INTERNATIONAL JOURNAL OF INFORMATIVE & FUTURISTIC RESEARCH (IJIFR)
ISSN: 2347-1697
VOLUME 13||ISSUE 8|| APRIL 2026
An Open Access, Peer Reviewed, Cross Reference DOI & Google Scholar Indexed International Research Journal
UGC Approved || IJIFR Impact Factor= 8.057
Estd Year: 2013 || Email: editors@ijifr.org
Availaible Online at: https://www.ijifr.org/

DOI: https://doi.org/10.64672/IJIFR/26.04.13.08.042 PUBLISHED ON: APRIL 21, 2026

INTELLIGENT COLLEGE PLACEMENT MANAGEMENT PLATFORM

K. Aparna [1], V.Vijayalakshmi [2]

[1]M.C.A. Student, [2] Assistant Professor
[1,2,] Department of Computer Applications,
Viswam Engineering College, Madanapalle, Andhra Pradesh, India

ABSTRACT

Campus placement plays a vital role in connecting engineering students with employers, but many institutions still rely on manual methods such as spreadsheets and emails, leading to inefficiencies. The Intelligent College Placement Management Platform is a full-stack web application developed using the Django framework to automate and streamline the entire placement process.The system supports three user roles: students, recruiters, and placement officers. Students can create profiles, upload resumes, apply for jobs, and track application status. Recruiters can post job openings, review applications, and issue offer letters. Placement officers manage the system through a centralized dashboard that provides real-time insights into placement activities.A key feature of the platform is the simulated AI-based matching engine, which uses natural language processing techniques to compare student skills with job requirements and generate a match score. This improves decision-making for both students and recruiters. The system also enforces placement policies such as the One-Student-One-Offer rule.Built using Python, Django, SQLite, and Bootstrap, the platform ensures efficient, transparent, and scalable placement management, improving coordination and reducing manual effort in campus recruitment processes.

KEYWORDS *Campus Placement System, Django Web Application, Placement Management, AI Matching Engine, Natural Language Processing (NLP), Resume Screening, Job Recommendation, Student Recruitment, Skill Matching, Full Stack Development*

PAPER CITATION:

Aparna, K., Vijayalakshmi, V. : " Intelligent College Placement Management Platform", International Journal of Informative & Futuristic Research (IJIFR), Vol. (13) (8), April 2026, pp. 1222-1230 ,https://doi.org/10.64672/IJIFR/26.04.13.08.042

1. INTRODUCTION

The relationship between academia and industry is most visibly realised during campus placements, where students transition into professional roles and institutions demonstrate their effectiveness as talent pipelines. This process involves coordination among placement officers, recruiters, and students, often under strict timelines and with rapidly changing information.

Traditionally, placement activities have been managed using tools like spreadsheets, emails, messaging apps, and notice boards. While functional individually, these methods create fragmented information systems where critical data—such as eligibility criteria, deadlines, and application status—is scattered. As a result, placement officers spend significant time on repetitive administrative tasks rather than

focusing on strategic responsibilities like industry engagement and performance analysis.Students also face challenges in this system due to inconsistent communication and lack of transparency. Job opportunities may not reach all eligible candidates, and there is often no systematic enforcement of eligibility rules. Furthermore, without structured job-skill matching, students may apply for unsuitable roles, leading to inefficiencies for both candidates and recruiters. Recruiters, in turn, must manually sift through large volumes of applications without tools to identify the best-fit candidates quickly.

The Intelligent College Placement Management Platform addresses these issues by providing a unified digital system that integrates all placement-related activities. It centralises student registration, job postings, application tracking, eligibility verification, document management, and reporting into a single web-based platform. This consolidation improves transparency, reduces administrative workload, and enables real-time monitoring of the placement process.

The system is developed using the Django framework, which offers robust features such as built-in authentication, an efficient ORM, and a ready-to-use administrative interface. Its Model-View-Template architecture ensures clear separation of concerns, making the application maintainable and scalable. Bootstrap 5 is used for the front-end to ensure responsiveness across devices.A key feature of the platform is the Simulated AI Matching Engine, which performs skill-based matching between students and job roles. Using Python-based string processing and set operations, the system calculates a match percentage based on overlapping skills and identifies gaps that students can work on. This approach is lightweight, interpretable, and does not require external AI services, while still providing meaningful decision support.

