HOUSEHOLD SERVICES APPLICATION

Modern application and devolopement-I,
SEPTEMBER,2024 TERM
Project report by Sohini Ghosh
Roll no. 23f2004366

STUDENT DETAILS:

Name: Sohini Ghosh
Roll: 23f2004366
Email: 23f2004366@ds.study.iitm.ac.in

About me: **I'm Sohini Ghosh currently a student in IITM's BS in Data Science program while also pursuing an offline BSc degree in Mathematics. In addition to my academic pursuits, I have a strong passion for coding, and this Household Services Application allows me to express my skills on a blank canvas, I've learned a lot from this project. As this is my first web-based application, I have put my best effort into shaping it. I enjoy solving complex real-world problems with simple, efficient code. This** project marks the first step in my coding journey, and I am eager to continue improving my skills.

PROJECT DESCRIPTION:

Project Statement

Develop a **Household Services Application**, a multi-user platform designed to connect **customers** seeking home services with **service professionals** offering those services. An **admin** oversees user management and service requests.

Problem Approach:

First, I made some dummy data storages even before creating any database table. Then I create the login condition based on this tables I started to create my structure one by one. Then, I create the database after I made a basic structure and routes for login and customer, professional dashboard. After that I connect my db tables with the routes and then based on the current user I query the databases. After that I create a admin dashboard and also add one more fields 'is_admin' to the customer table to check whether the customer is admin or not during login, then I redirect them to their dashboard based on this 'is_admin' field and then I added 'is_blocked' field to both customer and professional dashboard since admin has the power to block the users. Then I create a separate table service_request that will save the customer service request and then I used that table to the professional dashboard to show the service requests based on their professional_id. Then I modify the customer table so that it can send a request a service that is not able to the admin. After that I create summary for three dashboards where in the customer dashboard we can able to see bar graphs of the total requests, completed, pending and rejected requests. Also in the professional I use bar graphs for total, pending, cancelled, completed requests in the summary. Now in the admin dashboard I use a bar graph to show total users of the website and no. of customers and no. of professionals also added a pie chart to show the number of total, pending, completed and cancelled requests.

FRAMEWORKS AND LIBRARIES USED:

(1) Flask: The backend framework used for building the web application. It handles routing, templates, and managing HTTP requests and responses.
(2) SQLAlchemy: An ORM (Object-Relation Mapping) library used for interacting with the database. It simplifies database operations and integrates seamlessly with Flask.
(3) SQLite: A lightweight, file-based relational database used to store application data, such as customer information, service details, orders, and feedback.
(4) Jinja2: A template engine built into Flask, used to dynamically render HTML pages withdata from the backend.
(5) Logging: Python's built-in logging module is used for monitoring and debugging the application by recording runtime events.
(6) Datetime: A Python module used to handle and format date and time information, such as timestamps for orders or feedback.
(7) os: Python's built-in os module is used for file and directory management, such as setting up the uploads folder for storing files.
(8) HTML/CSS/JavaScript: Frontend technologies for user interface design and interactivity. Also used Tailwind CSS apart from vanilla CSS.
(9) ChartJS: Used for creating different types of charts on the admin, customer and professional dashboards.

DB SCHEMA DESIGN:

1. Customers Table (customers)**:**

id (PK), name (String, 100), username (String, 100, Unique), password (String, 100), email (String, 100, Unique), address (String, 200), pincode (String, 10), is_admin (Boolean, Default: False), blocked (Boolean, Default: False)

2. Cart Table (cart)**:**

id (PK), customer_id (FK), service_id (FK), service_name (String, 80), quantity (Integer, Default: 1), price (Float), total (Float), time_required (Integer)

3. Services Table (services):

id (PK), name (String, 80, Unique), price (Integer), timerequired (Integer), description (String, 200), allowed (Boolean, Default: False), status (String, 20, Default: "Pending")

4. Professionals Table (professionals)**:**

id (PK), name (String, 32), username (String, 120, Unique), password (String, 200), email (String, 120, Unique), address (String, 200), pincode (String, 10), service_type (FK), experience (Integer), description (Text), document (String, 200, Nullable), is_approved (Boolean, Default: False), blocked (Boolean, Default: False), date_created (DateTime, Nullable)

