MAD-1 Project Report

Project Description:

The project involves building a multi-user application, *A-Z Household Services*, that connects customers, service professionals, and an admin. It facilitates service management for household solutions like plumbing, cleaning, and more. The app includes features like user registration/login, service creation, booking, and review management.

My approach was aligned with the project requirements, implementing core functionalities as outlined. As a student with prior theory knowledge of *Modern Application Development (MAD-1)*, I applied concepts learned in class to build the application. For development, I primarily relied on Flask and SQLite, supplemented with Jinja2 templates and Bootstrap for the frontend.

While working on the project, I utilized official documentation, online resources, and ChatGPT for guidance. However, the implementation was primarily my own effort, adhering to the required functionality and ensuring a fully operational website. The core features, such as user role differentiation, service management, and service request handling, have been implemented successfully.

Technologies Used:

- **Frameworks**: Flask (with SQLAlchemy), Bootstrap
- **Database**: SQLite
- **Frontend**: HTML, CSS (styling, gradient background, star animations), JavaScript

Flask was used for backend logic, SQLAlchemy for managing the database, and Bootstrap for responsive design. Additional features like animated backgrounds and visual enhancements were added for a better user experience.

DB Schema Design:

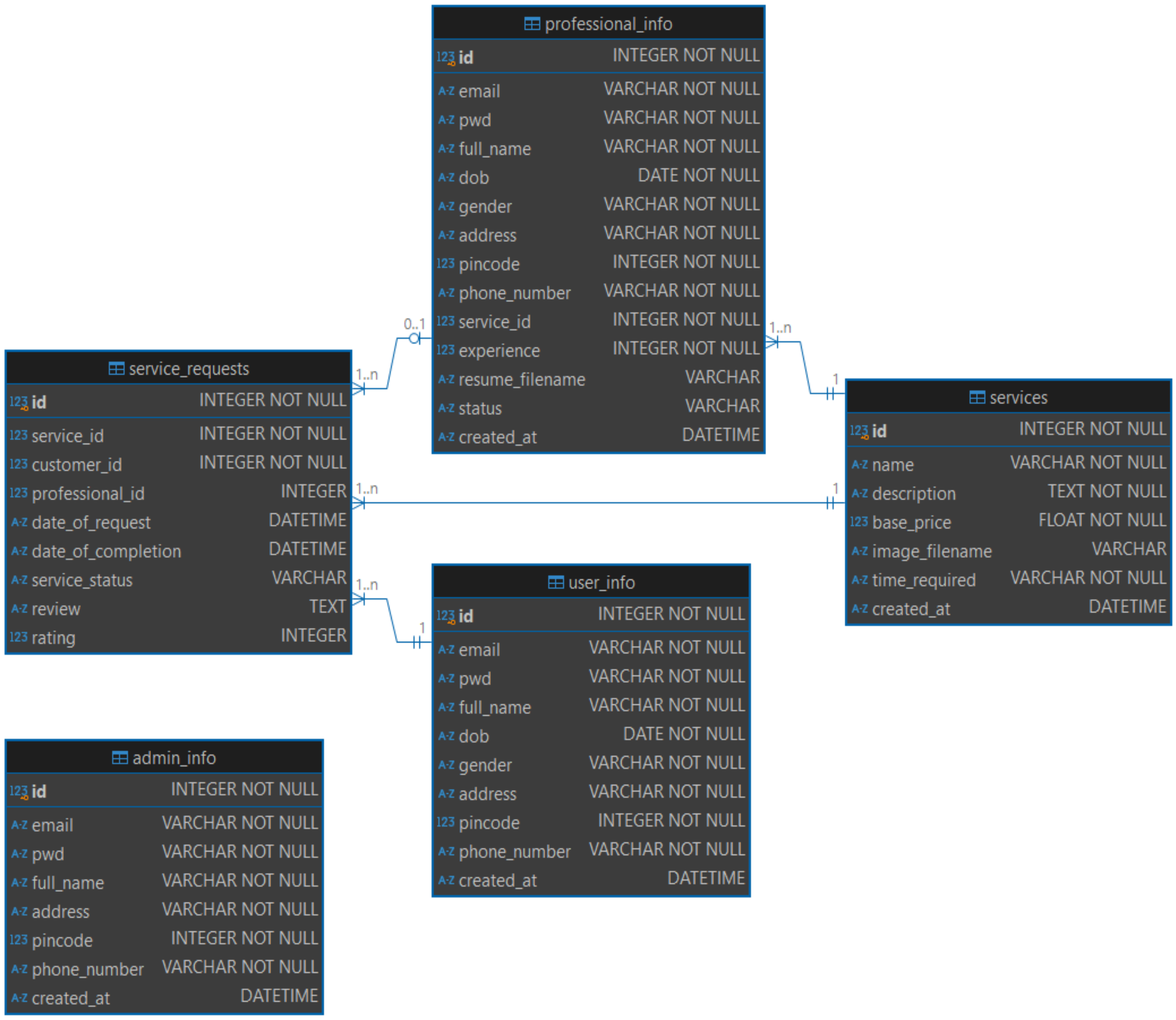
The database schema includes the following:

