MAD2 Project Report

A-Z Household Services Application V2

Author:

Name: Soumyadip Adhikari
Roll No.: DS23F2003716
E-mail: 23f2003716@ds.study.iitm.ac.in

I am a BS student from IIT Madras passionate about Programming, Data Science and their Implementations.

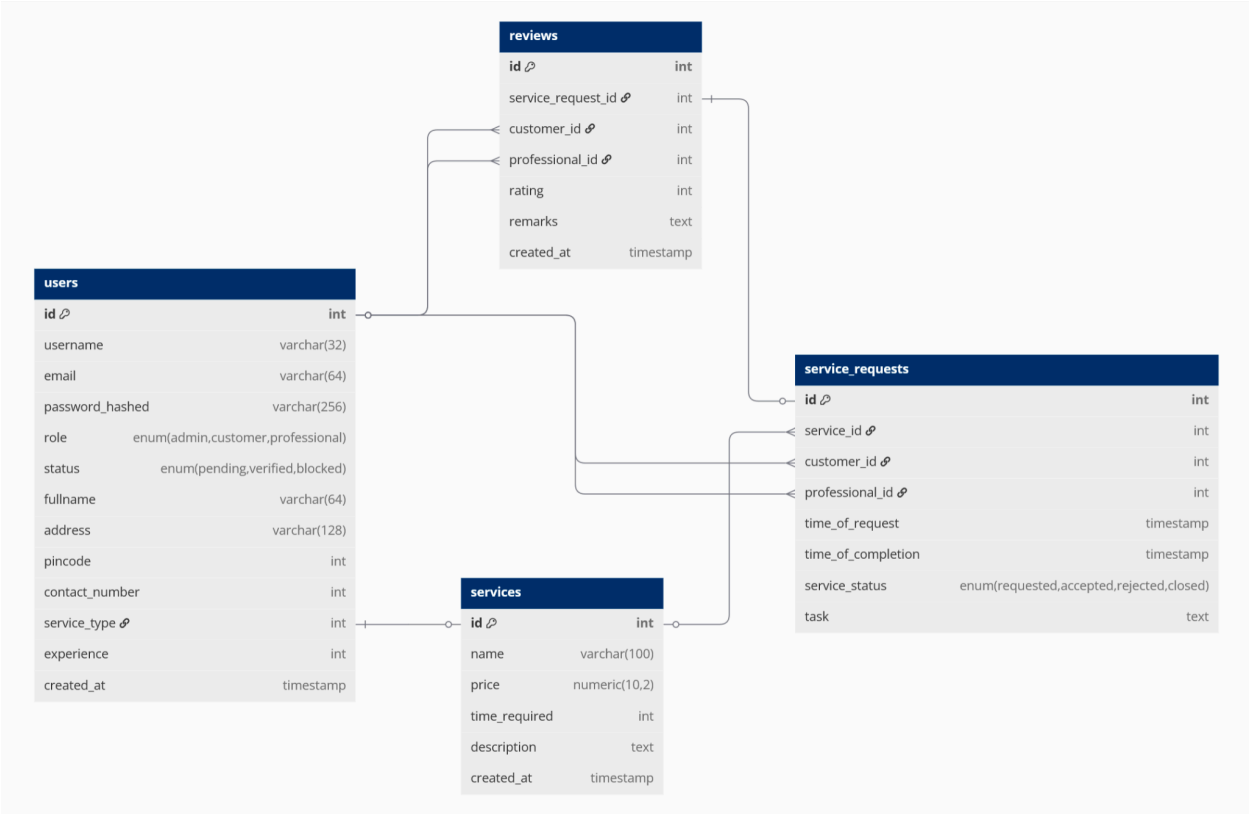
Description:

App Name: ServiceSphereV2
Description: It is a multi-user app (requires one admin and other service professionals/ customers) which acts as a platform for providing comprehensive home servicing and solutions, made for the project of the MAD - II course (Jan 2025 term).

Technologies Used:

- **Flask:** Backend framework in Python for building the web application.
- **Flask-Restful:** For building restful APIs in Flask.
- **Flask-JWT-Extended:** For authentication of users handling JSON Web Tokens.
- **Flask-Caching:** For caching.
- **Vue JS:** JavaScript framework for building Frontend of the web application.
- **Vite:** For Frontend tooling.
- **SQLAlchemy:** ORM (Object-Relational Mapping) tool for database interactions.
- **SQLite3:** Database management system for storing application data.
- **CSS + Bootstrap5:** For design and styling.
- **Werkzeug:** For hashing passwords.
- **ChartJS:** For making charts.
- **Celery:** For asynchronous background tasks.
- **Redis:** For Caching and as a message broker.