5. Orders Table (orders)**:**

id (PK), customer_id (FK), provider_id (FK), service_id (FK), date_requested (DateTime, Default: Current Timestamp), date_completed (DateTime, Nullable), status (String, 20, Default: "Pending"), notes (Text, Nullable), work_completed (Boolean, Default: False), customer_approval (Boolean, Default: False), professional_approval (Boolean, Default: False), feedback_given (FK), cart_id (FK), quantity (Integer, Default: 1)

6. Service Requests Table (service_requests)**:**

id (PK), customer_id (FK), professional_id (FK), service_id (FK), pincode (String, 10), status (String, 20, Default: "Pending"), date_of_request (DateTime, Default: Current Timestamp), date_completed (DateTime, Nullable), work_completed (Boolean, Default: False), customer_approval (Boolean, Default: False), professional_approval (Boolean, Default: False)

7. Feedbacks Table (feedbacks)**:**

id (PK), customer_id (FK), service_id (FK), professional_id (FK), rating (Integer), comment (Text, Nullable), created_at (DateTime, Default: Current Timestamp), feedback_given (Boolean, Default: False)

Database-Relations:

Customer ↔ Cart:

One-to-Many (Customer.id ↔ Cart.customer_id)

Cart ↔ Service:

Many-to-One (Cart.service_id ↔ Service.id)

Customer ↔ Order:

One-to-Many (Customer.id ↔ Order.customer_id)

Professional ↔ Order:

One-to-Many (Professional.id ↔ Order.provider_id)

Service ↔ Order:

One-to-Many (Service.id ↔ Order.service_id)

Customer ↔ Feedback:

One-to-Many (Customer.id ↔ Feedback.customer_id)

Service ↔ Feedback:

One-to-Many (Service.id ↔ Feedback.service_id)

Professional ↔ Feedback:

One-to-Many (Professional.id ↔ Feedback.professional_id)

Customer ↔ ServiceRequest:

One-to-Many (Customer.id ↔ ServiceRequest.customer_id)

Professional ↔ ServiceRequest:

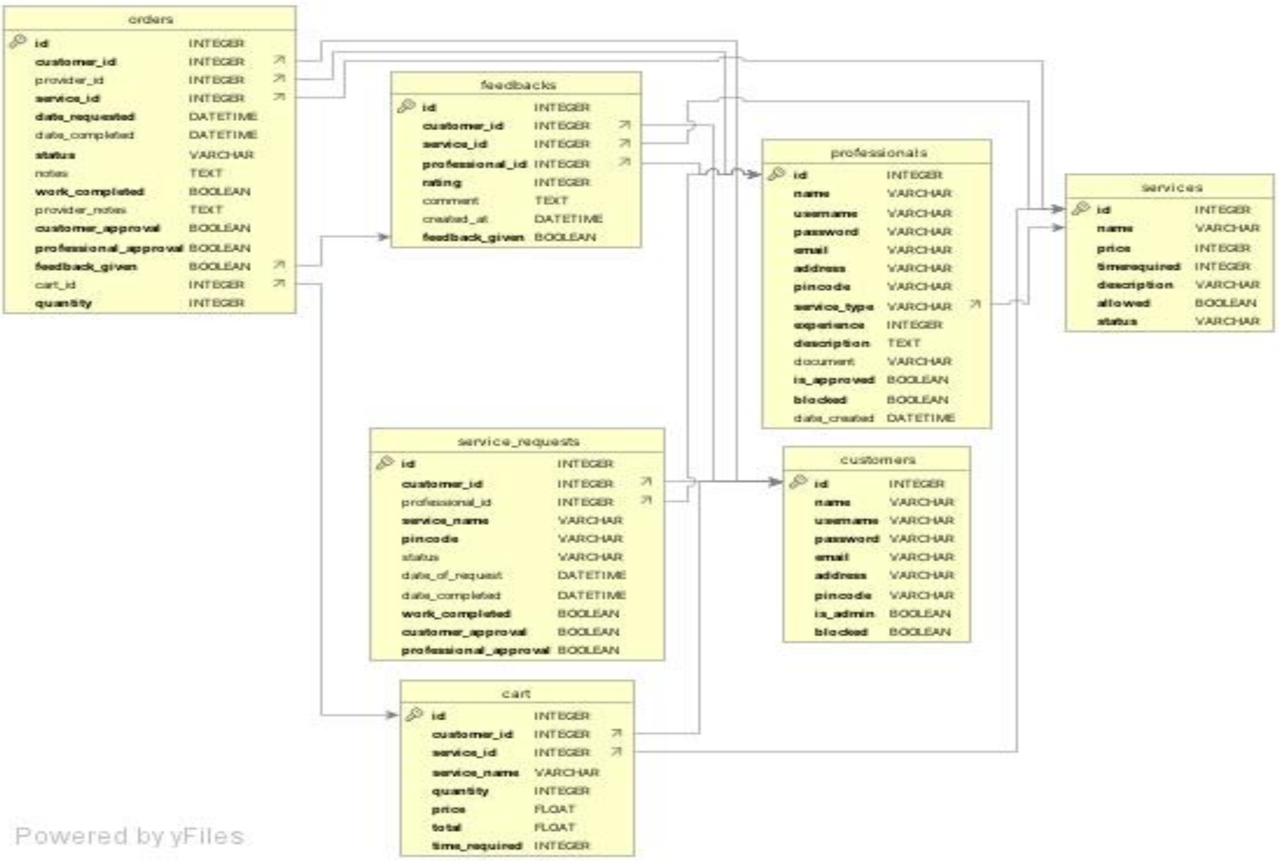
One-to-Many (Professional.id ↔ ServiceRequest.professional_id)