The platform emphasises usability for all users. Students can easily explore opportunities and track applications, recruiters can manage postings efficiently, and placement officers can monitor performance through intuitive dashboards. This focus on user experience ensures effective adoption and sustained usage.

1.2 Project Objectives

The primary objective of this project is to digitise and centralise the entire placement process, replacing fragmented manual systems with a unified platform. This enhances efficiency, accuracy, and communication.

Another objective is to implement role-based access control, ensuring that students, recruiters, and placement officers have tailored interfaces and restricted access to relevant data.

The system also aims to enforce placement policies automatically, such as the One-Student-One-Offer rule, ensuring fairness and consistency.

Additionally, the platform provides intelligent job matching to help students identify suitable roles and improve their skills, while assisting recruiters in finding relevant candidates.

It also supports document management, including resumes and offer letters, integrated within the application workflow.

Finally, the system generates analytics and reports, enabling placement officers to track performance, identify trends, and support institutional decision-making.

2.LITERATURE SURVEY

The development of the Intelligent College Placement Management Platform is supported by existing research and technologies in web development, software engineering, and artificial intelligence. This section reviews key contributions from literature that influence the design and implementation of the system. The foundation of the system is built on the Django framework, as described by Adrian Holovaty and Jacob Kaplan-Moss (2009). Their work highlights Django's emphasis on rapid development, clean design, and reusable components. The official documentation provided by the Django Software Foundation (2023) further explains its robust features such as authentication, ORM, and scalability, making it suitable for building data-driven web applications like placement systems.

Python serves as the core programming language for the project. According to Mark Lutz (2013), Python is valued for its simplicity, readability, and extensive library support. Similarly, Jeff Forcier et al. (2008) demonstrate how Python integrates effectively with Django to create scalable web applications.

For front-end design, Mark Otto and Jacob Thornton (2021) describe Bootstrap as a responsive and mobile-first framework. It ensures consistent user interfaces across devices, which is essential for accessibility in placement platforms.The system design is influenced by software engineering principles outlined by Ian Sommerville (2016) and Roger S. Pressman (2020). These works emphasise structured development processes, requirement analysis, and maintainable system architecture. Additionally, Martin Fowler (2002) and the design patterns introduced by Erich Gamma et al. (1994) provide guidance on building scalable and reusable software systems.

Database management is another critical aspect. The project utilises SQLite for development due to its lightweight nature, as documented by the SQLite Consortium (2023). Advanced database concepts are supported by Mike Bayer (2012) through SQLAlchemy and by the PostgreSQL Global Development Group (2023), which provides enterprise-level database solutions.The concept of intelligent matching is inspired by advancements in Natural Language Processing (NLP). Jacob Devlin et al. (2019) introduced BERT, a deep learning model for language understanding. Similarly, Nils Reimers and Iryna Gurevych (2019) proposed Sentence-BERT for generating sentence embeddings. While these advanced models require significant computational resources, they inspire the simplified AI matching approach used in this project.

Basic NLP techniques referenced from Steven Birdet al. (2009) support text processing and skill extraction. Additionally, data handling concepts from Wes McKinney (2017) contribute to efficient data manipulation in Python.Modern web applications also rely on asynchronous processing and APIs. Tools like Celery and Redis (2023) enable background task management, while Django REST Framework supports the creation of scalable APIs for system integration.

3. SYSTEM PROPOSAL

3.1 Existing System

Campus placement management in many engineering colleges ranges from fully manual methods to partially digital systems using general-purpose tools. The most common approach is spreadsheet-based management, where placement officers maintain Excel files for companies, job postings, student eligibility, and results. Although easy to use and low-cost, this method does not scale well. Large datasets become difficult to manage, formulas break easily, and there is no reliable support for multiple users or version control, leading to confusion and inefficiency.

Email is another widely used tool for communication. Job notifications, applications, and updates are shared through email, creating a distributed record across multiple inboxes. However, this system lacks centralisation, making it difficult to track application statuses or retrieve information efficiently. Both students and recruiters must manually search through emails, which becomes time-consuming as data grows.

