HOUSE HOLD SERVICE

- **STUDENT NAME:** PRUTHVI PRASAD S
- **STUDENT MAIL ID :** 23f1002103@ds.study.iitm.ac.in

DESCRIPTION: This is a multi user application that has with three main user roles: Admin, Customer, and Professional. Admin manage categories, approve professionals, and monitor activity. Customers can book services, track requests, and give feedback. Professionals can accept requests and manage their schedules after admin approval. The app is built using Vue.js (frontend) and Python Flask (backend), with Celery and Redis for handling background tasks like reminders and reports.

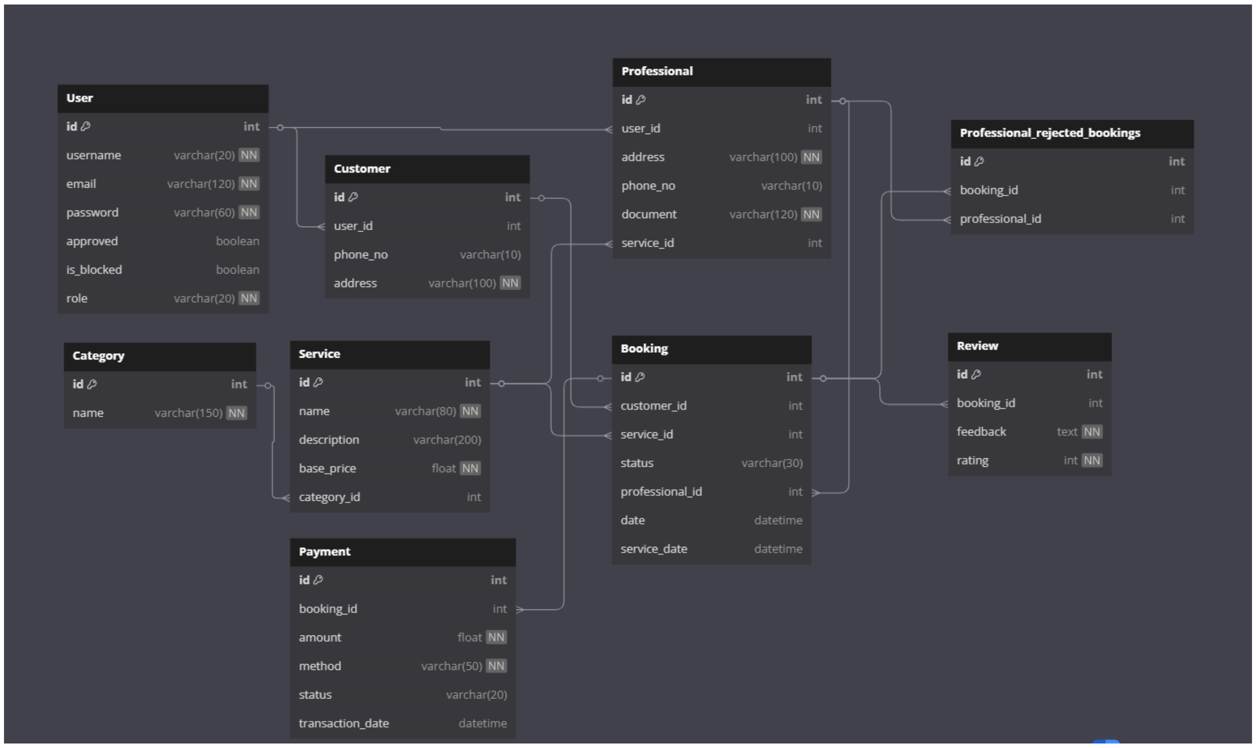
TOOLS WHICH USED:

- **FRONTEND**: Vue.js(for building user interfaces)
- **BACKEND**: Python Flask(for handling API and business logic)
- **DATABASE**:SQLite
- **Task Scheduling**:Celery (for handling background tasks)
- **Authentication:** Flask-JWT-Extended (for secure user login)
- **Message Broker:** Redis (for managing task queues)
- **Email Testing:** MailHog (for testing email functionality)

