A Major Project Report

On

PLACEMENT DATA FOR JOB CLASSIFICATION

Submitted to CMREC (UGC Autonomous)

In Partial Fulfilment of the requirements for the Award of Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted

By

G.RISHWANTH	**(228R1A66E8)**
G.RITHWIK	**(228R1A66F2)**
P.HARSHINI	**(228R1A66H4)**
V.HARISH KUMAR	**(228R1A66J9)**

Under the Esteemed guidance of

Mrs.M. SOUJANYA

Assistant Professor, Department of CSE(AI&ML)

Department of Computer Science and Engineering(AI&ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE, New Delhi, Affiliated to JNTU, Hyderabad)
(Kandlakoya, Medchal Road, Medchal-Malkajgiri Dist., Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NAAC&NBA, Approved by AICTE New Delhi, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering (AI & ML)

CERTIFICATE

This is to certify that the Major project entitled **" PLACEMENT DATA FOR JOB CLASSIFICATION"** is a bonafide work carried out by

G.RISHWANTH	**(228R1A66E8)**
G.RITHWIK	**(228R1A66F2)**
P.HARSHINI	**(228R1A66H4)**
V.HARISH KUMAR	**(228R1A66J9)**

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI&ML) from CMR Engineering College, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Mrs. M. Soujanya	Mr. G. Venkateswarlu	Dr. Madhavi Pingili
Assistant Professor	Assistant Professor	Professor & HOD
Department of CSE (AI & ML)	Department of CSE (AI & ML)	Department of CSE (AI & ML)

External Examiner: _____

DECLARATION

This is to certify that the work reported in the present Major project entitled "**PLACEMENT DATA FOR JOB CLASSIFICATION** " is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI & ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our Major project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

G.RISHWANTH (228R1A66E8)
G.RITHWIK (228R1A66F2)
P.HARSHINI (228R1A66H4)
V.HARISH KUMAR (228R1A66J9)

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, Professor & HOD, Department of CSE(AI&ML), CMR Engineering College for their constant support.

We are extremely thankful to **Mrs. M. Soujanya,** Assistant Professor, Internal Guide, Department of CSE(AI&ML), for her constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Major project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the Major project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

G.RISHWANTH (228R1A66E8)
G.RITHWIK (228R1A66F2)
P.HARSHINI (228R1A66H4)
V.HARISH KUMAR (228R1A66J9)

CONTENTS

ABSTRACT

The Placement Portal is a web application desgined to streamline and enhance the placement process for students , alumni and Training and Placement officers (TPOs). It serves as a centralized platform, automating placement-related activities and offering final-year students access to essential information about opportunities, recruitment schedules, and preparatory resources. TPOs can efficiently manage student records, coordinate with recruiters, and analyze placement trends. The portal maintains comprehensive records of placement statuses, higher education pursuits, and alumni updates, enabling institutions to track success and build long-term connections.Built with user-friendly features akin to popular social media platforms, the portal fosters seamless interaction among students, TPOs, and recruiters. Its digitized workflow minimizes manual effort, reduces errors, and supports real-time decision-making. With a focus on transparency, scalability, and data security, the system ensures a robust and adaptable solution for institutions of all sizes. By modernizing campus recruitment processes, the Placement Portal bridges the gap between academia and industry, fostering efficient, transparent, and impactful career development.

Keywords: Placement Portal, automation, recruitment, Training and Placement Officers, students, alumni, career development, data security, scalability, digital workflow, user-friendly, campus recruitment.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction

This project focuses on developing a comprehensive web-based application for the Training and Placement (T&P) Department to streamline and enhance placement-related activities within colleges. As placement processes grow more complex and student numbers increase, traditional manual methods of managing profiles, records, and recruitment details become inefficient, time- consuming, and prone to errors. To overcome these challenges, the proposed system provides a centralized digital platform that ensures secure and seamless access for all authorized users through unique login credentials.

The system also benefits students by providing timely access to job oppurtunities, company details, eligibility criteria , and recruitment schedules ,enabling better preparation and decision- making. For administrators, it offers tools to manage recruitment drives,communicate with companies, and generate placement reports. By digitizing the entire workflow,the application reduces manual effort, improves transparency , and enhances coordination between students and the T&P department.Overall, this web application modernizes the placement process, making it more efficient, reliable, and scalable. By bridging the gap between academia and industry, the system contributes to improved career development for students and strengthens the institution's reputation for successful placements.

Furthermore, the platform incorporates advanced data analytics to track student progress and identify skill gaps, allowing for more targeted training sessions.Integrated communication modules ensure that urgent notifications regarding interview shifts or venue changes reach students instantly via automated alerts.The system also maintains an exhaustive historical database of previous recruitment cycles, providing valuable insights for predicting future hiring trends. To ensure data integrity, robust encryption protocols are implemented to protect sensitive student documents and personal information from unauthorized access. The application's responsive design ensures that both recruiters and students can manage their profiles effectively across various mobile and desktop devices.

1.2 Project Objectives

Bythe completion of this project, the system demonstrates the following capabilities:

1.Streamline the Placement Process – Automate and simplify the placement activities for students,

The capability focuses on replacing cumbersome,manual workflows with a cohensive automated system.automated system. By digitizing registration, document verification, and interview scheduling, the platform minimizes human error and administrative bottlenecks. It ensures that both students and TPOs can focus on quality preparation rather than paperwork. This transition to a digital environment results in a faster, more organized recruitment cycle that can handle large volumes of data with ease.

2. Centralized Access to Information – Provide final-year students with easy access to placement opportunities, recruitment schedules, and preparatory resources. The system acts as a "single source of truth" where final-year students can find everything they need for their career journey. Instead of tracking multiple emails or notice boards, students can view real-time job openings, detailed company profiles, and specific eligibility criteria in one dashboard. Additionally, the inclusion of preparatory resources and recruitment schedules helps students stay ahead of deadlines.

3. Efficient Record Management – Enable TPOs to manage student data, coordinate with recruiters, and track placement trends effectively. For Training and Placement Officers, the system provides a powerful set of tools to organize and filter vast amounts of student data. TPOs can instantly generate shortlists based on specific recruiter requirements, such as GPA or technical skills, saving hours of manual sorting. The platform also tracks placement trends over time, allowing the department to visualize success rates through automated reporting.

4. Track Alumni & Higher Education Progress – Maintain comprehensive records to monitor student outcomes, alumni updates, and higher education pursuits. This project focuses on developing a comprehensive web-based application for the Training and Placement (T&P) Department to streamline and enhance placement-related activities within colleges. As placement processes grow more complex and student numbers increase, traditional manual methods of managing profiles, records, and recruitment details become inefficient, timeconsuming, and prone to errors.

5. Enhance Interaction – Foster seamless communication among students, TPOs, and recruiters through user-friendly, social media-like features.The application bridges the communication gap by incorporating intuitive, social media-like features that encourage engagement. Students can interact with TPOs for guidance, while recruiters can provide direct updates or feedback through integrated messaging modules. This modern approach to communication ensures that urgent notifications are never missed and fosters a supportive community environment.

1.3 Purpose of the Project

The proposed system streamlines placement activities through a centralized platform for students, alumni, and TPOs. Students can manage profiles and track recruitment easily, while TPOs gain better visibility for planning. The platform strengthens alumni networking and mentoring, and supports higher-education aspirants with tailored program and scholarship information. Overall, it improves placement efficiency, enhances alumni engagement, and supports academic and professional growth.

1.4 Problem Statement

The Placement activities in many educational institutions are still managed manually, leading to errors, delays, and difficulty in updating or tracking student and recruitment information. Students lack timely access to opportunities, while Training and Placement Officers face challenges in handling large volumes of data, coordinating drives, and maintaining accurate records. There is a need for a centralized and automated system that simplifies data management, improves communication, and streamlines the entire placement process.

1.5 Existing System

The existing placement system is mainly based on manual processes, which leads to several inefficiencies. A large amount of work is done manually, increasing the chances of errors and inconsistencies in data handling. There is a lack of proper standardization, making it difficult to maintain uniform records across the system. The process is also time-consuming, as data collection, verification, and management require significant effort. Additionally, insufficient data analysis makes it hard to extract useful insights from placement data.

Furthermore, the system lacks a proper hierarchical structure, leading to poor organization of information. Updating records is difficult and often results in outdated or incomplete data. Duplication of files is another major issue, causing redundancy and confusion. The system also has limited scalability and cannot efficiently handle large volumes of data. Overall, these limitations highlight the need for an automated and efficient placement management system**.**

Disadvantages

- Most placement activities are handled manually, making the process time-consuming and inefficient.
- Maintaining records in multiple formats (spreadsheets, documents) leads to errors and repeated data..
- Students and administrators do not get timely updates on placement activities and opportunities.
- It is hard for TPOs to filter eligible candidates and generate accurate reports due to unorganized data.
- The system cannot efficiently handle large volumes of student and recruitment data as the number of users increases.

1.6 Proposed System

A placement management system is a software solution designed to help the students. The purpose of the proposed system is to provide Information about Companies, placements and Higher Education. Placement coordinators can edit/modify and students can access the data in the system quickly. The system provides an efficient way to maintain student records and the company details in the database, ensuring the correctness as well as the integrity of the data. The system also reduces the manual record maintenance time and provides an effective This system consists of different modules to interact with initiating with logging into system redirects to the home page. It also showcases the objectives of the placement portals, companies that are available for recruitment process.

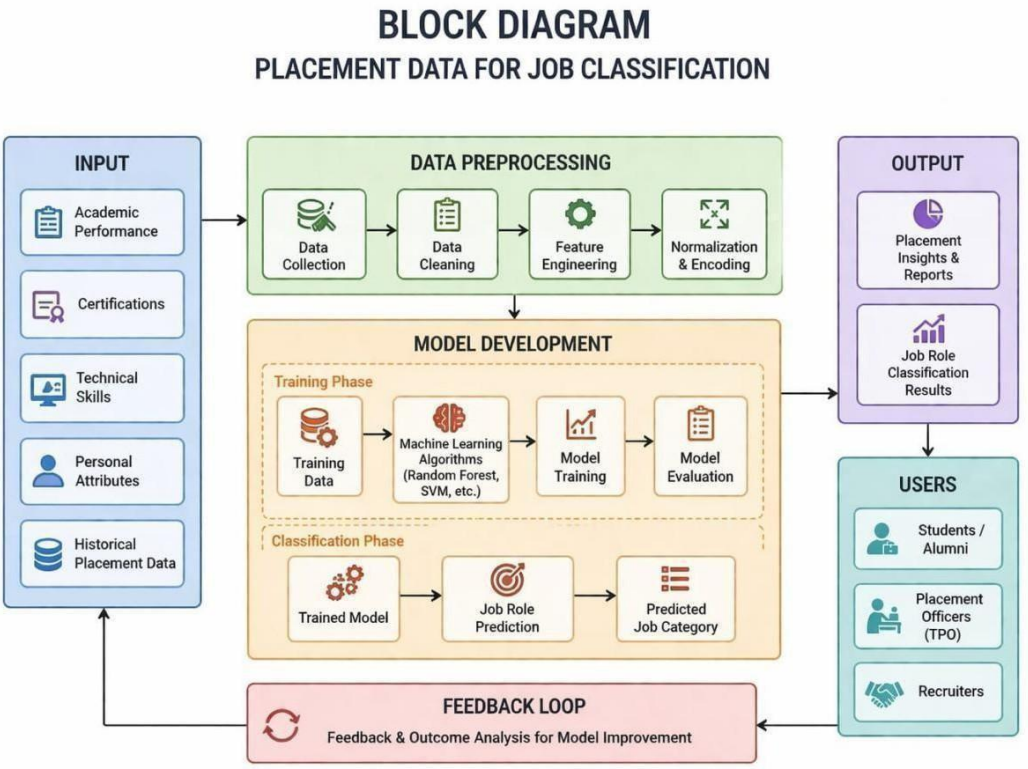


Fig 1.6.1: Block diagram of proposed system.

Advantages

- The system generates detailed reports and insights on student performance, placement trends, and recruitment outcomes, helping in better decision-making.
- Reduces administrative burden by automating repetitive tasks like eligibility, filters document verification, and notification broadcasting.
- Students receive timely updates about job opportunities, eligibility criteria, and recruitment schedules, ensuring they do not miss any important information.
- Ensures data security through role-based access controls and encrypted storage, protecting sensitive student and company information.
- All student, company, and placement-related data are stored in a single platform, making it easy to access, manage, and maintain consistency.
- The system maintains detailed student profiles including academic records, skills, and achievements, enabling accurate job matching and classification.
- Provides a scalable infrastructure that easily accommodates increasing numbers of students and corporate partners as the institution grows.

1.7 Input and Output Design

1.7.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple.

creation of a user-centric interface that minimizes cognitive load while maximizing data accuracy. By implementing intuitive forms with predefined dropdowns, checkboxes, and radio buttons, the system limits the necessity for free-text entry, which is often the primary source of data inconsistency. This structured approach ensures that student profiles and company requirements are captured in a standardized format, making subsequent processing and filtering significantly more efficient. Furthermore, the design incorporates real-time field validation to catch errors at the point of entry, providing immediate feedback to the user and preventing the submission of incomplete or incorrectly formatted information.

To further streamline the user experience, the input architecture prioritizes logical sequencing and grouped data fields, ensuring that the flow of information follows a natural progression. This organization reduces the time required for students to upload their resumes and for administrators to post new job descriptions, effectively eliminating unnecessary delays in the placement timeline. By integrating automated data extraction features—such as parsing basic details from uploaded documents—the system minimizes the manual effort required from the user, thereby reducing fatigue and the likelihood of clerical mistakes. Security remains a cornerstone of the input design, where every entry point is shielded by rigorous sanitization protocols to prevent common vulnerabilities like SQL injection or cross-site scripting. Access to specific input modules is strictly governed by user roles, ensuring that a student cannot accidentally or intentionally modify administrative records or recruiter data. This balance of accessibility and restriction ensures that while the system remains easy to navigate for the average user, the underlying data remains uncompromised and private, maintaining the overall credibility of the Training and Placement ecosystem.

The input is desgined in such a way so that it provides security and ease of use with retaining the privacy. Input design consider the following things.

1. What data should be given as input?
2. How the data should be arranged or coded?
3. The dialog to guide the operating personnel in providing input.
4. Methods for preparing input validations and steps to follow when error occur.

Objectives

- Ensure the data entered is correct and minimizes errors through validation and checks.
- Protect sensitive information by implementing proper access controls and secure input methods.
- Detect and prevent incorrect or incomplete data using validation rules and prompts..

1.7.2 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

Select methods for presenting information.,Create document, report, or other formats that contain information produced by the system.The output form of an information system should accomplish one or more of the following objectives. Convey information about past activities, current status or projections of the Future, Signal important events, opportunities, problems, or warnings.,Trigger an action.,Confirm an action.

2. LITERATURE SURVEY

1. **Naga Swaroopa , " Encryption Approach for Securing Educational Data Using Attribute-Based Methods," Scopes Conference, CSM, 2025**,
   This paper focuses on securing educational and placement-related data using the attribute encryption techniques. It ensures data privacy and controlled access

2. **Bele V., Pingle S., Wanwe A., Vibhandik A., "Training and Placement Cell," 2024.**
   The authors design a system to streamline Training and Placement Cell operations by facilitating efficient student data management and real-time tracking of recruitment events. The framework enhances operational transparency and supports timely decision-making.

3. **Mehar M., Dhoke S., Chahande A., Lambat M., Tete S., "Campus Recruitment Management (Online) System," 2024.**
   This work proposes an online recruitment management system enabling seamless interaction between students, TPOs, and recruiters. It reduces paperwork, improves communication, and ensures structured handling of recruitment activities on campus.

4. **Kumar B. N., Kandula V., Ambiti P., Hema K., Buddha K., "Student Analysis System for Training and Placement," 2024.**
   The authors propose a system focused on analyzing student performance to assess placement readiness. It offers analytical insights to TPOs, helping them monitor trends, evaluate student preparedness, and optimize placement strategies.

5. **Chaware S., Kshirsagar K., Bankar G., Ramtekkar P., Lautre B., "Web Based Information System for Training and Recruitment at Industry," 2023.**
   This paper presents a web-based information system aimed at improving communication between students and recruiters. The system centralizes placement data, allowing effective tracking of recruitment processes and enhancing accessibility of placement-related information.

6. **Basha C. Z., Tasneem S., Miriyala P., "Enhanced Technique for Placement Monitoring Using ServiceNow Portal," 2013.**
   This study introduces an advanced placement monitoring solution built on the ServiceNow platform. The system provides structured tracking mechanisms that enhance efficiency, improve information flow, and support systematic management of placement records.