1. **Admin Table**:
   Columns: `id`, `email`, `pwd` (hashed), `full_name`, `address`, `pincode`, `phone_number`, `created_at`,
2. **Users Table**:
   Columns: `id`, `email`, `pwd` (hashed), `full_name`, `dob`, `gender`, `address`, `pincode`, `phone_number`, `created_at`,

3. **Professionals Table**:
   Columns: `id`, `email`, `pwd` (hashed), `full_name`, `dob`, `gender`, `address`, `pincode`, `phone_number`, `service_id (FK Services), experience`, `resume_filename`, `status, created_at`.
4. **Services Table**:
   Columns: `id`, `name`, `description`, `base_price`, `image_filename`, `time_required`, `created_at`.
5. **Service Requests Table**:
   Columns: `id`, `service_id (FK Services)`, `customer_id (FK Users)`, `professional_id (FK Professionals)`, `image_filename`, `time_required`, `created_at`.

Design Justification:
The schema is normalized to minimize redundancy. Relations between tables ensure efficient data management, enabling smooth service assignments and updates.

ER Diagram:



professional_info	
id	INTEGER NOT NULL
email	VARCHAR NOT NULL
pwd	VARCHAR NOT NULL
full_name	VARCHAR NOT NULL
dob	DATE NOT NULL
gender	VARCHAR NOT NULL
address	VARCHAR NOT NULL
pincode	INTEGER NOT NULL
phone_number	VARCHAR NOT NULL
service_id	INTEGER NOT NULL
experience	INTEGER NOT NULL
resume_filename	VARCHAR
status	VARCHAR
created_at	DATETIME

service_requests	
id	INTEGER NOT NULL
service_id	INTEGER NOT NULL
customer_id	INTEGER NOT NULL
professional_id	INTEGER
date_of_request	DATETIME
date_of_completion	DATETIME
service_status	VARCHAR
review	TEXT
rating	INTEGER

services	
id	INTEGER NOT NULL
name	VARCHAR NOT NULL
description	TEXT NOT NULL
base_price	FLOAT NOT NULL
image_filename	VARCHAR
time_required	VARCHAR NOT NULL
created_at	DATETIME

user_info	
id	INTEGER NOT NULL
email	VARCHAR NOT NULL
pwd	VARCHAR NOT NULL
full_name	VARCHAR NOT NULL
dob	DATE NOT NULL
gender	VARCHAR NOT NULL
address	VARCHAR NOT NULL
pincode	INTEGER NOT NULL
phone_number	VARCHAR NOT NULL
created_at	DATETIME

admin_info	
id	INTEGER NOT NULL
email	VARCHAR NOT NULL
pwd	VARCHAR NOT NULL
full_name	VARCHAR NOT NULL
address	VARCHAR NOT NULL
pincode	INTEGER NOT NULL
phone_number	VARCHAR NOT NULL
created_at	DATETIME

ER Diagram of my Database! (household.db)
Look at the Relationship of my Database in this diagram…..

API Design:

I haven't used any API on this Project, All the CRUD operations have been implemented directly using flask and SQLite etc…..

Architecture and Features:

The project folder structure is:

- **Root Folder (Zip Folder)**: Contains Code Folder - `household-services-application`.
- **Code Folder**: Contains some *folders:* (`backend, static, templates,` and `instance.`) and *files:* (`app.py`, `requirements.txt`, `README.md`, `.gitignore`, and `project-report.pdf`.)
- **Insides Folders**:
  - `backend`: Contains controllers and models (Python files).
  - `static`: Includes CSS files, 1 JS file, images (folder), and uploaded files (folder).
  - `templates`: All HTML files for the UI.
  - `instance`: Contains the SQLite database (`household.db`).

