**Author**

Vikash Prasad  
21F3002277  
 [21f3002277@ds.study.iitm.ac.in](mailto:21f3002277@ds.study.iitm.ac.in)  
As an online degree student of IIT Madras with a deep passion for programming and data science, I thoroughly enjoyed working on this project “Household Services Application”. The journey was both exhilarating and educational, presenting numerous challenges that tested my skills. Each obstacle I encountered was an opportunity for growth, and the sense of accomplishment I felt upon overcoming these hurdles was immensely rewarding. This project not only reinforced my technical abilities but also fueled my enthusiasm for tackling complex problems in the field of software development.

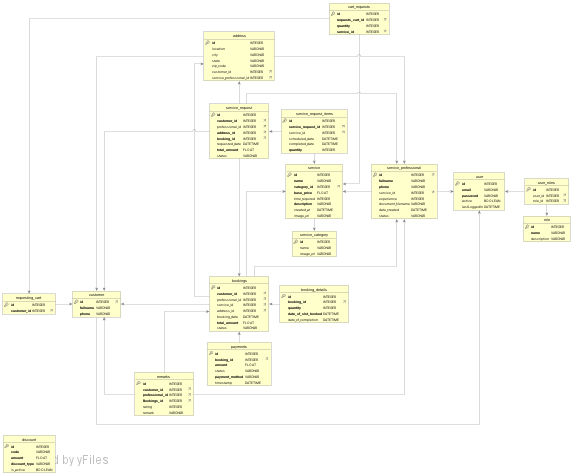
**Description**

The Household Services Application is a multi-user platform designed to provide comprehensive home services by connecting customers with skilled service professionals. The platform operates with three user roles: **Admin**, **Professional**, and **Customer**, ensuring streamlined service management and delivery. Built using frameworks like **SQLite**, **Flask**, **VueJS** and **VueCLI** , the app prioritizes simplicity, performance, and a seamless user experience. role differentiation.

**Technologies Used**

* **SQLite**: Database management.
* **Flask**: Backend API framework.
* **VueJS**: Frontend development for interactive UI.
* **VueCLI**: VueJS Advanced with CLI
* **Redis**: Caching and job queuing.
* **Bootstrap**: Responsive design and styling.
* **Jinja2**: Template rendering for emails and reports.
* **Celery**: Handling asynchronous tasks such as reminders and reporting

**DB Schema Design**

****

**Architecture and Features**

The project follows a layered architecture:

* Views (src/views/) :LandingPage.vue,RegisterPage.vue,LoginPage.vue,,HomePage.vue,SearchPage.vue,

StatisticsPage.vue,CartPage.vue,CheckoutPage.vue,ConfirmationPage.vue,BookingPage.vue

* Components(src/components/admin): CardPayment.vue, CashPayment.vue, Footer.vue, NavBar.vue, NetBankingPayment.vue, PayPage.vue, UpiPayment.vue

(admin/) AdminDashboard.vue, CategoryModal.vue, Customers.vue, Professionals.vue, SearchComponent.vue, SearchModal.vue, ServiceModal.vue, Services.vue, StatisticsComponent.vue

(customers/) AddressModal.vue, CustomerBookings.vue, CustomerDashboard.vue, CustomerSearch.vue, CustomerStat.vue, OrderConfirmation.vue, RegisterCustomer.vue,

RemarkModal.vue, SlotModal.vue

(professionals/) ProfessionalBookings.vue,ProfessionalDashboard.vue, ProfessionalSearch.vue, ProfessionalStat.vue, RegisterProfessional.vue, RequestModal.vue

*Features Implemented*:

### **Role-Based Access Control (RBAC)**

The platform supports RBAC with tailored dashboards for each role:

1. Admin Dashboard: Manage users, approve/reject professionals, create/update services.
2. Professional Dashboard: View and manage assigned service requests.
3. Customer Dashboard: Search services, create/edit/close service requests.

### **Admin Functionalities**

User Management:

* Approve/reject service professionals post verification.
* Block/unblock users based on activity and reviews.

Service Management:

* Create, update, and delete services with attributes like price, time required, and description.

Performance Monitoring:

* Search for professionals or customers based on filters like location or reviews.
* Trigger asynchronous CSV exports of service request details.

### Service Professional Functionalities

1. View and manage all assigned service requests.
2. Accept or reject service requests.
3. Close service requests post-completion.

### Customer Functionalities

1. Register and search for services by name, pin code, or location.
2. Create, edit, and close service requests.
3. Submit reviews and ratings for completed services.

## **Backend Jobs and Scheduling**

1. **Daily Reminders**:
   * Notify professionals of pending requests via email/SMS using **Google Chat Webhooks**.
2. **Monthly Reports**:
   * Generate detailed activity reports (HTML or PDF) summarizing requested and completed services.
   * Automatically send reports on the first day of each month.
3. **CSV Export**:
   * Admin-triggered batch jobs to generate and send CSV files of closed service requests.

## **Performance and Caching**

1. **Redis Caching**:
   * Implemented for frequently accessed data such as service listings.
   * Includes cache expiry to ensure data freshness.
2. **Optimized API Performance**:
   * Backend APIs are optimized to handle concurrent user requests efficiently.

**Video**

[**ServXpert-MAD II.mp4**](https://drive.google.com/file/d/1hNjhnNNd546ExkpGqBfgWEXnKbMevo6k/view?usp=sharing)