Project Report

Modern Application Development - I

Himanshu Saini
22f3002094
22f3002094@ds.study.iitm.ac.in

A-Z HouseHold Service Application

The Household Services Platform **Helpify** is a dynamic, multi-user application that bridges the gap between customers and service professionals, all under the oversight of a dedicated admin. Designed to optimize service delivery, booking, and management, the platform ensures seamless operations with transparency and reliability at its core.

Key Features

- **User Roles:** Supports three distinct roles—Admins, Customers, and Service Professionals, each with tailored functionalities.
- **Admin Capabilities:** Comprehensive dashboard to manage users, approve professionals, oversee service listings, and track activities.
- **Customer Features:** Create, modify, and complete service requests; and provide feedback through reviews.
- **Professional Tools:** Allows service professionals to review and respond to service requests, either accepting or declining them, and close completed requests while maintaining credibility through customer ratings.
  Professionals can create new plans under the category they are registered in.

Technologies and Frameworks

I used Below mentioned stack to create my application.

- **Flask:** Forms the backbone of the web application, handling backend logic and server-side operations.

- **Jinja2 Templates + Bootstrap:** Merges dynamic content generation with responsive and aesthetically pleasing UI elements.
- **SQLite:** Serves as the lightweight, reliable database solution for organizing and accessing user and service-related data.
- **Flask-SQLAlchemy:** Simplifies database interactions, enabling efficient data management within the app.
- **Flask-Login:** Manages user authentication and session persistence, providing seamless login and logout functionalities.
- **Flask-RESTful:** Streamlines API development for managing endpoints and facilitating communication between components.
- **Requests:** Handles HTTP communication, ensuring smooth interaction between different parts of the application.

Database Schema Summary

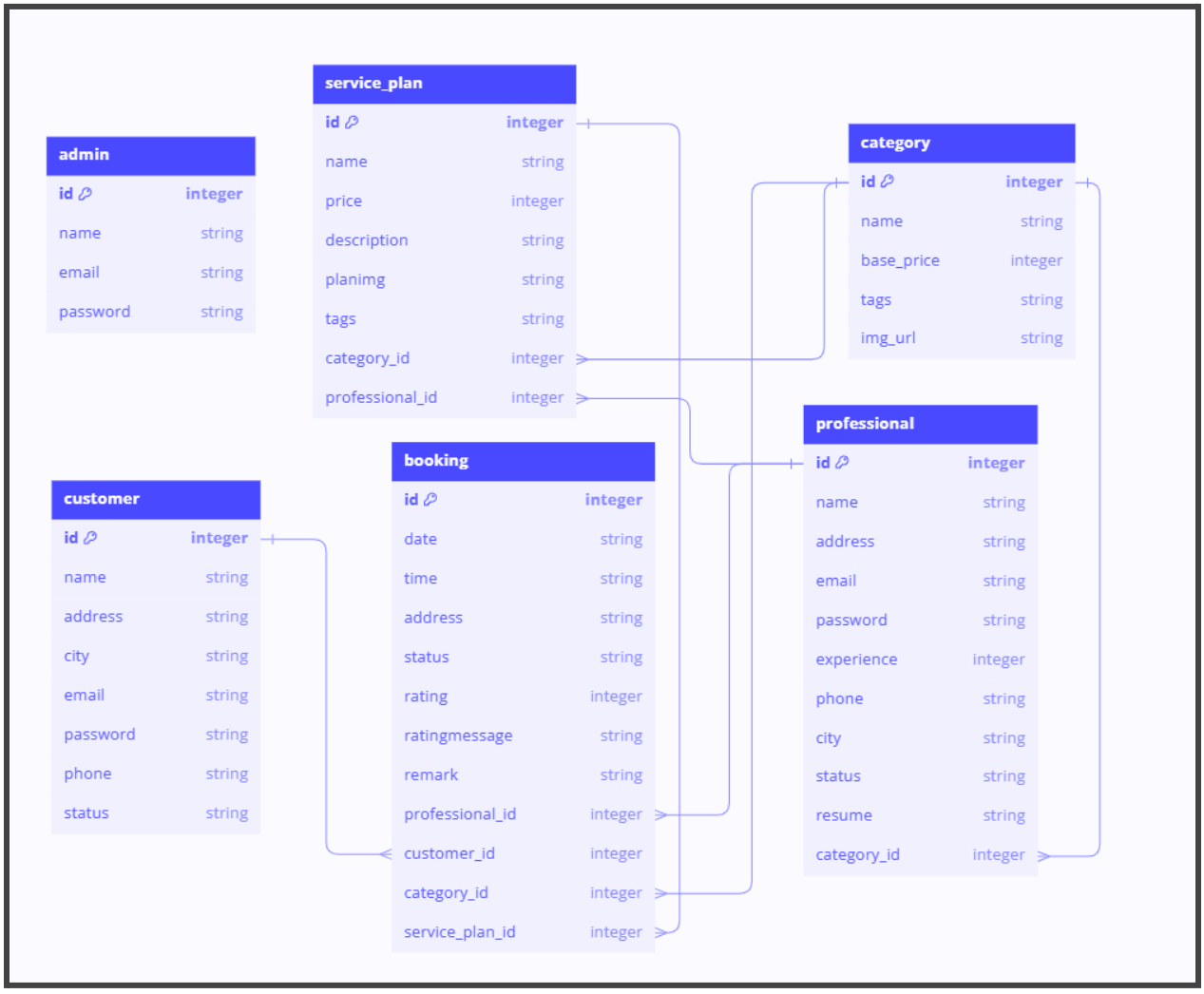
The database schema consists of the following core models:

