**Students Details**

**Name : -** UNNAYAN SRIVASTAVA

**Roll No. : -** 23F3003811

**Email : -** 23f3003811@ds.study.iitm.ac.in

**Level : -** Diploma Level

**Subject : -** Modern Application Development 1 Project

**Phone No. : -** 9336991922

**Project Details**

**Project Statement: -** Household Services Application :It is a multi-user app (requires one admin and other service professionals/ customers) which acts as a platform for providing comprehensive home servicing and solutions.

**Frameworks Used: -** Python Flask, Jinja template, HTML, CSS, Bootstrap, SQLite, Restful API,

Werkzeug for exception handling.

**Libraries Used: -** flask, flask\_restful, werkzeug, datetime, sqlalchemy, flask\_sqlalchemy.

**Approach: -**

In this project, it was very clear that I would be requiring 3 different tables to store the details of customers, service professionals and one common table to store the Email ID and password of all the users including admin too. For the services, 2 tables were required to store the details of services and service required initiated by the user.

My first first objective was to develop the login page and to route the users based on their roles i.e. Admin, User or Service Professional. For this, I used basic login search based on email ID and password and then based on their roles, I redirected them to their respective dashboard. Also to register new uses.

My second task was to develop a basic layout for each role so that I don’t have to write the same code again and again for different functionality of a particular user. For this purpose, I used html and css to create layouts and then used block command to fill the body of the html depending upon the role and its functionality.

After that, I developed the home page for Admin using jinja templates as placeholders. I also developed routes in python flask for proper transfer between the urls and to fetch all the services for the database and also to add / edit the services. Similarly, to fetch the professionals details and service request details and their current status. All these things were queried from the database and then with the help of jinja templates displayed on the web application.

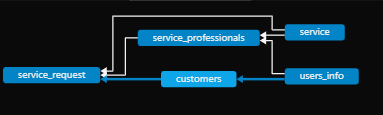
Later, I developed the Search functionality for the admin, where the admin can search based on service name, service request status, customer pincode and service professional service type. Also the admin can delete the user and service professional if he feels something inappropriate. For this I created a query to fetch all the details and then checked the query text in the fetched data. Advance sql command also helped me to do these things easily and quickly.

At last, I developed the summary page, where the admin can see how many services are requested, accepted and closed etc details about the service requests and the overall liking and disliking of the web application. For this, I wrote the query to fetch the remarks and based on some specific keywords, I counted them as liking and disliking. Similarly, Sql query was made to

To fetch the count of requested, accepted and closed requests and this entire data was returned to the html page. With the help of Chartist, the charts were dynamically generated on the web application and displayed on the screen.

FInally, a logout route was also developed for returning back to the login page. Similar approach is used in developing the customer and service professionals dashboard and their functionality. From the home page to search and also the summary page. For proper application and view API is also used to view datas at the backend easily.

**ER Diagram and Relation**



Here, **User\_info** table (id, email\_id, password, role)

**Customers** table(id, email\_id \\ foreign key user\_info, first\_name, last\_name, address, pincode, contact\_number)

**Service\_professionals** table(id, email\_id \\ foreign key user\_info, first\_name, last\_name, address, pincode, contact\_number, service\_type, documentname, date\_created, description, experience, status, rating)

**Service** table(id, name, base\_price, time\_required, description, photo)

**Service\_request** table(id, service\_id \\ foreign key service, customer\_id \\ foreign key customers, professional\_id \\ foreign key service\_professional, date\_of\_request, date\_of\_completion, service\_type, rating., remark)

**API Endpoints: -**

"/api/get\_services", "/api/post\_services", "/api/put\_services/<id>", "/api/delete\_services/<id>"

**Presentation LInk: -** <https://drive.google.com/file/d/1WxELGc0jWg5f-OZNbOsjdhqFCcHGXon9/view?usp=drive_link>