7. **Mythili M., Aishwarya R., Shenbagam P., Sandhiya, "e-Placement Management," 2022.**
   The authors present an electronic placement management system focused on automating student profile handling, interview scheduling, and real-time status updates. The system improves data accuracy and supports smooth coordination during placement drives.

8. **Anand K., Retheesh D., Hemalatha J., Karishma S., Logeswari R., "Application for Training and Placement Cell," 2022.**
   This paper introduces an automated software solution designed for Training and Placement Cells. It enables TPOs to manage student records, coordinate recruitment activities, and generate analytical reports. The system enhances administrative efficiency and reduces manual dependency in placement operations.

9. **TrimukheS., Todmal A., Pote K., Gite M., Pophale S. S., "Online Training and Placement System,"IRJET, 2021.**
   The authors develop an online portal aimed at automating various placement-related tasks. Their approach provides real-time access to placement updates and reduces manual processing, ensuring a more effective and accessible placement environment for students and administrators.

10. **Anjali et al., "Web Based Placement Management System," 2021.**
    The authors propose a web-based placement management system that supports student profile maintenance, interview scheduling, and placement tracking. The system simplifies the workflow of placement processes by offering structured data handling and improving coordination between students and placement officers.

Focused Area / Title	**Key Findings**	**Reference**
A Review on Placement Management System**[1]**	Highlights advancements, challenges, and the need for efficient digital tools to improve placement processes.	Spoorthi M. S., Kavana V., Koushik M. N., Veena M .Review paper 2024.
Placement cell automation and tracking **[2]**	Enhances transparency and enables real-time tracking of recruitment activities.	Bele V., Pingle S., Wanwe A., Vibhandik A. T&P cell 2024.
Campus Recruitment Management (Online) System **[3]**	Reduces paperwork, improves communication, and enables structured recruitment management.	Mehar M., Dhoke S., Chahande A Lambat M., Tete S., "Campus Recruitment Management (Online System," 2024.
Student Analysis System for Training and Placement **[4]**	Provides analytical insights to evaluate placement readiness and improve strategies	Kumar B. N., Kandula V., Ambiti P., Hema K., Buddha K., "Student Analysis System for Training and Placement," 2024.
Web Based Information System for Training and Recruitment at Industry **[5]**	Improves interaction between students and recruiters and centralizes placement data	Chaware S., Kshirsagar K., Banka G., Ramtekkar P., Lautre B., "Web Based Information System fo Training and Recruitment a Industry," 2023.

Table no. 2 Literature Review Summary

Focused Area/ Title	**Key Findings**	**Reference**
Enhanced Technique for Placement Monitoring Using ServiceNow Portal **[6]**	Improves efficiency, structured tracking, and information flow in placement activities	Basha C. Z., Tasneem S., Miriyala P., "Enhanced Technique for Placement Monitoring Using ServiceNow Portal," 2023.
e-Placement Management **[7]**	Improves data accuracy and supports real-time updates and coordination	Mythili M., Aishwarya R., Shenbagam P., Sandhiya, "e-Placement Management," 2022.
Application for Training and Placement Cell **[8]**	Reduces manual work and improves administrative efficiency with reporting features	Anand K., Retheesh D., Hemalatha J., Karishma S., Logeswari R., "Application for Training and Placement Cell," 2022.
0nline Training and Placement System**[9]**	Provides real-time updates and reduces manual processing	Trimukhe S., Todmal A., Pote K., Gite M., Pophale S. S., "Online Training and Placement System," IRJET, 2021.
Placement workflow automation**[10]**	Simplifies placement processes through structured data handling and coordination	Anjali et al., "Web Based Placement Management System," 2021.

Table no. 2 Literature Review Summary

3. SOFTWARE REQUREIMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.1.1 Data Analysis

The analysis of the existing placement process revealed that student details, academic records, resumes, and company requirements are mostly maintained manually in spreadsheets, emails, and physical documents, leading to data inconsistency, duplication, and difficulty in updating information. Training and Placement Officers struggle to filter eligible students, track placement progress, and generate accurate reports due to scattered and unorganized data sources. Students also lack real-time access to recruitment schedules, company criteria, and placement updates, causing delays and confusion during the placement season. These issues highlight the need for a centralized, digital system that can organize data efficiently, ensure accuracy, support quick retrieval, and streamline communication between students, administrators, and recruiters.

3.1.2 Data Preprocessing

Data preprocessing involves organizing and preparing student and company information to ensure accuracy, consistency, and usability within the Placement Management System. Since raw data collected from students—such as academic details, resumes, skills, and personal information—may contain errors, missing values, or inconsistencies, preprocessing starts with validating and cleaning the data. Duplicate entries are removed, incomplete fields are flagged for correction, and formats for attributes like phone numbers, email IDs, and CGPA are standardized. Company data, including eligibility criteria, job descriptions, and recruitment schedules, is also structured to maintain uniformity. All validated data is then categorized and stored in a systematic format to enable efficient filtering, profile matching, and report generation.

3.1.3 Machine Learning Algorithm for Prediction

Machine learning algorithms help predict student placement outcomes and job suitability by analyzing academic and skill-related data. Models like Logistic Regression, Decision Tree**s**, and Random Forest are effective for classifying whether a student is likely to be placed.Support Vector Machines (SVM) and K-Nearest Neighbors (KNN**)** support accurate eligibility and company-fit predictions.

For complex patterns, Neural Networks provide higher accuracy. These algorithms enable better insights for students and TPOs, improving decision-making during the placement process.

3.2 Functional Requirements

The Functional requirements for a system describe the functionality or the services that the system is expected to provide.These are the statements of services the system should provide and how the system should react to particular inputs and how the system should behave in particular situation.

- User Registration :User Register with their registration details.
- User Login:User Login their account using password.
- Live inputs: inputs given by the User requirement.
- Load Model:Trained or Tested model will be load.
- Predict Output: output will be predict based on parameters.

3.3 Non-Functional Requirements

The non-functional requirements describe the system constraints. Performance: The application should have better accuracy and should provide prediction in less time. Scalability: The system must have the potential to be enlarged to accommodate the growth.Capability: The capability of the storage should be high so the large amount of data can be stored in order to train the model

- The system shall ensure high reliability and stability during all stages of text processing and classification.
- The system shall support scalable performance as dataset size and category complexity increase.
- The system shall maintain efficient processing with minimal latency in generating predictions.

3.4 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Beyond simple cost-benefit analysis, this phase evaluates the strategic alignment of the software with the institution's long-term digital transformation goals. By identifying potential constraints—whether financial, technological, or human-centric—early in the lifecycle, the study minimizes the risk of project failure For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations are involved in the feasibility analysis are

1.Economic Feasibility

2.Technical Feasibility

3. Social Feasibility

3.4.1 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. The long-term economic benefit is even more significant, as the transition from paper-based, labor-intensive workflows to an automated digital platform reduces the operational costs associated with printing, physical storage, and administrative man-hours. This makes the system a high-ROI (Return on Investment) asset for the institution, as the marginal cost of adding new users or data remains negligible.

3.4.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.4.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Social Feasibility hinges on the human element, ensuring that the transition from manual to digital is met with enthusiasm rather than resistance. To achieve high user adoption, the system features an interface that mimics familiar social and professional platforms, reducing the learning curve for students and staff. A comprehensive change management strategy is integrated into the rollout, including interactive workshops and user manuals that demystify the technology. By positioning the system as a tool that empowers students with better opportunities and relieves TPOs of tedious tasks, the project fosters a sense of ownership among users.

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The proposed system is developed using Windows 7 Ultimate as the operating system and Python as the primary coding language. The application uses Python for the front-end logic and Django ORM for the back-end framework to handle server-side operations efficiently. For designing the user interface, standard web technologies such as HTML, CSS, and JavaScript are utilized to create a responsive and user-friendly experience. The system stores and manages data using MySQL, implemented through the WAMP server, ensuring reliable and structured database management.

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.x
- Core Libraries: Scikit-learn, NumPy, Pandas
- Development Environment: VS Code / PyCharm / Jupyter Notebook
- Documentation Tools: MS Word / LaTeX

4.2 Hardware Requirements

The hardware requirements define the minimum system configuration needed to ensure smooth operation, reliable data handling, and support for future enhancements within the placement management system. A Pentium-IV or a more modern processor such as an Intel Core i3 or i5 is recommended to handle application processes efficiently. The system should include at least 8 GB of RAM to manage multiple tasks, process user requests, and maintain stable performance during database operations. A storage capacity between 250 GB and 512 GB, either on an HDD or SSD, is required for storing application files, datasets, logs, and backup information. Standard input devices like a Windows keyboard and a two- or three-button mouse provide adequate support for interacting with the system. A display such as an SVGA monitor or any screen with a minimum size of 14 inches ensures clear visibility of the user interface and system outputs. Additionally, internet connectivity, while optional, is recommended to facilitate dataset downloads, system updates, and potential integration with cloud services in future expansions.

- Processor: Intel Core i3/i5 or equivalent
- Memory (RAM): Minimum 8 GB
- Storage: 250 GB HDD/SSD
- Optional: Internet connectivity for dataset access and future cloud integration.

5. SOFTWARE DESIGN

5.1 System Architecture

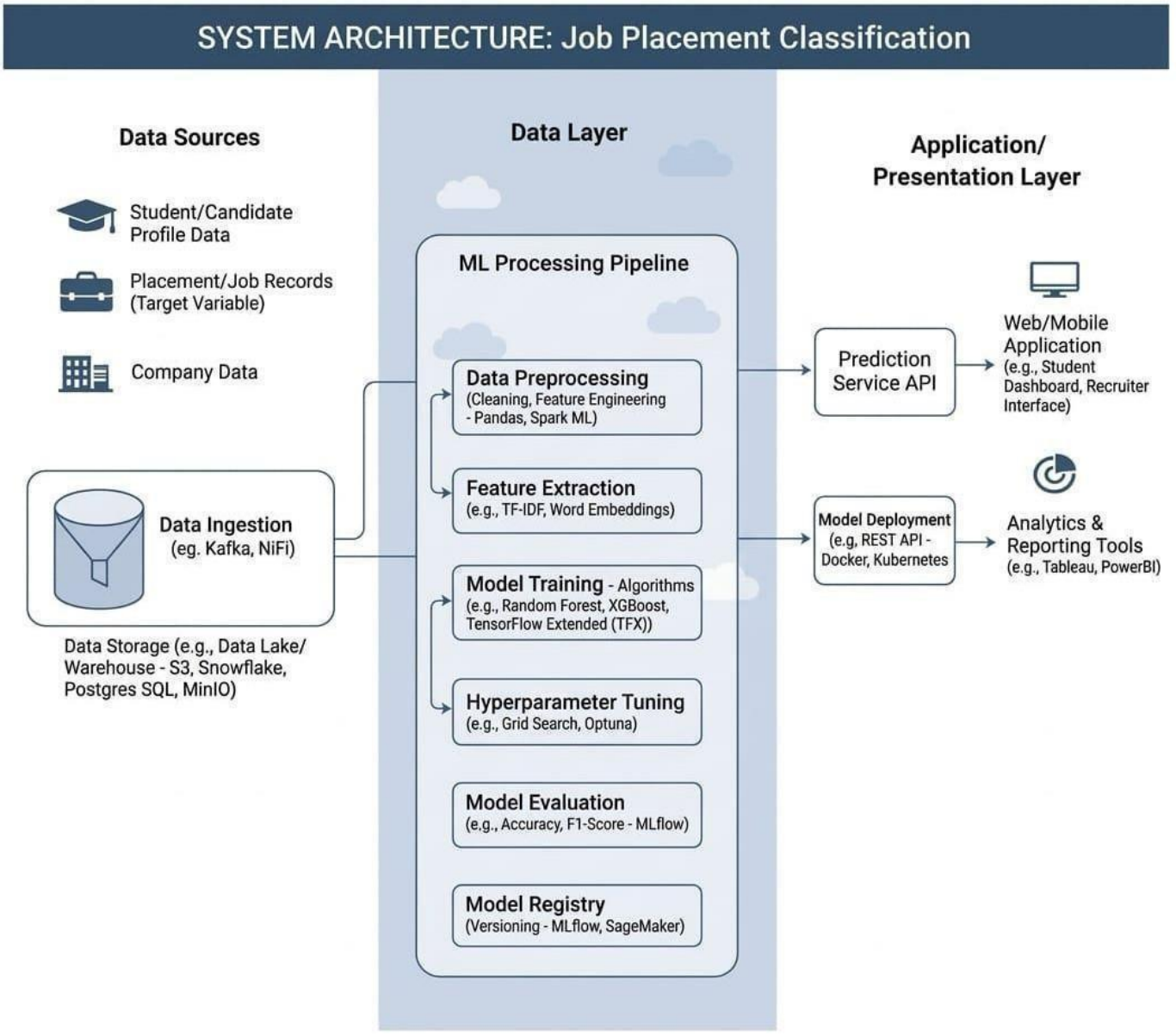


Fig: 5.1 System Architecture

The system architecture This diagram illustrates the architecture of a comprehensive Placement Management System, which is organized into four major modules: Drives, Trainings, Higher Education, and Alumni. Each module plays a significant role in managing different aspects of student development and placement activities within an institution. The Drives module deals with all recruitment-related operations by storing information about upcoming placement drives, keeping track of students who have been successfully placed, and maintaining details of their joining orders. This helps both students and placement officers follow the entire recruitment process in an organized and efficient manner.

The Trainings module focuses on improving student readiness for placements by offering programs such as Campus Recruitment Training (CRT), which enhances aptitude, communication, and problem-solving skills. It also includes companyspecific training designed to match the expectations and requirements of individual recruiters, thereby strengthening student employability. The Higher Education module supports students who prefer continuing their studies instead of seeking immediate employment, recording the exams they attempt, their scores, and any allotment orders received from universities or institutions.

This ensures that students aiming for postgraduate or international opportunities are properly guided and tracked. The Alumni module maintains detailed information about former students, including their personal and academic details, year of passing, higher education pursuits, job roles, company information, photographs, and salary packages. This data is valuable for building strong alumni networks and offering guidance to current students. Overall, the architecture of this system showcases a well-structured approach that not only simplifies placement management but also supports training, higher education planning, and alumni tracking, providing complete and holistic assistance to students throughout their career development journey.

5.2 Dataflow Diagram

The A Data Flow Diagram (DFD) is a structured analysis and design tool used to represent the flow of data within a system. It visually illustrates how data enters the system, how it is processed, stored, and how it moves between different components. The DFD breaks down a system into processes, data stores, data flows, and external entities, showing both the logical and physical aspects of the data movement.

- External Entities: Represent outside sources or destinations of data (e.g., users, external systems, or organizations).
- Processes: Represent activities or transformations that take input data and produce output data (e.g., classification, filtering).
- Data Stores: Represent repositories where data is stored for later retrieval (e.g., message database,

corpus storage).

- Data Flows: Represent the path and direction of data as it moves through the system.

The DFD provides a clear and simplified view of the entire system without going into implementation details. It helps stakeholders understand how data is captured, processed, and delivered. Typically, Level 0 (context diagram) gives the overall system view, while Level 1 and Level 2 diagrams break down processes into finer details.

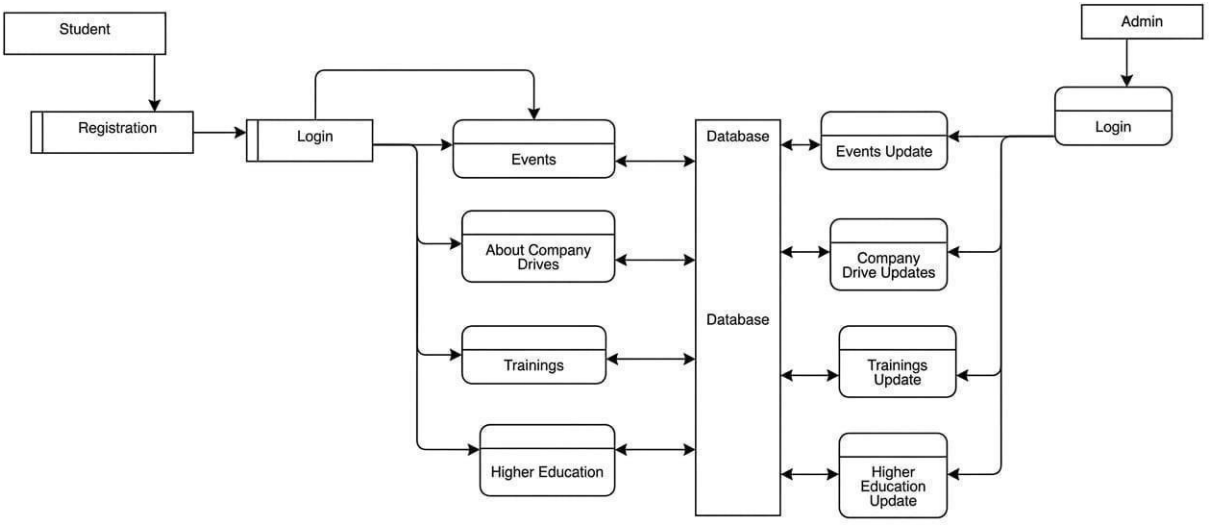


Fig 5.2 Dataflow Diagram

5.3 UML Diagrams

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that has proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using UML helps project teams communicate, explore potential designs and validate the architectural design of the software.

