Project Report

A-Z Household Services Application

Author:

Name: Soumyadip Adhikari
Roll No.: DS23F2003716
E-mail: 23f2003716@ds.study.iitm.ac.in

I am a BS student from IIT Madras passionate about Data Science and its Implementations.

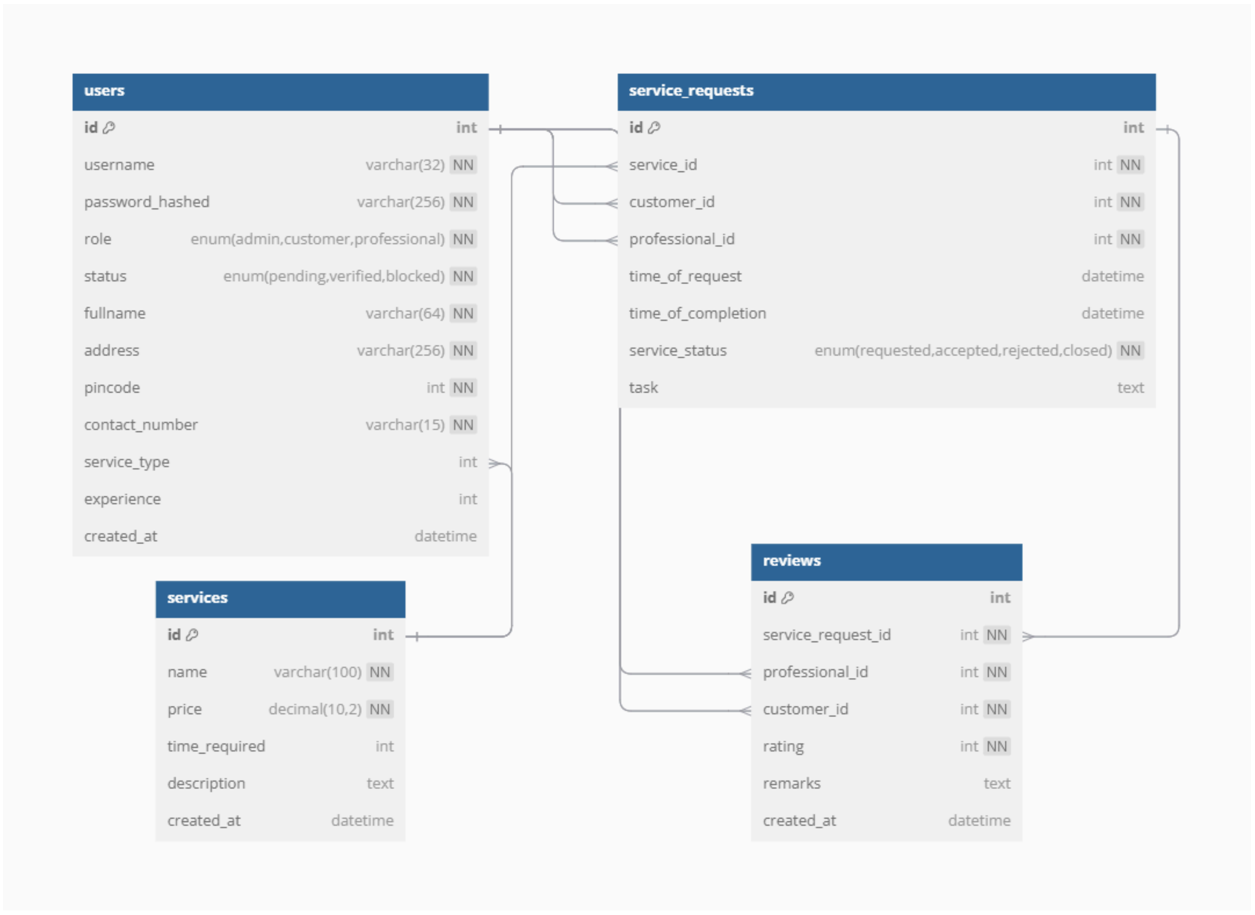
Description:

App Name: ServiceSphere
Description: A multi-user app which acts as a platform for providing comprehensive home servicing and solutions, for the final project of the MAD - I course (Sep 2024 term).

Technologies Used:

- Flask for backend
- Flask-SQLAlchemy as ORM
- Flask-Login for authentication of users
- Werkzeug for hashing passwords
- Chartjs for charts
- Jinja2 for HTML generation
- Vanilla CSS + Bootstrap for styling

DB Model:



The schema of the database consists of four relations for users, services, service_requests, and reviews. All of them have one-to-many relationships with the others, as can be inferred from the diagram.

I have added the services relation to allow for defining various service offerings that can be requested by customers and fulfilled by professionals.

The username, fullname, and name fields have maximum lengths of 32, 64, and 100 characters respectively. Additional constraints, such as minimum lengths, are enforced in the application controllers.

The username field must be unique and non-empty. Passwords are hashed, so their maximum length is set to 256 in the schema to accommodate the hash.

Each entity (user, service, service_request, and review) captures its creation time in the created_at field at the moment of creation

Architecture:

```
root/
│   .env
│   app.py
│   README.md
│   requirements.txt
│
├───application/
│       config.py
│       init_db.py
│       models.py
│       validations.py
│
├───instance/
│       db.sqlite3
│
├───static/
│   ├───css/
│   │       style.css
│   │
│   └───img/
│           favicon.ico
│           ServiceSphere.png
│
└───templates/
├──  │   base.html
├──  │   edit_profile.html
├──  │   index.html
├──  │   login.html
├──  │   navbar.html
├──  │   profile.html
├──  │
├──  ├───admin/
├──  │       a_add_service.html
├──  │       a_edit_service.html
├──  │       a_home.html
├──  │       a_summary.html
├──  │       a_user_details.html
├──  │
├──  ├───customers/
├──  │       c_book_service.html
├──  │       c_home.html
├──  │       c_register.html
├──  │       c_remarks.html
├──  │       c_select_professional.html
├──  │       c_summary.html
├──  │
└──  └───professionals
     ├── p_home.html
     ├── p_register.html
     └── p_summary.html
```

Above is the directory structure of the flask application.

Features at a Glance:

Roles:

- **Admin**: Root access; no registration required. Manage users and services, approve professionals, block users based on reviews or fraudulent activity.
- **Service Professional**: Register/Login, accept/reject service requests, visible based on reviews, handle service completion.
- **Customer**: Register/Login, view services, create/close service requests, provide reviews.

Services:

- **Create/Edit/Delete a Service**: Admin creates services with a base price. After that he can edit or delete the service along with the professionals associated with the service.

Service Requests:

- **Create/Edit/Delete a Service Request**: Customers create requests based on available services.
- **Accept/Reject Requests**: Professionals manage assigned service requests.

Core Functionalities:

- **Admin Dashboard**: Manage all users and professionals, approve or block based on profile or activity.
- **Service Management**: Admin handles creation and modification of services.
- **Customer & Professional Interaction**: Customers create service requests, professionals accept or close them.

Other Functionalities:

- **Form Validation**: Frontend with HTML5/JavaScript and backend validations.
- **Styling**: Aesthetic frontend using CSS/Bootstrap.
- **Login System**: Secure access with Flask extensions like Flask-Login.