Author

Suyog Parkhi
22f3000976
22f3000976@ds.study.iitm.ac.in

I am a full-stack developer with expertise in both frontend and backend technologies. I enjoy building scalable applications and have a keen interest in designing user-friendly interfaces.

Description

This project involves developing a comprehensive household services platform that connects customers with professional service providers. The application manages user registration, service requests, professional verification, and review systems through a Flask backend API and Vue.js frontend.

Technologies Used

Backend

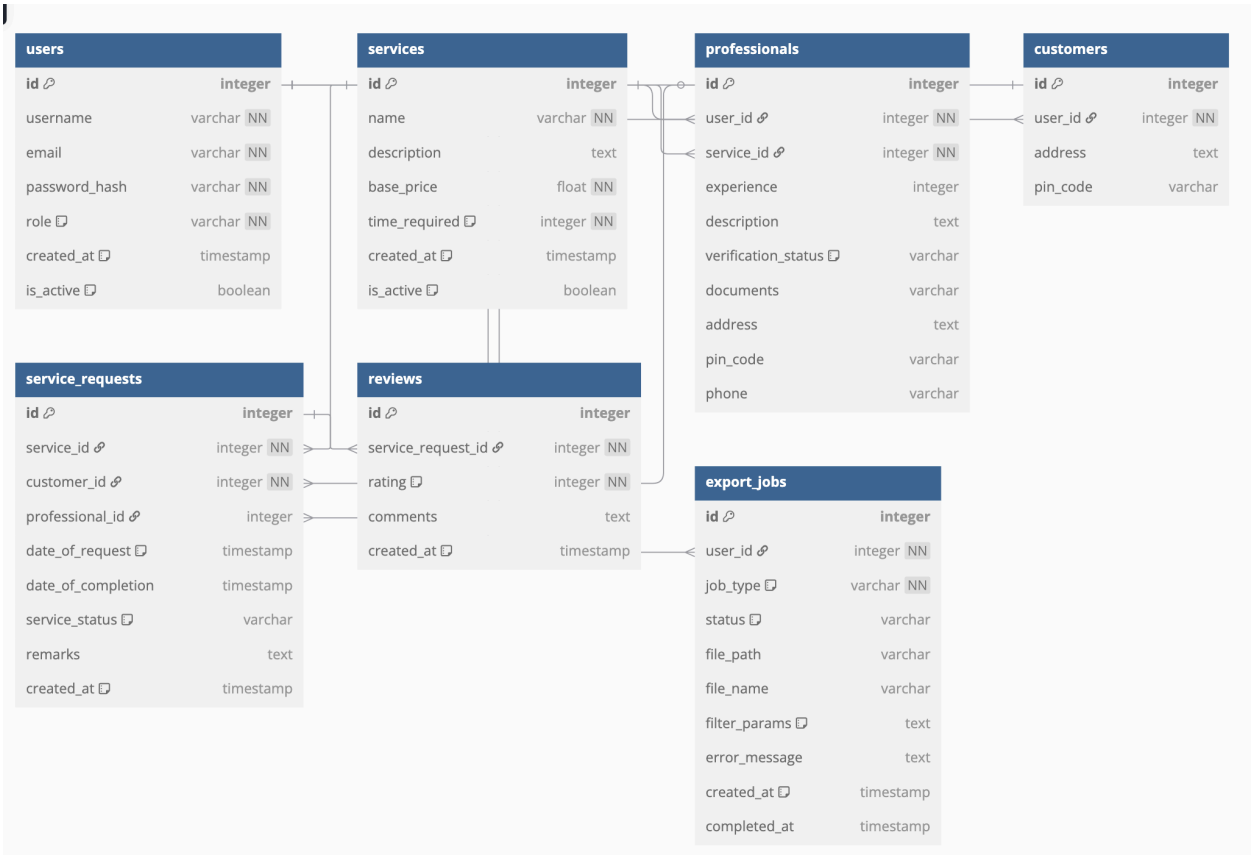
- Flask: Core web framework for building the REST API
- SQLAlchemy: ORM for database interactions
- Flask-JWT-Extended: For authentication and authorization
- Flask-Caching: For performance optimization through caching
- Celery: For background task processing and scheduling
- Redis: As a message broker for Celery and caching solution

Frontend

- Vue.js: Frontend framework for building single-page application
- Vuex: For centralized state management
- Vue Router: For client-side routing
- Bootstrap: For responsive UI components
- Axios: For HTTP requests to the backend API

These technologies were chosen to create a scalable, maintainable application with clear separation of concerns. The backend handles data processing and business logic, while the frontend provides an intuitive user interface.

DB Schema Design



The database schema follows a relational model with tables for users, services, service requests, and reviews, ensuring efficient data management, role-based access, and lifecycle tracking while maintaining data integrity through foreign key relationships.

API Design

The API follows a RESTful design pattern organized around user roles:

- Authentication APIs: /api/auth/ endpoints for login and registration
- Admin APIs: /api/admin/ endpoints for platform management and professional verification
- Customer APIs: /api/customer/ endpoints for service requests and reviews
- Professional APIs: /api/professional/ endpoints for managing service requests

JWT authentication is implemented with role-based access control. Each endpoint verifies the user's role before processing requests. API responses use consistent JSON formatting with appropriate HTTP status codes.

Architecture and Features

Architecture

- Backend: Follows a modular blueprint pattern
- models/: Database models with SQLAlchemy
- api/: Route blueprints (admin, auth, customer, professional)
- services/: Core business logic
- tasks/: Background jobs using Celery
- utils/: Utility functions and helpers
- Frontend: Component-based architecture
- views/: Page components organized by user role
- components/: Reusable UI components
- store/: Vuex modules for state management
- services/: API integration layer
- router/: Routes with role-based access control

Features

- Role-Based Access: Different interfaces for customers, professionals, and administrators
- Service Request Lifecycle: Complete flow from creation to completion
- Professional Verification: Admin approval process for professionals
- Review System: Rating and feedback for completed services
- Background Processing: Asynchronous task handling for exports and notifications
- Caching: Performance optimization for frequently accessed data
- Responsive Design: Mobile-friendly interface using Bootstrap

Additional features include service availability settings for professionals, export capabilities for administrators, and a comprehensive dashboard for all user types.

Video

Project Demo Video