1. **Admin (admin)**
    - Stores admin details like id, name, email, and password. Admin manages the system but doesn't directly interact with other entities.
2. **Professional (professional)**
    - Contains details of professionals like id, name, email, password, experience, status, etc.
    - Relationships: Linked to multiple **Service Plans** and **Bookings**.
    - References **Service Category**.
3. **Customer (customer)**
    - Stores customer information such as id, name, email, address, etc.
    - Relationships: A customer can make multiple **Bookings**.
4. **Service Category (category)**
    - Defines service categories like id, name, base_price, etc.
    - Relationships: Linked to multiple **Service Plans** and **Professionals**.
5. **Service Plan (service_plan)**
    - Contains details of service plans such as id, name, price, description, etc.

- Relationships: Linked to one **Service Category** and one **Professional**.
- Has multiple **Bookings**.

6. **Booking (booking)**
   - Captures booking information like id, date, time, status, rating, etc.
   - Relationships: Linked to a **Customer**, **Professional**, **Service Category**, and **Service Plan**.

ERD



File Structure Overview

1. **Root Level**
    - **app.py**: Main entry point for the app.
2. **Backend (backend/)**
    - **api.py**: API endpoints for app functionality.
    - **models.py**: Defines the database models.
    - **routes.py**: Manages routing of requests.
3. **Instance (instance/)**
    - **my_app_db.sqlite3**: SQLite database file.
4. **Static (static/)**
    - Contains static assets (images, CSS, JS).
5. **Templates (templates/)**
    - **Admin**: HTML pages for admin dashboard, service management, customer, and professional details.
    - **Customer**: HTML pages for customer booking, dashboard, and profile settings.
    - **ServiceProfessional**: HTML pages for service professional dashboard, bookings, and service plans.

Api Endpoints Used

```
api.add_resource(Apistatistics,"/api/extractstats")
```

Controllers

```
@app.route("/register/<user_type>",methods=["GET","POST"])
@app.route("/login", methods=["GET", "POST"])
@app.route("/dashboard/profile-setting",methods=["GET","POST"])
@app.route("/dashboard/search", methods=["GET", "POST"])
@app.route("/dashboard/service-categories", methods=["GET", "POST"])
```

```
@app.route("/dashboard/professional-detail/<int:id>", methods=["GET",
"POST"])
@app.route("/dashboard/customer-detail/<int:id>", methods=["GET", "POST"])
@app.route("/dashboard/serviceplans", methods=["GET", "POST"])
@app.route("/dashboard/<cat_name>", methods=["GET", "POST"])
@app.route("/book", methods=["GET", "POST"])
@app.route("/dashboard/booking/<int:id>", methods=["GET", "POST"])
@app.route("/dashboard/statistics", methods=["GET", "POST"])
@app.route("/logout" ,methods=["GET", "POST"])
```

Video:

Link:
https://drive.google.com/file/d/1PV_ULYB_3_pUIfkYgFulPcTnAeRNe21i/view?usp=sharing