The Behavioral UML diagrams describe the behavior of the system, its actors, and the interaction between the components. On the other hand, Structural UML diagrams depict the static structure of the system, showing its components and relationships. UML has been integrated as a standard by OMG, and its primary goals are to provide a formal basis for understanding modeling languages, offer a ready-to-use expressive language for system developers, and encourage the growth of object-oriented tools.

Goals of UML:

- To provide a expressive visual modeling language for developing and exchanging the meaningful models.
- To Encourage the growth of object oriented tools.
- To integrate best practices into system development.
- To model both structural and behavioral aspects of the system.
- To support object-oriented design and development practices.
- To document the system clearly for future reference and maintenance.

Types of UML Diagrams:

1. Sequence Diagram:
2. Use Case Diagram:
3. Activity Diagram:
4. Class Diagram:

5.3.1. Sequence Diagram

A sequence diagram for placement data for job classification shows the step-by-step interaction between different components of the system in a time sequence. It illustrates how data flows from the user to the system and how the system processes it to produce job classification results. It typically represents interactions between actors such as Student/User, System, Database, and Machine Learning Model. The diagram shows how a user logs in, submits profile data (skills, academics, resume), the system validates and pre processes the data, sends it to the classification model, and receives the predicted job role. Finally, the system displays the result to the user.

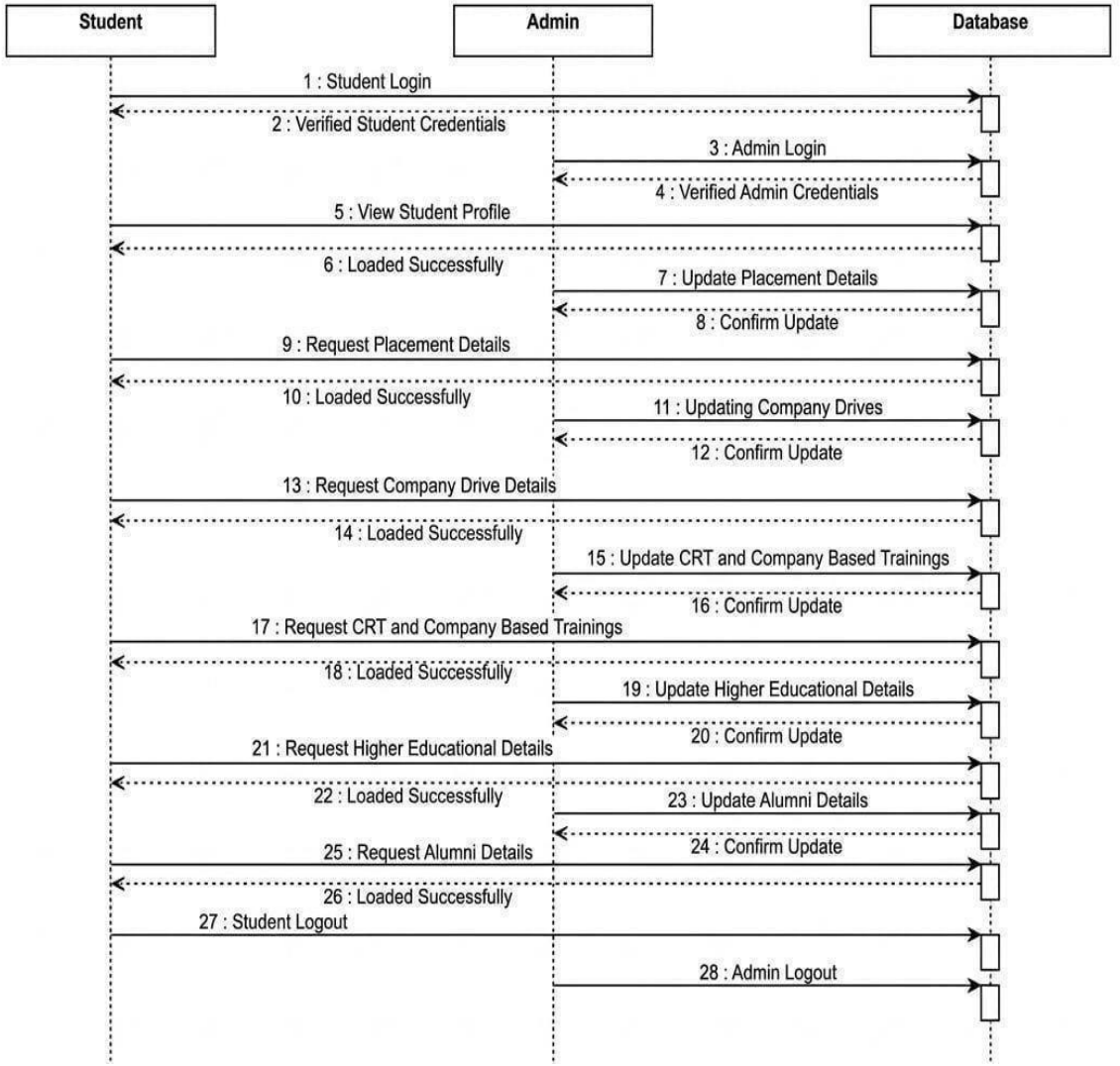


Fig 5.3.1: Sequence Diagram

List of actions

• User:

The user registers or logs into the system and provides input such as academic details, skills, and resume. This information is submitted for further processing and job classification.

• Database:

The system validates the user input, stores it in the database, and performs preprocessing like cleaning and formatting the data. It then prepares the data to be sent to the prediction model.

• Model:

The model receives the processed data and analyzes patterns based on training. It predicts the most suitable job role for the user based on their profile.

• Evaluation:

The evaluation component checks the prediction accuracy and generates the final result. The system then displays the classified job role to the user and stores it for future reference.

5.3.2 Use Case Diagram

This Unified Modeling Language Use Case Diagram illustrates the Admin interaction with a Placement Management System, specifically focusing on how data is funneled into a central database to eventually facilitate job classification.A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions areperformed forwhich actor. Rolesoftheactors in thesystem canbe depicted.

Hereisadetailedbreakdownofthecomponents andtheworkflow:

1. PrimaryActors

Admin(PrimaryActor):Thecentralauthorityresponsiblefordataentryandsystemmaintenance. Theytrigger all the actions (use cases) listed in the ovals.

Database(SecondaryActor): Representedhereas an actor,it signifiesthepersistentstoragelayer. Everyaction performedbytheadminresults in adatatransaction(creationorupdate) within thisdatabase.

2. Core Functional Use Cases

Thediagrambreaksdowntheadministrativeworkflowintoseveralkeymodules:

Authentication Layer

Register/Login: The entrypoints for theadminto access thesecuredashboard.

Logout: Ensuresthesessionisterminatedanddata remainssecureaftertheworkiscompleted.

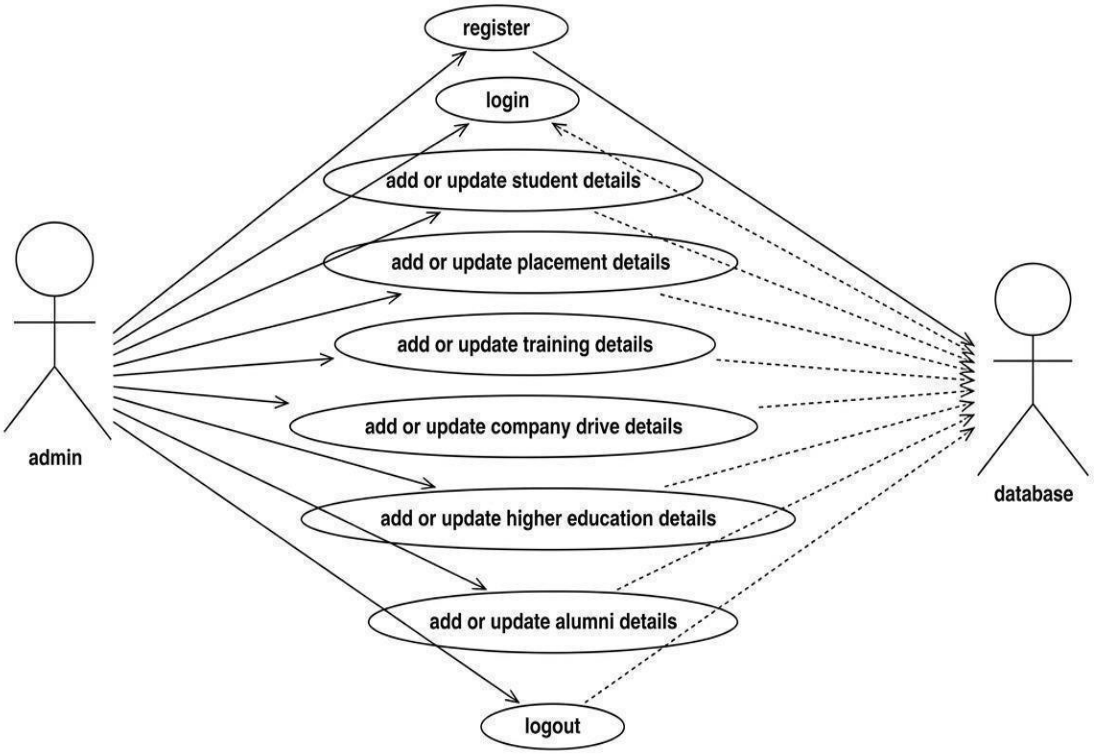


Fig 5.3.2 Use Case Diagram

5.3.3 Activity Diagram

The Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step work flows of components in a system. An activity diagram shows the overall flow of control. The work flow illustrates how two primary user role interact with data that would ultimately be used for job classification and placement tracking the student(Consumer role) Focuses on data retrieval and visualization.The Admin(Manager role) Focuses onentryandmaintenance.

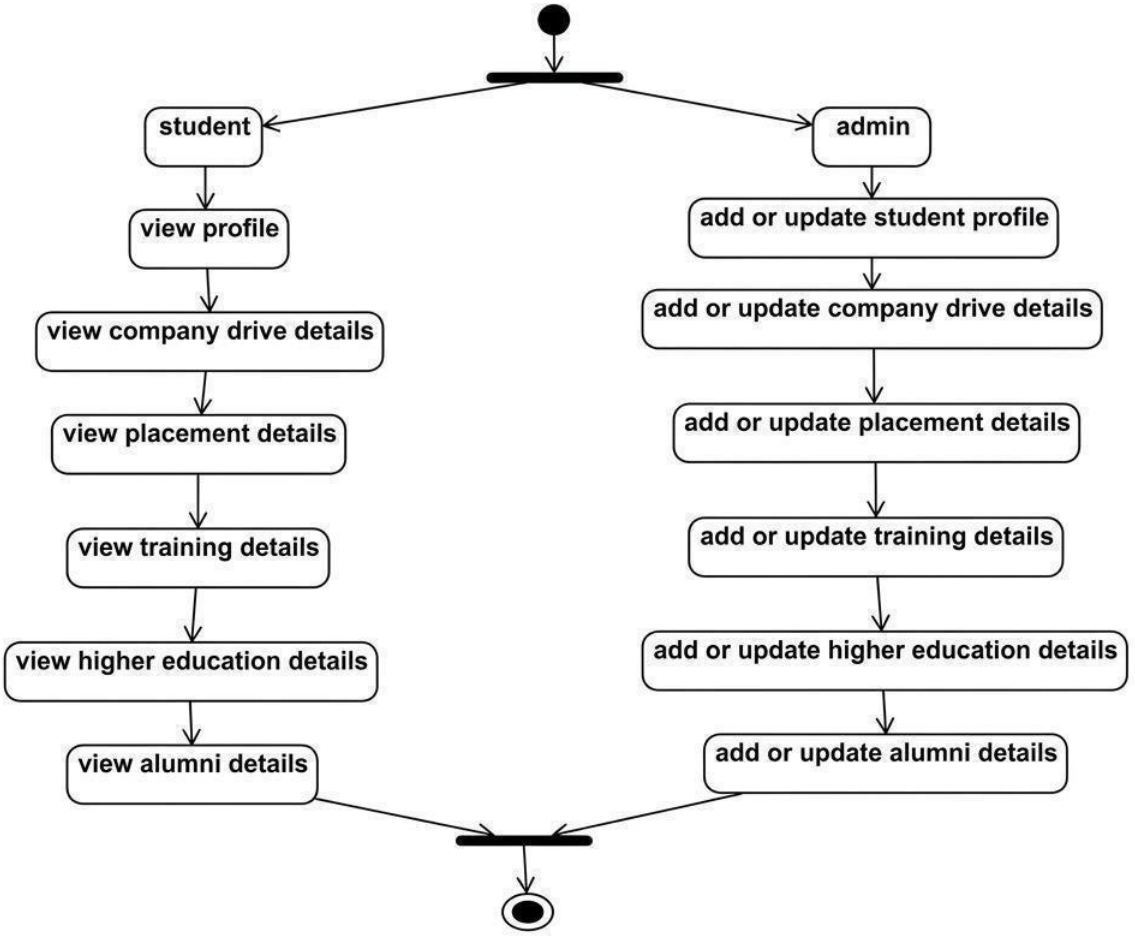


Fig 5.3.3 Activity Diagram

5.3.4. Class Diagram

The a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which classcontains information.

On This image represents a UML Class Diagram for a Placement Management System. It outlines the structural design of your project, specifically showing how different users (Actors) interact with the central data repository.The diagram is organized into three primary components: Student, Admin, and Database..

