Project Report
Modern Application Development - I

Tarika Subhash Hedaoo
22f3002758
22f3002758@ds.study.iitm.ac.in

Household Services Application-HomeEase

The *HomeEase* is a multi-user platform connecting customers with service professionals under admin supervision. It ensures efficient service delivery and user management through intuitive features.

Core Functionalities

- **Authentication**: Separate login/register for Admin, Customers, and Service Professionals.
- **Admin Dashboard**: Manage users, approve professionals, and handle services (create, update, delete).
- **Customer Features**: Search services by City, Service Name; manage service requests; provide feedback.
- **Professional Features**: View, accept/reject, and close service requests; maintain visibility through ratings.
- **Search**: Customers search services; Admin searches professionals and customers for management actions.

The platform ensures streamlined operations and a user-friendly experience, fostering transparency and reliability in household service management.

Technologies and Frameworks

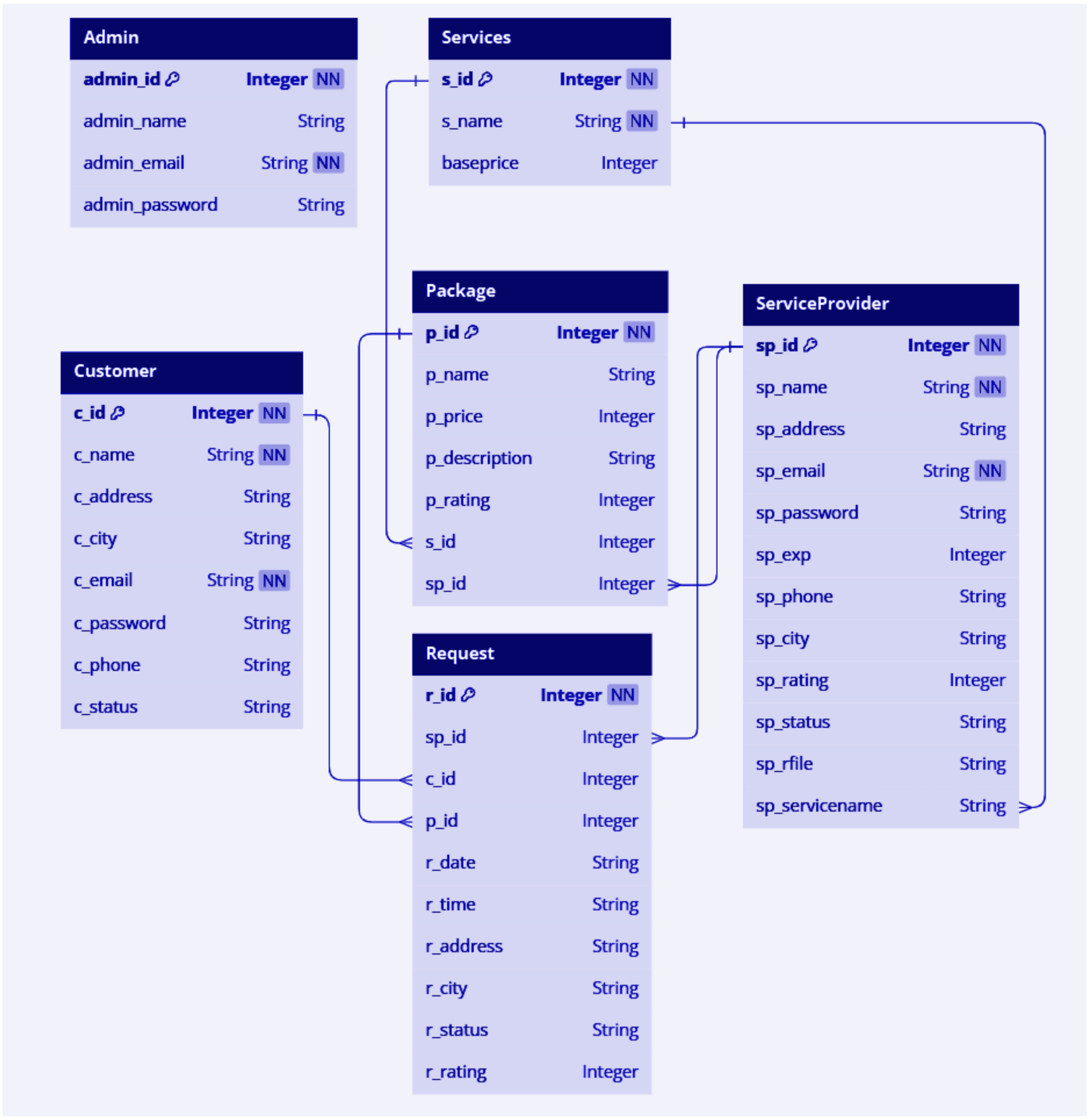
1. **Flask**: For building the core web application and backend logic.
2. **Flask-SQLAlchemy**: To manage database connections and operations across the app.
3. **Flask-RESTful**: Used to build and handle APIs for backend endpoints.
4. **Flask-Login**: Handles user session management, including login/logout, user data storage, and session persistence.
5. **Requests**: For managing HTTP requests and responses.
6. **Jinja2 Templates + Bootstrap**: For creating a responsive and visually appealing user interface.
7. **SQLite**: For storing and managing user and service-related data.

