Application Report: Services_4U

Roll no- 22f3003256

Name- Abhinav yadav

Gmail- 22f3003256@ds.study.iitm.ac.in

1. Introduction -: Service_4U CRUD application built using Flask and Vue.js (CDN). The application allows customers to request services from professionals, track service requests, and leave reviews. It also implements authentication, background task processing, and caching to enhance performance and user experience.

2. Technologies Used

- **Backend:** Flask (Python), Flask Security, SQLite3
- **Frontend:** Vue.js (CDN), Bootstrap
- **Database:** SQLite3
- **Authentication:** Flask Security
- **Background Tasks:** Celery, Redis
- **Caching:** Flask Caching
- **Email Notifications:** Flask Mail

3. Core Functionalities

- **User Authentication:**
  - Role-based authentication (Admin, Customer, Professional)
  - Token-based authentication using Flask Security
- **CRUD Operations:**
  - Customers can request services from professionals.
  - Professionals can accept or reject service requests.
  - Users can update their profiles.
  - Admins can manage users and services.
- **Service Request Management:**
  - Customers can create, track, and cancel service requests.
  - Professionals can accept, complete, or reject requests.
  - Status updates (Pending, Accepted, Completed, Canceled)
- **Background Tasks:**
  - Daily reminders sent via Flask Mail using Celery & Redis.
  - Automated database cleanup tasks.
- **Caching:**
  - Flask Caching used to optimize API responses.
  - Frequently accessed data stored in cache for performance improvements.

- **Review System:**
    - Customers can submit reviews and ratings for completed services..

4. System Architecture The application follows a client-server architecture:

- **Frontend:** Vue.js interacts with the Flask backend using AJAX requests.
- **Backend:** Flask serves API endpoints and handles authentication, database operations, and background tasks.
- **Database:** SQLite3 stores user information, service requests, and reviews.
- **Task Queue:** Celery and Redis handle background processing (email notifications, scheduled tasks).

5. Challenges Faced & Solutions

- **Session Management:** Implemented Flask Security for secure authentication.
- **Performance Optimization:** Used Flask Caching to reduce API response time.
- **Email Delivery Delays:** Configured Celery and Redis for asynchronous email processing.
- **Handling Concurrent Requests:** Optimized database queries to avoid bottlenecks.

6. Future Improvements

- Migrate from SQLite3 to PostgreSQL for scalability.
- Implement a real-time chat system between customers and professionals.
- Improve UI/UX with Vue components.
- Add multi-language support.

7. Conclusion This application successfully provides a platform for customers to connect with professionals, manage service requests, and leave feedback. The integration of Flask Security, Celery, Redis, and Vue.js ensures a smooth and efficient user experience. Further enhancements can be made to scale the application and improve its usability.

Video link
https://drive.google.com/file/d/1y4PQuolgP7o0HBm4p2VXlDHgn_9QjQTQ/view?usp=sharing