Service ↔ ServiceRequest:

One-to-Many (Service.id ↔ ServiceRequest.service_id)

Service ↔ Professional:

One-to-Many (Service.name ↔ Professional.service_type)



orders	
id	INTEGER
customer_id	INTEGER
provider_id	INTEGER
service_id	INTEGER
date_requested	DATETIME
date_completed	DATETIME
status	VARCHAR
notes	TEXT
work_completed	BOOLEAN
provider_notes	TEXT
customer_approval	BOOLEAN
professional_approval	BOOLEAN
feedback_given	BOOLEAN
cart_id	INTEGER
quantity	INTEGER

feedbacks	
id	INTEGER
customer_id	INTEGER
service_id	INTEGER
professional_id	INTEGER
rating	INTEGER
comment	TEXT
created_at	DATETIME
feedback_given	BOOLEAN

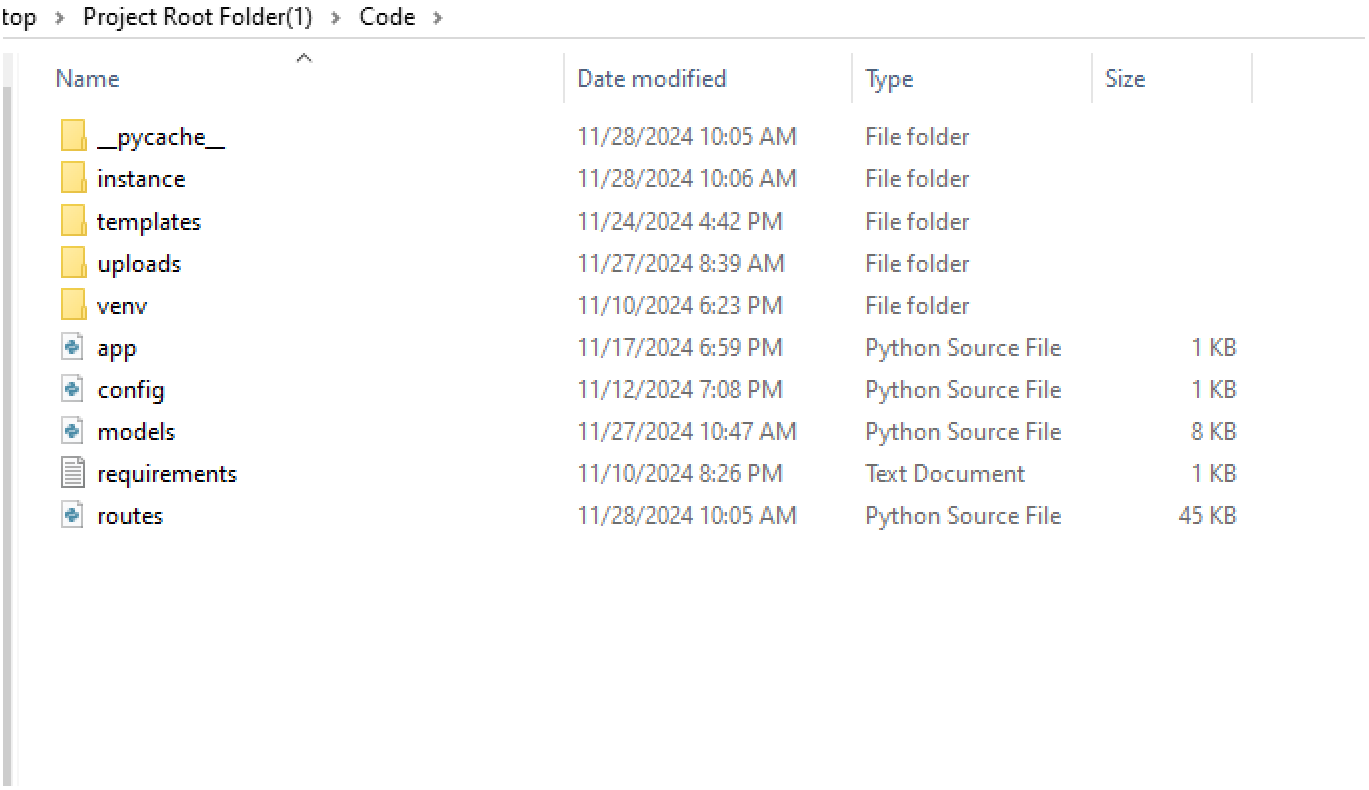
professionals	
id	INTEGER
name	VARCHAR
username	VARCHAR
password	VARCHAR
email	VARCHAR
address	VARCHAR
pincode	VARCHAR
service_type	VARCHAR
experience	INTEGER
description	TEXT
document	VARCHAR
is_approved	BOOLEAN
blocked	BOOLEAN
date_created	DATETIME

services	
id	INTEGER
name	VARCHAR
price	INTEGER
timerequired	INTEGER
description	VARCHAR
allowed	BOOLEAN
status	VARCHAR

service_requests	
id	INTEGER
customer_id	INTEGER