DATABASE DESIGN and MODELS:

- **User** – Stores user details like username, email, password, role (customer or professional), and approval status
- **Customer** – Contains customer-specific information such as phone number, address, and links to related bookings.
- **Professional** – Holds professional details like phone number, address, service offered, and document verification.
- **Service** – Defines available services with details like name, description, base price, and associated category.
- **Booking** – Manages customer bookings, including status, date, linked service, and assigned professional.
- **Category** – Groups services under categories for better organization.
- **Review** – Stores customer feedback and rating for completed bookings.
- **Professional Rejected Bookings** – Tracks bookings rejected by professionals.

- **Payment** – Records payment details for each booking.



API DESIGN:

The API handles all business logic and data management, allowing communication between the frontend (Vue.js) and the backend (Flask).

KEY API ENDPOINTS

1. /signup : Registers a new customer or professional. Professionals need admin approval to log in.
2. /login :Authenticates the user using email and password, generates a JWT token upon successful login.
3. /logout : Logs out the user by blacklisting the JWT token.
4. /categories : Creates a new category. Ensures the category name is unique.
5. /services : Creates a new service under an existing category.
6. /managecustomers : Retrieves a list of all customers.
7. /professionls: Retrieves a list of all registered professionals.
8. /bookingoverview : Retrieves a list of all service bookings, including customer and professional details.
9. /request_service :Create a new service booking.
10. /ongoingservices :View ongoing and pending service bookings.
11. /cancelbooking/<booking_id> : Cancel a pending booking.
12. /bookings/<booking_id>/update-date :Update the scheduled date of a booking.

13. /bookingstats :View customer-specific booking statistics.
14. /mybookings :View the customer's complete booking history.
15. /completedbookings :View the list of completed services.
16. /feedback/<booking_id>: Submit feedback and a rating for a completed booking.
17. /payments and /payments/<payment_id> : Make a payment for a service and View payment details.
18. /professional/bookings :Get new pending service requests.
19. /professional/bookings/update/<booking_id> :Accept or reject a booking.
20. /professional/scheduled_bookings : View all scheduled (accepted) bookings.
21. /servicehistory : View all completed services.
22. /professionalstats : View professional-specific booking statistics and average rating.
23. /export-closed-services :Admin can export closed service requests as a CSV file.
24. /searchservices : Customers can search for services by name or location.
25. /searchprofessionals :Admin can search for professionals by name, service, status, and rating.
26. /profile : Fetch or update user profile.
27. /create-csv : Trigger a CSV export using Celery.
28. /get-csv-data/<task_id>: Retrieve the CSV export data from Celery.

ARCHITETURE AND FEATURES:

The backend of Fixify is built using **Flask** following the **Model-View-Controller (MVC)** architecture. API endpoints are defined to handle user authentication, booking, search, profile updates, and admin operations. The database models are defined using SQLAlchemy. Authentication and authorization are handled using **JWT (JSON Web Tokens)**, ensuring secure and role-based access. Background tasks such as sending daily reminders to professionals, generating monthly reports, and exporting closed service requests are managed using **Celery** and Redis.Email notifications for booking confirmation and activity reports are generated using Flask-Mail and rendered with HTML templates stored in the templates folder. Flask configurations and Celery setup are managed through the config file. Implemented caching using flask_caching with Redis to improve performance and reduce database load. Cached booking data expires every **5 seconds**, ensuring real-time updates after cache expiry.

The frontend is developed using **Vue.js**, where components are organized under the src/components folder. Key views include the home page, login, signup pages, and role-based dashboards for admin, customer, and professional. State management is handled using **local storage** for storing authentication tokens and session data. Routing is managed using **Vue Router**, which ensures that users are directed to their respective dashboards based on their roles. The search feature allows customers to search for services by location and name, while admins can search for professionals by status and rating. Forms are validated using Vue's reactive state and native HTML validation

CLICK HERE FOR VIDEO LINK