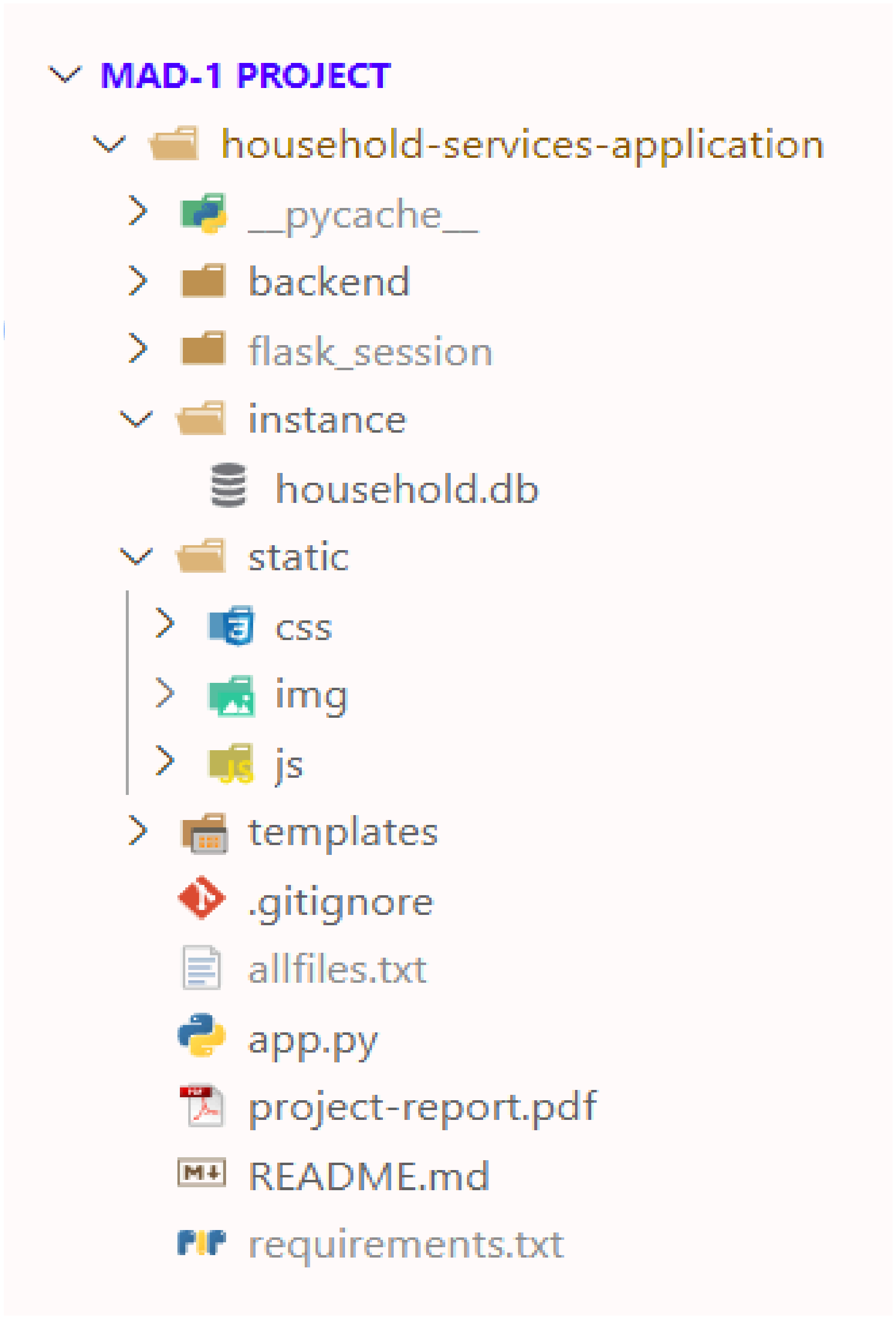
Features Implemented:

1. **Dynamic Routing**: Simplifies navigation.
2. **Authentication System**: Supports admin, user, and professional logins. (using session)
3. **Service Management**: Enables users to request services and track their status.
4. **Responsive Design**: Mobile-friendly UI with Bootstrap.
5. **Gradient Backgrounds and Star Animations**: Enhances aesthetics.
6. **File Upload Support**: Professionals can upload resumes and certifications.

Additional Features:

1. Default admin credentials for ease of testing.
2. Organized storage for uploaded files in the `static/uploads` folder.
3. Secure password handling and validation.

File Directory Image:



```
MAD-1 PROJECT
  household-services-application
    __pycache__
    backend
    flask_session
    instance
      household.db
    static
      css
      img
      js
    templates
    .gitignore
    allfiles.txt
    app.py
    project-report.pdf
    README.md
    requirements.txt
```

Project Video Link:

(Explanation video of Project)

Python Project || Household service Application