Student Details
Name: Vidhatri HR
Email: 23f2000575@ds.study.iitm.ac.in

Presentation video link: https://drive.google.com/file/d/1EX8Cwg3ggKHCVGgWZixEz5W7Hb4XIkWt/view?usp=sharing

---

Project Statement

This is a household services application.
It has three types of users: Admins, Professionals, and Customers.
Admins manage services and users.
Professionals offer services in various categories.
Customers book and close service requests.
Features include login, registration, CRUD operations, and a user-friendly UI.

Approach

1. Identify entities, attributes, and relationships.
2. Create an ER diagram.
3. Write SQLAlchemy model classes for entities.
4. Set up a Flask app with blueprints for modular routes.
5. Initialize the database, populate sample data, and set up `flask-login`.
6. Implement core features in stages: login, register, and admin, followed by professional and customer features.
7. Test all functionalities thoroughly.
8. Deploy online on Railway.app platform.

Live Demo
https://household-services-production.up.railway.app

Accounts:

- **Admin**: `admin1@example.com`, Password: `GgJI0lQ4X`
- **Customer**: `customer1@example.com`, Password: `12345`
- **Professional**: `professional1@example.com`, Password: `12345`

Libraries and Tools

- **Libraries**: Flask, Flask-SQLAlchemy, Flask-Login, Werkzeug, Matplotlib, Bootstrap CSS.
- **Tools**: Pony ORM Editor (ER diagrams), Excalidraw (diagrams).

Routes

Admin Blueprint (`admin_bp`)

- `/admin/home`: Admin dashboard.
- `/admin/search`: Search for users or services.
- `/admin/search-results/<search_type>`: Display search results.
- `/admin/summary`: View summary reports.
- `/admin/service-details/<int:service_id>`: View service details.
- `/admin/customer-details/<int:customer_id>`: View customer details.
- `/admin/professional-details/<int:professional_id>`: View professional details.
- `/admin/approve-professional/<int:professional_id>`: Approve a professional.
- `/admin/block-professional/<int:professional_id>`: Block a professional.
- `/admin/delete-professional/<int:professional_id>`: Delete a professional.
- `/admin/add-service`: Add a new service.
- `/admin/edit-service/<int:service_id>`: Edit service details.
- `/admin/delete-service/<int:service_id>`: Delete a service.

Customer Blueprint (`customer_bp`)

- `/customer/home`: Customer dashboard.
- `/customer/search`: Search for professionals or services.
- `/customer/search-results/<search_type>`: Display search results.
- `/customer/summary`: View customer activity summary.
- `/customer/profile`: View customer profile.
- `/customer/edit-profile`: Edit customer profile.
- `/customer/professional-details/<int:professional_id>`: View professional details.
- `/customer/book-service/<int:service_id>`: Book a service.
- `/customer/close-request/<int:request_id>`: Close a service request.

Professional Blueprint (`professional_bp`)

- `/professional/home`: Professional dashboard.
- `/professional/search`: Search for customers or requests.
- `/professional/search-results/<search_type>`: Display search results.
- `/professional/summary`: View professional activity summary.
- `/professional/profile`: View professional profile.
- `/professional/edit-profile`: Edit professional profile.
- `/professional/customer-details/<int:customer_id>`: View customer details.
- `/professional/accept-request/<int:request_id>`: Accept a service request.

