UrbanAid

Sujal Raj Pradhan
23f2004759@ds.study.iitm.ac.in
I am currently a Teaching Assistant for Statistic-II at IIT Madras. Along with that, I am doing research work under Dr. Rituparna Datta. I like to lift and I have picked up a new fondness for performing card tricks.

Project Description

This project involves developing UrbanAid, a Household Services Platform that connects customers with service professionals. It includes managing the customer lifecycle (service requests, tracking, feedback), professional lifecycle (profile management, request handling), and admin lifecycle (service oversight, user management). Additionally, it requires caching for performance optimization and scheduled messages for timely notifications.

Technologies used:

Vue.JS Flask SQLite Redis

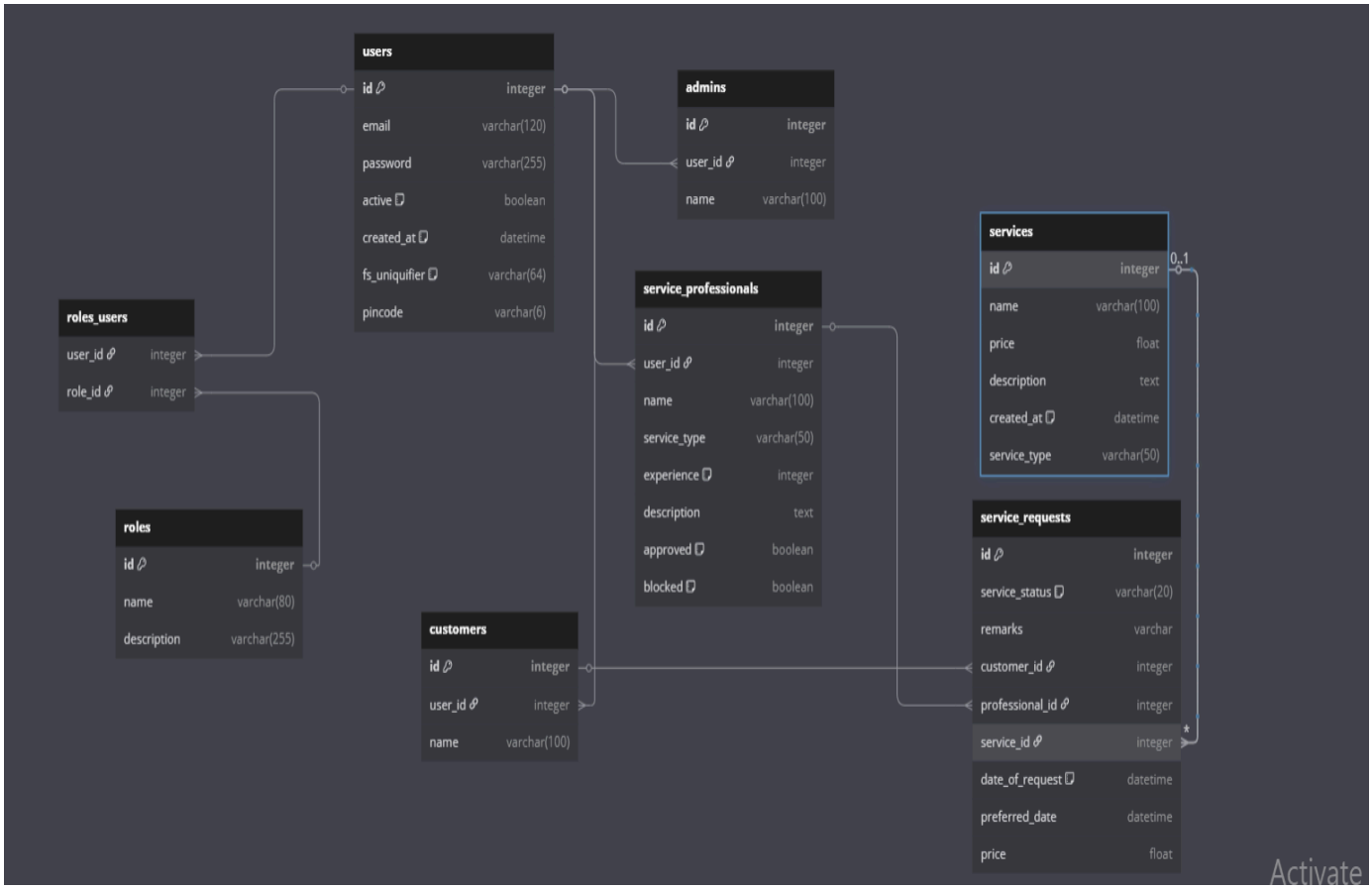
Celery

Chart.JS

Bootstrap

MailHog

DataBase Schema Design:



API Resource Endpoints in UrbanAid

UrbanAid provides a structured set of RESTful API endpoints categorized by user roles to facilitate seamless interaction between customers, service professionals, and administrators.

Authentication Endpoints

- POST /signup – Register a new user
- POST /signin – User login
- POST /signout – User logout

Customer Endpoints

- GET /customer/dashboard – Retrieve customer dashboard data
- POST /customer/services – Create a new service request
- PUT /customer/services/<request_id> – Modify an existing service request
- DELETE /customer/services/<request_id> – Close a service request
- GET /customer/search-services – Search for available services
- GET /service/<service_id>/professionals – View professionals offering a specific service

Professional Endpoints

- GET /professional/requests – View assigned service requests
- PUT /professional/requests/<request_id> – Update the status of a service request
- GET /professional/profile – Retrieve professional profile details

Admin Endpoints

- GET /admin/dashboard – View overall platform analytics
- GET /admin/service – Fetch all available services
- GET /admin/service/<service_id> – Retrieve details of a specific service
- POST /admin/service – Add a new service to the platform
- PUT /admin/service/<service_id> – Modify an existing service
- DELETE /admin/service/<service_id> – Remove a service from the system
- GET /admin/customers – Retrieve a list of all customers