Database Schema Description

The database schema for the *Household Services Application* is designed to effectively manage the interactions between admins, customers, service providers, services, packages, and requests. It ensures proper organization of data and maintains relationships between entities to support seamless operations.

1. **Admin Table**
   - Stores admin details such as ID, name, email, and password.
   - Admins oversee the entire platform, managing users, services, and requests.
2. **Services Table**
   - Contains service details, including name and base price.
   - Associated with both the **ServiceProvider Table** (to map providers to services) and the **Package Table** (to define packages under each service).
3. **ServiceProvider Table**
   - Maintains information about service providers, including their details, experience, and associated service.
   - Linked to the **Services Table** via a foreign key to associate each provider with a specific service.
   - Establishes relationships with the **Package Table** (to manage offered packages) and the **Request Table** (to handle service requests).
4. **Customer Table**
   - Captures customer details such as name, email, and address.
   - Connected to the **Request Table**, enabling customers to create and manage service requests.
5. **Package Table**
   - Stores package information such as price, description, and ratings.
   - Linked to the **Services Table** and **ServiceProvider Table** to define packages offered by a specific provider for a specific service.
   - Related to the **Request Table**, as each request is tied to a specific package.
6. **Request Table**
   - Handles service request details, including customer ID, service provider ID, package ID, and status.
   - Establishes relationships with the **Customer Table**, **ServiceProvider Table**, and **Package Table** to link requests to the respective entities.