Some institutions use platforms like Moodle or Google Classroom for placement activities. While these platforms support announcements and submissions, they are not designed for placement workflows. They lack features such as eligibility checks, application tracking, recruiter interfaces, and analytics dashboards, resulting in only partial automation.

Commercial platforms such as HirePro, AMCAT, and iXams provide advanced solutions including testing and analytics. However, they are often expensive and may not align with the specific policies or needs of smaller institutions.

Professional networking platforms like LinkedIn are also used to connect students with recruiters, but they do not support institutional requirements such as policy enforcement, document management, or administrative control.

3.1.1 Advantages of Existing System

(i) Easy to Use: Spreadsheet tools like Excel have a very low learning curve and are already familiar to most placement officers.

(ii) No Setup Cost: These systems can be used immediately without requiring additional infrastructure or software investment.

(iii) High Flexibility: Data can be easily modified, new columns can be added, and custom formulas or filters can be applied as needed.

(iv) Widely Accessible Communication: Email is universally used and ensures that all stakeholders receive important updates reliably.

(v) Formal and Traceable Records: Email communication provides an auditable history that can be referred to in case of disputes or clarification.

(vi) Quick Deployment: Manual and semi-digital systems can be implemented instantly without development time.

(vii) Human Decision-Making: Placement officers can apply judgment in special cases, such as relaxing eligibility criteria or extending deadlines.

(viii) Adaptability: The system can be adjusted informally based on changing requirements without needing technical modifications.

3.2 Proposed System

The Intelligent College Placement Management Platform is designed as a unified and structured solution to replace fragmented placement processes. Built using the Django framework with a relational database, the system ensures efficient data management, scalability, and automation of placement activities.

The platform introduces a well-defined data model that organises entities such as students, companies, job postings, applications, and offer letters into structured database tables. This enables efficient querying, reporting, and management of placement data.

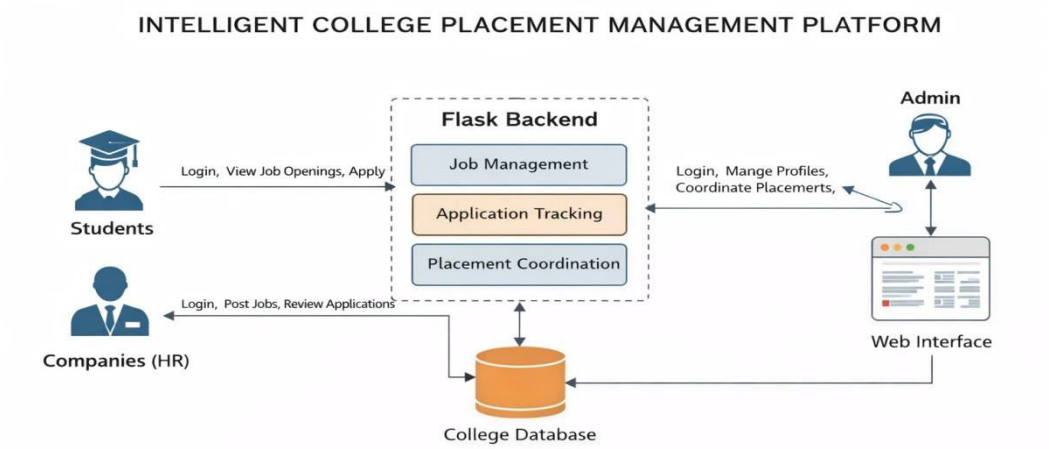

Figure 3.1: System architecture

A key feature is role-based access control, where students, recruiters, and placement officers interact only with relevant data. Students can view and apply for jobs, recruiters can manage postings and applicants, and placement officers have full system visibility and control.

The system also includes an AI-based matching engine that compares student skills with job requirements and generates match scores. This helps students identify suitable opportunities and allows recruiters to shortlist candidates more effectively.

Placement policies, such as the One-Student-One-Offer rule, are enforced automatically, ensuring fairness and consistency. The system also provides real-time notifications and a dashboard with analytics like placement statistics, company participation, and in-demand skills, enabling informed decision-making.

