# Project Report: MAD I


## Created By:

Name: Abhijit Nandi

Roll: 22F1001555

Email: 22f1001555@ds.study.iitm.ac.in


## 1. Project Overview

Root Services is a web-based platform designed to connect customers with professionals offering household-related services such as plumbing, electrical work, cleaning, and home repairs. The platform allows customers to browse available services, request assistance from highly rated skilled professionals for any household jobs. Meanwhile, professionals can find work in their neighbourhood and generate good income.

## 2. Approach

The development of this project was structured around a RESTful API-based architecture with a well-defined database schema. The key steps in our approach included:

- Identifying core entities such as Users (Customers & Professionals), Services, Service Requests, and Reviews.
- Designing an Entity-Relationship (ER) Diagram to define relationships among tables.
- Implementing authentication and role-based access control using Flask-Security.
- Developing RESTful APIs for CRUD operations on services, service requests, and user management.
- Creating an interactive frontend using Vue.js and Bootstrap.
- Integrating Chart.js to visualize statistical data for service professionals and administrators.
- Utilizing Celery & Redis for background task processing (e.g., report generation).
- Implementing Flask-Caching to optimize API response times.

# 3. Frameworks and Libraries

## 3.1 Backend Technologies

- Flask: web framework for handling API requests.
- Flask-Security: User authentication and role-based access control.
- Flask SQLAlchemy: To Define models, perform query operations and commit changes to the database.
- Flask Restful: For creation of API resources.
- SQLite: Used as Database for the application.
- Celery + Redis:  For handling background tasks such as report generation.
- Flask-Caching: To cache frequently accessed data and improve performance.

## 3.2 Frontend Technologies

- Vue.js: JavaScript framework for building UI components.
- Bootstrap: For responsive and aesthetic UI design.
- Chart.js: For displaying graphical statistics on service requests.

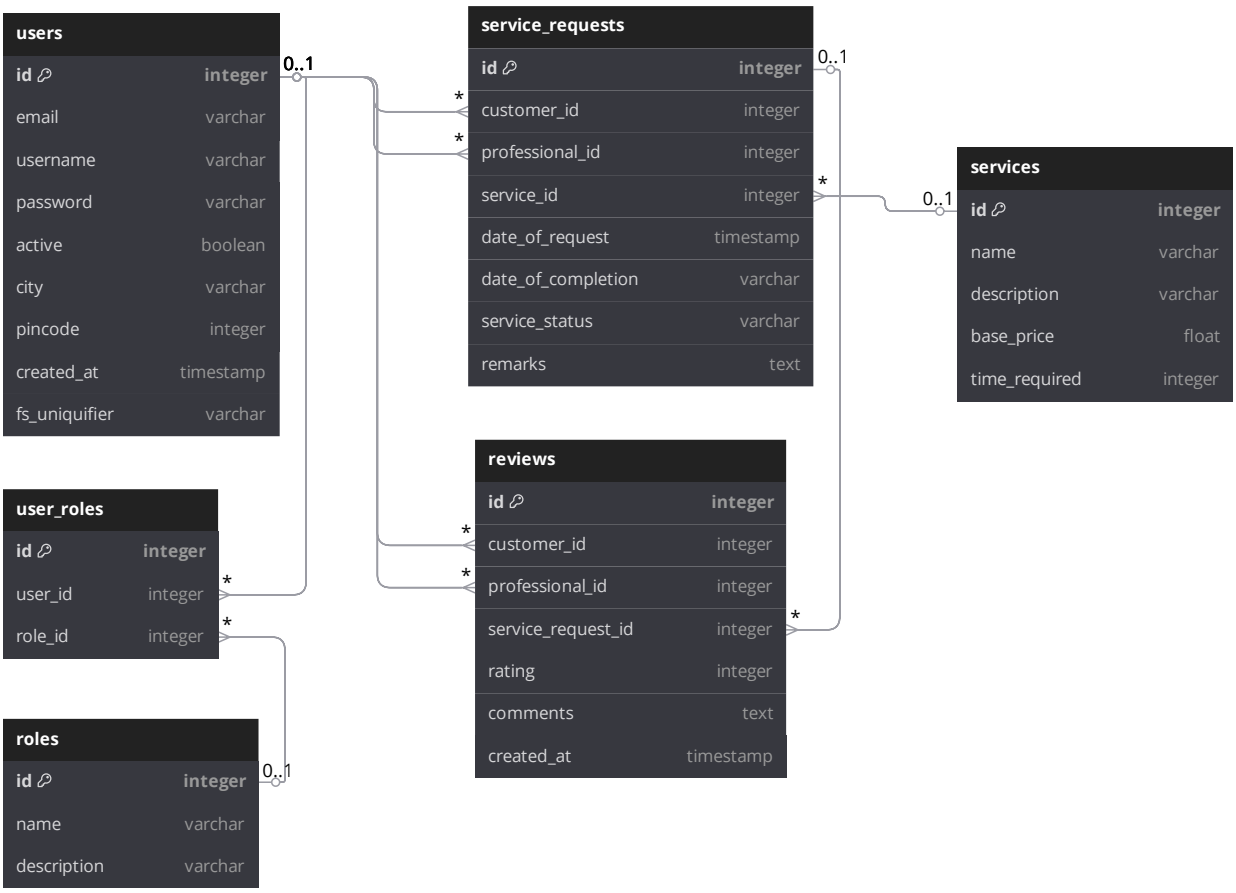
## 3.3 Database & Storage:

- SQLite: Used as the relational database for structured data storage.

# 4. API Resources Endpoints

| Method | Endpoint | Description |
|---|---|---|
| GET | /api/service-request/get | Retrieve all service requests |
| POST | /api/service-request/create | Create a new service request |
| PUT | /api/service-request/update/<id> | Update service request status |
| DELETE | /api/service-request/delete/<id> | Delete a service request |
| GET | /api/service | Fetch available services |
| POST | /api/service | Add a new service (Admin only) |
| PUT | /api/service/update/<id> | Update service details (Admin only) |
| DELETE | /api/service/delete/<id> | Delete a service (Admin only) |
| GET | /api/users | Retrieve list of users |
| GET | /api/admin-stats | Fetch statistics for admin dashboard |
| GET | /api/prof-stats | Fetch statistics for professionals |
| GET | /api/customer-stats | Fetch statistics for customers |

## 5. ER Diagram




## 6. Key Features

- Role-based Access Control: Customers and professionals have distinct access levels.
- Service Request Management: Customers can book, track, and review services.
- Professional Dashboard: Service providers can manage their assigned tasks efficiently.
- Admin Panel: Provides an overview of system performance and user activity.
- Data Visualization: Graphical representation of service trends using Chart.js.
- Automated Report Generation: Celery and Redis handle scheduled reports.
- Caching for Performance: Frequently accessed data is cached using Flask-Caching.

## 7. Presentation Link

MADII_22f1001555_project_demonstration