MAD-2 Project Report

Author

- **Roll Number:** 22f2001166
- **Name:** Lakshya Manglani
- **Student Email:** 22f2001166@ds.study.iitm.ac.in

Project Description

The Household Services App is a web-based platform that connects Service Professionals with Customers for household services. Developed using Flask, SQLite, VueJS, Redis, and Celery, it supports three distinct user roles: Admin, Professional, and Customer. Admin manage users and services while monitoring service requests through graphical insights. Customers can create and monitor service requests, and Professionals browse and accept service requests. The platform offers daily reminders, monthly activity reports, CSV exports, and role-based access control (RBAC) to ensure efficient and secure user interactions. The app aims to optimize the household services process, focusing on performance, responsive design, and an intuitive user interface.

Technologies used

- Python is the core programming language used in the project.

- Frameworks and Libraries:

    - Flask is the main web framework used for developing the API.
    - VueJS is used for building the user interface.
    - Bootstrap is used for styling and responsive design.

- Database:
    - SQLite is the database used for storing and managing application data.
    - Flask-SQLAlchemy is utilized as the ORM for interacting with the SQLite database.

- Caching and Asynchronous Tasks:
    - Redis is used for caching to enhance performance.
    - Celery is implemented for handling asynchronous tasks and scheduled jobs.

Project Structure

1) Main File and Requirements:
    a) app.py: The core of the application, handling the initialization of the Flask app.
    b) requirements.txt: Contains all the required libraries and frameworks.

2) Backend Folder:
    a) models.py: Defines the SQLAlchemy models used for database interactions.
    b) resources.py: Implements the RESTful API resources.
    c) routes.py: Manages the routing of the application.
    d) celery_worker.py: Defines Celery tasks for handling asynchronous jobs.
    e) celery_config.py: for celery configurations.
    f) config.py: for configurations.
    g) extensions.py: for cache initialization.

h) create_initial_data.py: for creating roles (“admin”, “professional”, “customer”) and admin credentials.

3) Frontend Folder:

a) templates/index.html: contains an HTML template used by the Flask application.
b) static/js: Contains VueJS pages for different views within the application.
c) static/css: Custom basic CSS for styling the application.

Features

1. Role-Based Access Control system.
2. Admin can manage users, services, and flagged content with the approval of professionals.
3. Admin can see the graphs and search for users, services, and service requests.
4. Customers can create, search, manage service requests, and edit profiles.
5. Professionals can browse, search, manage service requests, and edit profiles.
6. Daily reminders and monthly activity reports sent via email.
7. Redis caching for faster data access with background task management via Celery.
8. Admin can export Service requests details as CSV files.

Video Link

MAD-II Project Video

API Endpoints

1. User Management:

a. GET /users
b. DELETE /users/<user_id>
c. PUT /users/<user_id>

2. Service Management:
    a. GET /services
    b. POST /services
    c. DELETE /services/<service_id>
    d. PUT /services/<service_id>

3. Service Requests Management:
    a. POST /requests
    b. DELETE /requests/<request_id>
    c. GET /requests
    d. PATCH /requests/<request_id>
    e. PUT /requests/<request_id>/accept

4. Service Requests Rating Management:
    a. POST /requests/<request_id>/rate

5. Profile Management:
    a. PUT /user-profile

6. CSV Exports and Download:
    a. POST /export-csv
    b. GET /download/<filename>

ER Diagram

Role:
One-to-Many: A role can be assigned to multiple users.

User:

Belongs to one Role (Foreign Key: role_id)
Can provide a Service (Foreign Key: service_id)
Can request multiple Service Requests (One-to-Many)
Can be assigned as a Professional to multiple Service Requests.

Service:
Can be requested in multiple Service Requests (One-to-Many)
Can have a User (provider)

Service Request:
Belongs to one User (Customer)
Assigned to one User (professional) (optional)
Associated with one Service.

Revoked Token:
Stores revoked JWT tokens (Independent Entity)