IBM Project Report

On

Named Entity Recognition Using Transformers Architecture

Developed By: -

Raj Kariya (21162101011)

Dushyant Vyas (22162172006)

Jhanvi Patel (22162122008)

Guided By:-

Prof. Pritesh Andharia(Internal)

Mr Nirav Rajgor (External)

Submitted to

Faculty of Engineering and Technology

Institute of Computer Technology

Ganpat University

Year - 2025

CERTIFICATE

This is to certify that the **IBM** Project work entitled **"Named Entity Recognition using Transformers Architecture"** by Dushyant Vyas (Enrollment No: 22162172006) of Ganpat University, towards the partial fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering. The findings contained in this Project have not been submitted in part or full to any other University for the award of any other Degree/Diploma.

Name & Signature of Guide

Name & Signature of Head of Department

Place: ICT - GUNI

Date: 22nd Feb, 2025

ACKNOWLEDGEMENT

The IBM/Industry Internship project has been an invaluable opportunity for both learning and personal growth. I feel truly privileged to have received support from numerous incredible individuals throughout the journey of completing this work. I would like to begin by expressing my sincere gratitude to Dr. Rohit Patel, Principal of ICT and Prof. Dharmesh Darji, Head of ICT, for granting us the chance to engage in this project. I am deeply thankful to Prof. Pritesh Andharia and Mr. Nirav Rajgor, our internal and external guides, for their mentorship on the project titled Named Entity Recognition using Transformer Architecture. Despite their demanding academic commitments, they generously devoted their time to guide us, offer feedback, and steer us in the right direction. Our heartfelt appreciation also goes to the CSE department for consistently monitoring our progress and ensuring all necessary facilities were in place to support our efforts. We take this opportunity to acknowledge their valuable contributions with gratitude.

RAJ KARIYA (Enrollment No:21162101011)

DUSHYANT VYAS (Enrollment No:22162172006)

JHANVI PATEL (Enrollment No:22162122008)

ABSTRACT

The project "Resume NER" is a web-based platform that aims to streamline the recruitment process using AI and machine learning technologies. It allows HR professionals to create job postings dynamically and share them publicly for applicants to apply. The system uses a transformer-based Named Entity Recognition (NER) model trained on a resume dataset to extract important details like skills, qualifications, and experience. After applying, candidates' resumes are analyzed, and a matching score is provided based on the job description. The platform is built with Django, HTML, CSS, JQuery, and Python, while Firebase handles authentication and MongoDB is used for data storage with optimized indexing. Resumes are stored in Firebase Storage. The website also features an AI Assistant powered by the Gemini Pro model to guide users. The complete system is hosted on an AWS EC2 instance with domain management through Route53 and SSL security using Certbot, ensuring a secure and reliable experience for HR departments.

INDEX

1.	**INTRODUCTION**		**1**
	1.1	Overview of the Project	2
	1.2	Problem Statement	2
	1.3	Objectives	2
	1.4	Scope of the Project	3
2.	**BACKGROUND & LITERATURE REVIEW**		**4**
	2.1	What is NER	5
	2.2	Importance of NER in Resume Evaluation	5
	2.3	Evolution of NER Models	6
	2.4	Overview of Transformer Models	6
	2.5	Existing Solutions & Their Limitations	8
3.	**PROJECT PLAN**		**9**
	3.1	Research & Initial Development (Weeks 1-6)	10
	3.2	Backend & Model Integration	10
	3.3	System Optimization & Testing	10
	3.4	Deployment & Final Enhancements	10
4.	**SYSTEM DESIGN**		**12**
	4.1	Architecture Diagram	13
	4.2	Data Flow Diagarm	14
	4.3	Module Overview	16

5.	**SYSTEM REQUIREMENTS**		**18**
	5.1	Hardware Requirements	19
	5.2	Software Requirements	19
	5.3	Tools & Libraries Used	20
6.	**IMPLEMENTATION**		**21**
	6.1	Development Environment	22
	6.2	Step by Step Implementation	22
	6.3	Deployment Environment	22
	6.4	Deployment Steps	23
	6.5	Post-Deployment Monitoring	23
7.	**CONCLUSION**		**24**
8.	**REFERENCES**		**26**

CHAPTER: 1
INTRODUCTION

1.1 Overview of the Project

"Resume NER" is a platform designed to simplify the hiring process using AI and ML technologies. It focuses on automating resume screening by applying a transformer-based Named Entity Recognition (NER) model. The system extracts important candidate information like skills, experience, and education from resumes, helping HR departments identify suitable candidates more efficiently. The project combines Django for the frontend, Python for the backend and model development, and MongoDB and Firebase for secure and structured data handling. It also includes an AI Assistant powered by the Gemini Pro model to enhance user interaction.

