Author

Name: Rushil Gupta

Roll no.: 21f1006728

Email: 21f1006728@ds.study.iitm.ac.in

About me: My name is Rushil Gupta. I am a software engineer and a 3rd year student of IIT-M BS in Data Science Program. I am based in Lucknow.

Description

Housefix is an application which connects household service providers to customers. A service professional can register themselves, and on approval of the admin, they can provide their services to customers. Customers can register and avail the service provided by these professionals by raising a service request. A professional can approve or reject a service request, and on approval, he can also mark it as completed. A user can withdraw their pending service request and give review and rating to their completed service request. Both types of users can search for services or service requests based on their characteristics. A single admin manages the entire application and he has the authority to block or unblock a customer or professional, or even remove them. He can also edit or delete a service.

Technologies used

Python: For writing the backend code.

Flask: For creating the backend API's

Flask-security: For Authentication and Authorization

Vue.js: JavaScript Framework for user interface

Bootstrap: For styling

SQLite: Database used for storing application data

Celery and Redis: For batch jobs and caching

Mailhog: For sending scheduled mails

DB Schema Design

User: Stores user data with columns: id (Integer), email (string), password (string), fs_uniquifier (string), active (Boolean)

Role: Stores three roles i.e admin, professional and customer with columns: id (integer), name (string)

UserRoles: Links the user with the role. Contains columns: id (integer), user_id (integer) and role_id (integer)

Professional: Stores the basic information of a service professional. Has columns: professional_id (integer), user_id (integer), name (string), service_type (string), experience (string) and pin(integer)

Customer: Stores the basic information of a customer. Has columns: customer_id (integer), user_id (integer), name(string), address (string) and pin (integer)

Service: Stores the service information which is provided by a professional. Contains columns: id (integer), name (string), price (integer), time_req (integer) and description (string)

ServiceRequest: Stores the request info of a service which is raised by a customer. Contains columns: id (integer), service_id (integer), customer_id (integer), professional_id (professional), req_date (datetime), comp_date (datetime), status (string), rating (integer) and review (string)

API Design

The APIs are designed to fetch and create/update/delete a particular service (or services), service request (or requests), to fetch customers and professionals and delete them (by admin only), and also to fetch reviews and ratings and give them (for customers). Following are the API endpoints used in my project –

- ServiceAPI: '/services/<int:service_id>'
- ServiceListAPI: '/services'
- ServiceReviewsAPI: '/services/<int:service_id>/reviews'
- ServiceRequestAPI: '/service_requests', '/service_requests/<int:request_id>'
- ServiceRequestFeedbackAPI: '/service_requests/<int:request_id>/feedback'
- ProfessionalListAPI: '/professionals', '/professionals/<int:professional_id>/toggle_block'
- CustomerListAPI: '/customers', '/customers/<int:customer_id>/toggle_block'
- UserDeleteAPI: '/users/<int:user_id>'

Architecture and Features

The project follows a Model-View-Controller (MVC) architecture, with a clear separation between the backend and frontend. Key features include: -

User Registration and Authentication: Both customers and professionals can register and log in securely and cannot access any route which was not meant for them.

Service Management: Professionals can create, edit, and delete their services, ensuring an up to date service detail.

Service Request Management: Professionals can accept or reject pending service requests and mark an accepted service request as completed, ensure proper tracking of a service request.

Feedback mechanism: Customers can give their feedback to a completed service request which was availed by them, in the form of rating and review.

Search feature: Professionals can search for a service request by customer name, pin etc and customers can search for a particular service, based on service name or service provider name or service provider pin code.

Timely Reports and Reminders: Automated email notifications are sent to users, providing monthly reports and daily reminders.

Video

https://drive.google.com/file/d/1KaRs9KM2q5Kepc1L3Eyyq3J83fbYpmQ4/view?usp=sharing