**MINI PROJECT REVIEW -3**

**College Placement Portal**

Submitted To: Submitted By:

Ms.Sreeshma Mohan Albin Sam Thomas(MZC23MCA-2003)

HOD, MCA Department MCA,S3

Mount Zion College of Engineering Batch: 2023-2025

Kadamanitta, Pathanamthitta.

Guide:

Ms.Sruthy Chandran

DATE: 04/10/2024

# **Abstract**

# The **College Placement Portal** is developed using Python with the Django framework, leveraging the Model-View-Template (MVT) architecture. The platform features user authentication and role-based access control, enabling students to register, view job listings, and apply for positions, while allowing admins to post and manage job opportunities. A streamlined approval system ensures that all listings receive admin oversight before publication. The project follows Agile development practices, facilitating continuous improvement and feature refinement. Future enhancements may include real-time notifications, advanced user analytics, and additional tools to enhance student-employer interactions.

# **1. Introduction**

# **1.1. About the project**

# The **College Placement Portal** is an essential platform that connects students with job opportunities, fostering career development and professional growth within the college community. Designed specifically for educational institutions, this web-based application enables students to easily access job listings, submit applications, and engage with potential employers. The portal serves as a centralized hub for students to explore internships, part-time, and full-time positions, while providing admins with tools to post and manage job opportunities effectively. This project aims to streamline the job search process, facilitate connections, and enhance employability for students. By creating a supportive environment for career exploration, the College Placement Portal contributes significantly to the overall success of students in their professional journeys.

# **1.2. Overview**

## The College Placement Portal aims to be an essential resource for students, enhancing their access to job opportunities while streamlining the placement process for educational institutions. Utilizing the Django framework, a robust and secure Python-based web framework, this platform is designed to provide a reliable and efficient user experience for both students and administrators. Django’s extensive set of tools and features ensures high performance and scalability, allowing the portal to accommodate a growing number of users and job postings without compromising responsiveness. By leveraging Django's capabilities, the College Placement Portal will facilitate seamless interactions between students and employers, ultimately supporting students in their career development and enhancing employability.

## **1.2.1 Modules**

1. **Admin Module**:
   * User Management: Admins have the authority to manage user accounts, including approving new registrations and addressing account-related issues.
   * Job Posting Management: Admins can create, edit, or delete job postings, ensuring that all listings are accurate and up to date.
   * **Application Management:** Admins can review applications submitted by students, tracking their statuses and providing feedback where necessary.
   * Notifications and Announcements: Admins can send notifications and announcements to students, keeping them informed about new job opportunities, deadlines, and important updates.
2. **User Module:** 
   * User Registration and Authentication: Students can register and log in using their email IDs, ensuring that only verified users can access the platform.
   * Profile Management: Students can create and manage their profiles, including personal information, resumes, and application history.
   * Job Search and Application: Students can browse job listings, view details, and apply directly through the portal, attaching necessary documents.
   * Application Tracking: Users can track the status of their job applications and receive updates on any changes.

**1.2.2. Technology Stack:**

* Backend: Python with Django Framework
* Frontend: HTML, CSS, JavaScript
* Database: SQLite (development)
* Hosting: Any suitable cloud service provider

## **1.2.3 Languages Using**

* Backend: Python with Django Framework
* Frontend: HTML, CSS, JavaScript
* Database: SQLite (development)

# **1.3. Scope of the Project**

# The scope of the College Placement Portal project is to create a secure, user-friendly web-based application designed to streamline the placement process for college students and administrators. The platform will allow for posting and managing job opportunities, enable students to apply for placements, and provide real-time tracking of application statuses. By leveraging the Django framework, the portal will ensure scalability, security, and efficiency, ultimately replacing traditional manual processes, enhancing student engagement, and simplifying overall placement management. Additionally, the portal aims to foster better communication between students and administrators, creating a more transparent and accessible placement experience.

# **2. System Analysis**

# **2.1. Introduction**

The College Placement Portal is designed to address the need for an efficient system to manage the placement process for college students and administrators. The system analysis focuses on identifying the requirements, functionalities, and constraints needed to develop a secure and user-friendly platform that facilitates job postings, student applications, and communication between stakeholders, ensuring a seamless and effective placement experience.

**Functional Requirements:**

* **User Management:** The platform requires a user authentication system, allowing students and administrators to register, log in, and manage their profiles.
* **Admin Functions:** Administrators can create, update, and delete job postings. Administrators can view and manage student applications, including status updates.
* **Student Functions:** Students can browse and filter job postings based on criteria such as location, company, and eligibility. Students can create and edit their profiles, including uploading resumes and cover letters.
* **Search and Filter Functionality:** search feature to allow students to find relevant job opportunities easily. Filters to narrow down job postings based on various parameters like industry, job type, and deadlines.
* **Application Management:** Students should have a dashboard to view their applied positions, along with statuses such as pending, interview, or rejected. Options for students to withdraw applications if needed.