1.2 Problem Statement

Manually screening large numbers of resumes is time-consuming and often prone to human error. HR teams face challenges in quickly identifying the best candidates due to the unstructured and varying formats of resumes. Existing solutions may not accurately extract all necessary information or adapt well to specific job requirements. There is a need for a system that automates this process, improves accuracy, and saves time by using advanced NLP models like transformers.

1.3 Objectives

The primary objectives of this project are:

1. To build a transformer-based NER model fine-tuned for extracting key information from resumes.
2. To create a user-friendly portal for HR professionals to manage job postings and applications.
3. To automatically score candidates based on their resumes and job descriptions for easier shortlisting.
4. To provide secure authentication and fast database operations using Firebase and MongoDB.
5. To ensure smooth hosting and availability using AWS cloud services.

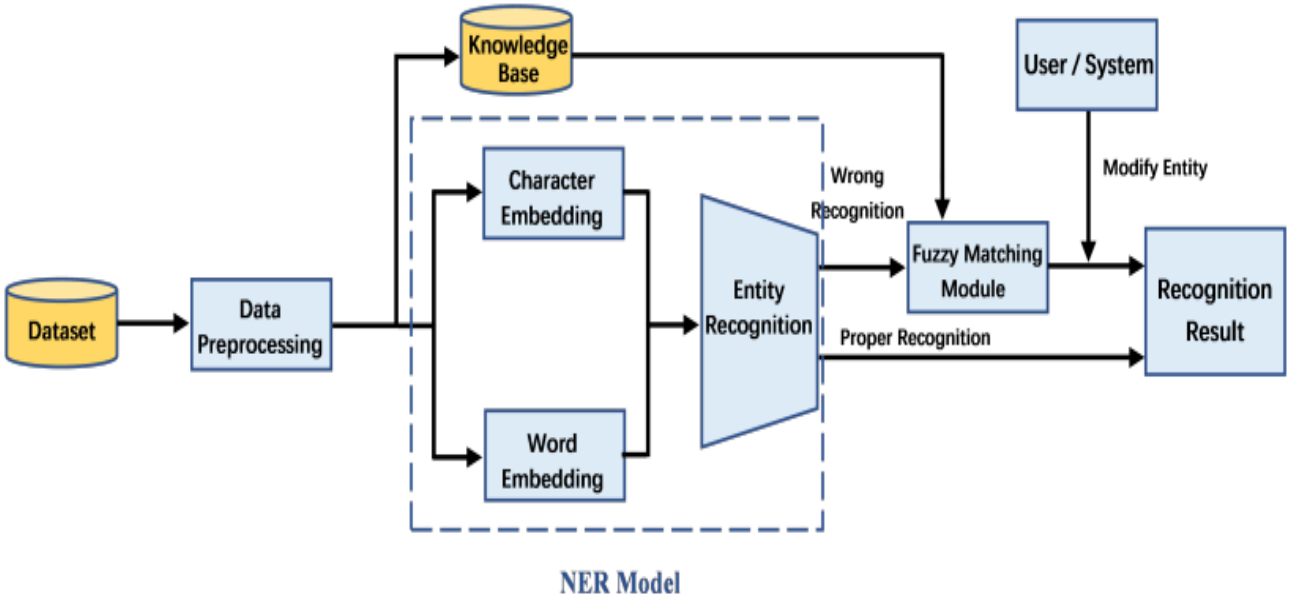
1.4 Scope of the Project

The project mainly targets HR departments that want to automate resume screening. It allows HR to create and manage jobs, collect applications through a public link, and view the most relevant candidates scored by the ML model. It covers resume uploads, AI-based extraction, candidate management, and dashboard-based analysis. The platform is hosted on AWS, secured with SSL, and optimized for fast performance. Future improvements could extend language support and model capabilities, but the current focus is on English-language resumes.

CHAPTER: 2
BACKGROUND & LITERATURE REVIEW

2.1 What is Named Entity Recognition (NER)?

Named Entity Recognition (NER) is a subfield of Natural Language Processing (NLP) that focuses on identifying and classifying key information within unstructured text. It tags entities like names of people, organizations, locations, dates, and other specific terms into predefined categories. NER plays an important role in extracting structured information from raw text, making it easier to process and analyze for different applications.



