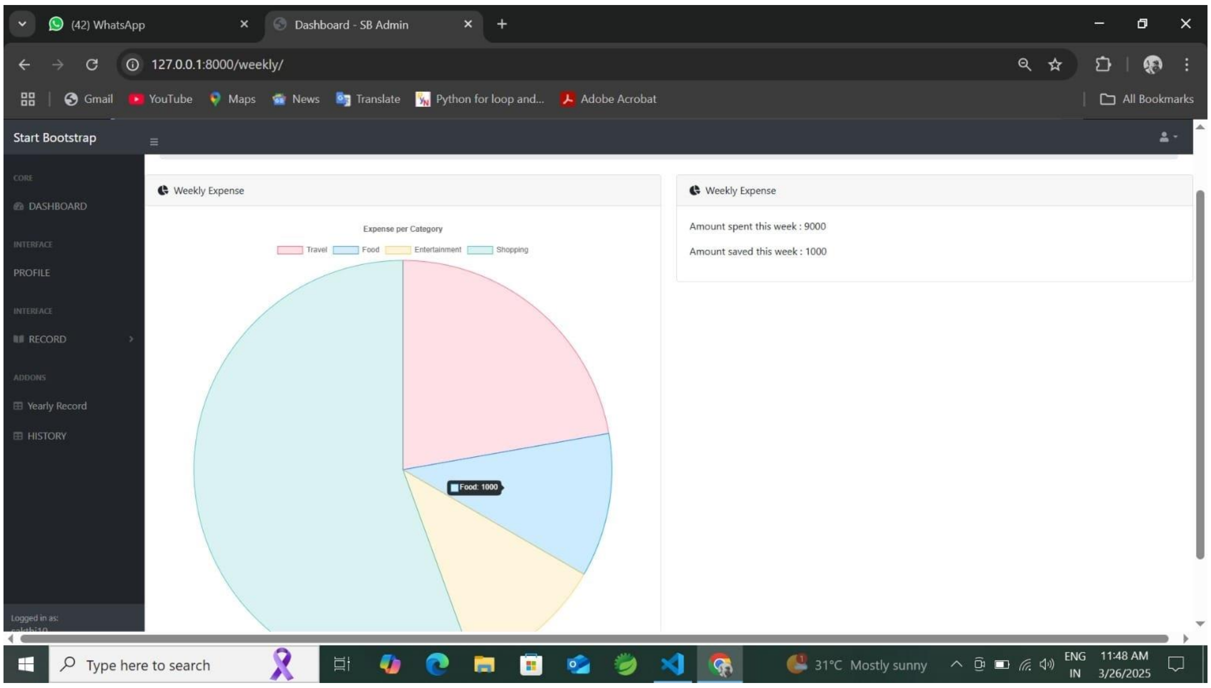
Showing page 3 of 3

Figure 8. ADDING EXPENSE 3



What you added	Amount	Category	Date
Expense	1000	Food	March 25, 2025
Expense	5000	Shopping	March 23, 2025
Expense	1000	Entertainment	March 20, 2025
Expense	2000	Travel	March 19, 2025
Expense	4000	Necessities	March 18, 2025
Expense	1000	Others	March 3, 2025
Expense	5000	Others	March 2, 2025
Income	5000	Others	Feb. 3, 2025