3.2.1 Disadvantages of the Proposed System

(i) Limited AI Accuracy: The matching engine relies on basic string matching and cannot capture semantic similarities (e.g., "ML" vs "Machine Learning").

(ii) No Advanced NLP Integration: Modern techniques like BERT or semantic embeddings are not implemented due to resource constraints.

(iii) Synchronous Processing: All operations run during request-response cycles, which may slow down performance as user load increases.

(iv) Lack of Background Task Handling: Tools like Celery and Redis are not integrated for asynchronous processing.

(v) Database Limitation: The use of SQLite restricts scalability and concurrent access in large deployments.

(vi) No External Notifications: The system lacks email or SMS alerts; users must log in to check updates.

(vii) Limited Real-World Scalability: Requires migration to PostgreSQL and additional optimisations for large-scale usage.

(viii) Dependency on Internet Access: Being a web-based system, it requires continuous connectivity or operation.

4. IMPLEMENTATION

4.1 Modules

The system is divided into seven main modules:

- User Authentication and Role Management
- Student Profile and Dashboard
- Company Profile and Job Management
- Placement Management and Policy Enforcement
- AI Skill Matching
- Administrative Dashboard and Reporting
- Configuration and Project Setup

These modules are implemented using the Django architecture and together form the complete system.

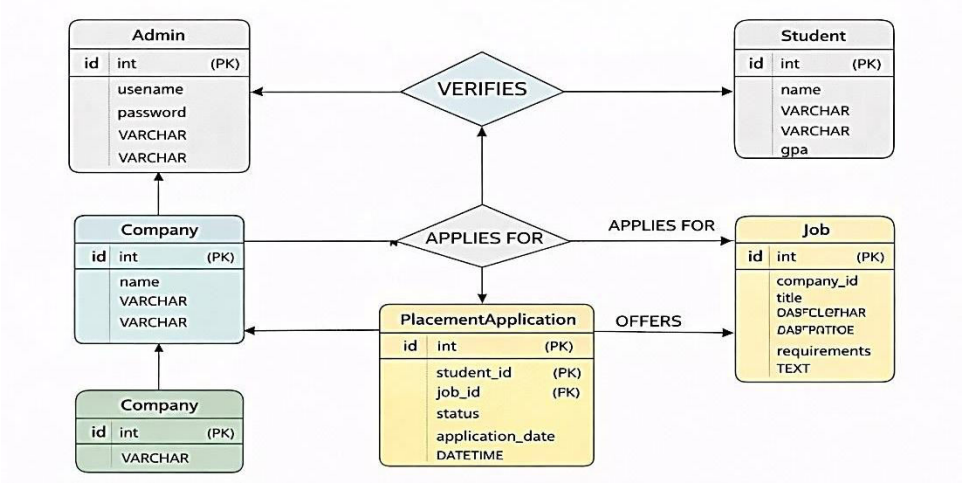

Figure 4.1 : ER Diagram(Working mechanism of the system)

4.2 Module Description

4.2.1 User Authentication and Role Management

This module manages registration, login, and user roles. A CustomUser model extends Django's authentication system with roles: Student, Company, and Admin. Users are redirected to role-specific dashboards after login, ensuring secure and personalised access.

4.2.2 Student Profile and Dashboard

This module manages student data such as academic details, skills, resume, and placement status. Students can view eligible jobs, apply, and track applications. Job recommendations are sorted using AI match scores.

It also includes an Interview Experience feature where students share interview details, helping others prepare.

Table 4.1: Student profile table

Field Name	Data Type	Description
id	Integer (PK)	Unique student ID
user_id	Foreign Key	Links to user account
full_name	CharField	Name of the student
Department	CharField	Branch/department
batch_year	Integer	Year of study batch
Cgpa	Decimal	Academic performance
Skills	TextField	List of technical skills
Resume	FileField	Uploaded resume file
is_placed	Boolean	Placement status

4.2.3 Company Profile and Job Management

Companies can register, create profiles, and post job opportunities. Each job includes required skills, CGPA criteria, salary, and deadline. Recruiters can view applications sorted by match score, enabling efficient candidate shortlisting.