2.2 Importance of NER in Resume Evaluation

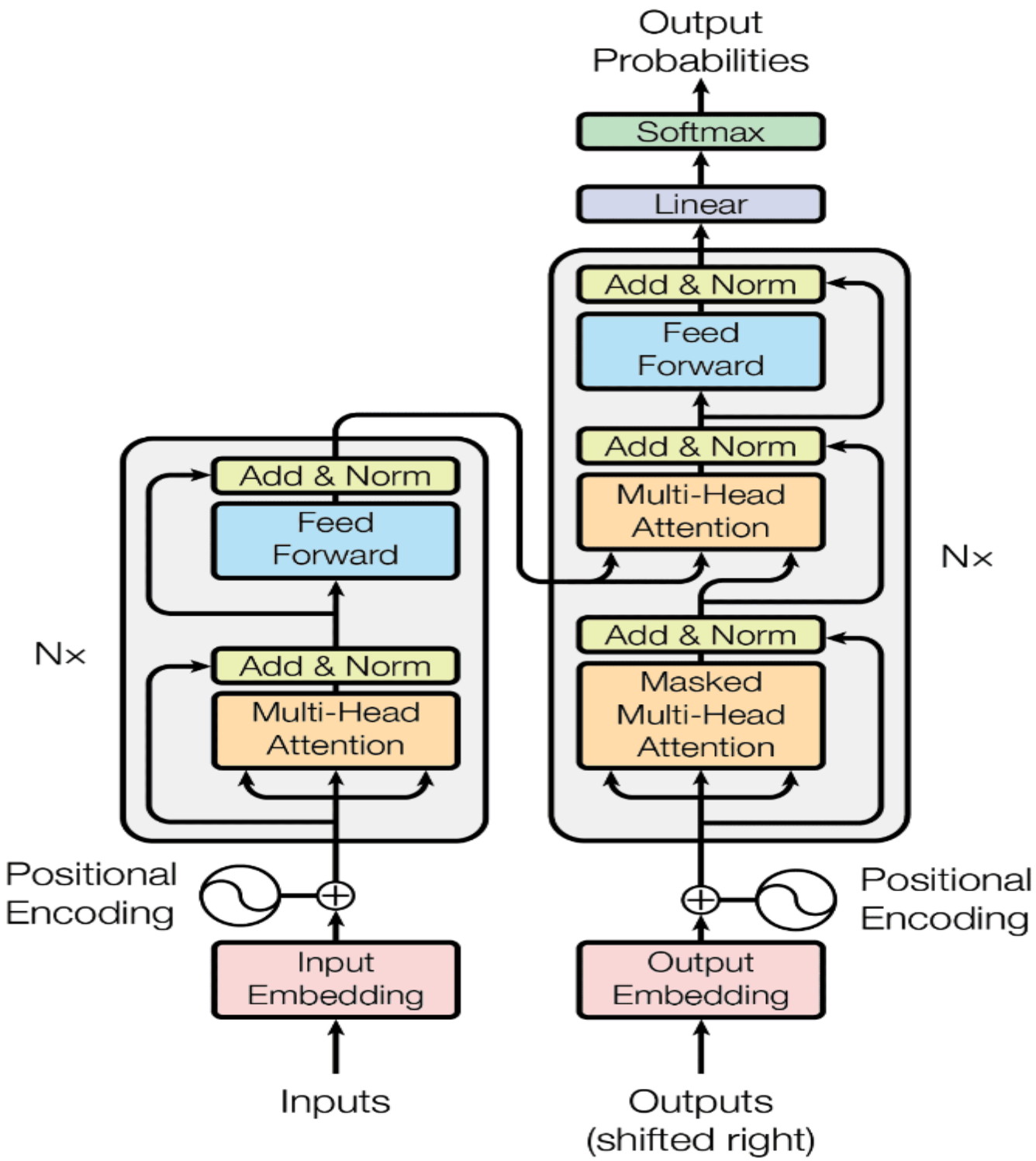
In resume evaluation, NER helps by automatically pulling out crucial details such as names, contact information, educational background, skills, and work experience. This automation reduces the manual effort required by HR departments, ensures consistent data extraction, and improves the efficiency and objectivity of candidate screening. By making information retrieval faster and more accurate, NER greatly enhances the recruitment process.

2.3 Evolution of NER Models

NER techniques have evolved significantly over time. Initially, rule-based systems that relied on predefined patterns were used, but they lacked scalability. Later, machine learning approaches like Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs) allowed models to learn from labeled datasets, improving flexibility. The introduction of deep learning, especially models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs), further advanced NER by capturing better context. More recently, transformer-based models have pushed NER to new levels by handling context more effectively through self-attention mechanisms.

2.4 Overview of Transformer Models (e.g., BERT, RoBERTa)

Transformer models, such as BERT and RoBERTa, are designed to process sequential data efficiently using self-attention mechanisms. They understand the relationship between words in a sentence in both directions, providing a deeper context compared to traditional models. Transformers are first pretrained on large amounts of text and then fine-tuned for specific tasks like NER, allowing them to achieve state-of-the-art performance in many NLP applications, including resume parsing.


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

2.5 Existing Solutions and Their Limitations

Several existing NER tools like spaCy, Stanford NER, and Hugging Face Transformers offer powerful entity extraction capabilities. However, these tools often face limitations when applied to specific domains like resume evaluation. Pretrained models may not perform accurately without additional fine-tuning, and some solutions struggle to scale efficiently for large datasets. Customization options are also limited, making integration with different HR workflows difficult. Furthermore, most tools primarily support English, reducing their effectiveness for multilingual resumes.

