Engineering and Scientific International Journal (ESIJ)
Volume 13, Issue 2, April – June 2026
ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

Smart Job Portal With AI-Powered Skill Matching, ATS Scoring and Intelligent Career Assistance using Django Framework

P. Hema[*1], B. Shireesha[2]

[1]*Student, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*
[2]*Assistant Professor, Department of Computer Applications, Viswam Engineering College, Andhra Pradesh, India*

Abstract —The rapid expansion of digital recruitment platforms has intensified the challenges associated with efficient candidate-job matching, manual resume screening, and lack of personalized career guidance. This paper presents a Smart Job Portal with AI-Powered Skill Matching and Career Assistance, a full-stack web-based system designed to streamline recruitment processes and enhance candidate experience through intelligent automation. The system is developed using the Django framework with Python as the core programming language and SQLite as the backend database. It incorporates an advanced Applicant Tracking System (ATS) that computes match scores by comparing candidate skills from both profile data and resume text against job requirements. The system further integrates AI-powered resume advice and chatbot assistance using large language model APIs such as OpenAI and Google Gemini. The platform supports multi-role access for candidates and HR professionals, providing dedicated dashboards and workflows. Candidates can build resumes, apply for jobs, track applications, and receive personalized suggestions, while recruiters can post jobs, evaluate applicants using ATS scores, and manage interview scheduling within a unified system. Experimental evaluation demonstrates improved recruitment efficiency, reduced manual effort, and enhanced candidate engagement. The proposed system offers a scalable and intelligent alternative to traditional job portals by combining automation, artificial intelligence, and full-stack development principles.

Keywords: *Smart Job Portal; ATS Score; Skill Matching; Artificial Intelligence; Resume Builder; Django; Recruitment Automation.*

1. Introduction

The evolution of digital recruitment systems has significantly transformed the employment landscape, yet challenges such as inefficient resume screening, lack of intelligent matching, and fragmented recruitment workflows persist. Traditional recruitment methods rely heavily on manual evaluation, leading to delays, inconsistencies, and missed opportunities in identifying suitable candidates. To address these challenges, this paper proposes a Smart Job Portal with Core Technologies, integrating automated skill matching, AI-driven resume advice, and an interactive chatbot into a unified recruitment system. By combining full-stack web technologies with artificial intelligence, the proposed system enhances recruitment efficiency, reduces manual workload, and democratizes access to career guidance.

2. Literature survey

Existing recruitment systems can be broadly categorized into general job portals, enterprise Applicant Tracking Systems (ATS), and AI-driven recruitment tools. General platforms such as LinkedIn and Naukri provide large-scale job listings but lack advanced matching capabilities. Enterprise systems offer workflow automation but are expensive and complex to deploy. AI-based recruitment tools focus on specific functionalities such as candidate assessment but lack integration across the recruitment lifecycle.

Research in Natural Language Processing has enabled techniques such as keyword extraction and similarity analysis for resume evaluation. Machine learning frameworks such as scikit-learn have been widely used fortext processing and classification tasks. Recent advancements in large language models have introduced AI-powered career guidance systems, enabling personalized recommendations. However, most existing solutions lack integration into a unified recruitment platform. The proposed system bridges this gap by combining ATS-based matching, AI advisory features, and full-stack implementation into a single cohesive system.

3. System architecture and design

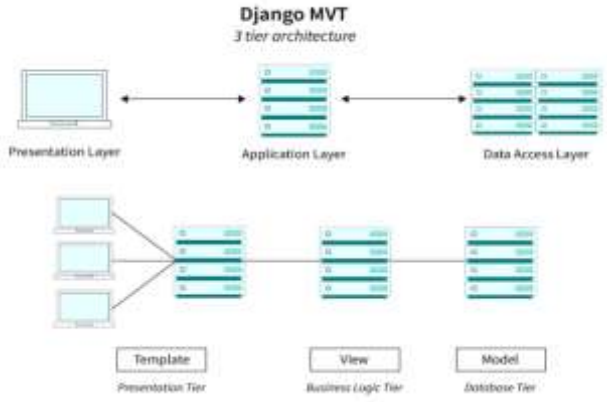

Fig.1: System Architecture

DOI: **10.30726/esij/v13.i2.2026.13200237**

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

The Smart Job Portal with Core Technologies is designed using a three-tier web application architecture, ensuring modularity, scalability, and efficient separation of concerns. The architecture consists of the following layers:

Presentation Layer: The presentation layer is responsible for user interaction and interface rendering.
Technologies used:
- HTML5
- CSS3
- Bootstrap
- JavaScript

Features:
- Candidate Dashboard
- HR Dashboard
- Job Search Interface
- Resume Builder UI
- Chatbot Interface

Django templates dynamically render content based on user roles.