Table 4.2 : Job Posting Table

Field Name	Data Type	Description
id	Integer (PK)	Unique job ID
company_id	Foreign Key	Linked company
job_title	CharField	Title of the job
required_skills	TextField	Required technical skills
min_cgpa	Decimal	Minimum eligibility CGPA
expected_ctc	Float	Salary package (LPA)
job_tier	CharField	Standard / Dream / Super Dream
deadline	CharField	Application closing date
is_active	Boolean	Job availability status

4.2.4 Placement Management and Policy Enforcement

This module manages applications, offer letters, and recruitment events.

It enforces rules like the One-Student-One-Offer policy by checking eligibility before allowing applications. Application statuses (e.g., Applied, Shortlisted, Offered) are tracked systematically.

Table 4.3 Application Table

Field Name	Data Type	Description
id	Integer (PK)	Unique application ID
student_id	Foreign Key	Applied student
job_id	Foreign Key	Applied job
status	CharField	Applied / Shortlisted / Offered
match_score	Integer	AI-based skill match percentage
applied_on	DateTime	Application timestamp

4.2.5 Simulated AI Skill Matching

This module calculates match scores between student skills and job requirements.

- Skills are processed using Python sets
- Match score is based on skill overlap percentage
- Missing skills are identified and suggested to students

This provides simple but effective decision support.

4.2.6 Administrative Dashboard and Reporting

- The admin dashboard (via Django Admin) allows placement officers to manage all data.

- It provides real-time analytics such as:
- Total students and companies
- Placement count
- Active job postings
- In-demand skills

This replaces manual reporting with automated insights.

4.2.7 Configuration and Project Setup

This module includes project settings, URL routing, and database configuration.

- Uses SQLite for development
- Custom user model configured
- Static and media files managed
- Database schema handled using migrations

5. Results and Discussions

The Intelligent College Placement Management Platform was successfully implemented using the Django framework, providing a unified solution for managing campus placements. The system effectively integrates student registration, job posting, application tracking, and administrative monitoring into a single platform.

The user authentication system ensured secure access with role-based dashboards for students, recruiters, and administrators. Students were able to create profiles, upload resumes, and apply for jobs based on eligibility criteria such as CGPA. Job listings were displayed along with AI-generated match scores, helping students prioritise suitable opportunities.

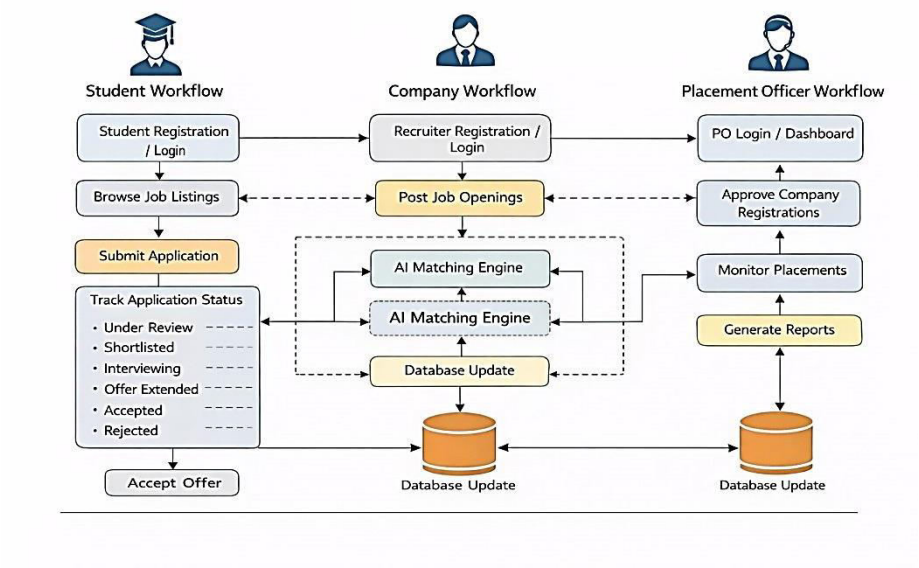

Figure 5.1 : Results of the system

Recruiters could post jobs and view applicants sorted by match score, significantly reducing manual screening effort. The placement management module successfully enforced policies like the One-Student-One-Offer rule by restricting invalid applications and providing clear feedback.