CHAPTER: 3
PROJECT PLAN

3.1 Research & Initial Development (Weeks 1-6) – COMPLETED

- Conducted detailed research on Named Entity Recognition (NER) techniques and transformer-based architectures.
- Started frontend development using Django for building a user-friendly web interface.
- Set up MongoDB as the database and drafted a basic API structure to manage resume data.

3.2 Backend & Model Training (Weeks 7-12)

- Fine-tuned transformer models like BERT/RoBERTa specifically for resume parsing tasks.
- Developed Django REST Framework (DRF) APIs and connected them with MongoDB for structured data storage.
- Enhanced the frontend dashboard to display extracted resume information dynamically.
- Milestone: Achieve a fully working NER model integrated with backend APIs and UI.

3.3 Model Integration and Optimize backend.(Weeks 13-16)

- Optimized backend APIs and database queries for faster performance.
- Implemented asynchronous processing to handle multiple resume uploads efficiently.
- Conducted unit and integration testing across all modules.
- Organized beta testing with HR users and fixed bugs based on the feedback received.

3.4 Deployment & Final Enhancements (Weeks 17-18)

- Deployed the website and model on AWS EC2 instance, using MongoDB Atlas for cloud database services.
- Implemented SSL certification for secure communication.
- Finalized UI/UX improvements and set up monitoring tools for post-deployment system tracking.
- Milestone: Live system deployment with a fully functional Resume NER platform.

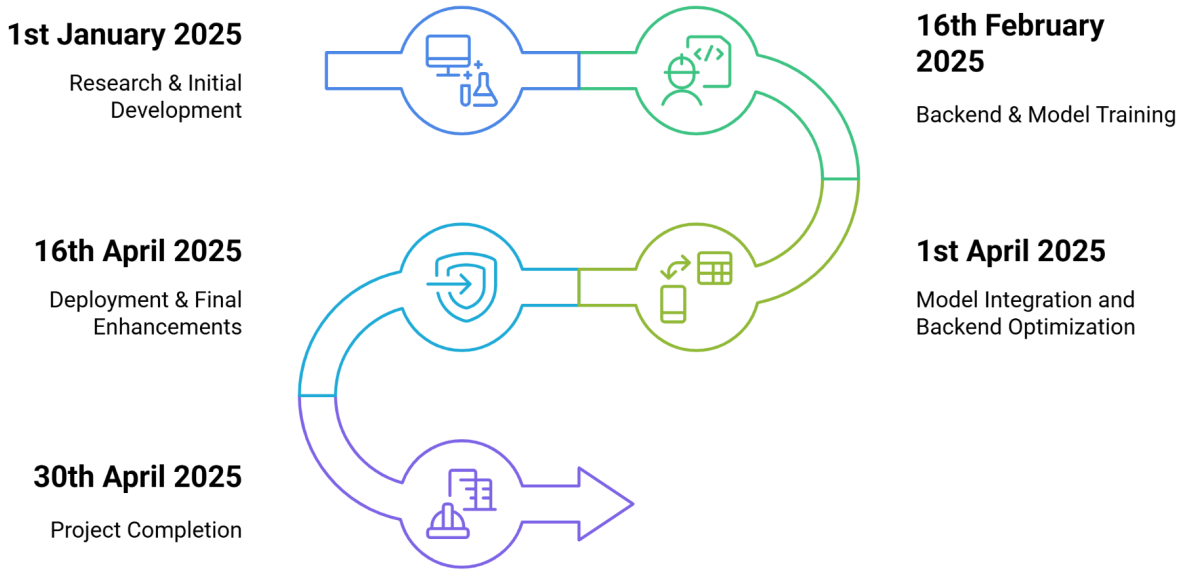


Figure: Timeline Chart

CHAPTER: 4
SYSTEM DESIGN

4.1 Architecture Diagram

The architecture of the Resume NER system is composed of four primary components:

1. **Frontend:**

   Developed using Django, HTML, CSS, and JQuery. It provides a user-friendly platform for HR users to create jobs, manage candidates, and view extracted information from resumes.

2. **Backend:**

   Built using Python and Django REST Framework (DRF), it integrates the NER model, handles the business logic, manages API endpoints, and communicates between the frontend and database.

3. **Database:**

   MongoDB is used as the primary database for storing structured candidate information such as names, skills, education, and work experience. Firebase Storage is used specifically for storing uploaded resumes.

4. **NER Model:**

   A transformer-based model (fine-tuned BERT) that extracts relevant entities from resumes and evaluates candidate-job matching scores.