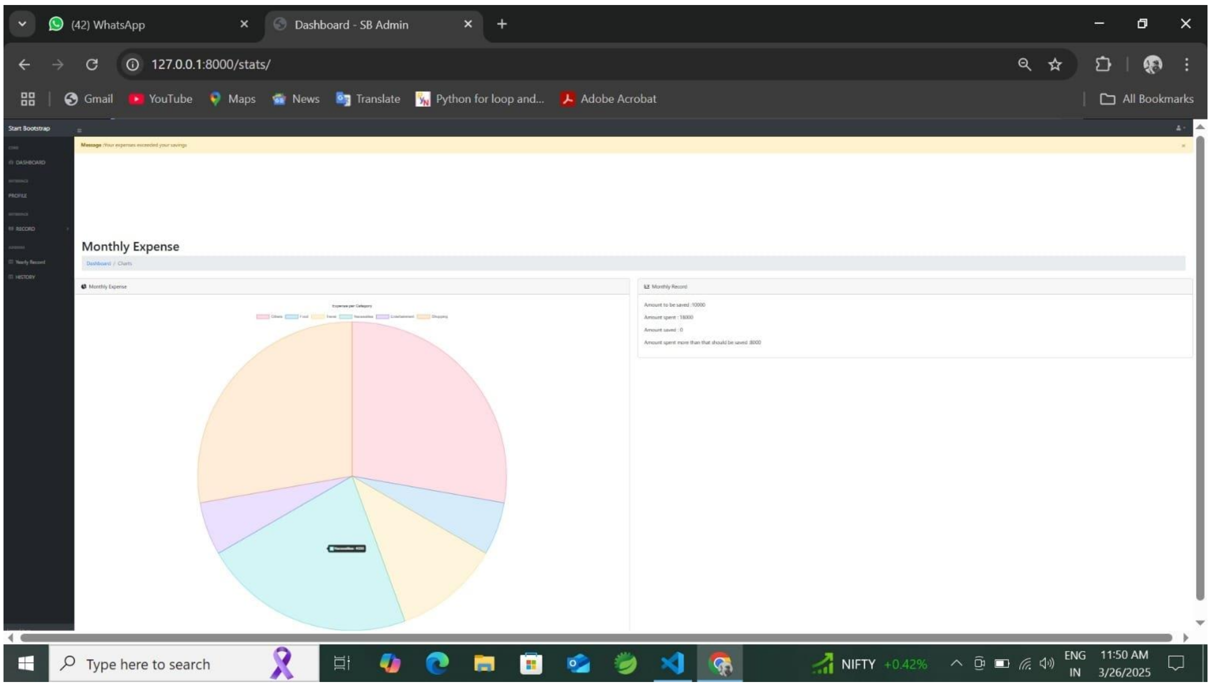
Figure 9. HISTORY



Amount spent this week : 9000

Amount saved this week : 1000

Figure 10: WEEKLY EXPENSE



Message :Your expenses exceeded your savings

Amount to be saved :10000

Amount spent : 18000

Amount saved : 0

Amount spent more than that should be saved :8000

Figure 11. MONTHLY EXPENSE

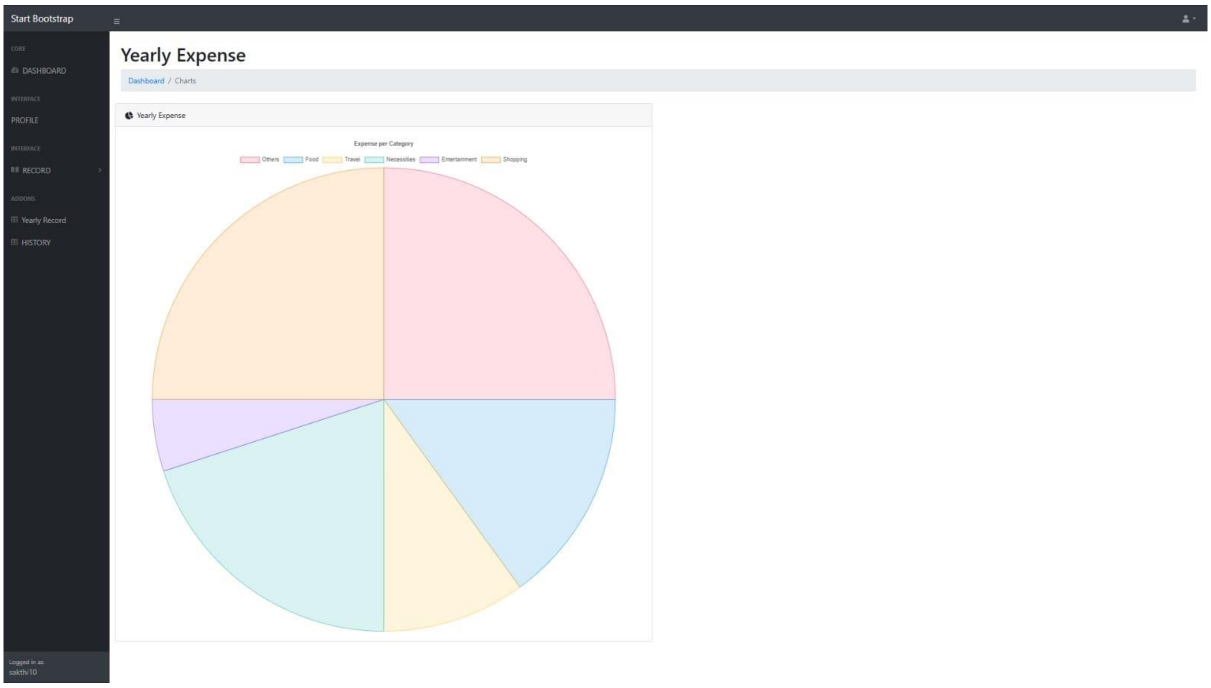


Figure 12. YEARLY EXPENSE

10.2 SAMPLE CODING

```
Pip install django

django-admin startproject ExpenseTracker

python mange.py startapp home
```

MODELS.PY

```
from django.db import models

from django.utils.timezone import now

from django.contrib.auth.models import User

from django.conf import settings

from django.db.models.signals import post_save

from django.dispatch import receiver

from django.db.models import Sum

#Create your models here.

SELECT_CATEGORY_CHOICES = [

    ("Food","Food"),

    ("Travel","Travel"),

    ("Shopping","Shopping"),

    ("Necessities","Necessities"),

    ("Entertainment","Entertainment"),

    ("Other","Other")

 ]

ADD_EXPENSE_CHOICES = [

     ("Expense","Expense"),

     ("Income","Income")

 ]

PROFESSION_CHOICES =[

    ("Employee","Employee"),
    ("Business","Business"),

    ("Student","Student"),

    ("Other","Other")

]

