Project Report

About the Author

Name: Bhavya Aditya
Roll no.: 21f3001442
Email: 21f3001442@ds.study.iitm.ac.in
About: A graduate of Delhi University, I am an AI engineer with experience in working in early stage startups as well as MSMEs. I am also a swimmin and water sports enthusiast.

Description

This project is a household services application called Tradie Match, that allows users to request services from professionals (called tradies), tradies to accept and complete jobs, and admins to manage the users, tradies and service categories. It is a reactive app with analytics and scheduled email tasks.

Technologies Used

Backend: Flask, Flask-SQLAlchemy, Flask-Security, Flask-Mail, Flask-Migrate, Flask-CORS
Frontend: Vue 3, Vue Router, Bootstrap 5, ChartJS
Task Queue: Celery and Redis
Email Testing: MailHog
PDF Generation: xhtml2pdf
CSV Export: Pandas
Database: SQLite

Flask Security has been used to have a refined RBAC based user experience and MailHog has been used to demonstrate the scheduled emails functionality.

DB Schema Design

The database has the following entities: User, Service, and ServiceRequest. The names of Service and ServiceRequest are suggestive of what they are. Flask-Security enables the use of a single entity “User” for all the roles in the app.This is an umbrella entity which includes all the “users”, “tradies”, and the “admin”.

The detailed schema is accessible from the link below:

https://drive.google.com/file/d/1a7GajnKB6CC33aiK227YkQHytTE5kEXH/view?usp=sharing

API Design

API endpoints have been separated in different blueprints. The blueprints include admin, user, tradie, auth; each with its functionalites and APIs. APIs for each role include:

-Auth: Loggin in and out, user/tradie registration

- Admin: Dashboard data, analytics, CRUD for services and user/tradie activation, Celery-based CSV downloads.
- User: Service creation, cancellation, tradie selection, review system, profile edit, search/filter.
- Tradie: Dashboard data, apply to request, complete request, analytics, profile editing.

Architecture and Features

The app followes the MVC pattern with four Flask Blueprints: admin, user, tradie, and auth. Use of blueprints ensure the code is clean and readable and functionalities are separated as per the role.

The Vue frontend consists of components routed via Vue router. These components have been styled using Bootstrap 5 and some CSS.

There are scheduled backend jobs implemented using Celery. These jobs include daily tradie update emails reminding them about open, due, accepted, jobs; tradie analytics emails; and monthly summary reports for the users

An important feature is the analytics feature. Admin has a separate analytics dashboard where the weekly, monthly, and location wise breakdowns of services and revenue is presented using ChartJS charts. The admin can view the analytics for individual tradies as well. The tradies too, can view their analytics.

Another key feature of the app is that users can search for indivudual tradies and send service requests directly to them. Such requests separately visible to the respective tradies and are visible normally to other tradies as well. This ensures that the user can get more offers/applications for such requests and then make the choice, while tradies can prioritize between open service requests and request directly made to them.

The user always finalizes a tradie from among the applicants and doing so “books” a service request. Once a service request has been marked as complete by the tradie, the user can leave a rating plus review for that tradie/request.

In addition to searching for tradies, the user can also search by location or service. Appropirate filters have been provided for the same.

Demonstration Video

https://drive.google.com/file/d/19keVbA4FfUkRLuaxA5HeG7ZD6y4kCXaa/view?usp=sharing