The entire system is deployed on an AWS EC2 instance, secured with SSL certification, and uses MongoDB Atlas for cloud database services.

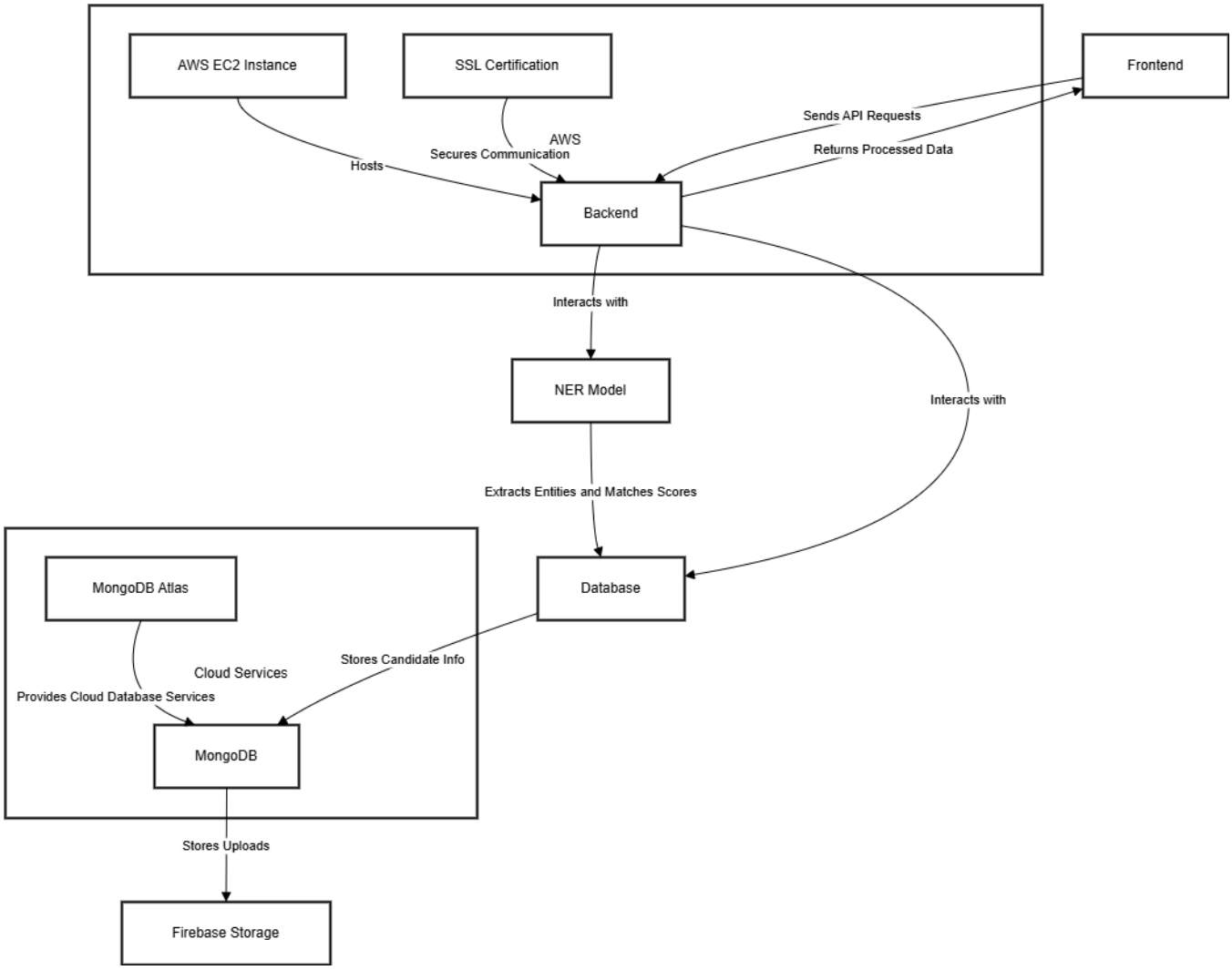


Figure: Architecture Diagram

4.2 Data Flow Diagram

The data flow for the project is structured as follows:

1. **Job Creation:**
    - HR users create job posts through the Django-based frontend.
    - Each job generates a dynamic application link.
2. **Application Submission:**
    - Candidates access the link, fill out the application form, and upload their resumes.
3. **Resume Processing:**
    - Uploaded resumes are sent to the backend server where the NER model processes them.
    - Entities like names, contact details, skills, and qualifications are extracted.

4. **Data Storage:**
    - Extracted information is stored in MongoDB collections.
    - Uploaded resume files are stored in Firebase Storage.
5. **Candidate Management:**
    - HR users can log in to view, sort, and shortlist candidates based on the matching score generated by the model.