Application Logic Layer: The core logic is implemented using the Django framework with Python.
Responsibilities:
- URL routing and request handling
- Business logic execution
- ATS score computation
- Resume parsing and skill extraction
- AI integration for advice and chatbot

Key components:
- views.py → request handling
- utils.py → ATS matching logic
- ai_utils.py → AI integration

Data Layer: The system uses SQLite for persistent data storage.
Database entities include:
- User
- CandidateProfile
- Job
- Application
- InterviewSchedule
- ResumeData

Django ORM ensures secure and efficient database operations.

External Services Layer: The system integrates with AI APIs:
- OpenAI
- Google Gemini

Used for:
- Resume advice generation
- Chatbot responses

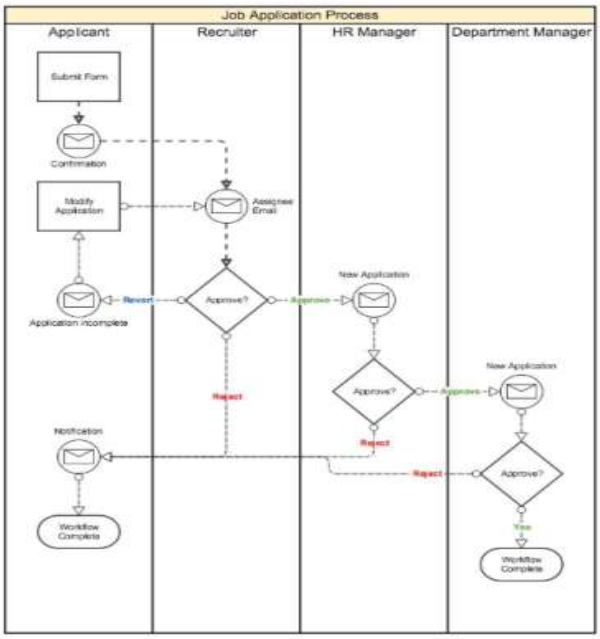

Fig.2: System Flow Diagram

The system operates through two major workflows:

Candidate Workflow
- Register and login
- Update profile (skills + resume)
- Search and browse jobs
- Apply for job
- ATS analysis performed:
  - Skill extraction
  - Matching
  - Score calculation
- View results and suggestions
- Request AI advice
- Track application status
- Build resume and download PDF
- Use chatbot for career guidance

HR Workflow
- Register and login
- Post job listings
- View applicants
- Analyze ATS scores
- Shortlist candidates
- Schedule interviews
- Update application status

System Processing
- Resume text extraction
- Skill normalization
- Matching algorithm execution
- AI API interaction

DOI: **10.30726/esij/v13.i2.2026.13200237**

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

3.1 UML Diagrams

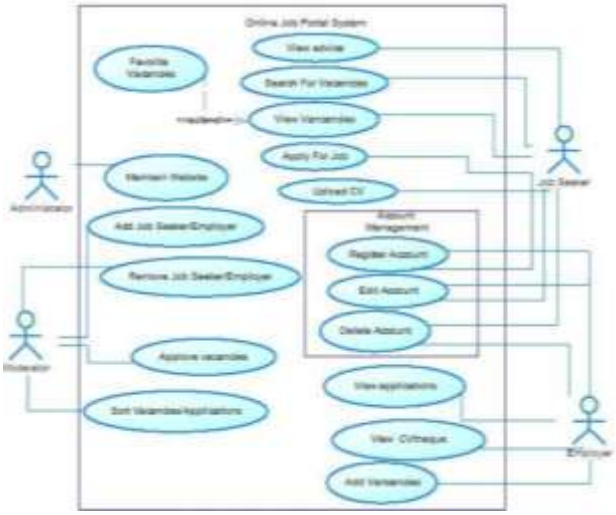

Fig.3: Use Case Diagram

Actors
- Candidate
- HR Professional

Candidate Use Cases
- Register/Login
- Update Profile
- Upload Resume
- Search Jobs
- Apply for Job
- View ATS Score
- Request AI Advice
- Use Chatbot
- Build Resume

HR Use Cases
- Register/Login
- Post Job
- View Applications
- Analyze ATS Score
- Shortlist Candidates
- Schedule Interview

Core Classes
- User (with role: Candidate / HR)
- CandidateProfile
- Job
- Application
- InterviewSchedule
- ResumeData

Key Relationships
- User ↔ CandidateProfile (One-to-One)
- User → Job (One-to-Many)
- Job ↔ Application (One-to-Many)
- Application → InterviewSchedule (One-to-One)
- User → ResumeData (One-to-Many)