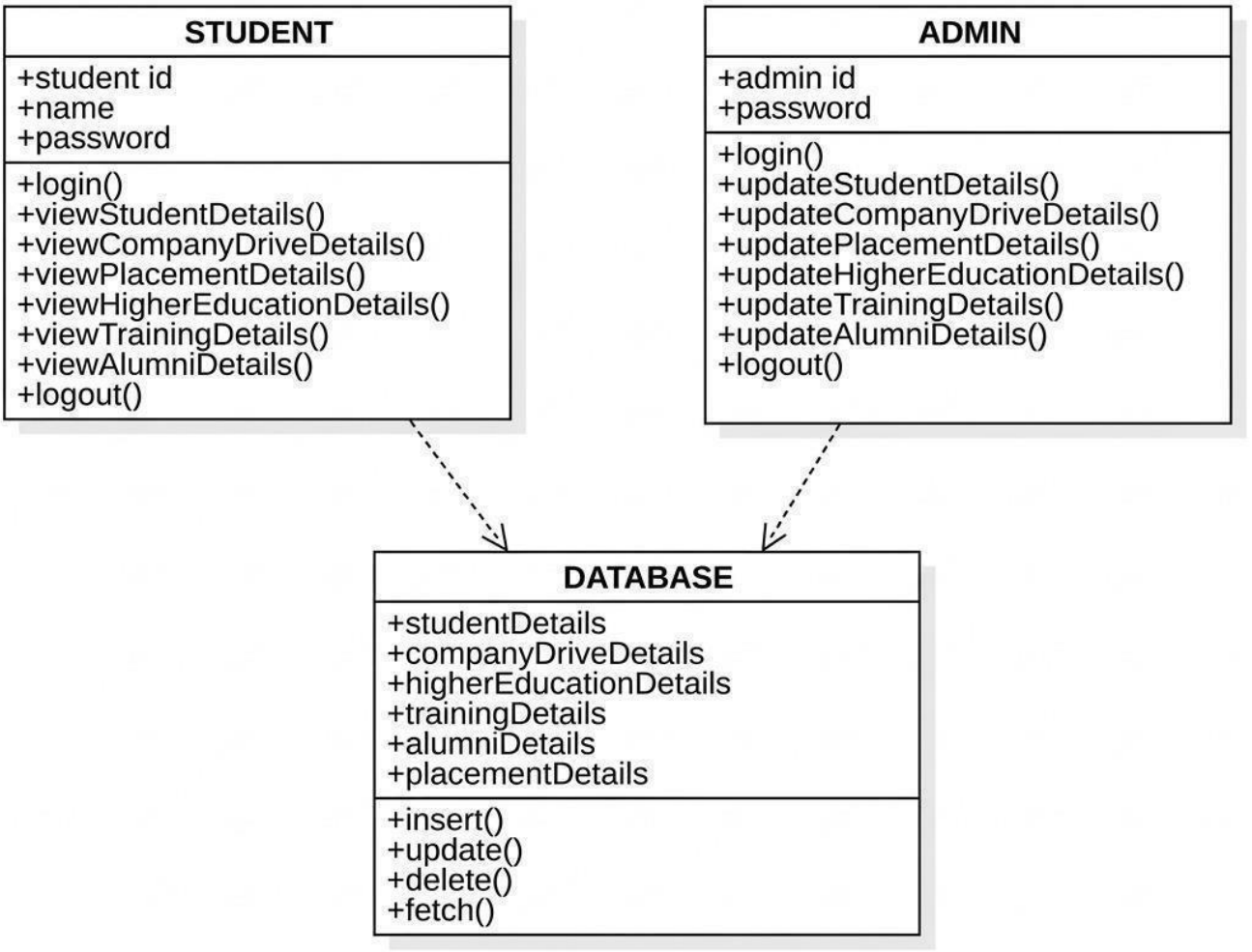


Fig 5.3.4 Class Diagram

6. CODING AND IMPLEMENTATION

6.1 Source Code

app.py:

```
import os
from flask import Flask, flash, redirect, render_template, request, session, abort
from models.keras_first_go import KerasFirstGoModel
from clear_bash import clear_bash
import numpy as np
from flask import Flask, request, jsonify, render_template
from flask import Flask, render_template, flash, url_for, request, session, redirect, jsonify
import os
import secrets
from PIL import Image
import tensorflow as tf
import keras
from keras.models import load_model
from tensorflow.python.keras.backend import set_session
from tensorflow.python.keras.models import load_model
from tensorflow.keras.optimizers import Adam
import pickle as pickle
import joblib
model1 = joblib.load("models/model.sav")
scalerX = pickle.load(open("models/scalerX", "rb"))

app = Flask(__name__)
cleaner=clear_bash()

model = load_model('models/model1.h5')
graph = tf.compat.v1.get_default_graph()

def train_model():
    global first_go_model

    print("Train the model")
    first_go_model = KerasFirstGoModel()

@app.route("/")
def home():

    return render_template('index.html')

@app.route('/index1')
def index1():
    return render_template('index1.html')

@app.route('/index2')
def index2():
    return render_template('index2.html')
```

```
@app.route('/index3')
def index3():
    return render_template('index3.html')

@app.route("/main")
def main():
    return render_template('main.html')

def save_picture(form_picture):
    random_hex = secrets.token_hex(8)
    _, f_ext     = os.path.splitext(form_picture.filename)
    picture_fn  = random_hex + f_ext
    picture_path = os.path.join(app.root_path, 'static/profile_pics', picture_fn)
    output_size  = (125,125)
    i            = Image.open(form_picture)
    i.thumbnail(output_size)
    i.save(picture_path)

    return picture_fn

@app.route("/predict4", methods = ['POST'])
def predict4():

    int_features= [int(x) for x in request.form.values()]
    print(int_features,len(int_features))
    final4=[np.array(int_features)]

    prediction4 = model1.predict(scalerX.transform([int_features]))
    output4=round(prediction4[0],2)
    print(output4)


    if (int(output4)==0):
        prediction = "Either anyone of this job position: Database Developer,  "

    elif (int(output4)==1):
        prediction = "Either anyone of this job position: Systems Security AdministratoAssociate "

    elif (int(output4)==2):
        prediction = "Either anyone of this job position: Web Developer,Information
        Security Analyst, CRM Business Analyst, Project Managet"

    elif (int(output4)==3):
        prediction = "Either anyone of this job position: Design & UX, “

    elif (int(output4)==4):
```

```
        prediction = "Either anyone of this job position: E-Commerce Analyst,
        Technical Services/Help Desk/Tech Support, Information Technology Auditor,
         Database Manager, Applications Developer,Database Administrator  "

    elif (int(output4)==5):
        prediction = "Either anyone of this job position: Network Engineer, "

    else:
        prediction = "invaild!"

    return (render_template('index3.html', prediction_text = prediction))


@app.route('/prediction',methods=['POST'])
def prediction():

    os          = request.form["os"]
    aoa         = request.form["aoa"]
    pc          = request.form["pc"]
    se          = request.form["se"]
    cn          = request.form["cn"]
    ma          = request.form["ma"]
    cs          = request.form["cs"]
    hac         = request.form["hac"]
    interest    = request.form["interest"]
    cert        = request.form["cert"]
    personality = request.form["personality"]
    mantech     = request.form["mantech"]
    leadership = request.form["leadership"]
    team        = request.form["team"]
    selfab      = request.form["selfab"]

    myu = [77.00318789848731, 76.99831228903614, 77.07569696212026, 77.11301412676585,
76.9541817727216, 77.0150018752344, 77.060320040005, 5.002687835979497]
    sig = [10.071578660726848, 10.098653693844197, 10.137528173238477, 10.088164425588161,
10.018397202418788, 10.18533143324003, 10.095941558583263, 2.582645138598079]
    arr = [os,aoa,pc,se,cn,ma,cs,hac]

    for i in range(8):
        arr[i] = float(arr[i])
        arr[i] = (arr[i]- myu[i])/sig[i]

    inti     = [0,0,0,0,0,0,0,0,0,0,0,0,0]
    certi    = [0,0,0,0,0,0,0]

    if interest == "analyst":
        inti[0] = 1
    elif interest == "hadoop":
        inti[1] = 2
    elif interest == "cloud":
        inti[2] = 3
```

```
elif interest == "data":
    inti[3] = 4
elif interest == "hacking":
    inti[4] = 5
elif interest == "management":
    inti[5] = 6
elif interest == "networks":
    inti[6] = 7
elif interest == "programming":
    inti[7] = 8
elif interest == "security":
    inti[8] = 9
elif interest == "software":
    inti[9] = 10
elif interest == "system":
    inti[10] = 11
elif interest == "testing":
    inti[11] = 12
elif interest == "web":
    inti[12] = 13

if cert == "app":
    certi[0] = 1
elif cert == "full":
    certi[1] = 2
elif cert == "hadoop":
    certi[2] = 3
elif cert == "security":
    certi[3] = 4
elif cert == "machine":
    certi[4] = 5
elif cert == "python":
    certi[5] = 6
elif cert == "shell":
    certi[6] = 7

for i in certi:
    arr.append(i)

for i in inti:
    arr.append(i)

if leadership == "yesl":
    arr.append(0)
    arr.append(1)
else:
    arr.append(1)
    arr.append(0)

if team == "yest":
    arr.append(0)
    arr.append(1)
```

```
else:
    arr.append(1)
    arr.append(0)

if personality == "extrovert":
    arr.append(1)
    arr.append(0)
else:
    arr.append(0)
    arr.append(1)

if selfab == "nos":
    arr.append(1)
    arr.append(0)
else:
    arr.append(0)
    arr.append(1)

if mantech == "man":
    arr.append(1)
    arr.append(0)
else:
    arr.append(0)
    arr.append(1)


print ('arr ',arr)
y      = model.predict(np.array( [arr,]))
result = np.where(y == np.amax(y))
print(y)
print(result)

if result[0]==[0]:
    return render_template('index1.html', prediction_text='Business Intelligence')
    print('Business Intelligence Analyst')
elif result[0]==[1]:
    return render_template('index1.html', prediction_text='Database Administrator')
    print('Database Administrator')
elif result[0]==[2]:
    return render_template('index1.html', prediction_text='Project Manager')
    print('Project Manager')
elif result[0]==[3]:
    return render_template('index1.html', prediction_text='Security Administrator')
    print('Security Administrator')
elif result[0]==[4]:
    return render_template('index1.html', prediction_text='Software Developer')
    print('Software Developer')
else:
    return render_template('index1.html', prediction_text='Technical Support')
    print('Technical Support')

print("done2")
```

```
@app.route('/result',methods = ['POST', 'GET'])
def result():
  if request.method == 'POST':
    result = request.form.getlist('Job')
    train_model()
    processed_text = first_go_model.prediction(result[0])
    result = {'Job': processed_text}
    return render_template("result.html",result = result)

def clear_bash():
   os.system('cls' if os.name == 'nt' else 'clear')


if _name___== "_main_":
   app.run(debug=True)
```

clear_bash.py:

```
import os

class clear_bash(object):
   def _init_(self):
      self.clean()

   def clean(self):
      os.system('cls' if os.name == 'nt' else 'clear')
```

dataset_handler.py:

```
   import pandas as pd
   from sklearn.model_selection import train_test_split
   from sklearn.preprocessing import LabelBinarizer
   from keras.preprocessing.text import Tokenizer


   class DatasetSpliter(object):

      def __init__(self,dataset_path,vocab_size,max_length):
         self.data = pd.read_csv(dataset_path, header = 0, names = ['Query', 'Description'])
         self.split_data()
         self.vocab_size=vocab_size
         self.max_length=max_length
         self.split_data()

      def split_data(self):
         # Split data to train and test (80 - 20)
         train, test = train_test_split(self.data, test_size=0.2)

         self.train_descs = train['Description']
```

```
        self.train_labels = train['Query']

        self.test_descs = test['Description']
        self.test_labels = test['Query']

    def data_encode(self):

        # define Tokenizer with Vocab Size
        tokenizer = Tokenizer(num_words=self.vocab_size)
        tokenizer.fit_on_texts(self.train_descs)
        x_train = tokenizer.texts_to_matrix(self.train_descs, mode='tfidf')
        x_test = tokenizer.texts_to_matrix(self.test_descs, mode='tfidf')

        encoder = LabelBinarizer()
        encoder.fit(self.train_labels)
        y_train = encoder.transform(self.train_labels)
        y_test = encoder.transform(self.test_labels)

        return x_train, y_train, x_test, y_test
```

keras_first_go:

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras import metrics
from sklearn.preprocessing import LabelBinarizer
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import one_hot

from dataset_handler import DatasetSpliter
from model_utils import conf_keras_first_go

class KerasFirstGoModel(object):
    def _init_(self):

spliter=DatasetSpliter(conf_keras_first_go.dataset_path,conf_keras_first_go.vocab_size,conf_keras_first_
go.max_length)
        split_data=spliter.data_encode()

        self.x_train=split_data[0]
        self.y_train = split_data[1]
        self.x_test = split_data[2]
        self.y_test = split_data[3]
        self.test_labels=spliter.test_labels

        self.create_model()

    def create_model(self):
        self.model = Sequential()
        self.model.add(Dense(conf_keras_first_go.dense, input_shape=
        (conf_keras_first_go.vocab_size,)))
        self.model.add(Activation(conf_keras_first_go.activation_function))
```

```
        self.model.add(Dense(conf_keras_first_go.dense))
        self.model.add(Activation(conf_keras_first_go.activation_function))
        self.model.add(Dropout(conf_keras_first_go.dropout))
        self.model.add(Dense(conf_keras_first_go.labels))
        self.model.add(Activation(conf_keras_first_go.last_activation_function))

        #Compile the model
        self.compile_model()

    def compile_model(self):
        self.model.compile(loss = conf_keras_first_go.loss,
                           optimizer = conf_keras_first_go.optimizer,
                           metrics = [metrics.categorical_accuracy, 'accuracy'])
        # summarize the model
        # print(self.model.summary())

    def create_history(self):
        self.model.fit(self.x_train, self.y_train,
                       batch_size=conf_keras_first_go.batch_size,
                       epochs=conf_keras_first_go.nb_epoch,
                       verbose=conf_keras_first_go.verbose,
                       validation_split=conf_keras_first_go.validation_split)

        score = self.model.evaluate(self.x_test, self.y_test,
                            batch_size=batch_size, verbose=1)

        # print('\nTest categorical_crossentropy:', score[0])
        # print('Categorical accuracy:', score[1])
        # print('Accuracy:', score[2])


    def prediction(self,user_text):

        # Encode the text
        encoded_docs = [one_hot(user_text, conf_keras_first_go.vocab_size)]
        # pad documents to a max length
        padded_text = pad_sequences(encoded_docs, maxlen=conf_keras_first_go.max_length,
padding='post')
        # Prediction based on model
        prediction = self.model.predict(padded_text)
        # Decode the prediction
        encoder = LabelBinarizer()
        encoder.fit(self.test_labels)
        result = encoder.inverse_transform(prediction)

        return result[0]
```

.ipynb_checkpoints:

```
import numpy as np #used to import mathematical operations
import matplotlib.pyplot as plt #used to plot different things in python
import pandas as pd #import data sets and manage data sets
```
[23] Python

Data Exploration

markdown

```
dataset = pd.read_csv('para_data_train_shuffle.csv')
```
[24] Python

```
dataset.head()
```
[25] Python

Feature Selection for training and testing Datas

```
X_train = dataset.iloc[: , :38].values  #independant variable vector
```
[26] Python

```
Y_train = dataset.iloc[: , 38:].values
```
[27] Python

```
dataset1 = pd.read_csv('para_data_test_shuffle.csv')
```
[28] Python

```
dataset1.head()
```
[36] Python

```python
X_test = dataset1.iloc[: , :38].values  #independant variable vector
```

[29] Python

```python
Y_test = dataset1.iloc[: , 38:].values
```

[30] Python

Data Scaling

```python
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

[31] Python

Neural Network - CNN

Generate | + Code | + Markdown

```python
from sklearn.decomposition import PCA

from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense, Dropout, GaussianNoise, Conv1D
from keras.preprocessing.image import ImageDataGenerator
from keras import regularizers
import seaborn as sns
```

Python

```python
model = Sequential()
layers = 10
units = 15
```

Python

```python
model.add(Dense(units, input_dim=38, activation='relu', kernel_regularizer=regularizers.l1(0.1)))
```

Python

```python
for i in range(layers):
    model.add(Dense(units, activation='relu'))
```

Python

```
model.add(Dense(6, activation='softmax'))
```
[14] Python

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
```
[15] Python

```
model.summary()
```
[16] Python

```
history = model.fit(X_train, Y_train, epochs=1000, batch_size=512, validation_split=0.30, verbose=2)
```
[17] Python

```
_, test_acc = model.evaluate(X_test, Y_test, verbose=1)
print('Test: ',test_acc)
```
[18] Python

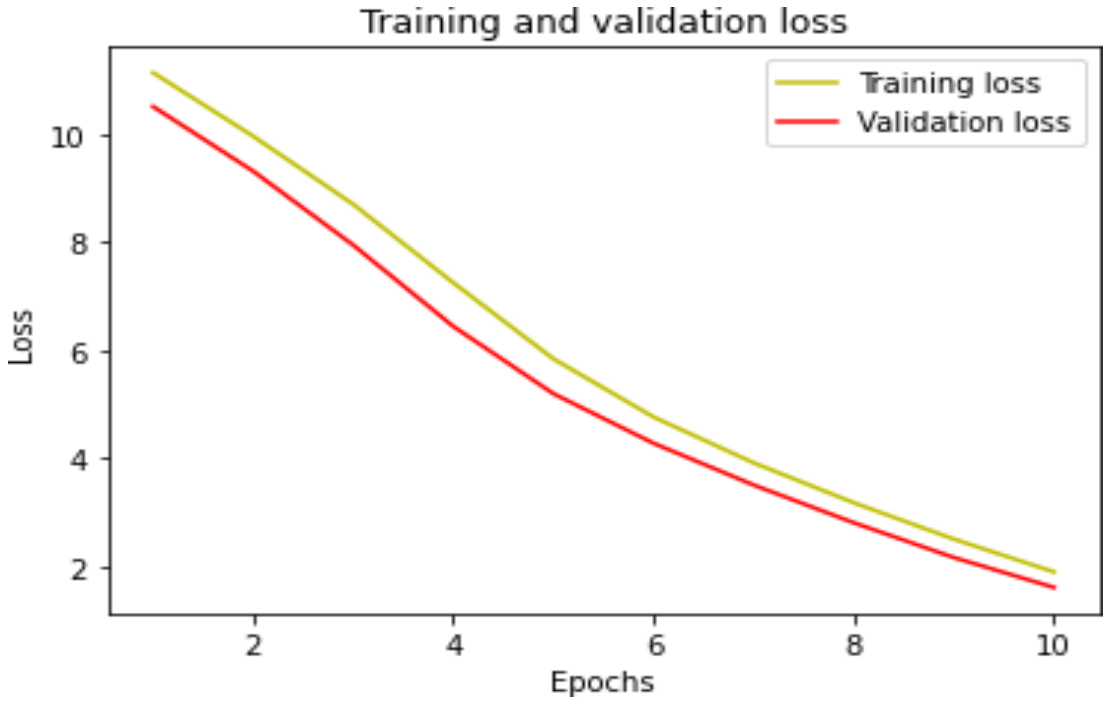
Saving Model