The AI Skill Matching module generated match percentages based on skill overlap and highlighted missing skills, offering useful guidance for students to improve their profiles. The administrative dashboard provided real-time insights such as total placements, active job postings, and in-demand skills, eliminating the need for manual report generation.

The results demonstrate that the system improves efficiency, transparency, and coordination compared to traditional methods. Centralised data management reduces errors and ensures consistent communication among stakeholders.

However, some limitations exist. The AI matching approach relies on exact keyword matching and lacks advanced semantic understanding. The use of SQLite may limit scalability in large deployments, and the absence of asynchronous processing can affect performance under heavy load.

6. CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The Intelligent College Placement Management Platform provides a complete solution to the challenges of campus placement management by integrating all activities into a single system. It replaces fragmented manual processes with a structured and automated platform that improves efficiency, transparency, and coordination among students, recruiters, and placement officers.Developed using the Django framework, the system leverages features such as authentication, ORM, and admin interface to build a scalable and maintainable application. The Model-View-Template architecture ensures clean separation of components, making future enhancements easier.A key feature of the system is the AI Skill Matching Engine, which calculates match scores between student skills and job requirements. This helps students choose suitable jobs and enables recruiters to shortlist candidates efficiently. The system also identifies missing skills, guiding students in improving their profiles.The platform enforces placement policies such as the One-Student-One-Offer rule automatically, ensuring fairness and reducing manual errors. Integrated document management for resumes and offer letters further streamlines the placement workflow.

6.2 Future Enhancement

- Advanced AI Matching: Integrate semantic models like Sentence-BERT to improve skill matching accuracy beyond keyword comparison.
- Resume Parsing: Automate skill extraction from resumes using NLP tools to reduce manual data entry and improve consistency.
- Database Upgrade: Replace SQLite with PostgreSQL for better scalability and concurrency.
- Asynchronous Processing: Use Celery with Redis to handle background tasks and improve performance.
- Notification System: Add email and SMS alerts using services like Twilio for real-time updates.
- Mobile Application: Develop a mobile app using frameworks like React Native or Flutter for better accessibility and push notifications.
- Predictive Analytics: Implement machine learning models to predict placement outcomes and identify students needing support.

7. REFERENCES

[1] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*, 2nd ed. New York, NY, USA: Apress, 2009.
[2] Django Software Foundation, "Django Documentation, Version 4.2," 2023. [Online]. Available: https://docs.djangoproject.com/en/4.2/
[3] J. Forcier, P. Bissex, and W. J. Chun, *Python Web Development with Django*. Upper Saddle River, NJ, USA: Addison-Wesley, 2008.
[4] M. Lutz, *Learning Python*, 5th ed. Sebastopol, CA, USA: O'Reilly Media, 2013.
[5] M. Otto and J. Thornton, "Bootstrap 5 Documentation," 2021. [Online]. Available: https://getbootstrap.com/docs/5.0/
[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992.
[8] SQLite Consortium, "SQLite Documentation," 2023. [Online]. Available: https://www.sqlite.org/docs.html
[9] M. Bayer, "SQLAlchemy," in *The Architecture of Open Source Applications, Vol. II*, A. Brown and G. Wilson, Eds., 2012. [Online]. Available: http://aosabook.org/en/sqlalchemy.html
[10] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley, 1994.
[12] Python Software Foundation, "Python 3.10 Documentation," 2023. [Online]. Available: https://docs.python.org/3.10/
[13] I. Sommerville, *Software Engineering*, 10th ed. Harlow, U.K.: Pearson, 2016.
[14] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2020.
[15] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. Sebastopol, CA, USA: O'Reilly Media, 2009.
[16] W. McKinney, *Python for Data Analysis*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2017.
[17] Celery Project, "Celery: Distributed Task Queue Documentation," 2023. [Online]. Available: https://docs.celeryq.dev/
[18] Redis Ltd., "Redis Documentation," 2023. [Online]. Available: https://redis.io/docs/
[19] PostgreSQL Global Development Group, "PostgreSQL 15 Documentation," 2023. [Online]. Available: https://www.postgresql.org/docs/15/
[20] Django REST Framework, "Django REST Framework Documentation," 2023. [Online]. Available: https://www.django-rest-framework.org/