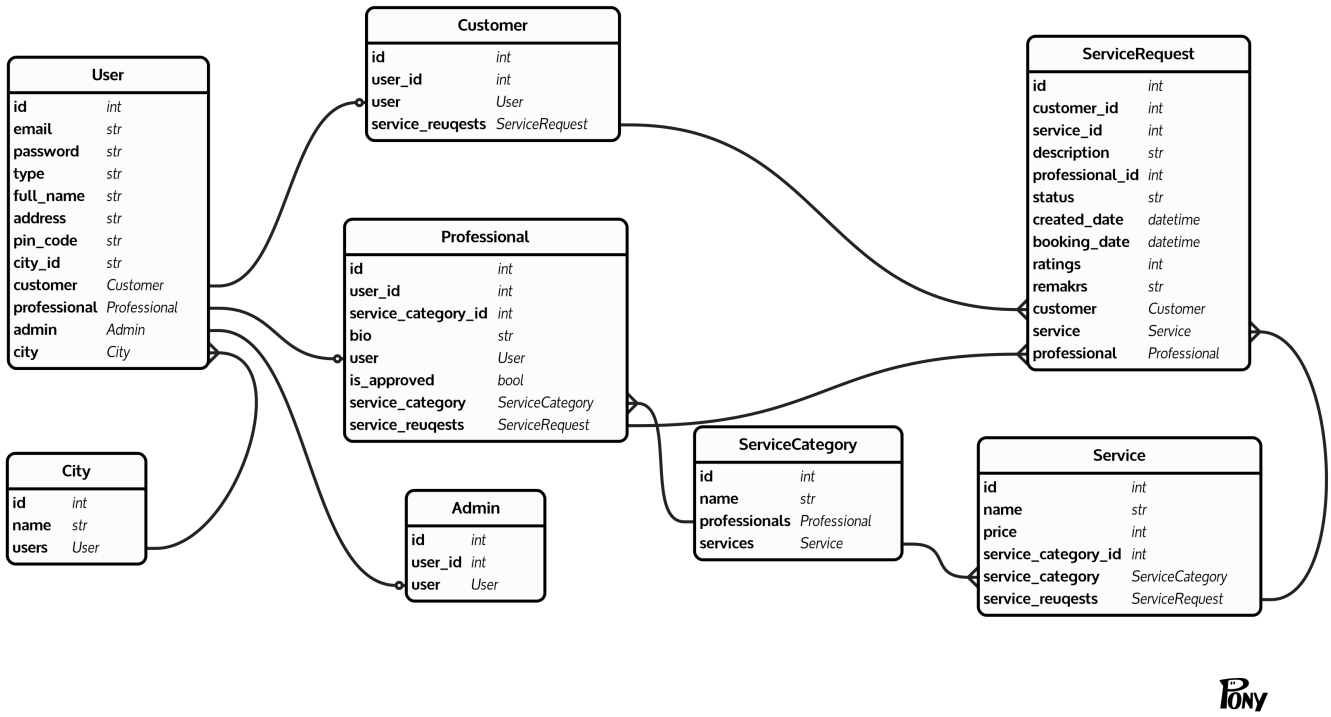
Auth Blueprint (`auth_bp`)

- `/login`: Log in as a user.
- `/logout`: Log out of the application.
- `/register`: Register a customer.
- `/register/professional`: Register a professional.

Root Blueprint (`root_bp`)

- Redirecting authenticated users to their respective dashboards.

Database Models



1. **User**: Represents all users in the system.
2. **City**: Represents cities linked to users.
3. **Admin**: Represents admin users.
4. **Customer**: Represents customers and their requests.
5. **Professional**: Represents professionals offering services.
6. **ServiceCategory**: Represents service categories (e.g., plumbing, cleaning).
7. **Service**: Represents individual services within categories.
8. **ServiceRequest**: Represents service requests from customers to professionals.

Folder Structure

- `routes/`: Contains Flask blueprints for modular route management.
- `templates/`: Contains HTML templates for the UI.
- `static/`: Contains static files like CSS and images.
- `instance/`: Contains the SQLite database file.
- `docs/`: Documentation, ER diagrams, and development notes.

Running the App

1. Create a virtual environment: `python -m venv venv`.

2. Activate the venv: `source venv/bin/activate` (Linux/Mac) or `venv\Scripts\activate` (Windows).
3. Install dependencies: `pip install -r requirements.txt`.
4. Run the app: `python app.py`.