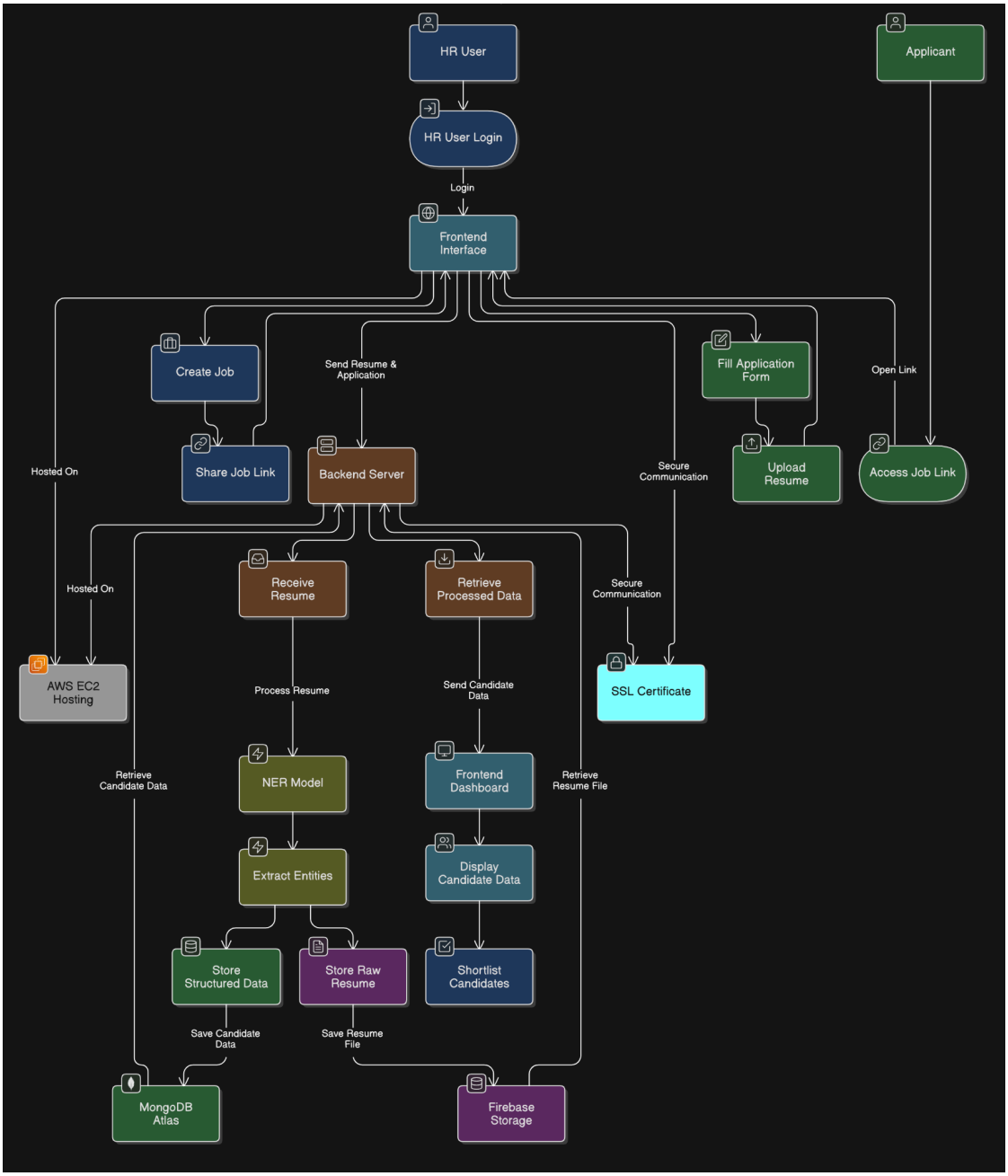


Figure: Data Flow Diagram

4.3 Module Overview

4.3.1 Frontend (Django)

1. **User Interface:**
    - Provides pages for dashboard, job creation, candidate management, and company details.
    - Features an AI Assistant on the landing page using Gemini Pro for user interaction.
2. **Authentication:**
    - Handled by Firebase Authentication with session management linked to Django.

4.3.2 Backend (Python and NER Model)

1. **API Services:**
    - Django REST Framework APIs manage resume uploads, job creation, candidate retrieval, and scoring.
2. **Model Integration:**
    - Fine-tuned BERT model processes resumes to extract entities and calculate matching scores.
3. **Security:**
    - Django Session Authentication for user validation.
    - APIs are secured based on logged-in user sessions.

4.3.3 Database (MongoDB and Firebase)

1. **MongoDB:**
    - Stores extracted candidate data with indexed collections for faster query responses.
2. **Firebase Storage:**
    - Handles the storage of uploaded resume files securely.