DB Model:



The schema of the database consists of four relations for users, services, service_requests, and reviews. All of them have relationships with the others, as can be inferred from the diagram.

Project Directory Structure:

```
root/
|    .gitignore
|    README.md
|
|---BackendFlask
|    |    .env
|    |    app.py
|    |    celerybeat-schedule
|    |    requirements.txt
|    |
|    |---application
|    |        config.py
|    |        init_db.py
|    |        models.py
|    |        tasks.py
|    |        workers.py
|    |
```

```
├──instance
│       db.sqlite3
│
└──templates
        c_monthly.html
        p_daily.html

└──FrontendVue
    │   index.html
    │   jsconfig.json
    │   package-lock.json
    │   package.json
    │   vite.config.js
    │
    ├──public
    │       favicon.ico
    │       ServiceSphere.png
    │
    └──src
        │   App.vue
        │   main.js
        │
        ├──assets
        │       style.css
        │
        ├──components
        │       alert.js
        │       Alert.vue
        │       Modal.vue
        │       Navbar.vue
        │       StarRating.vue
        │
        └──views
            │   errorpage.vue
            │   home.vue
            │   login.vue
            │   profile.vue
            │
            ├──admin
            │       a_home.vue
            │       a_search.vue
            │       a_summary.vue
            │
            ├──customers
            │       c_home.vue
            │       c_register.vue
            │       c_search.vue
            │       c_select_professional.vue
            │       c_summary.vue
            │
            └──professionals
                    p_home.vue
                    p_register.vue
                    p_search.vue
                    p_summary.vue
```

Above is the directory structure of the application.

API Design:

User Registration

- POST /api/register/customer
- POST /api/register/professional

Authentication

- POST /api/login

User Profile Management

- GET /api/profile/<username>
- PUT /api/profile/edit/<username>

Admin Dashboard

- GET /api/admin

Services Management

- GET /api/services
- POST /api/service/add
- PUT /api/service/<service_id>/edit
- DELETE /api/service/<service_id>/delete

User Management by Admin

- PUT /api/user/<user_id>/approve
- PATCH /api/user/<user_id>/block
- DELETE /api/user/<user_id>/delete

Admin Search

- GET /api/admin/search/<search_by>/<search_text>

Admin Summary

- GET /api/admin/summary

Export Data

- GET /api/export

Customer Dashboard

- GET /api/customer

Retrieves professional-related data.

- GET /api/<service_id>/select_professional

Service Requests Management by Customer

- POST /api/book/<service_id>/<professional_id>
- PUT /api/edit/<request_id>
- PATCH /api/close/<request_id>

Service Reviews

- POST /api/review/<request_id>

Customer Search

- GET /api/customer/search/<search_by>/<search_text>

Customer Summary

- GET /api/customer/summary

Professional Dashboard

- GET /api/professional

Services Requests Management by Professional

- PUT /api/service_request/<request_id>/accept
- PATCH /api/service_request/<request_id>/reject

Professional Search

- GET /api/professional/search/<search_by>/<search_text>

Professional Summary

- GET /api/professional/summary

Features at a Glance:

Roles:

- **Admin**: Root access; no registration required. Manage users and services, approve professionals, block users based on reviews or fraudulent activity.
- **Service Professional**: Register/Login, accept/reject service requests, handle service completion.
- **Customer**: Register/Login, view services, create/close service requests, provide reviews.

Services:

- **Create/Edit/Delete a Service**: Admin creates services with a base price. After that he can edit or delete the service along with the professionals associated with the service.

Service Requests:

- **Create/Edit/Delete a Service Request**: Customers create requests based on available services.
- **Accept/Reject Requests**: Professionals manage assigned service requests.

Core Functionalities:

- **Admin Dashboard**: Manage all users and professionals, approve or block based on profile or activity.
- **Service Management**: Admin handles creation and modification of services.
- **Customer & Professional Interaction**: Customers create service requests, professionals accept or reject them.
- **Daily Reminders**: Professionals get daily reminder mails about pending service requests.
- **Monthly Activity Report**: Customers get monthly report mails of their activity.

Other Functionalities:

- **Single Page Application**: Using VueJS, Vue Router and Vite.
- **Styling**: Aesthetic frontend using CSS/Bootstrap.
- **Login System**: Authentication with JWT with Flask extensions like Flask-JWT-Extended.
- **Analytics**: Pie charts and Bar charts using chart.js.

Video:

2025-03-28 17-54-26.mkv
https://drive.google.com/file/d/1AtdZ7S2efpjB6iW_AAqbe3vcv7gCtxpN/view