class Addmoney_info(models.Model):

    user = models.ForeignKey(User,default = 1, on_delete=models.CASCADE)

    add_money = models.CharField(max_length = 10 , choices =
ADD_EXPENSE_CHOICES )

    quantity = models.BigIntegerField()

    Date = models.DateField(default = now)

    Category = models.CharField( max_length = 20, choices =
SELECT_CATEGORY_CHOICES , default ='Food')

    class Meta:

        db_table:'addmoney'

class UserProfile(models.Model):

    user = models.OneToOneField(User,on_delete=models.CASCADE)

    profession = models.CharField(max_length = 10, choices=PROFESSION_CHOICES)

    Savings = models.IntegerField( null=True, blank=True)

    income = models.BigIntegerField(null=True, blank=True)

    image = models.ImageField(upload_to='profile_image',blank=True)

    def __str__(self):

       return self.user.username
```

Admin.py

```
# Register your models here.

from .models import Addmoney_info

From django.contrib import admin

class Addmoney_infoAdmin(admin.ModelAdmin):
    list_display=("user","quantity","Date","Category","add_money")
admin.site.register(Addmoney_info,Addmoney_infoAdmin)
from django.contrib.sessions.models import Session
admin.site.register(Session)
from .models import UserProfile
admin.site.register(UserProfile)
```