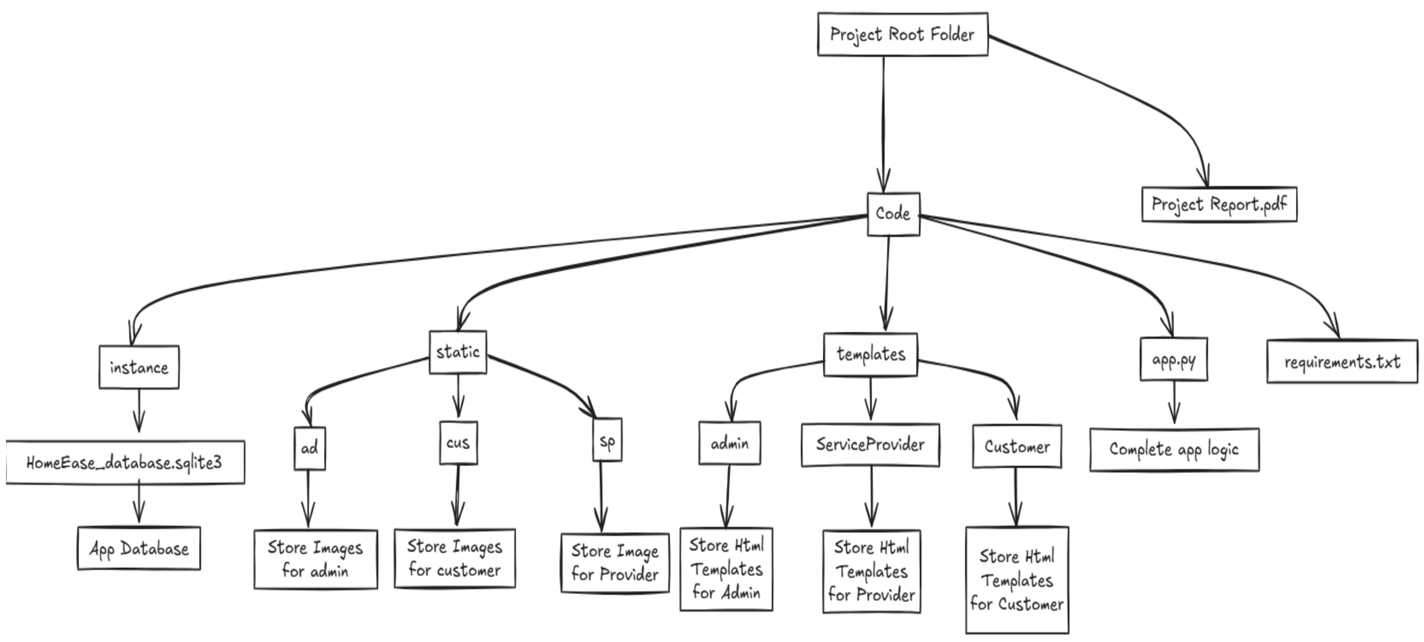
ERD


Admin
admin_id Integer NN
admin_name String
admin_email String NN
admin_password String
Services
s_id Integer NN
s_name String NN
baseprice Integer
Package
p_id Integer NN
p_name String
p_price Integer
p_description String
p_rating Integer
s_id Integer
sp_id Integer
ServiceProvider
sp_id Integer NN
sp_name String NN
sp_address String
sp_email String NN
sp_password String
sp_exp Integer
sp_phone String
sp_city String
sp_rating Integer
sp_status String
sp_rfile String
sp_servicename String
Customer
c_id Integer NN
c_name String NN
c_address String
c_city String
c_email String NN
c_password String
c_phone String
c_status String
Request
r_id Integer NN
sp_id Integer
c_id Integer
p_id Integer
r_date String
r_time String
r_address String
r_city String
r_status String
r_rating Integer

Summary Of API Endpoints

API Resource	Method	Functionality
/api/service/create	POST	Create a new service
/api/service/update	PUT	Update an existing service
/api/service/delete	DELETE	Delete an existing service
/api/book	POST	Create a new service request
/api/book/edit	PUT	Edit an existing service request
/api/flag	POST	Flag a user for inappropriate behavior
/api/package/create	POST	Create a new service package
/api/package	GET	Get a list of available service packages
/api/admin/update	PUT	Update admin details
/api/customer/register	POST	Register a new customer
/api/customer/update	PUT	Update customer details
/api/serviceprovider/register	POST	Register a new service provider
/api/serviceprovider/update	PUT	Update service provider details
/api/stats/<u_type>	GET	Get statistics for a specific user type (customer, service provider, admin)

Architecture and Features



Controllers

The **controllers** that were used:

- @app.route ("/register", methods=["GET","POST"])
- @app.route("/", methods=["GET","POST"])
- @app.route("/profile-update", methods=["GET","POST"])
- @app.route("/login",methods=["GET","POST"])
- @app.route("/customer/Dashboard",methods=["GET","POST"])
- @app.route("/customer/book", methods=["GET","POST"])
- @app.route("/rating",methods=["GET","POST"])
- @app.route("/customer/search",methods=["GET","POST"])
- @app.route("/customer/stats",methods=["GET","POST"])
- @app.route("/serviceprovider/dashboard", methods=["GET","POST"])
- @app.route("/serviceprovider/create",methods=["GET","POST"])
- @app.route("/serviceprovider/stats",methods=["GET","POST"])
- @app.route("/admin/dashboard/",methods=["GET","POST"])
- @app.route("/admin/search" , methods=["GET", "POST"])
- @app.route("/flag",methods=["GET","POST"])
- @app.route("/admin/stats",methods=["GET","POST"])
- @app.route("/service" , methods=["GET","POST"])
- @app.route('/logout')

Video

Link: https://drive.google.com/file/d/1zw0UhEBnfrzmAQKbcimFaFaWWWVBSQUn/view?usp=sharing