YASH VAISHNAV
Backend Developer

9343881830
codegeek004@gmail.com
linkedin.com/in/codegeek004
Indore, Madhya Pradesh, India
github.com/codegeek004/

SUMMARY

I am an experienced backend developer with a strong foundation in Django, Django REST Framework (DRF), and Flask, and hands-on experience with MySQL, PostgreSQL, and MongoDB. Driven by a passion for system design, I focus on building clean, efficient APIs and robust backend systems, with practical experience deploying applications on AWS and managing backend infrastructure.

SKILLS AND TOOLS

Python | Django | DRF | Flask | HTML | CSS | Bootstrap | Git | JavaScript | AWS Cloud | MySQL | MongoDB | PostgreSQL | OAuth 2.0 | OIDC | SAML-SSO | JWT | Redis | Celery | Docker | Apache Airflow | Postman | Sockets | Vim/Nano | Nginx & gunicorn | Linux | Tensorflow | OCR | System Design | Operating System

HACKATHONS

- Health hackathon organized by John Hopkins University
- 12 hour hackathon organized by Codemos Institute Indore

CERTIFICATIONS

- HTML, CSS, & JavaScript Course by John Hopkins University

BLOGS

- Authorization or Authentication using OAuth-2.0 Part-1
- Message Framing in TCP Sockets

EDUCATION

- B.Tech (Specialization in Cloud Computing and Automation)
- VIT Bhopal University
- CGPA: 7.62

PERSONAL INFORMATION

- Languages : English and Hindi
- Hobbies : Reading and Sports

EXPERIENCE

Backend Developer Intern **08/2024 - 01/2025**

Techritzy Pvt Ltd
Indore, Madhya Pradesh, India

- Developed web applications API in **DRF** and secured it using **JWT**, **OAuth 2.0** and **Role-Based Authorization**.
- Used **Celery Beat** to schedule reminders for each user according to the schedule from the Gemini API, placing tasks into **Redis**, acting as a message broker, which queued and forwarded them to Celery workers. The workers processed these tasks and sent emails through **Gmail's SMTP service**, configured within the application, with automatic retries to resend any failed notifications.
- Used **Apache Airflow** to manage and prioritize tasks for sending email notifications, ensuring critical emails were delivered first, automatically retrying failed attempts, and monitoring every task through the **Airflow UI** where I could visualize task status, review execution logs, track retry counts, and receive failure alerts, giving complete visibility and control over all scheduled actions.
- Deployed an application on **AWS** by on Ubuntu EC2 instance to run the application, linking it to a **MySQL** database on **RDS** placed in a Private Subnet, and setting up **VPC** and **Security Group** to control network access, with **Application Load Balancer** to distribute traffic across servers.
- Configured **Nginx** as the web server to manage all incoming HTTP/HTTPS requests, handle client connections efficiently, serve static assets such as CSS, JavaScript, and images directly, and forward dynamic requests to **Gunicorn** working as the application server, which processes Python application logic, executes Django routes, communicates with the database, and returns responses through Nginx
- Integrated **HTTPS** encryption on the domain by installing an **SSL Certificate** from **Let's Encrypt** and scripted automatic renewal every three months, ensuring secure data transmission and contributing to the overall SDLC lifecycle by enforcing security best practices during deployment and operations.

PROJECTS

PhotosPurge **source code**

- Developed a web application to address 15 GB Google Account Storage challenges, enabling users to delete category wise emails and migrate all Google Photos from one Google Account to another using **Django** and **MySQL**.
- Developed Gmail app for category wise email deletion and recovery. Implemented **OAuth 2.0** with Django Allauth to access user emails and offloaded tasks to **Redis Queues** processed by multiple **Celery Workers** asynchronously. Added custom middleware to refresh expired tokens automatically and **SMTP** notifications for task success or failure.
- Created Photos app for migrating Google photos. Fetched source account photos via Django Allauth authentication and implemented manual authentication for the destination account using credentials.json to obtain write access. Batched 100 photos per task into **Redis Queues** processed asynchronously by **Celery Workers**, with Custom Middleware to refresh tokens and **SMTP** for sending notifications.

Authentication API **source code**

- Developed a secure web API using **Django REST Framework(DRF)** for authentication and authorization implementing **JWT** authentication with SimpleJWT library, issuing access and refresh tokens and automatically refreshing them for active users to maintain uninterrupted sessions.
- Integrated **Multi Factor Authentication(MFA)** with Google Authenticator and Authy, added **Forgot Password** functionality, and Block IP feature after three failed login attempts to protect against attacks.
- Managed session and access control, enforcing Single Device Login by logging out from all other devices, and implemented **Role Based Authorization** with **Django Permissions** to define access levels for users and admins.

ParkEasy **source code**

- Built a parking management system using **Flask** and **MySQL** with **Session Based Authentication** and **OAuth 2.0** Authorization Code Grant flow with Google Sign-In for authentication without manually entering credentials.
- Added **Role Based Authorization** to manage access between users and admin. Configured **SMTP** to send payment receipts after each booking.
- Added APScheduler to run a background job every 5 minutes that clears expired bookings from the database managing slot availability and **Dockerized** application with deployment on Heroku.