PROJECT REPORT

A-2-Z Household Services Web App (MAD – 2)

AUTHOR

Student Name: Shibashish Banerjee

Roll Number: 22f3003226

Email: 22f3003226@ds.study.iitm.ac.in

Hello! I am a diploma level student, who has always been curious and fascinated by coding and how all our servers and apps work. This was my first ever full-stack app, built completely on own and I'm proud of my efforts. I encountered many challenges that pushed me to expand my skillset and expertise.

PROJECT DESCRIPTION

The project involved developing a Household Services Application using Flask, SQLAlchemy, JS, Vue JS, HTML and CSS. It is a multi-user app (requires one admin and other service professionals / customers) which acts as platform for providing comprehensive home servicing and solutions. It incorporates features such as User authentication, Role based access, User Management for admin, Creation and management of service requests for professionals and customers.

TECHNOLOGIES USED

1.) **Flask:** Micro web framework for its simplicity and extensibility.
2.) **Flask-JWT-Extended**: User token management and authentication.
3.) **Flask-SQLAlchemy:** Simplifies database operations with SQLAlchemy support.
4.) **Vue Js:** JavaScript framework for building the user interface.
5.) **Flask-Restful:** for creating RESTful APIs**.**
6.) **Bootstrap:** Framework for responsive design.
7.) **Flask-Caching:** Adds caching for improved performance.
8.) **Celery**: Asynchronous task queue for background jobs like reminders and monthly reports.
9.) **Redis**: Caching layer and supports background jobs. Redis: Caching layer and supports background jobs.

10.) **Matplotlib**: for plotting graphs and charts in summary section of admin and user

API Endpoints

1. api.add_resource(LoginApi, '/api/login')

2. api.add_resource(SignupApi, '/api/signup')

3. api.add_resource(GetServicesForSignupApi, '/api/signup/services')

4. api.add_resource(DashboardApi, '/api/dashboard', '/api/dashboard/<int:userId>')

5. api.add_resource(ServiceApi, '/api/service', '/api/service/<int:service_id>')

6. api.add_resource(ServiceRequestApi, '/api/get_request/<int:service_id>', '/api/create_request/<int:service_id>')

7. api.add_resource(CustomerRequestApi, '/api/customer_request/<int:request_id>')

8. api.add_resource(ServiceRequestUpdateApi, '/api/request_status/<int:request_id>')

9. api.add_resource(CloseServiceRequestApi, '/api/close_request/<int:request_id>')

10. api.add_resource(ExportAPI, '/api/export')

11. api.add_resource(ProfessionalsApi, '/api/professionals')

12. api.add_resource(SummaryApi, '/api/summary')

DATABASE ARCHITECTURE

1.) **User Table**

Schema: id(Primary key), user_name, password, address, pincode, role, status, avg_rating, rating_count, service_proffessional_experience, last_seen, service_id

Foreign keys: service_id(References householdServices table)

Relationships: One-to-Many with householdServices and householdServiceRequest tables.

2.) **HouseholdServices Table**

Schema: id(Primary key), service_name, service_description, base_price, time_required.

Relationships: One-to-Many with User and HouseholdServiceRequest tables.

3.) **HouseholdServiceRequest Table**

Schema: id(Primary key), service_id, customer_id, service_proffessional_id, request_description, request_status, date_created, date_closed, rating_by_customer, review_by_customer.

Foreign keys: service_id(References householdServices), customer_id(References user table), service_proffessional_id(References user table)

Relationships: Many-to-One with householdServices and user tables.

Explanatory Video: VIDEO