professional_id	INTEGER
service_name	VARCHAR
pincode	VARCHAR
status	VARCHAR
date_of_request	DATETIME
date_completed	DATETIME
work_completed	BOOLEAN
customer_approval	BOOLEAN
professional_approval	BOOLEAN

customers	
id	INTEGER
name	VARCHAR
username	VARCHAR
password	VARCHAR
email	VARCHAR
address	VARCHAR
pincode	VARCHAR
is_admin	BOOLEAN
blocked	BOOLEAN

cart	
id	INTEGER
customer_id	INTEGER
service_id	INTEGER
service_name	VARCHAR
quantity	INTEGER
price	FLOAT
total	FLOAT
time_required	INTEGER



top › Project Root Folder(1) › Code ›

Name	Date modified	Type	Size
__pycache__	11/28/2024 10:05 AM	File folder	
instance	11/28/2024 10:06 AM	File folder	
templates	11/24/2024 4:42 PM	File folder	
uploads	11/27/2024 8:39 AM	File folder	
venv	11/10/2024 6:23 PM	File folder	
app	11/17/2024 6:59 PM	Python Source File	1 KB
config	11/12/2024 7:08 PM	Python Source File	1 KB
models	11/27/2024 10:47 AM	Python Source File	8 KB
requirements	11/10/2024 8:26 PM	Text Document	1 KB
routes	11/28/2024 10:05 AM	Python Source File	45 KB

PROJECT VIDEO:

https://youtu.be/AvYAoJ__oxo?si=CCPSpWPZQeyqMo3t