```
from keras.models import load_model

model.save('model.h5')
```
[35] Python

Accuracy & Loss Visualization

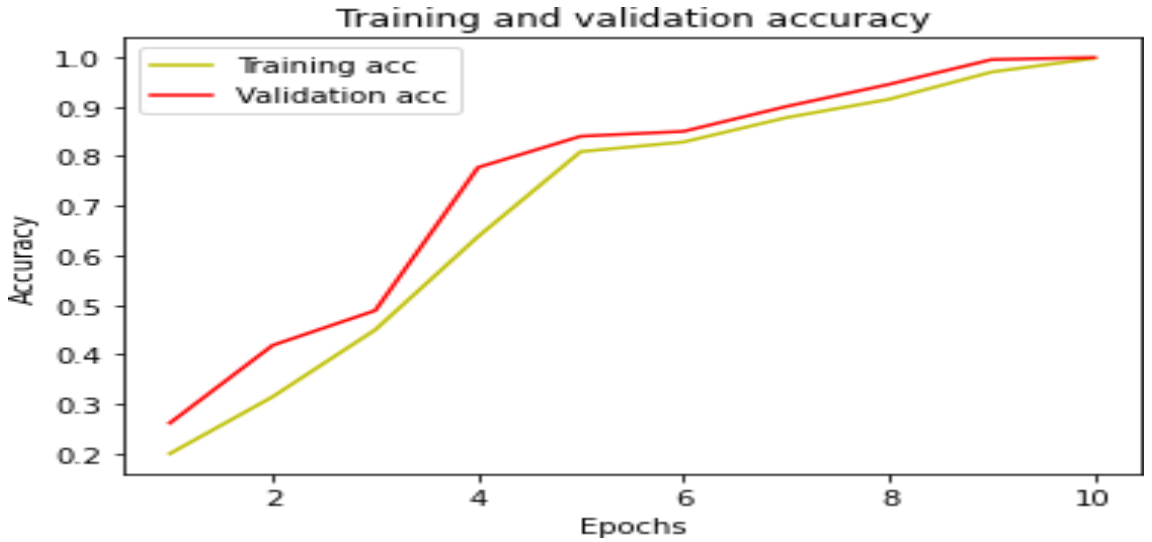
Generate + Code + Markdown

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```
[19] Python




```
acc = history.history['categorical_accuracy']
val_acc = history.history['val_categorical_accuracy']
plt.plot(epochs, acc, 'y', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

[22] Python



Notebook.ipynb:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.utils import resample
import joblib
import warnings
%matplotlib inline
```

[1] Python

Data Exploration

```
df = pd.read_csv("data/data.csv")
df.head(5)
```

[2] Python

Preprocessing

```
missing_data = df.isnull().sum()
print(missing_data)
```

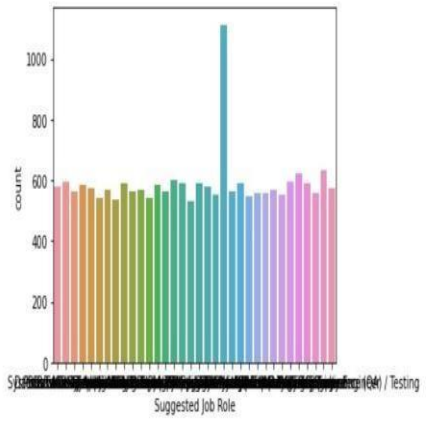
[3] Python

Visualization

```
import seaborn as sns
sns.countplot(x="Suggested Job Role", data = df)
```

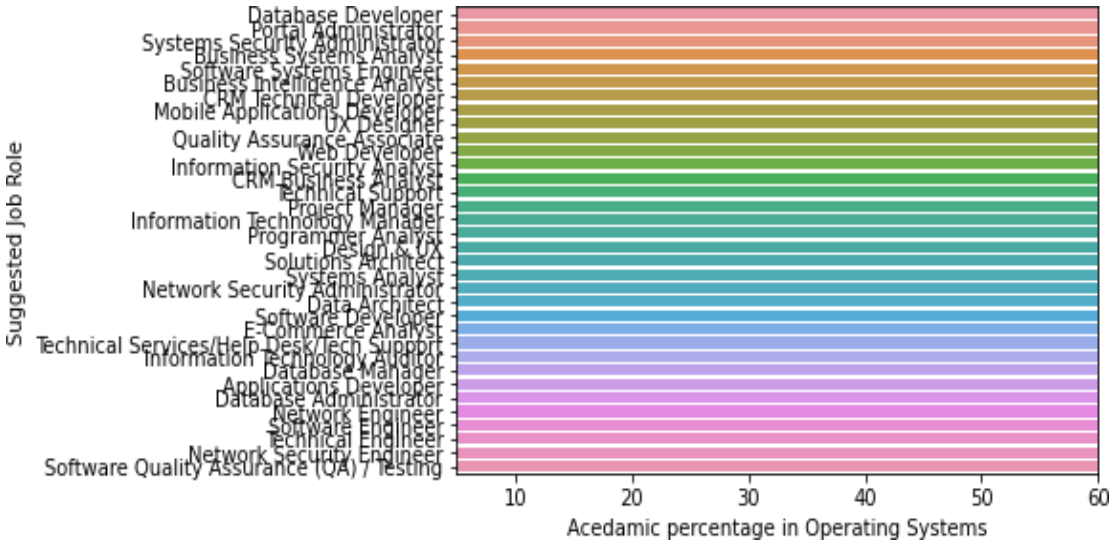
[4] Python

```
<AxesSubplot:xlabel='Suggested Job Role', ylabel='count'>
```




```
sns.barplot( df['Acedamic percentage in Operating Systems'],df['Suggested Job Role'])
plt.xlim(5,60)
plt.show()
```

[5] Python



Data Cleaning

```
df.info()
```
[6] Python

```
df.drop(['memory capability score', 'Interested subjects','interested career area ','Job/Higher Studies?','Type of company want to settle in?','Taken inputs from seniors or elders',
```
[7] Python

```
df['reading and writing skills'].unique()
```
[8] Python

```
array(['excellent', 'poor', 'medium'], dtype=object)
```

```
df['Suggested Job Role'] = df['Suggested Job Role'].replace({'Database Developer': 0, 'Technical Support': 0, 'Business Intelligence Analyst': 0, 'Business Systems Analyst': 0, 'Por
```
[9] Python

Generate + Code + Markdown

Start Chat to Generate Code (Ctrl+I)

```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
```
[10] Python

```
df['can work long time before system?']= label_encoder.fit_transform(df['can work long time before system?'])
df['self-learning capability?']= label_encoder.fit_transform(df['self-learning capability?'])
df['Extra-courses did']= label_encoder.fit_transform(df['Extra-courses did'])
df['certifications']= label_encoder.fit_transform(df['certifications'])
df['workshops']= label_encoder.fit_transform(df['workshops'])
df['talenttests taken?']= label_encoder.fit_transform(df['talenttests taken?'])
df['olympiads']= label_encoder.fit_transform(df['olympiads'])
df['reading and writing skills']= label_encoder.fit_transform(df['reading and writing skills'])
df['Management or Technical']= label_encoder.fit_transform(df['Management or Technical'])
df['Introvert']= label_encoder.fit_transform(df['Introvert'])
```
[11] Python

```
df['can work long time before system?'].unique()
df['self-learning capability?'].unique()
df['Extra-courses did'].unique()
df['certifications'].unique()
df['workshops'].unique()
df['talenttests taken?'].unique()
df['olympiads'].unique()
df['reading and writing skills'].unique()
df['Management or Technical'].unique()
df['Introvert'].unique()
```

[12] Python

```
array([0, 1])
```

```
df.info()
```

[13] Python

Generate + Code + Markdown

```
print(df.shape)
```

[14] Python

```
(20000, 25)
```

Feature Selection

```
X = df.iloc[:, 0:24]
y = df.iloc[:,24]
```

[15] Python

Data Splitting & Scaling

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

[16] Python

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
sc_y = StandardScaler()
X_train = sc_x.fit_transform(X_train)
X_test = sc_x.transform(X_test)
```

[17] Python

```
import pickle
pickle.dump(sc_x, open("scalerX", "wb"))
```

[18] Python

```
X_scaler = StandardScaler().fit(X_train)
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

[19] Python

Machine Learning

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

[20] Python

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
RF.fit(X_train_scaled, y_train)
predictions = RF.predict(X_test_scaled)
print("*Confusion Matrix for RF: ")
print(confusion_matrix(y_test, predictions))
```

[21] Python

```
confusion = confusion_matrix(y_test, predictions)
TP = confusion[0, 0]
TN = confusion[0, 1]
FP = confusion[1, 0]
FN = confusion[1, 1]
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print(classification_error)
val1 = accuracy_score(y_test, predictions) *100
print(val1)
from sklearn import metrics
RF_sensitivity = ((TP / float(FN + TP)))* 100

print(RF_sensitivity)
RF_specificity = ((TN / (TN + FP))) * 100

print(RF_specificity)
```

(variable) confusion: ndarray

[22] Python

Voting Classifier

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
clf1 = SVC(gamma='auto')
clf2 = RandomForestClassifier(n_estimators=50, random_state=1)
clf3 = DecisionTreeClassifier()
eclf1 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('dt', clf3)], voting='hard')
eclf1.fit(X_train, y_train)
predictions = eclf1.predict(X_test)
print("*Confusion Matrix for Voting Classifier: ")
print(confusion_matrix(y_test, predictions))
```

[22] Python

```
confusion = confusion_matrix(y_test, predictions)
TP = confusion[0, 0]
TN = confusion[0, 1]
FP = confusion[1, 0]
FN = confusion[1, 1]
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print(classification_error)
val2 = accuracy_score(y_test, predictions) * 100
print(val2)
from sklearn import metrics
VOT_sensitivity = ((TP / float(FN + TP))) * 100

print(VOT_sensitivity)
VOT_specificity = ((TN / (TN + FP))) * 100

print(VOT_specificity)
```

[23] Python

Decision Tree Classifier

Generate + Code + Markdown

```
from sklearn import tree
DT = tree.DecisionTreeClassifier()
DT.fit(X_train_scaled, y_train)
predictions = DT.predict(X_test_scaled)
print("*Confusion Matrix for DT: ")
print(confusion_matrix(y_test, predictions))
```

[24] Python

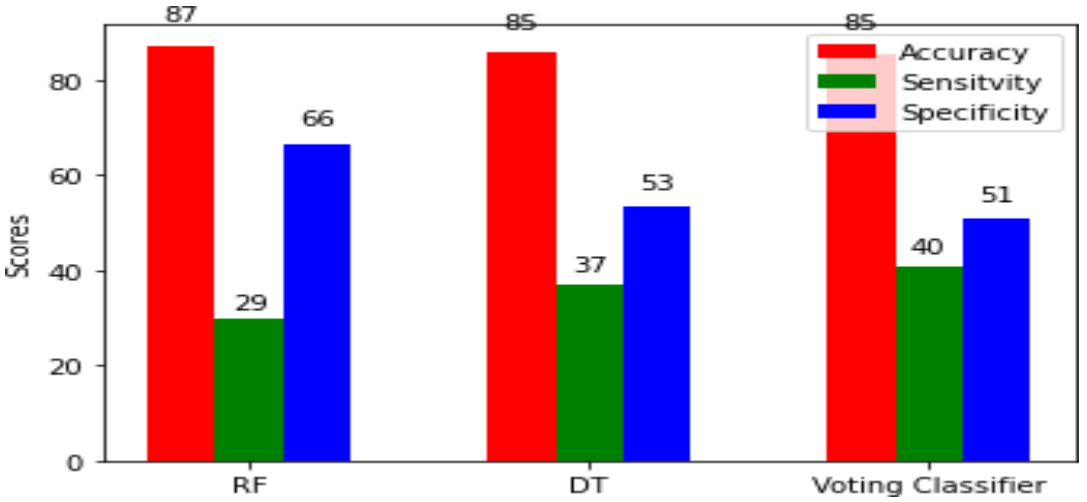
```
confusion = confusion_matrix(y_test, predictions)
TP = confusion[0, 0]
TN = confusion[0, 1]
FP = confusion[1, 0]
FN = confusion[1, 1]
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print(classification_error)
val3 = accuracy_score(y_test, predictions) * 100
print(val3)
from sklearn import metrics
DT_sensitivity = ((TP / float(FN + TP))) * 100

print(DT_sensitivity)
DT_specificity = ((TN / (TN + FP))) * 100

print(DT_specificity)
```

[25] Python

Accuracy Comparison

```
import numpy as np
import matplotlib.pyplot as plt

N = 3
ind = np.arange(N)  # the x locations for the groups
width = 0.2      # the width of the bars

fig = plt.figure()
ax = fig.add_subplot(111)

yvals = [val1*5,val2*5,val3*5]

rects1 = ax.bar(ind, yvals, width, color='r')
zvals = [RF_sensitivity,VOT_sensitivity,DT_sensitivity]
rects2 = ax.bar(ind+width, zvals, width, color='g')
kvals = [RF_specificity,VOT_specificity,DT_specificity]
rects3 = ax.bar(ind+width*2, kvals, width, color='b')

ax.set_ylabel('Scores')
ax.set_xticks(ind+width)
ax.set_xticklabels( ('RF','DT','Voting Classifier') )
ax.legend( (rects1[0], rects2[0], rects3[0]), ('Accuracy', 'Sensitvity', 'Specificity') )

def autolabel(rects):
    for rect in rects:
        h = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.05*h, '%d'%int(h),
                ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

plt.show()
```

Python



Model Building

```
import joblib
filename = 'model.sav'
joblib.dump(RF, filename)
```

[23] Python