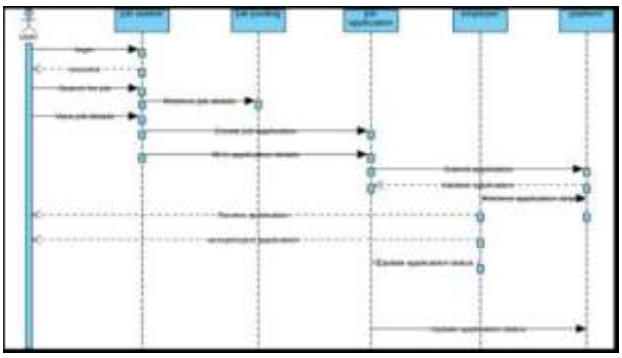

Fig.4: Sequence Diagram

Sequence Flow
- Candidate applies for job
- System retrieves profile and resume
- ATS analysis function executes
- Match score calculated
- Data stored in database
- Response returned to user
- AI advice generated (optional)

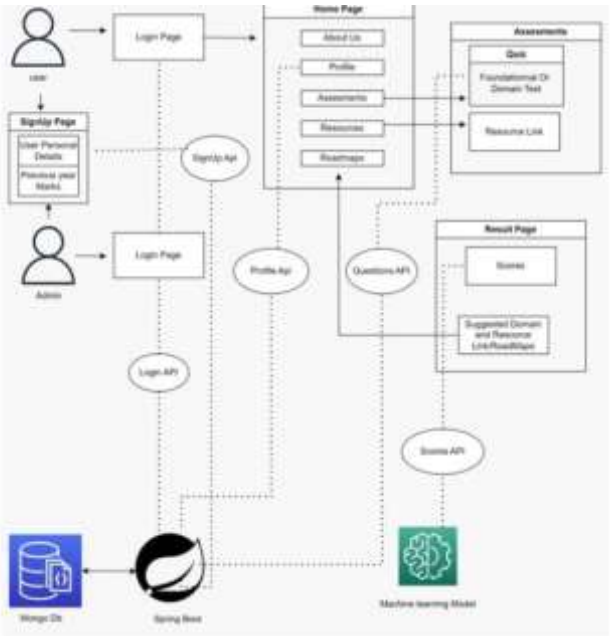

Fig.5: Activity Diagram

Activities Covered
- User registration
- Profile update
- Job application
- ATS matching

DOI: 10.30726/esij/v13.i2.2026.13200237

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

- AI advice generation
- Interview scheduling

4. Implementation

4.1 System Modules

The Smart Job Portal is implemented using a modular architecture within the Django framework. Each module is designed to perform a specific function, ensuring scalability, maintainability, and efficient workflow execution.

Authentication and User Management Module
This module manages:
- User registration and login
- Role-based access (Candidate / HR)
- Session handling and security

Django's built-in authentication system ensures secure password storage and access control.
Candidate Profile Module
This module enables candidates to:
- Update personal details
- Add skills and experience
- Upload resumes

The system stores structured candidate data for efficient matching.
Job Management Module
Handles:
- Job posting by HR users
- Job description and skill requirements
- Job listing display

Each job acts as an input for ATS matching.
Application Management Module
Responsible for:
- Job applications
- ATS score storage
- Application tracking

Candidates can view application status in real time.
ATS Matching Module
Core intelligent module that performs:
- Skill extraction from resume
- Matching with job requirements
- Score calculation

The ATS score is computed based on:
- Profile skills
- Resume content
- Job requirements

AI Resume Advice Module
This module integrates AI APIs from OpenAI to:
- Analyze resumes
- Suggest improvements
- Provide personalized recommendations

Chatbot Assistance Module
Provides:
- Career guidance
- Job-related queries
- Resume suggestions

This enhances user engagement and usability.
Interview Scheduling Module
Enables HR users to:
- Schedule interviews
- Update interview details
- Notify candidates

4.2 Detailed Implementation

4.2.1 ATS Score Calculation Logic

The ATS scoring mechanism is implemented using rule-based matching:
- Extract candidate skills
- Compare with job-required skills
- Compute match percentage

Formula:
ATS Score=Matched SkillsTotal Required Skills×100ATS\ Score = \frac{\text{Matched Skills}}{\text{Total Required Skills}} \times 100ATS Score=Total Required SkillsMatched Skills×100

Additional weightage is given to:
- Resume content relevance
- Experience level

4.2.2 Resume Parsing

The system extracts:
- Skills
- Keywords
- Experience

Text preprocessing includes:
- Lowercasing
- Stop-word removal
- Tokenization

4.2.3 AI Integration

AI APIs are used for:
- Resume feedback generation
- Chatbot responses

The system sends structured prompts and receives contextual outputs.

