## Author

**AMAN KUSHWAHA**
**23f2001036**
**23f2001036@ds.study.iitm.ac.in**
I am also a Student at Jawaharlal Nehru University, New Delhi pursuing B. Tech in Electronics & Communication Engineering. My primary programming language is Python.

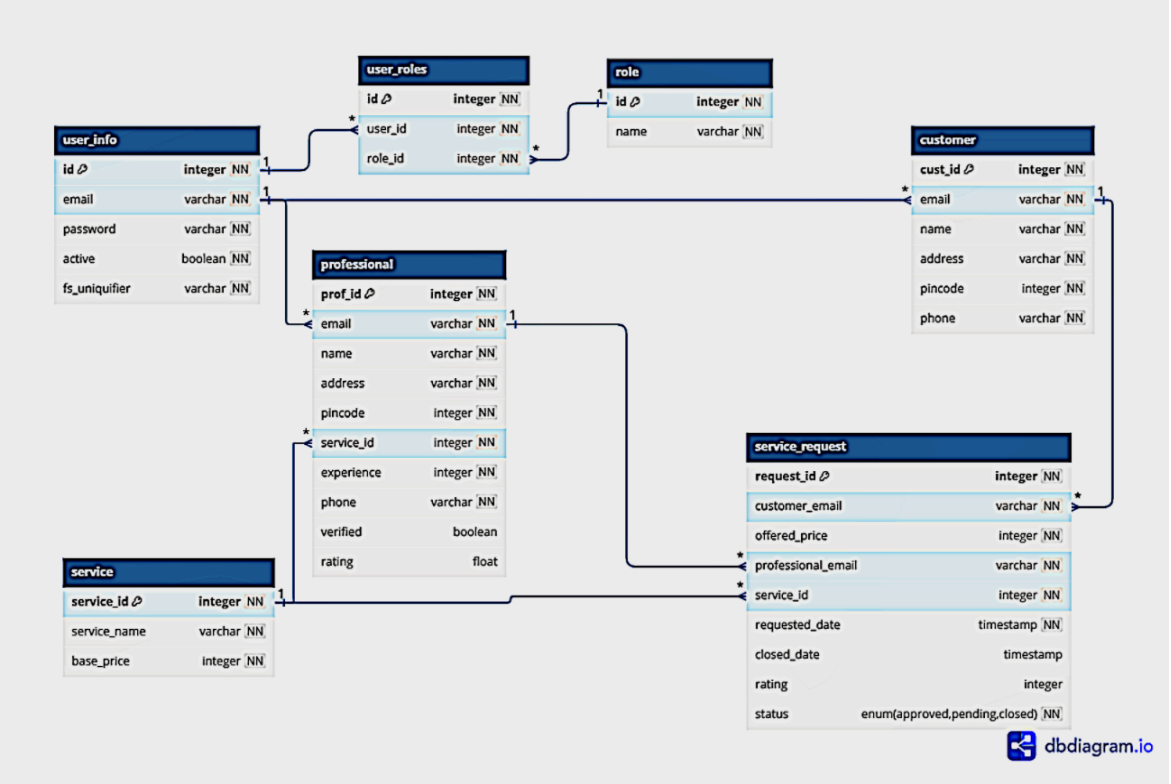
## Description

This project involves creating a service management platform with three user roles: Admin,Customer, and Service Professional. Users can log in or register through a forms for each role, and a database identifies each type of user. Admins have a dashboard where they can manage users, approve service professionals after verifying them, and block or unblock users. Admins can also create, update, or delete services with details like price . Customers can search for services using filters like location or pin code and create, or close service requests and customers can give rating to the service. Service professionals can view customer requests, accept or reject them, and mark them as completed after service delivery. The platform includes easy-to-use search features for customers and admins, making it simple to find services or manage users. Admin have ability to send mail to Customer and Professional and can generate a CSV file report of requests for himself.Overall, the system provides a clear and organized way to handle services and user interactions.

## Technologies used

- **Flask**: Flask has been used as a framework to manage the routes of the app
- **flask_sqlalchemy**: Flask SQL Alchemy has been used to access the ORM (Object Relation Model) database in SQLite (DB Browser).
- **flask_security:** used for authentication of user.
- **Flask-RESTful** – Extension used for building RESTful APIs in Flask and the Api endpoints created using this is used for CRUD operation.
- **Bootstrap**- Used bootstrap for styling.
- **Celery, Redis, MailHog,flask_caching:** Used caching for better performance and use redis,celery and mailhog for scheduling a mail.
- **VueJS:**Used for creating UI.
- **flask_excel**

## DB Schema Design




This database is designed for a **service booking platform** where customers can request services from professionals. It has a user management system that stores user details in the **`user_info`** table, and roles like "customer" or "professional" are assigned through the **`role`** and **`user_roles`** tables. Customers and professionals have their own separate tables **customer** and **professiona**l to store their specific details. The **`service`** table keeps a list of available services, while the **`service_request`** table tracks service bookings, including pricing, status, and customer ratings. This design helps manage users, roles, and service transactions efficiently, making it easy to use .

## API Design

Flask_RESTful api is used in this project.
This  API helps manage **services** and **service requests** between **customers** and **professionals**. The `ServiceApi` lets users add, update, or delete services, while the `ServiceRequestApi` allows customers to request services and professionals to respond. User roles control access, so professionals see only their relevant requests, customers track their own, and admins see everything. Security is ensured with authentication (`@auth_required('token')`), and caching (`@cache.memoize(timeout=5)`) improves speed. The system also updates professional ratings based on completed requests, making the platform  efficient.

## Architecture and Features

Project is in the mad2_project folder, which contains all the necessary components for the web application. The **backend** folder includes Python files such as **resources.py**, which manages CRUD operations and **routes.py**, which handles the application's routes for login and registration & **models.py** file defines the database schema and the relationships between different tables.It have subfolder celery which have all necessary files required for scheduling a email. The **instance** folder contains the **database.sqlite3** file, which is the SQLite database storing all the data. In the **frontend** folder, there are subfolders like **components** , which contains reusable component like Navbar and **styles** folder for CSS files (including **main.css**), The **pages** folder holds Javascript(VueJs) files , which are used to design different pages for the users. The application is run using the command **python3 app.py**, where the routes and application logic are executed. The models.py file handles the database schema,  users can log in or register here, with login credentials verified against the database. Admins can manage users (approve, block/unblock), services (create, update, delete), and professionals (approve proposals and verify profiles). Professionals can manage service requests, accept/reject them. Customers can search services, create or close service requests, and view their request status.

## Video

Mad2 project demo.mp4