```
['model.sav']
```

Python

6.2 Implementation:

The implementation phase of the proposed system focuses on transforming the designed architecture into a fully functional application capable of predicting cryptocurrency market performance using machine learning and deep learning techniques. The system is developed as a web-based application using Flask, integrating a trained CNN-LSTM deep learning model, along with database management and user interaction modules.

The implementation follows a modular approach, ensuring that each component such as user interface, backend processing, database handling, and predictive modeling operates independently yet cohesively. The system ensures efficient data handling, secure user authentication, and accurate prediction generation, thereby providing a complete end-to- end solution for market forecasting.

6.2.1 Development Environment

The development environment plays a crucial role in ensuring smooth implementation and execution of the system. The proposed system is developed using Python as the primary programming language due to its extensive support for machine learning and web development libraries.

The backend framework used is Flask, which provides lightweight and flexible web application development. The system also utilizes TensorFlow and Keras for implementing deep learning models, while libraries such as NumPy, Pandas, and Joblib are used for data processing and model handling.

The development tools include Visual Studio Code and Jupyter Notebook for coding, testing, and experimentation. SQLite is used as the database due to its simplicity and ease of integration. The system is designed to run on standard operating systems such as Windows or Linux, without requiring high-end hardware resources, making it accessible and cost-effective.

6.2.2 Frontend Implementation

The frontend of the system is designed to provide a user-friendly and interactive interface that allows users to easily interact with the application. The frontend is implemented using HTML templates integrated with Flask's Jinja2 templating engine.

Several web pages are developed to handle different functionalities of the system. These include the index page, login page, registration page, home page, prediction page, history page, and analytics page. Each page is designed with clear navigation and structured input fields to ensure ease of use.

The prediction page allows users to input various cryptocurrency-related features such as price, volume, sentiment scores, and market indicators. The results are displayed in a structured format, providing the predicted closing price along with relevant insights.

Overall, the frontend ensures that the system is accessible to both technical and non- technical users by providing a clean and intuitive interface.

6.2.3 Backend Implementation

The backend of the system is implemented using the Flask web framework, which handles the core functionality and business logic of the application. The backend is responsible for managing routes, processing user inputs, handling sessions, and integrating the predictive model.

Each functionality of the system is mapped to specific routes such as login, register, predict, history, and logout. The backend processes HTTP requests from the frontend and returns appropriate responses.

Session management is implemented to ensure secure user authentication, allowing only authorized users to access prediction features. Flash messages are used to provide feedback

to users, such as successful login, registration errors, or prediction results.

The modular structure of the backend ensures maintainability and scalability, allowing future enhancements to be easily integrated into the system.

6.2.4 Database Implementation

The system uses SQLite as the database for storing user information and prediction history. The database is initialized at runtime, and tables are created if they do not already exist.

The Users table stores user details such as username, email, phone number, and encrypted password. Password security is ensured using hashing techniques provided by the Werkzeug library.

The Predictions table stores details of each prediction made by users, including input data, predicted values, and timestamps. Each prediction is linked to a specific user through a foreign key relationship.

This database structure ensures efficient data storage and retrieval, enabling users to track their prediction history and analyze past results.

6.2.5 Data Preprocessing Implementation

Data preprocessing is an essential step in ensuring that the input data is clean, consistent, and suitable for model prediction. The system performs preprocessing operations similar to those used during model training.

The dataset is first cleaned by removing missing values and invalid entries. Irrelevant columns such as identifiers are dropped to reduce noise in the data. String-based missing values are converted into appropriate formats, and numerical consistency is maintained across all features.

Normalization and scaling are applied using pre-trained scalers to ensure that input values fall within the same range as the training data. Feature selection is performed based on the model requirements, ensuring that only relevant attributes are used for prediction.

Visualization techniques such as histograms, line plots, and scatter plots are also used during preprocessing to understand data distribution and relationships between features.

6.2.6 Model Implementation (CNN-LSTM)

The predictive model used in the system is a hybrid CNN-LSTM model, which combines the strengths of Convolutional Neural Networks and Long Short-Term Memory networks.

The CNN layers are responsible for extracting spatial features and identifying patterns within the input data. These layers help in capturing relationships between different features such as price, volume, and sentiment indicators.

The LSTM layers are used to capture temporal dependencies and sequential patterns in the data. Since cryptocurrency prices are time-series data, LSTM plays a crucial role in modeling trends and predicting future values.

The model is trained using historical data and saved as a pre-trained file. During implementation, the model is loaded into the system along with the corresponding scalers and metadata. The model uses Mean Squared Error as the loss function and Adam optimizer for training.

6.2.7 Prediction Module Implementation

The prediction module is responsible for generating output based on user input. This module follows a structured workflow to ensure accurate predictions.

When a user submits input data, the system converts the data into a numerical vector based on predefined features. The data is then scaled using the input scaler to match the training data distribution.

The scaled data is reshaped into a time-window format, which is required for the CNN-LSTM model. The model processes this input and generates a predicted value, which is then inverse scaled to obtain the actual predicted closing price.

The prediction result is displayed to the user and simultaneously stored in the database for future reference.

6.2.8 User Authentication and Security Implementation

Security is an important aspect of the system, and several mechanisms are implemented to ensure data protection and user privacy.

User authentication is handled through login and registration modules. Passwords are securely stored using hashing techniques, preventing unauthorized access to sensitive data.

Session management ensures that users remain authenticated throughout their interaction with the system. Unauthorized users are restricted from accessing prediction and history features.

Input validation is also performed to prevent incorrect or malicious data from affecting system performance.

6.2.9 Prediction History and Data Management

The system maintains a record of all predictions made by users. This feature allows users to view their past predictions and analyze trends over time.

The history module retrieves prediction data from the database and displays it in a structured format. Users can review input values, predicted results, and timestamps, enabling better understanding and evaluation of the system's performance.

This functionality enhances the usability of the system by providing transparency and traceability of predictions.

6.2.10 System Integration and Workflow

The complete system integrates all components into a unified workflow. The process begins with user authentication, followed by data input, preprocessing, prediction, and result display.

The backend communicates with the database to store and retrieve information, while the frontend provides an interface for user interaction. The predictive model operates seamlessly within this pipeline, ensuring real-time prediction capability.

This integration ensures that the system operates efficiently and delivers accurate results in a user-friendly manner.

6.2.11 Implementation Advantages

The implementation of the proposed system offers several advantages. It provides an end-to-end solution combining web development and deep learning. The system ensures accurate predictions through advanced modeling techniques and structured preprocessing.

It offers real-time prediction capability, secure user authentication, and efficient data management. The modular architecture allows easy scalability and future enhancements such as real-time data integration and advanced analytics.

Overall, the implementation demonstrates a practical and efficient approach to solving the problem of cryptocurrency market prediction using modern technologies.

7. SYSTEM TESTING

System testing is a crucial phase in the software development lifecycle that ensures the complete and integrated system functions correctly according to the specified requirements. It involves evaluating the entire application as a whole rather than focusing on individual components, with the primary objective of identifying defects, inconsistencies, and performance issues before deployment. In the context of the proposed cryptocurrency market prediction system, system testing plays a vital role in validating the reliability, accuracy, and efficiency of all modules, including user authentication, data preprocessing, model prediction, database management, and user interface.

The testing process is designed to verify that the system behaves as expected under various conditions, including valid and invalid inputs, different user interactions, and varying data scenarios. Since the application integrates multiple technologies such as Flask for web development, SQLite for database management, and a CNN-LSTM deep learning model for prediction, it is essential to ensure seamless interaction between these components. System testing confirms that the data flows correctly from the user interface to the backend, is processed accurately by the model, and the results are displayed properly to the user.

In addition to functional correctness, system testing also evaluates non-functional aspects such as performance, security, usability, and reliability. The system must be able to handle multiple user requests efficiently, maintain data integrity, and ensure secure access through proper authentication mechanisms. Testing also ensures that the prediction module produces consistent and accurate results based on the input data, reflecting the effectiveness of the trained model in real-world scenarios.

Furthermore, system testing helps in identifying edge cases and unexpected behaviors that may not have been considered during development. By simulating real-time usage conditions, the system can be assessed for its robustness and stability. This phase also ensures that all integrated modules work cohesively without conflicts, thereby delivering a smooth and uninterrupted user experience.

Overall, system testing serves as a validation step that guarantees the developed application meets both functional and non-functional requirements. It ensures that the cryptocurrency prediction system is dependable, secure, and ready for deployment, providing users with accurate predictions and a reliable platform for analyzing market trends.

7.1 Types of System Testing

System testing for the proposed cryptocurrency market prediction system involves multiple levels of testing to ensure that the entire application functions correctly, efficiently, and securely. Since the system integrates various components such as a web interface, backend processing using Flask, a deep learning prediction model, and a database, it is essential to perform different types of testing to validate each aspect of the system. The following types of system testing are carried out to ensure the reliability and robustness of the application. The testing strategy is designed to simulate real-world trading environments, ensuring the system can handle volatile market fluctuations and high-frequency data updates without lag. It encompasses rigorous validation of the data pipeline, from API ingestion to the final visual output, to guarantee that the predictive insights remain accurate and actionable.

The following are the major types of system testing performed for the proposed system:

7.1.1 Functional Testing

Functional testing is conducted to verify whether each module of the system operates according to the specified requirements. It ensures that all functionalities such as user registration, login, prediction generation, and history retrieval are working correctly.

In this system, functional testing includes validating user inputs, checking correct navigation between pages, ensuring proper interaction between frontend and backend, and verifying that the prediction module generates accurate outputs. It also ensures that the system correctly stores and retrieves user data and prediction results from the database.

This type of testing confirms that the system performs all expected operations without errors

and meets the intended objectives.It encompasses rigorous validation of the data pipeline, from API ingestion to the final visual output, to guarantee that the predictive insights remain accurate and actionable.

7.1.2 Unit Testing

Unit testing focuses on testing individual components or modules of the system independently. Each function in the application, such as database connection, model loading, input processing, and prediction generation, is tested separately to ensure correctness. unit tests are designed to cover a wide range of edge cases, such as handling null inputs, invalid data formats, and database timeouts. By utilizing testing frameworks like Py test or Unit test, developers can automate these checks, ensuring that any future modifications to the code do not inadvertently break existing logic.

For example, functions responsible for handling user authentication, data scaling, and model prediction are tested in isolation. This helps in identifying errors at an early stage and ensures that each module performs its intended task accurately before integration.

Unit testing improves code quality and reduces the chances of failures during later stages of development

7.1.3 Integration Testing

Integration testing is performed to verify the interaction between different modules of the system. Since the proposed system consists of multiple components such as frontend interface, backend logic, database, and machine learning model, it is essential to ensure that these components communicate effectively. This testing phase specifically focuses on the data flow between interconnected layers, ensuring that no information is lost or corrupted as it moves through the pipeline. For instance, the system validates that the preprocessed features generated by the Flask backend align perfectly with the input dimensions required by the deep learning model.

In this system, integration testing checks whether user inputs from the frontend are correctly processed by the backend, whether the processed data is properly passed to the prediction model, and whether the results are stored in the database and displayed to the user. It also verifies that the asynchronous communication between the server and the database remains stable, preventing synchronization issues when multiple users request predictions simultaneously. By simulating end-to-end transaction paths, integration testing confirms that the disparate technological stacks—from the UI elements to the complex