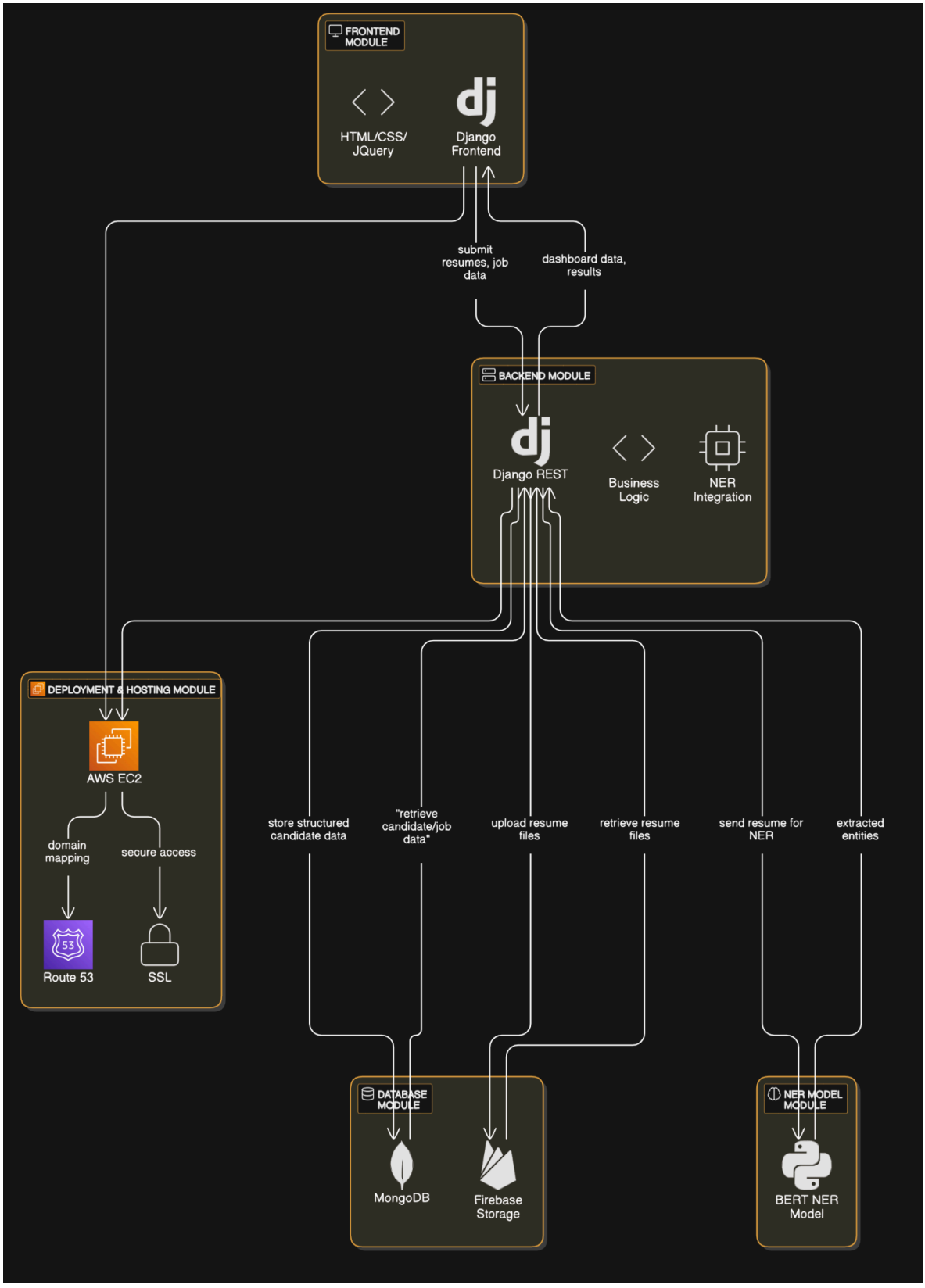


Figure: Module Overview

CHAPTER: 5
SYSTEM REQUIREMENTS

5.1 Hardware Requirements

Component	Specification
Processor	Minimum 2 GHz Dual-Core (Recommended: 4-Core or higher)
RAM	Minimum 8 GB (Recommended: 16 GB)
Storage	Minimum 50 GB available
Cloud Server	AWS EC2 (Instance Type: t3.large, 8 GiB RAM, 50 GB Boot Volume)

5.2 Software Requirements

Software Component	Specification
Operating System (Local)	Windows 10/11, MacOS, Linux (All Distros)
Operating System (Server)	Amazon Linux
Web Server	Nginx
Programming Language	Python 3.9+
Web Framework	Django 4.0

Database	MongoDB Atlas (cloud) + Firebase Storage
Authentication Service	Firebase Authentication
Hosting Platform	AWS EC2
DNS Management	AWS Route 53