Urls.py

```
from django.contrib import admin
from django.urls import path
from django.urls import include
from . import views
from django.contrib.auth import views as auth_views
urlpatterns = [
    path('', views.home, name='home'),
    path('index/', views.index, name='index'),
    path('register/',views.register,name='register'),

    path('handleSignup/',views.handleSignup,name='handleSignup'),
    path('handlelogin/',views.handlelogin,name='handlelogin'),
    path('handleLogout/',views.handleLogout,name='handleLogout'),
    path('reset_password/',auth_views.PasswordResetView.as_view(template_name =
"home/reset_password.html"),name='reset_password'),
path('reset_password_sent/',auth_views.PasswordResetDoneView.as_view(template_name="
home/reset_password_sent.html"),name='password_reset_done'),
path('reset/<uidb64>/<token>/',auth_views.PasswordResetConfirmView.as_view(template_n
ame ="home/password_reset_form.html"),name='password_reset_confirm'),
path('reset_password_complete/',auth_views.PasswordResetView.as_view(template_name
="home/password_reset_done.html"),name='password_reset_complete'),

    path('addmoney/',views.addmoney,name='addmoney'),

path('addmoney_submission/',views.addmoney_submission,name='addmoney_submission'),

    path('charts/',views.charts,name='charts'),

    path('tables/',views.tables,name='tables'),

    path('expense_edit/<int:id>',views.expense_edit,name='expense_edit'),

    path('<int:id>/addmoney_update/', views.addmoney_update, name="addmoney_update") ,

    path('expense_delete/<int:id>',views.expense_delete,name='expense_delete'),

    path('profile/',views.profile,name = 'profile'),

    path('expense_month/',views.expense_month, name = 'expense_month'),

    path('stats/',views.stats, name = 'stats'),

    path('expense_week/',views.expense_week, name = 'expense_week'),

    path('weekly/',views.weekly, name = 'weekly'),

    path('check/',views.check,name="check"),

    path('search/',views.search,name="search"),

    path('<int:id>/profile_edit/',views.profile_edit,name="profile_edit"),

    path('<int:id>/profile_update/',views.profile_update,name="profile_update"),

    path('info/',views.info,name="info"),

    path('info_year/',views.info_year,name="info_year"),

]
```

Views.py A:

```
from django.shortcuts import render,HttpResponse,redirect

from django.contrib import messages

from django.contrib.auth import authenticate ,logout

from django.contrib.auth import login as dj_login

from django.contrib.auth.models import User
from .models import Addmoney_info,UserProfile
from django.contrib.sessions.models import Session
from django.core.paginator import Paginator, EmptyPage , PageNotAnInteger
from django.db.models import Sum
from django.http import JsonResponse
import datetime
from django.utils import timezone
```

B:

```
def home(request):
    if request.session.has_key('is_logged'):
        return redirect('/index')
    return render(request,'home/login.html')
   # return HttpResponse('This is home')
def index(request):
    if request.session.has_key('is_logged'):
        user_id = request.session["user_id"]
        user = User.objects.get(id=user_id)
        addmoney_info = Addmoney_info.objects.filter(user=user).order_by('-Date')
        paginator = Paginator(addmoney_info , 4)
        page_number = request.GET.get('page')
        page_obj = Paginator.get_page(paginator,page_number)
        context = {
            # 'add_info' : addmoney_info,
           'page_obj' : page_obj
        }
    #if request.session.has_key('is_logged'):
        return render(request,'home/index.html',context)
    return redirect('home')
```

C:

```
def addmoney(request):
    return render(request,'home/addmoney.html')


def profile(request):
    if request.session.has_key('is_logged'):
        return render(request,'home/profile.html')
    return redirect('/home')
def profile_edit(request,id):
    if request.session.has_key('is_logged'):
        add = User.objects.get(id=id)
        return render(request,'home/profile_edit.html',{'add':add})
    return redirect("/home")
```

D:

```
def profile_update(request,id):
    if request.session.has_key('is_logged'):
        if request.method == "POST":
            user = User.objects.get(id=id)
            user.first_name = request.POST["fname"]
            user.last_name = request.POST["lname"]
            user.email = request.POST["email"]
            user.userprofile.Savings = request.POST["Savings"]
            user.userprofile.income = request.POST["income"]
            user.userprofile.profession = request.POST["profession"]
            user.userprofile.save()
            user.save()
            return redirect("/profile")
    return redirect("/home")
```