- PUT /admin/customers/<customer_id> – Update customer account status
- GET /admin/professionals – Retrieve a list of all professionals

- PUT /admin/professionals/<professional_id> – Approve or block a professional
- GET /admin/search-professionals – Search for service professionals
- GET /admin/service/<service_id>/requests – Fetch service requests for a specific service
- GET /admin/service-requests – Retrieve all service requests across the platform

Background Tasks & Reports

- GET /downloadcsv – Initiate CSV report generation
- GET /getcsv/<task_id> – Download the generated CSV report

The application enforces authentication and role-based access control using @auth_required('token') and @roles_accepted() decorators to ensure secure and restricted access to endpoints based on user roles.

UrbanAid Application Overview Architecture:

UrbanAid follows a layered architecture ensuring clear separation of concerns:

- Core: Flask configuration (app.py)
- Models: SQLAlchemy ORM schemas (models.py)
- Controllers: RESTful API resources (routes.py) organized by roles
- Background Processing: Celery tasks (tasks.py, celery_instance.py)
- Configuration: Environment settings (config.py)
- Caching: Performance optimization (caching.py)
- Frontend: Vue.js components (views/) with global styling

Key Features

- Authentication: Role-based access control with token authentication
- User Roles: Admin, Customer, and Professional workflows
- Service Management: Full CRUD operations for services
- Service Lifecycle: Multi-state workflow (Requested → Accepted → Completed → Closed)
- Professional Approval: Admin verification of service providers
- Reporting: Background tasks for CSV exports and monthly reports
- Notifications: Automated email alerts for pending tasks
- Caching: Optimized response caching for performance
- Search: Service and professional filtering
- Responsive UI: Interactive and status-driven interface

UrbanAid ensures scalability, asynchronous processing, and robust API design, delivering an efficient platform for household services.

How to start the app?

In terminal execute these commands:

- python app.py (backend)
- npm run server (frontend)
- celery -A app:celery_app worker --loglevel=INFO --pool=solo (backend)
- celery -A app:celery_app beat --loglevel=INFO (backend)

Project Video Presentation Link:

MAD 2 Video Presentation.mkv