5.3 Tools and Libraries Used

Category	**Tools & Libraries**
Frontend	HTML5, CSS3, JQuery, Django Templates
Backend	Django REST Framework (DRF) , FastAPI , Uvicorn
Model Development	Transformers (Hugging Face), Sentence Transformers, NLTK, Wordcloud
Deployment & Security	Nginx, Certbot (SSL Certification), AWS EC2, AWS Route 53
Others	Git & GitHub (Version Control), MongoDB Indexing, Firebase Authentication

CHAPTER: 6
IMPLEMENTATION

6.1 Development Environment

- **Local Machine OS:** Windows 10, macOS
- **Server OS:** Amazon Linux (AWS EC2 Instance)
- **Programming Language:** Python 3.9+
- **Frameworks:** Django 4.x, Django REST Framework
- **Database:** MongoDB (local for development, MongoDB Atlas for production)
- **Version Control:** Git and GitHub
- **Frontend:** HTML5, CSS3, JQuery, Django Templates

6.2 Step-by-Step Implementation

1. **Frontend Development:**
    - Created Django templates for pages like Landing Page, Sign-Up/Sign-In, Dashboard, Create Job, Manage Candidates, and Company Details.
    - Integrated an AI Assistant on the landing page using Gemini Pro API.
2. **Backend Development:**
    - Set up Django REST Framework APIs to handle operations like job creation, candidate management, and resume uploads.
    - Integrated Firebase Authentication for user login and session management.
3. **Model Training and Integration:**
    - Fine-tuned a BERT-based transformer model on the resume dataset from Kaggle.
    - Developed APIs to send uploaded resumes to the model for entity extraction and matching.
4. **Database Integration:**
    - Linked Django APIs with MongoDB for storing structured resume data.
    - Used Firebase Storage for storing uploaded resume files.
5. **Testing and Debugging:**
    - Test the system using various resume formats.
    - Debug and optimize the code for performance and accuracy.

6.3 Deployment Environment

- **Cloud Platform:** AWS EC2 (t3.large instance with 8 GiB RAM, 50GB storage)
- **Domain Management:** AWS Route 53
- **Web Server:** Nginx
- **Database:** MongoDB Atlas (Cloud-hosted)
- **SSL Certification:** Using Certbot for HTTPS security
- **Hosting Subdomain:** resumener.rajkariya.study

6.4 Deployment Steps

1. Launched an AWS EC2 instance with Amazon Linux OS.
2. Installed required software: Python, Nginx, MongoDB drivers, Git, and Certbot.
3. Cloned the project from GitHub repository onto EC2.
4. Set up a Gunicorn server for running Django application.
5. Configured Nginx as a reverse proxy to serve the Django app securely.
6. Configured Route 53 to link the subdomain to EC2's public IP.
7. Generated and installed SSL certificates using Certbot for secure HTTPS communication.

6.5 Post-Deployment Monitoring

- **Monitoring Tools:** AWS CloudWatch for instance monitoring.
- **Database Monitoring:** MongoDB Atlas dashboard for real-time query and storage analysis.
- **Application Monitoring:** Django server logs and Nginx logs.
- **Security Measures:** Enabled firewall rules (Security Groups) on AWS, SSL encryption enabled.

CHAPTER: 7
CONCLUSION

The "Resume NER" project successfully demonstrates the integration of advanced natural language processing techniques with practical HR needs. By leveraging transformer-based models, particularly fine-tuned BERT, the system accurately extracts essential information from resumes and automates the candidate shortlisting process. The platform offers HR departments a powerful, user-friendly web application built using Django, with secure backend support and scalable database management through MongoDB.

Through cloud deployment on AWS and the addition of features like asynchronous processing, Django session authentication, and SSL security, the platform ensures reliability, speed, and data protection. The dynamic job creation and resume evaluation workflow significantly reduce manual effort and enhance decision-making for recruiters. Overall, this project lays a strong foundation for future extensions, such as multilingual resume processing and deeper AI-driven candidate analysis, making the recruitment process more intelligent and efficient.

CHAPTER: 8
REFERENCES

1. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

2. Datta, S. (2022, January 31). How to fine-tune BERT for NER using HuggingFace. *freeCodeCamp*.

3. Hugging Face. (n.d.). Transformers documentation: Token classification. Retrieved May 2, 2025,

4. Hugging Face. (n.d.). BERT model documentation. Retrieved May 2, 2025,

5. IBM. (n.d.). Named entity recognition with IBM Watson natural language processing.

6. MongoDB, Inc. (n.d.). MongoDB documentation.

7. Django Software Foundation. (n.d.). Django documentation.

8. Amazon Web Services. (n.d.). Amazon EC2 documentation.

9. Google. (n.d.). Firebase authentication documentation.

10. Singh, S. (n.d.). Resume-NER: Applying BERT for named entity recognition on resumes. GitHub.