This testing ensures smooth data flow across the system and eliminates issues related to module interaction.

7.1.4 System Testing

System testing evaluates the complete system as a whole to ensure that all components work together correctly in a real-world environment. It validates the overall behavior of the system under different scenarios.Beyond functional verification, system testing rigorously assesses how the application behaves under extreme conditions, such as sudden spikes in user traffic or unexpected interruptions in third-party data feeds. This involves testing the system's resilience to ensure that even if one component—like an external market API—experiences downtime, the rest of the application remains operational or fails gracefully with appropriate user notifications.

The process also includes cross-browser compatibility checks and mobile responsiveness testing to guarantee a consistent user experience regardless of the device being used to access the predictions. Furthermore, non-functional aspects such as latency and throughput are measured to ensure that the transition from data input to prediction output occurs within an acceptable timeframe for real-time decision-making.

For the proposed system, system testing includes verifying end-to-end workflows such as user login, data input, prediction generation, and result display. It also ensures that the system handles different types of inputs and conditions without failure.

This testing confirms that the system meets both functional and non-functional requirements and is ready for deployment.

7.1.5 Performance Testing

Performance testing is conducted to evaluate the system's responsiveness, speed, and stability under different workloads. It ensures that the system can handle multiple users and large datasets efficiently. To ensure the system remains viable for professional use, performance testing also delves into stress testing and endurance testing, which measure the application's ability to maintain peak efficiency over extended periods of continuous operation.

In this system, performance testing checks the time taken for model prediction, database operations, and page loading. It also evaluates how the system behaves when multiple users access the prediction module simultaneously. By simulating high-concurrency environments, developers can observe how memory management and CPU utilization scale as the volume of simultaneous prediction requests increases. This proactive analysis allows for the fine-tuning of server configurations and the optimization of database queries

This testing helps in identifying bottlenecks and ensures that the system provides fast and efficient responses.

7.1.6 Security Testing

Security testing ensures that the system is protected against unauthorized access and potential threats. It verifies that user data is safe and that proper authentication mechanisms are in place.the integrity of session management is scrutinized to prevent session hijacking or fixation, ensuring that user tokens are securely handled and expire appropriately after periods of inactivity. Robust access control lists (ACLs) are tested to confirm that users can only view their own history. These measures collectively transform the application into a resilient fortress capable of maintaining privacy and operational continuity in an increasingly hostile digital landscape.

In the proposed system, security testing includes validating password encryption, session management, and access control. It ensures that only authenticated users can access sensitive features such as prediction and history modules.

This testing helps in maintaining data confidentiality and preventing security breaches.

7.1.7 Usability Testing

Usability testing focuses on evaluating the user-friendliness and ease of use of the system. It ensures that users can interact with the application without confusion or difficulty.

In this system, usability testing involves checking the design of web pages, clarity of input forms, readability of prediction results, and ease of navigation between different modules. Feedback from users is used to improve the interface and enhance user experience.

This testing ensures that the system is accessible and convenient for both technical and non-technical users.

7.1.8 Regression Testing

Regression testing is performed to ensure that new updates or changes in the system do not affect existing functionalities. Whenever modifications are made, previously tested features are re-tested to confirm that they still work correctly.

In this system, regression testing ensures that updates in the prediction model, database, or frontend design do not impact core functionalities such as login, prediction, or data storage.

This testing helps in maintaining system stability and reliability over time.

7.1.9 Validation Testing

Validation testing ensures that the developed system meets the user requirements and fulfills its intended purpose. It confirms that the system delivers accurate predictions and provides meaningful insights to users. To ensure the system provides genuine value to its stakeholders, validation testing involves a rigorous comparison between the model's predicted price points and actual historical market data. This process, often referred to as back testing, allows developers to calculate precision metrics such as Mean Absolute Error (MAE) or Root Mean Square Error (RMSE), ensuring that the predictions stay within a reliable margin of error.

For the proposed system, validation testing includes verifying prediction accuracy, checking consistency of outputs, and ensuring that the system provides useful results for decision-making.

This testing confirms that the system is suitable for real-world applications and meets user expectations.

7.2 Test Strategies

Test strategies define the systematic approach adopted to evaluate the performance, functionality, and reliability of the proposed cryptocurrency market prediction system. A well-planned testing strategy ensures that all components of the system are thoroughly examined under different scenarios, thereby minimizing errors and enhancing overall system quality. Since the system integrates multiple technologies such as web development, database management, and deep learning models, a combination of different testing approaches is required to ensure complete validation.

The testing strategy for the proposed system follows a structured methodology that includes planning, execution, monitoring, and evaluation of test cases. It ensures that both functional and non-functional requirements are satisfied, and the system performs efficiently in real-world conditions.

7.2.1 Black Box Testing Strategy

Black box testing focuses on testing the functionality of the system without considering its internal structure or implementation details. The system is tested based on input and output behavior.

In the proposed system, users provide input values such as market features, and the system generates predicted outputs. The tester verifies whether the output is correct and meaningful without analyzing how the prediction is internally computed. This strategy is useful for validating user interface functionality, input handling, and prediction results.

7.2.2 White Box Testing Strategy

White box testing involves examining the internal logic, code structure, and execution paths of the system. It ensures that all statements, conditions, and loops in the program are executed correctly.

In this system, white box testing is applied to backend components such as Flask routes, database operations, and prediction logic. It verifies that functions for data preprocessing, scaling, model loading, and prediction execution work correctly. This strategy helps in identifying logical errors, code inefficiencies, and security vulnerabilities.

7.2.3 Top-Down Testing Strategy

Top-down testing is an approach where testing starts from the top-level modules and gradually moves to lower-level modules. High-level functionalities are tested first, followed by detailed components.

In the proposed system, testing begins with user interface modules such as login and prediction pages. Once these are verified, lower-level components like backend processing, database interaction, and model prediction are tested. This strategy ensures early detection of major functional issues and improves system reliability.

7.2.4 Bottom-Up Testing Strategy

Bottom-up testing follows the opposite approach, where testing starts from the lowest-level components and gradually integrates higher-level modules.

For this system, testing begins with individual components such as database connectivity, preprocessing functions, and model prediction modules. These components are tested independently before integrating them with the frontend interface. This approach ensures that foundational modules are stable and error-free before building the complete system.

7.2.5 Integration Testing Strategy

Integration testing strategy focuses on validating the interaction between different modules of the system. It ensures that data flows correctly between components without loss or corruption.

In this system, integration testing verifies that user inputs from the frontend are correctly passed to the backend, processed by the prediction model, and stored in the database. It also ensures that the output is accurately displayed to the user. This strategy helps in identifying issues related to module communication and data consistency.

7.2.6 Validation and Verification Strategy

Validation and verification are essential to ensure that the system meets both design specifications and user requirements.

Verification ensures that the system is developed correctly according to design specifications, while validation ensures that the system fulfills user needs. In the proposed system, verification checks include correctness of code implementation and model integration, whereas validation ensures that the prediction results are meaningful and useful for users.

7.2.7 Performance Testing Strategy

Performance testing strategy focuses on evaluating the system's speed, responsiveness, and stability under different workloads.

In this system, performance testing ensures that prediction results are generated within an acceptable time frame. It also checks system behavior when multiple users access the application simultaneously. This strategy helps in optimizing system performance and ensuring smooth user experience.

7.2.8 Security Testing Strategy

Security testing strategy ensures that the system is protected against unauthorized access and data breaches. It focuses on safeguarding user information and maintaining system integrity.

In the proposed system, this includes testing password encryption, session handling, and access control mechanisms. It ensures that only authenticated users can access prediction and history modules. This strategy is crucial for maintaining user trust and system reliability.

7.2.9 Error Handling and Recovery Strategy

Error handling strategy ensures that the system can handle unexpected inputs and failures gracefully without crashing.

In this system, proper validation is implemented to handle invalid inputs, missing values, or incorrect data formats. Error messages are displayed to guide users in correcting inputs. Recovery mechanisms ensure that the system continues to function smoothly even when errors occur.

7.2.10 Regression Testing Strategy

Regression testing strategy is used to ensure that modifications or updates in the system do not affect existing functionalities.

Whenever changes are made to the code, model, or database, previously tested modules are re-tested to confirm that they still work correctly. This strategy ensures system stability and prevents the introduction of new errors during updates.

7.2.11 User Acceptance Testing Strategy

User Acceptance Testing (UAT) ensures that the system meets the expectations of end users and is ready for deployment.

In this system, users interact with the application by performing tasks such as registration, login, prediction, and viewing history. Feedback is collected to evaluate system usability, accuracy, and overall satisfaction. This strategy ensures that the system is practical and effective for real-world usage.

Overall, the testing strategies adopted for the proposed system ensure comprehensive evaluation of all components and functionalities. By combining multiple testing approaches, the system achieves high reliability, accuracy, and performance, making it suitable for real-world deployment and usage.

7.3 Sample Test Cases

S No.	Test Case	Input	Expected Result	Result
01.	Home Page Display	Open application	Home page with navigation bar displayed	Pass
02.	Navigation Menu Functionality	Click on menu options (Home, Prediction)	Corresponding pages are loaded correctly	Pass
03.	Prediction Input Form	Enter academic/skill details	Input fields accept data without errors	Pass
04.	Form Submission	Click submit button	Data processed and prediction initiated	Pass
05.	Skills Input Module	Enter skills in textbox	Skills accepted and processed	Pass
06.	Learning/Resources Page	Open coding/resources section	Tutorials and learning content displayed	Fail

Table no 7.3 Test Cases

Fig 7.3.1: Home Page Interface

The screenshot shows the initial landing page of the system with the navigation bar containing options such as Home, Prediction, and Skills. This verifies that the application loads successfully and the user interface is properly displayed.

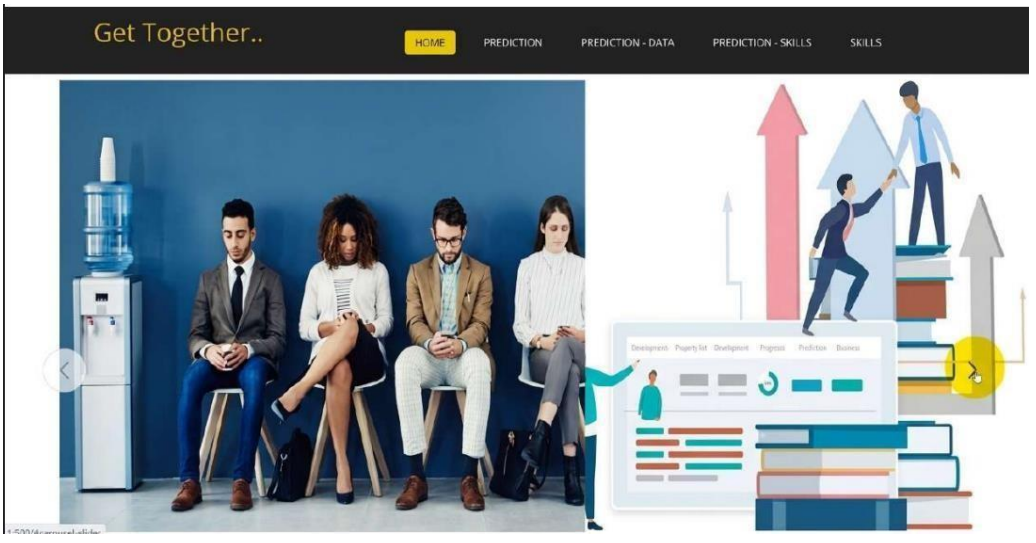


Fig 7.3.2: Navigation Menu Functionality

This figure demonstrates the working of the navigation bar. When the user clicks different menu options, the system redirects to the respective pages, confirming proper routing and page linking.

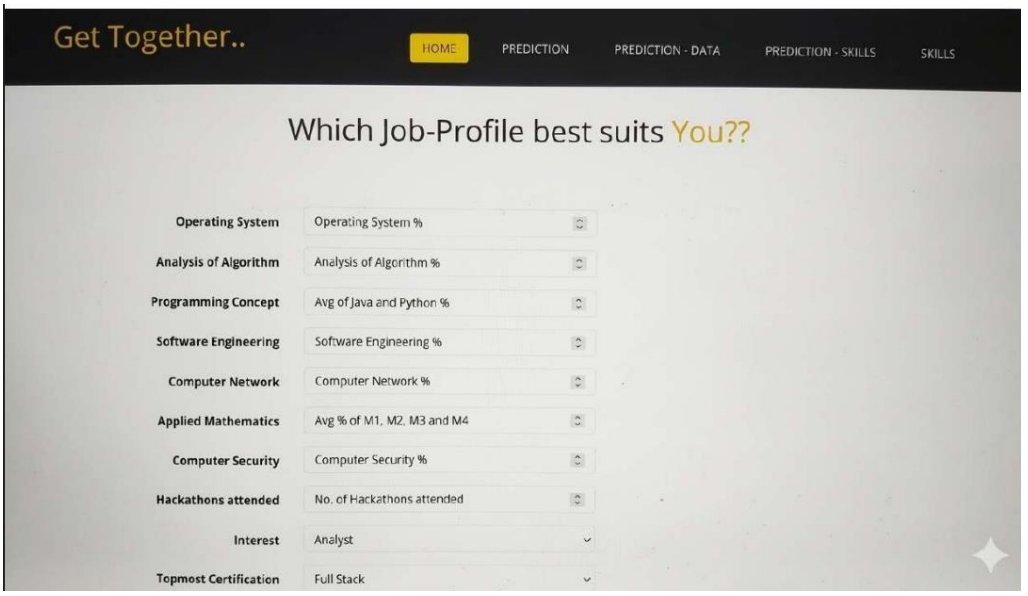


Fig 7.3.3: Prediction Input Form

The figure displays the input form where users enter details such as operating system knowledge, programming skills, and other parameters. This validates that the system correctly accepts user inputs.

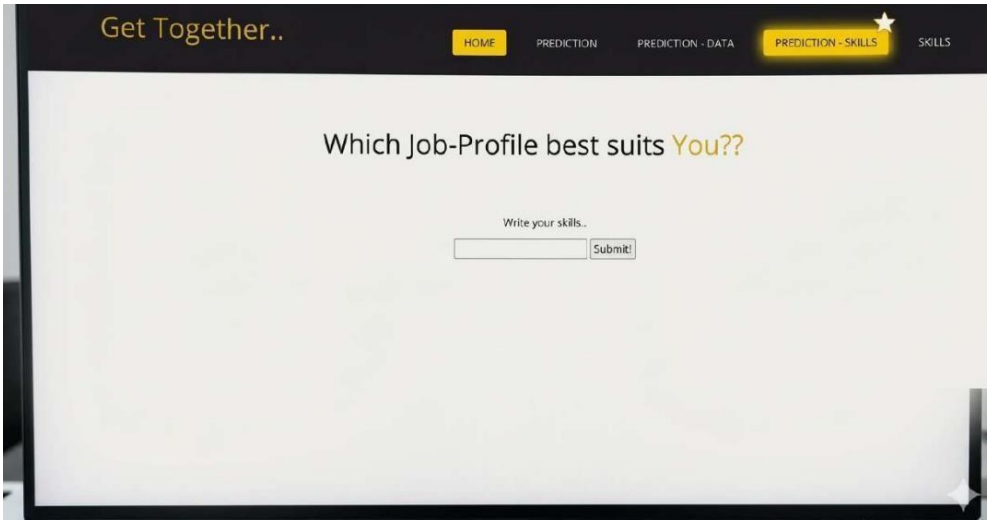


Fig 7.3.4: Form Submission and Processing

This screenshot represents the system after clicking the submit button. It confirms that the backend processes the entered data and initiates prediction without errors.

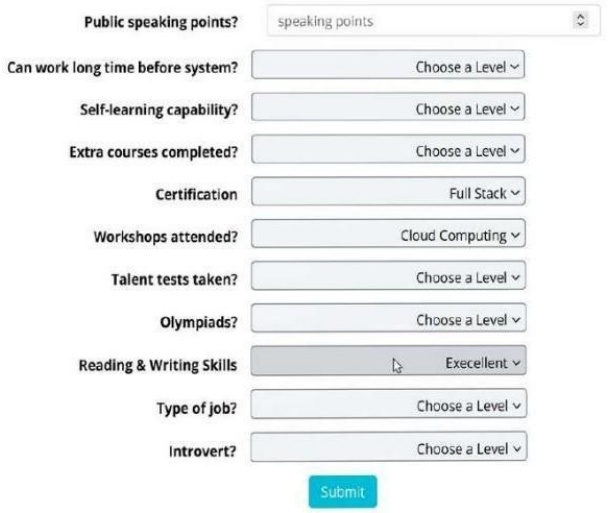


Fig 7.3.5: Skills Input Module

This figure shows the module where users can manually enter their skills. It ensures that the system captures skill-based input correctly for prediction purposes.

Fig 7.3.6: Learning Resources / Coding Section

The screenshot displays the learning resources section containing programming tutorials such as C, Java, and Python. This verifies that additional system modules are functioning and displaying content properly.

8. RESULTS

The results of the proposed system demonstrate the successful implementation of a web-based application that assists users in identifying suitable job profiles based on their academic performance, technical skills, and personal attributes. The system effectively integrates user input modules, processing mechanisms, and output display components to provide meaningful and accurate results.

The outputs obtained from the system are represented through various interface screens, which validate the correct functioning of each module. These screenshots (as shown in the uploaded output document) highlight different stages of user interaction, including homepage navigation, input forms, skill entry, and learning resources display.

The system produces results in a structured and user-friendly format, ensuring that users can easily interpret the recommendations and insights generated. The following figures represent the key output screens of the system along with their descriptions.

Fig 8.1: Home Page Interface

The screenshot illustrates the landing page of the application, where the user is presented with a navigation bar containing options such as Home, Prediction, Prediction Data, Prediction Skills, and Skills. This confirms that the application loads successfully and provides a structured interface for navigation. The homepage serves as the entry point for all system functionalities.