E:

```
def handleSignup(request):
    if request.method =='POST':
            # get the post parameters
            uname = request.POST["uname"]
            fname=request.POST["fname"]
            lname=request.POST["lname"]
            email = request.POST["email"]
            profession = request.POST['profession']
            Savings = request.POST['Savings']
            income = request.POST['income']
            pass1 = request.POST["pass1"]
            pass2 = request.POST["pass2"]
            profile = UserProfile(Savings = Savings,profession=profession,income=income)
            # check for errors in input
            if request.method == 'POST':
                try:
                    user_exists = User.objects.get(username=request.POST['uname'])
                    messages.error(request," Username already taken, Try something else!!!")
                    return redirect("/register")
                except User.DoesNotExist:
```

```
            if len(uname)>15:
                messages.error(request," Username must be max 15 characters, Please try
again")
                return redirect("/register")


            if not uname.isalnum():
                messages.error(request," Username should only contain letters and numbers,
Please try again")
                return redirect("/register")


            if pass1 != pass2:
                messages.error(request," Password do not match, Please try again")
                return redirect("/register")


        # create the user
        user = User.objects.create_user(uname, email, pass1)
        user.first_name=fname
        user.last_name=lname
        user.email = email
        # profile = UserProfile.objects.all()


        user.save()
        # p1=profile.save(commit=False)
        profile.user = user
        profile.save()
        messages.success(request," Your account has been successfully created")
        return redirect("/")
    else:
        return HttpResponse('404 - NOT FOUND ')
    return redirect('/login')


def handlelogin(request):
    if request.method =='POST':
    # get the post parameters
    loginuname = request.POST["loginuname"]
    loginpassword1=request.POST["loginpassword1"]
    user = authenticate(username=loginuname, password=loginpassword1)
    if user is not None:
        dj_login(request, user)
        request.session['is_logged'] = True
        user = request.user.id
        request.session["user_id"] = user
        messages.success(request, " Successfully logged in")
        return redirect('/index')
    else:
        messages.error(request," Invalid Credentials, Please try again")
        return redirect("/")
    return HttpResponse('404-not found')
def handleLogout(request):
    del request.session['is_logged']
    del request.session["user_id"]
    logout(request)
    messages.success(request, " Successfully logged out")
    return redirect('home')
```

F:

```
def addmoney_submission(request):
    if request.session.has_key('is_logged'):
        if request.method == "POST":
            user_id = request.session["user_id"]
            user1 = User.objects.get(id=user_id)
            addmoney_info1 = Addmoney_info.objects.filter(user=user1).order_by('-Date')
            add_money = request.POST["add_money"]
            quantity = request.POST["quantity"]
            Date = request.POST["Date"]
            Category = request.POST["Category"]
            add = Addmoney_info(user = user1,add_money=add_money,quantity=quantity,Date
= Date,Category= Category)
            add.save()
            paginator = Paginator(addmoney_info1, 4)
            page_number = request.GET.get('page')
            page_obj = Paginator.get_page(paginator,page_number)
            context = {
                'page_obj' : page_obj
                }
            return render(request,'home/index.html',context)
    return redirect('/index')
def addmoney_update(request,id):
    if request.session.has_key('is_logged'):
        if request.method == "POST":
            add  = Addmoney_info.objects.get(id=id)
            add .add_money = request.POST["add_money"]
            add.quantity = request.POST["quantity"]
            add.Date = request.POST["Date"]
            add.Category = request.POST["Category"]
            add .save()
            return redirect("/index")
    return redirect("/home")
```

G:

```
def expense_edit(request,id):
    if request.session.has_key('is_logged'):
        addmoney_info = Addmoney_info.objects.get(id=id)
        user_id = request.session["user_id"]
        user1 = User.objects.get(id=user_id)
        return render(request,'home/expense_edit.html',{'addmoney_info':addmoney_info})
    return redirect("/home")


def expense_delete(request,id):
    if request.session.has_key('is_logged'):
        addmoney_info = Addmoney_info.objects.get(id=id)
        addmoney_info.delete()
        return redirect("/index")
    return redirect("/home")
```