4.2.4 Database Implementation

The system uses SQLite with the following tables:
- User
- CandidateProfile
- Job
- Application

DOI: **10.30726/esij/v13.i2.2026.13200237**

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

- InterviewSchedule
- ResumeData

Relationships are managed using Django ORM.

4.2.5 Resume Builder Implementation

The system provides:

- Structured form input
- Resume template generation
- PDF export functionality

This helps candidates create professional resumes directly.

5. Results and Performance Analysis

5.1 Performance Analysis

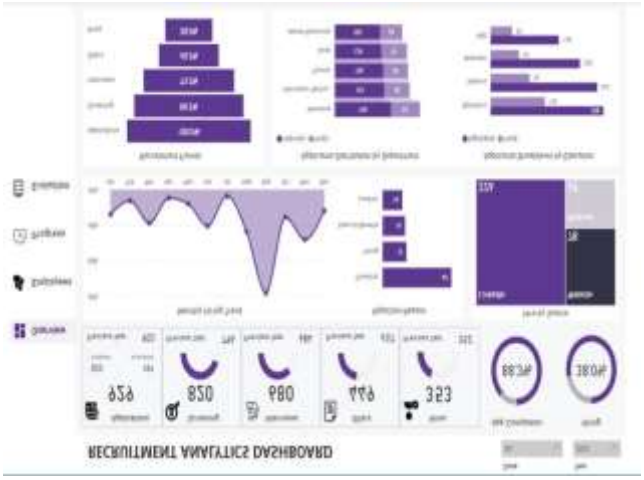

Fig.6: Performance Analysis 1

Fig.7: Performance Analysis 2

Fig.8: Performance Analysis 3

The system was evaluated using multiple candidate profiles and job postings to assess efficiency and accuracy.

Observations

- *Fast Processing*
  - ATS score generated instantly
- *Efficient Matching*
  - Accurate identification of relevant candidates
- *Scalable Design*
  - Supports multiple users simultaneously

5.2 Analytical Results

ATS Score Distribution

- High match candidates: 75–95%
- Medium match: 40–75%
- Low match: < 40%

Recruitment Efficiency

- Reduced screening time significantly
- Faster decision-making

User Experience

- Candidates receive real-time feedback
- Improved engagement through AI chatbot

Accuracy of Matching

- Effective for structured skill-based jobs
- Slight limitations for complex roles

5.3 Discussion

The Smart Job Portal system demonstrates the effectiveness of integrating AI and ATS-based matching into recruitment workflows.

Advantages

- Reduces manual effort
- Improves candidate-job matching
- Provides AI-driven insights
- Enhances user experience

DOI: **10.30726/esij/v13.i2.2026.13200237**

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

Limitations

- Rule-based ATS scoring
- Limited semantic understanding
- Dependency on resume quality

Despite these limitations, the system provides a strong foundation for intelligent recruitment automation.

6. Conclusion

The proposed Smart Job Portal with AI-Powered Skill Matching and Career Assistance successfully addresses the limitations of traditional recruitment systems by integrating intelligent automation, full-stack web technologies, and AI-driven features into a unified platform. The system effectively streamlines the recruitment lifecycle, from job posting and application submission to candidate evaluation and interview scheduling.

7. Future Enhancements

While the Smart Job Portal provides a robust framework for intelligent recruitment, several advanced enhancements can further improve its performance, scalability, and real-world applicability.

References

[1] Django Software Foundation, *Django Documentation*. [Online]. Available: https://docs.djangoproject.com/.
[2] OpenAI, *OpenAI API Documentation*. [Online]. Available: https://platform.openai.com/docs/.
[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186.
[4] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
[5] Scikit-learn Developers, *Scikit-learn Documentation*. [Online]. Available: https://scikit-learn.org/stable/documentation.html..
[6] LinkedIn Corporation, *LinkedIn Developer Documentation*. [Online]. Available: https://learn.microsoft.com/linkedin/.
[7] Indeed, *Indeed Hiring Lab: Labor Market Research and Hiring Trends*. [Online]. Available: https://www.hiringlab.org/.
[8] Info Edge (India) Ltd., *Naukri JobSpeak Report*. [Online]. Available: https://www.naukri.com/jobsspeak.
[9] Bootstrap Team, *Bootstrap Documentation*. [Online]. Available: https://getbootstrap.com/docs/.
[10] SQLite Consortium, *SQLite Documentation*. [Online]. Available: https://www.sqlite.org/docs.html.
[11] Python Software Foundation, *Python Documentation*. [Online]. Available: https://docs.python.org/3/.
[12] Google, *TensorFlow Documentation*. [Online]. Available: https://www.tensorflow.org/api_docs.

DOI: **10.30726/esij/v13.i2.2026.13200237**