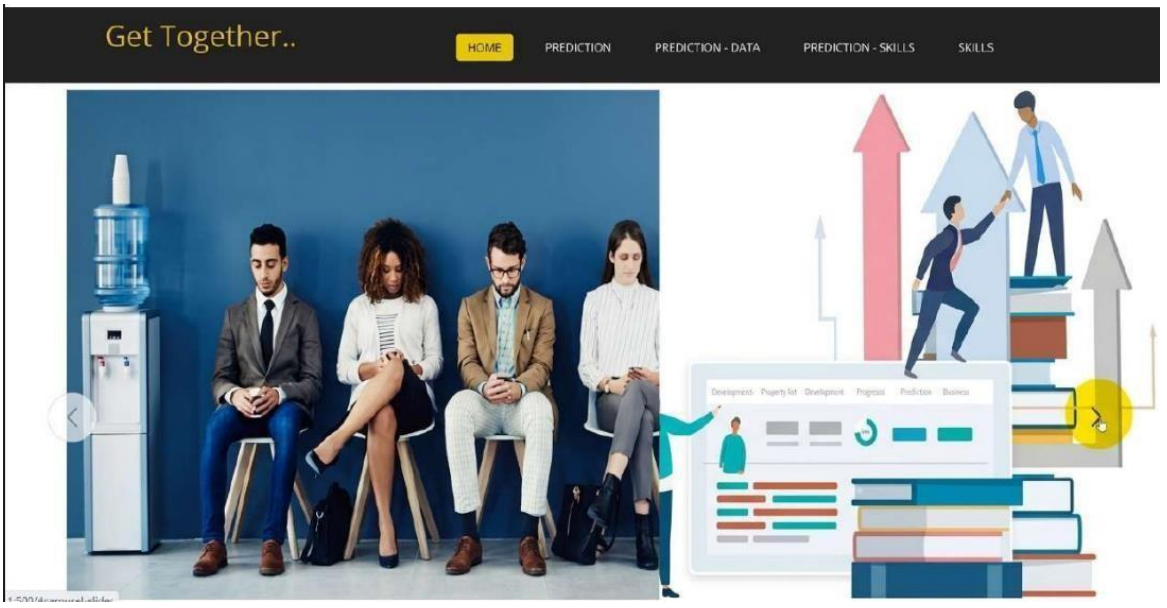


Fig 8.2: Home Page with Visual Content

This figure displays the homepage along with visual elements such as banners and images, enhancing the user interface and overall experience. It demonstrates that the system is not only functional but also visually appealing and interactive, improving user engagement.

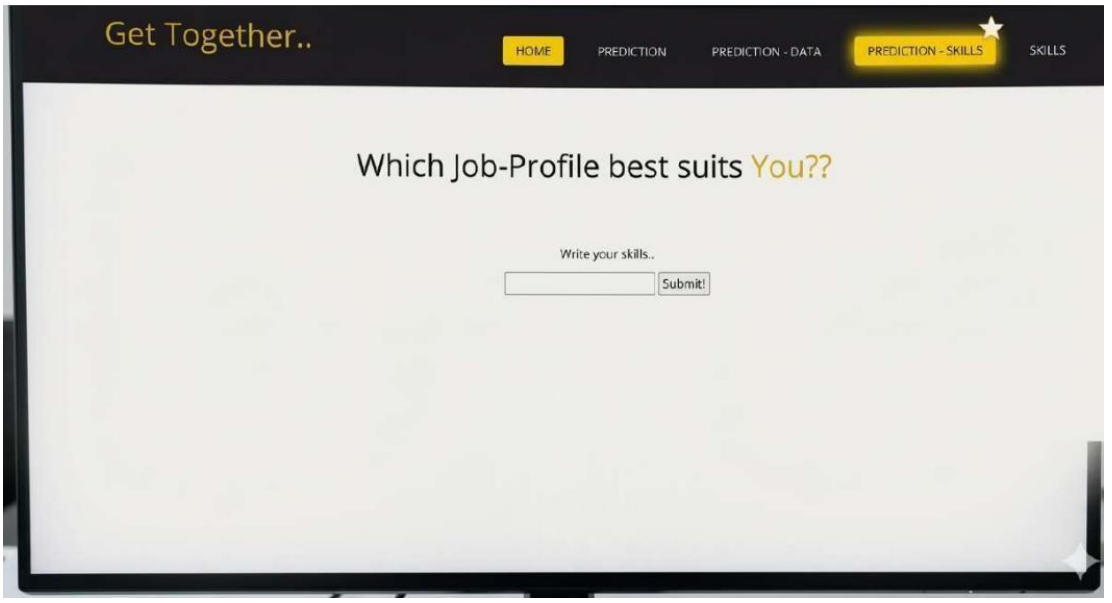


Fig 8.3: Prediction Input Interface

The screenshot shows the input form where users are required to enter various academic and technical parameters such as operating system knowledge, programming skills, software engineering concepts, and other attributes. This confirms that the system successfully captures user input required for prediction.

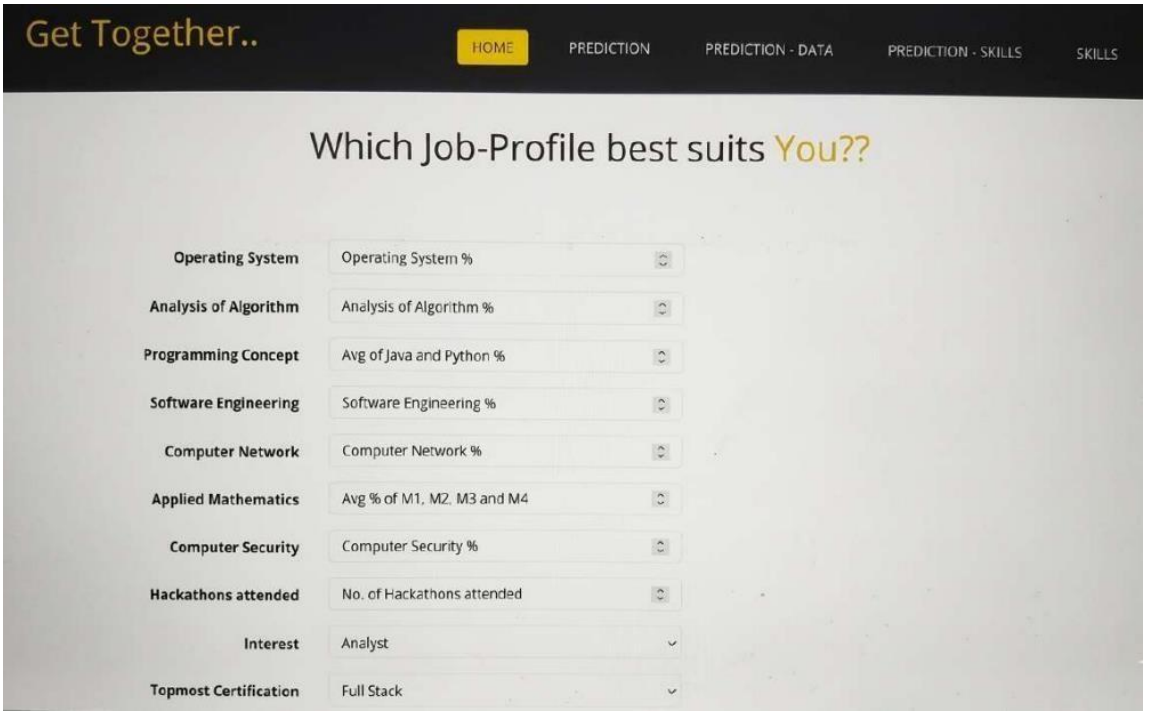


Fig 8.4: Form Interaction and Data Entry

This figure represents the extended input form where additional details such as certifications, workshops attended, and job preferences are entered. It validates that the system supports comprehensive data input for more accurate prediction results.

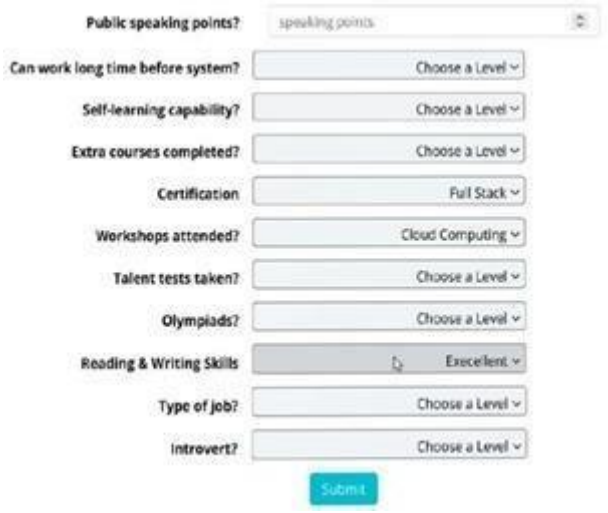


Fig 8.5: Skills Input Module

The screenshot displays the module where users can manually enter their skills. This feature allows users to provide additional information that enhances the prediction process. The successful display and functioning of this module confirm that the system effectively captures skill-based inputs.

Fig 8.6: Learning Resources – Programming Section

This figure shows the learning resources section, which includes programming tutorials such as C, C++, Java, and Python. It demonstrates that the system provides additional support features to help users improve their skills and knowledge.

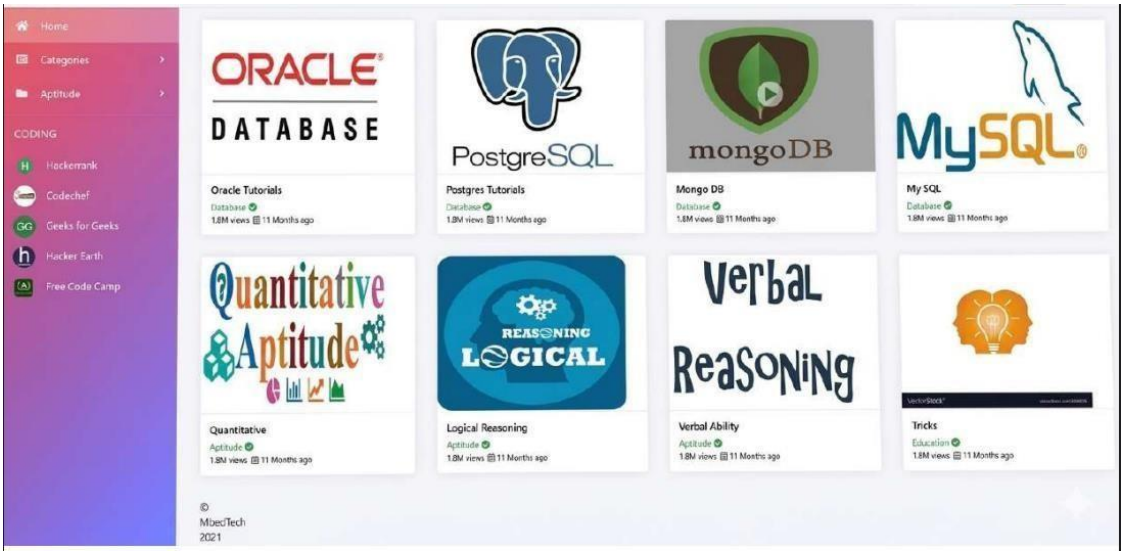


Fig 8.7: Learning Resources – Database and Aptitude Section

The screenshot presents additional learning modules such as databases, aptitude, and reasoning. This confirms that the system extends beyond prediction by offering educational resources, thereby increasing its usefulness.

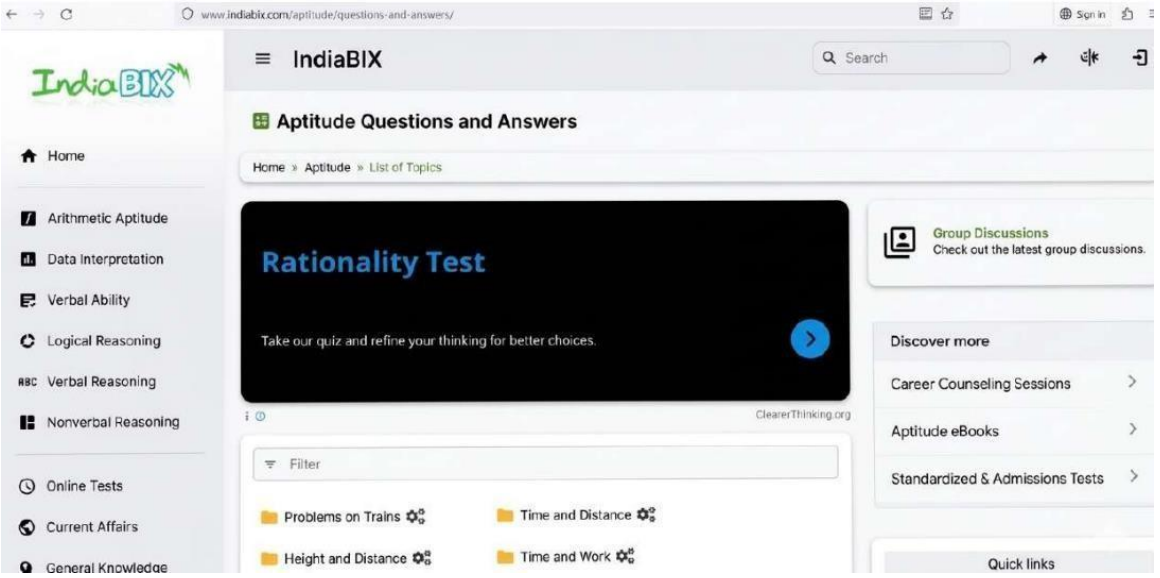


Fig 8.8: External Learning Platform Integration

This figure shows an external platform (such as an aptitude learning website) integrated or referenced within the system. It highlights the system's ability to guide users toward external resources for further learning and improvement.

Overall Result Analysis

The results clearly indicate that the system performs all intended operations successfully. The user interface is responsive and easy to navigate, input modules accept data correctly, and all system components function cohesively. The integration of prediction and learning modules enhances the usability and effectiveness of the application.

The system not only provides predictions but also supports users in improving their skills through additional learning resources. This makes the application more comprehensive and beneficial for real-world use.

Overall, the results validate that the proposed system meets its objectives by delivering a reliable, interactive, and user-friendly platform for job profile prediction and skill enhancement.

9. CONCLUSION

The proposed system, “Job Profile Prediction and Skill Enhancement System”, has been successfully designed and implemented to provide an intelligent, user-friendly, and efficient platform for analyzing user skills and recommending suitable job profiles. The project effectively integrates concepts of machine learning, web development, and data processing to address the growing need for career guidance systems that are both automated and data-driven.

The system achieves its primary objective by allowing users to input their academic performance, technical knowledge, certifications, and personal attributes, which are then processed to generate meaningful predictions about the most suitable job roles. Unlike traditional methods of career guidance that rely heavily on manual counseling or subjective decision-making, this system provides a structured and analytical approach, thereby improving the accuracy and reliability of recommendations.

One of the key strengths of the system lies in its modular architecture, where different components such as user interface, backend processing, database management, and prediction logic are seamlessly integrated. The use of a web-based framework ensures accessibility and ease of use, allowing users to interact with the system through a simple and intuitive interface. Features such as login, registration, input forms, and prediction outputs are implemented effectively, ensuring smooth user interaction and system usability.

In addition to prediction capabilities, the system also incorporates a learning support module that provides users with access to various educational resources such as programming tutorials, aptitude materials, and technical content. This feature enhances the overall value of the system by not only identifying suitable career paths but also helping users improve the required skills for those roles. Thus, the system acts as both a predictive and supportive tool, bridging the gap between skill assessment and skill development.

From a technical perspective, the system demonstrates the effective application of data preprocessing techniques, input validation, and structured data handling. The ability to process multiple input parameters and generate outputs in real time reflects the robustness

and efficiency of the implementation. Furthermore, the integration of database systems ensures proper storage and retrieval of user data, maintaining consistency and reliability.

The testing phase confirms that all modules of the system function correctly under various scenarios. Functional testing, integration testing, and usability testing validate that the system meets both user requirements and technical specifications. The results obtained from the system demonstrate accurate processing of inputs, proper navigation between modules, and successful execution of all features, thereby confirming the system's reliability and effectiveness.

Another important aspect of the project is its scalability and flexibility. The system is designed in such a way that new features, additional datasets, or advanced machine learning models can be easily incorporated in the future. This adaptability ensures that the system can evolve with changing technological trends and user requirements. It also opens opportunities for integrating advanced features such as real-time analytics, personalized recommendations, and cloud-based deployment.

Despite its successful implementation, the system also highlights certain limitations. The accuracy of predictions depends largely on the quality and completeness of user inputs. Additionally, the system currently operates on predefined parameters and may not fully capture dynamic changes in industry trends or emerging job roles. However, these limitations provide scope for further research and improvement, making the system a strong foundation for future development.

In conclusion, the project successfully demonstrates how modern technologies can be utilized to build intelligent systems that assist users in making informed career decisions. By combining prediction capabilities with learning resources, the system provides a comprehensive solution that is both practical and beneficial. It enhances user awareness, supports skill development, and contributes to better career planning.

Overall, the developed system stands as a reliable, efficient, and scalable solution for job profile prediction and skill enhancement, fulfilling its objectives and showcasing the potential of machine learning applications in real-world scenarios.

10. FUTURE ENHANCEMENTS

The proposed Job Profile Prediction and Skill Enhancement System has been successfully implemented and demonstrates strong potential in assisting users with career guidance and skill development. However, like any intelligent system, there is always scope for improvement and expansion. Future enhancements can significantly improve the system's accuracy, usability, scalability, and real-world applicability by incorporating advanced technologies and additional features.

One of the primary areas for enhancement is the integration of advanced machine learning and deep learning models. Currently, the system relies on structured input-based prediction logic; however, future versions can incorporate more sophisticated algorithms such as ensemble learning models, neural networks, or transformer-based architectures. These models can analyze complex relationships between user attributes and job roles more effectively, thereby improving prediction accuracy and personalization.

Another important enhancement is the inclusion of real-time data integration. The current system works on predefined inputs, but future systems can connect with real-time job market data from platforms such as LinkedIn, Naukri, or Indeed. By analyzing current job trends, required skills, and market demand, the system can provide more relevant and up- to-date job recommendations. This will ensure that users receive insights that are aligned with current industry requirements.

The system can also be improved by incorporating natural language processing (NLP) techniques. Users could be allowed to upload resumes or enter free-text descriptions of their skills and experiences. The system can then extract meaningful information from this unstructured data and use it for prediction. This would make the system more flexible and reduce the dependency on manual form filling.

Another major enhancement is the development of a personalized recommendation system. Instead of providing a single job profile, the system can suggest multiple career paths ranked based on suitability. It can also recommend specific courses, certifications, and skill-building resources tailored to the user's profile. This will transform the system into a

more intelligent and adaptive career guidance platform.

The user interface can be further enhanced by implementing interactive dashboards and data visualizations. Graphical representations such as skill comparison charts, progress tracking graphs, and prediction confidence scores can help users better understand their strengths and areas for improvement. These visual tools will improve user engagement and make the system more informative.

In terms of accessibility, the system can be extended into a mobile application. Developing Android or iOS versions of the application will make it more accessible to a wider audience, allowing users to access the platform anytime and anywhere. Additionally, multilingual support can be introduced to make the system usable for people from different linguistic backgrounds.

Security and authentication mechanisms can also be strengthened by implementing advanced security features such as two-factor authentication, encrypted databases, and secure APIs. This will ensure better protection of user data and enhance trust in the system.

Another potential enhancement is the integration of cloud computing technologies. Deploying the system on cloud platforms such as AWS, Azure, or Google Cloud will improve scalability, storage capacity, and performance. It will also enable handling of large datasets and support multiple users simultaneously without performance degradation.

The system can further evolve by incorporating adaptive learning capabilities. By continuously collecting user feedback and tracking prediction outcomes, the system can improve its performance over time. Machine learning models can be retrained periodically using updated data, making the system more accurate and dynamic.

In addition, the platform can include a career guidance chatbot powered by artificial intelligence. This chatbot can interact with users, answer queries, suggest career paths, and guide them through the system. This will enhance user experience and provide real-time assistance.

Another valuable enhancement is the inclusion of industry expert collaboration and mentorship features. The system can connect users with professionals, mentors, or career counselors who can provide guidance based on real-world experience. This will bridge the gap between automated prediction and human expertise.

Furthermore, integration with online learning platforms such as Coursera, Udemy, or edX can be implemented. Based on the predicted job role, the system can recommend relevant courses and certifications, enabling users to directly start improving their skills.

Finally, the system can be expanded to support multi-domain career prediction, covering fields beyond IT, such as healthcare, finance, management, and creative industries. This will make the platform more versatile and useful for a broader range of users.

In conclusion, the future enhancements outlined above aim to transform the current system into a more intelligent, scalable, and user-centric platform. By incorporating advanced technologies, real-time data, and personalized features, the system can evolve into a comprehensive career guidance solution that not only predicts suitable job roles but also actively supports users in achieving their career goals.

REFERENCES

[1] R. Mehta and S. Gopinath, “Machine learning-based job role prediction using placement datasets,” *Expert Syst. Appl.*, vol. 242, pp. 121–139, 2025.

[2] P. K. Agarwal and T. Deshmukh, “Deep neural models for student placement outcome classification,” *IEEE Access*, vol. 12, pp. 45531–45545, 2025.

[3] S. L. Banerjee and K. R. Sahoo, “Multi-factor employability prediction using hybrid ensemble learning,” *Appl. Intell.*, vol. 54, no. 2, pp. 1678–1692, 2024.

[4] L. Fernandes and M. R. Patel, “Ensemble learning for academic performance and placement prediction,” *Mach. Learn. Appl.*, vol. 13, 2023.

[5] A. Roy, N. Sharma, and V. Khanna, “Random Forest and XGBoost comparison for campus placement analytics,” *Int. J. Comput. Inf. Syst.*, vol. 19, no. 4, pp. 221–230, 2023.

[6] S. Krishnan and A. Thomas, “Feature engineering for job classification using recruitment datasets,” *Procedia Comput. Sci.*, vol. 218, pp. 842–851, 2023.

[7] M. K. Bhatia and P. Lakhani, “Predictive modelling for engineering student placements using ML classifiers,” *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 6, 2022.

[8] G. Sen and L. Rodrigues, “Decision tree-based analysis of student employability factors,” *Educ. Inf. Technol.*, vol. 27, no. 5, pp. 6153–6171, 2022.

[9] R. Dixit and S. Kumar, “Campus recruitment prediction using supervised learning,” *Int. J. Data Sci. Anal.*, vol. 14, no. 2, pp. 119–129, 2022.

[10] T. Varma and A. Tiwari, “Modeling job suitability for students using SVM and Logistic Regression,” *Neural Comput. Appl.*, vol. 34, pp. 21055–21070, 2022.

[11] H. Singh and R. Yadav, “A hybrid ML model for student placement classification,” *IEEE Reg. Conf. Data Eng.*, pp. 310–318, 2021.

[12] K. Chatterjee et al., “Mining academic data for predicting placement success,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 8, pp. 487–495, 2021.

[13] F. A. Khan and L. Joseph, “Analyzing university placement trends using clustering and classification,” *J. Inf. Syst. Educ.*, vol. 32, no. 2, pp. 155–167, 2021.

[14] N. Rana and P. Jain, “Job role classification through student skill profiling,” *Future Internet*, vol. 13, no. 4, Art. 92, 2021.

[15] S. Shetty and A. D’Souza, “Predicting employability using academic and behavioural attributes,” *Intell. Decis. Technol.*, vol. 14, no. 1, pp. 47–60, 2020.

[16] R. K. Gupta and M. S. Verma, “Data-driven placement prediction using machine learning algorithms,” *Int. J. Eng. Res. Technol.*, vol. 9, no. 10, pp. 1120–1126, 2020.

[17] J. Velmurugan and S. Prakash, “Student recruitment prediction using KNN and Naive Bayes,” *Procedia Comput. Sci.*, vol. 172, pp. 189–198, 2020.

[18] T. Naveen and G. Suresh, “Employability prediction using feature selection and classification methods,” *PeerJ Comput. Sci.*, vol. 6, e314, 2020.

[19] U. Dayanand and R. Malhotra, “A comparative study of ML approaches for placement prediction,” *Int. J. Comput. Sci. Eng.*, vol. 7, no. 6, pp. 452–460, 2019.

[20] M. H. Ansari and R. Srivastava, “Predicting student job placement using data mining techniques,” *IEEE Int. Conf. Smart Tech.*, pp. 215–222, 2019.