H:

```
def expense_month(request):
    todays_date = datetime.date.today()
    one_month_ago = todays_date-datetime.timedelta(days=30)
    user_id = request.session["user_id"]
    user1 = User.objects.get(id=user_id)
    addmoney = Addmoney_info.objects.filter(user =
user1,Date__gte=one_month_ago,Date__lte=todays_date)
    finalrep ={}


    def get_Category(addmoney_info):
        # if addmoney_info.add_money=="Expense":
        return addmoney_info.Category
    Category_list = list(set(map(get_Category,addmoney)))


    def get_expense_category_amount(Category,add_money):
        quantity = 0
        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
        for item in filtered_by_category:
            quantity+=item.quantity
        return quantity


    for x in addmoney:
        for y in Category_list:
            finalrep[y]= get_expense_category_amount(y,"Expense")


    return JsonResponse({'expense_category_data': finalrep}, safe=False)


def stats(request):
    if request.session.has_key('is_logged') :
        todays_date = datetime.date.today()
        one_month_ago = todays_date-datetime.timedelta(days=30)
        user_id = request.session["user_id"]
        user1 = User.objects.get(id=user_id)

        addmoney_info = Addmoney_info.objects.filter(user =
user1,Date__gte=one_month_ago,Date__lte=todays_date)

        sum = 0

        for i in addmoney_info:

            if i.add_money == 'Expense':

                sum=sum+i.quantity

        addmoney_info.sum = sum

        sum1 = 0

        for i in addmoney_info:

            if i.add_money == 'Income':

                sum1 =sum1+i.quantity

        addmoney_info.sum1 = sum1

        x= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum

        y= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum

        if x<0:

            messages.warning(request,'Your expenses exceeded your savings')

            x = 0

        if x>0:

            y = 0

        addmoney_info.x = abs(x)

        addmoney_info.y = abs(y)

        return render(request,'home/stats.html',{'addmoney':addmoney_info})


def expense_week(request):

    todays_date = datetime.date.today()

    one_week_ago = todays_date-datetime.timedelta(days=7)
    user_id = request.session["user_id"]

    user1 = User.objects.get(id=user_id)

    addmoney = Addmoney_info.objects.filter(user =
user1,Date__gte=one_week_ago,Date__lte=todays_date)

    finalrep ={}


    def get_Category(addmoney_info):

        return addmoney_info.Category

    Category_list = list(set(map(get_Category,addmoney)))




    def get_expense_category_amount(Category,add_money):

        quantity = 0

        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")

        for item in filtered_by_category:

            quantity+=item.quantity

        return quantity


    for x in addmoney:

        for y in Category_list:

            finalrep[y]= get_expense_category_amount(y,"Expense")


    return JsonResponse({'expense_category_data': finalrep}, safe=False)


def weekly(request):

    if request.session.has_key('is_logged') :

        todays_date = datetime.date.today()
        one_week_ago = todays_date-datetime.timedelta(days=7)

        user_id = request.session["user_id"]

        user1 = User.objects.get(id=user_id)

        addmoney_info = Addmoney_info.objects.filter(user =
user1,Date__gte=one_week_ago,Date__lte=todays_date)

        sum = 0

        for i in addmoney_info:

            if i.add_money == 'Expense':

                sum=sum+i.quantity

        addmoney_info.sum = sum

        sum1 = 0

        for i in addmoney_info:

            if i.add_money == 'Income':

                sum1 =sum1+i.quantity

        addmoney_info.sum1 = sum1

        x= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum

        y= user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum

        if x<0:

            messages.warning(request,'Your expenses exceeded your savings')

            x = 0

        if x>0:

            y = 0

        addmoney_info.x = abs(x)

        addmoney_info.y = abs(y)

    return render(request,'home/weekly.html',{'addmoney_info':addmoney_info})


def check(request):
    if request.method == 'POST':
        user_exists = User.objects.filter(email=request.POST['email'])
        messages.error(request,"Email not registered, TRY AGAIN!!!")
        return redirect("/reset_password")


def info_year(request):
    todays_date = datetime.date.today()
    one_week_ago = todays_date-datetime.timedelta(days=30*12)
    user_id = request.session["user_id"]
    user1 = User.objects.get(id=user_id)
    addmoney = Addmoney_info.objects.filter(user =
user1,Date__gte=one_week_ago,Date__lte=todays_date)
    finalrep ={}


    def get_Category(addmoney_info):
        return addmoney_info.Category
    Category_list = list(set(map(get_Category,addmoney)))



    def get_expense_category_amount(Category,add_money):
        quantity = 0
        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
        for item in filtered_by_category:
            quantity+=item.quantity
        return quantity


    for x in addmoney:
        for y in Category_list:
            finalrep[y]= get_expense_category_amount(y,"Expense")

    return JsonResponse({'expense_category_data': finalrep}, safe=False)

def info(request):
    return render(request,'home/info.html')
```