**Non-Functional Requirements:**

* **Performance:** The portal should provide quick response times for all operations, ensuring that job postings, application submissions, and status updates are processed within seconds.
* **Scalability:** The system must be scalable to accommodate increasing numbers of users, job postings, and applications as the college community grows.
* **Security:** Strong security protocols should be implemented to ensure the confidentiality, integrity, and availability of user data

**Constraints:**

* **Technological:** The system is built using Django, meaning that all functionality must align with Django's capabilities and limitations.
* **Scalability:** The system may need to scale up to accommodate an increasing number of students and job postings, which could be constrained by available hardware or limitations in the current system architecture.

**Stakeholders:**

* **Students:** The primary users of the portal, who will use it to browse, apply for, and track the status of placement opportunities.
* **Placement Officers (Administrators):** Placement officers manage the entire placement process, including posting job opportunities, viewing student applications, and communicating with students.
* **College Management:** College management may use the platform for insights into placement statistics, such as student participation rates and employer engagement.

# **2.2. Existing System**

In many educational institutions, the existing placement management system relies heavily on traditional methods such as in-person meetings, paper-based records, notice boards, and mass email communication. These methods have significant limitations in terms of efficiency, scalability, and transparency.

**Key Characteristics:**

1. **Manual and Paper-Based Processes:**
   * **Limited Efficiency:** Managing records using paper-based documents or spreadsheets is time-consuming and error-prone.
   * **Difficult to Update:** Keeping records up-to-date manually can lead to inconsistencies and missed information.
   * **Data Loss Risks:** Paper records are at risk of loss or damage, compromising data integrity.
2. **Fragmented Communication Channels:**
   * **Inconsistent Information Flow:** Relying on emails, notice boards, or in-person meetings can result in information gaps or delays.
   * **Missed Opportunities:** Students may miss important announcements due to lack of centralized communication.
3. **Limited Accessibility:**
   * **Physical Presence Required**: Students must be on campus to access updates, limiting accessibility for remote users.
   * **Inflexible Access Hours:** Information is only available during office hours, reducing convenience.
4. **Lack of Real-Time Updates:**
   * **Delayed Information:** Updates on job postings or application statuses are not immediate, leading to uncertainty.
   * **Lack of Notification Mechanisms:** No automated notifications for students about important events or updates.

# **2.3. Proposed System**

The proposed College Placement Portal is a modern, web-based solution designed to overcome the limitations of traditional placement management methods in educational institutions. This portal will provide a secure, scalable, and user-friendly platform where students can easily explore job opportunities, apply for placements, and track their applications in real time. Placement officers will have a centralized system to post job openings, manage applications, and communicate efficiently with students, ensuring a streamlined and transparent placement process for all stakeholders.

**Key Features:**

1. **User Management:**
   * **Authentication:** Secure login and registration processes for students and administrators to ensure data integrity.
   * **Role-Based Access:** Students can create and manage their profiles and applications, while administrators oversee job postings and user management.
2. **Job Posting Management:**
   * **Posting Opportunities:** Placement officers can easily create, edit, and delete job postings, ensuring timely updates to students.
   * **Approval Workflow:** New job postings will go through an admin approval process to ensure accuracy and relevance before being made visible to students.
3. **Application Tracking:**
   * **Real-Time Updates:** Students can track the status of their applications, receiving notifications about changes in their application status.
   * **Dashboard:** A user-friendly dashboard for students to manage their applications and view all active opportunities in one place.
4. **Scalability** **and Performance:**
   * **Efficient Resource Management:** The system will be designed to efficiently handle an increasing number of users and job postings without compromising performance.

**Benefits:**

* **Enhanced User Experience:** A user-friendly interface and intuitive navigation make it easy for students and placement officers to access information and manage applications efficiently.
* **Improved Communication:** Integrated messaging and notification systems facilitate timely communication between students and placement officers, reducing misunderstandings and enhancing engagement.
* **Centralized Management:** Centralized platform for job postings and applications simplifies management for placement officers, allowing them to efficiently oversee the placement process.

# **3. System Design**

**3.1 DFD (Data Flow Diagram)**

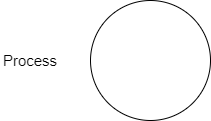
A Data Flow Diagram (DFD) is a graphical representation that illustrates how data moves within a system. It focuses on the flow of information between processes, data stores, and external entities. DFDs are commonly used in systems analysis to map out the flow of data in a clear, concise manner.

**Key Components of a DFD:**

1. **External Entities (Sources/Sinks):**
   * These represent the entities outside the system that interact with it. Examples include users (students, administrators), or external systems.
   * **Symbol:** Rectangles or squares.



1. **Processes:**
   * Processes represent actions or operations that transform incoming data into outgoing data. Each process is a function that takes inputs, processes them, and produces outputs.
   * **Symbol:** Circles or rounded rectangles.



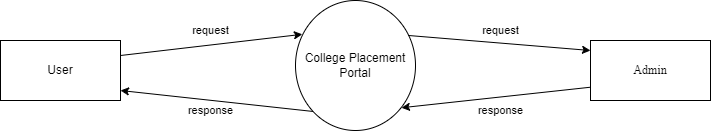
1. **Data Stores:**
   * Data stores represent where data is stored within the system, such as databases or files. They serve as repositories where data can be held for later use by processes.
   * **Symbol:** Open-ended rectangles.



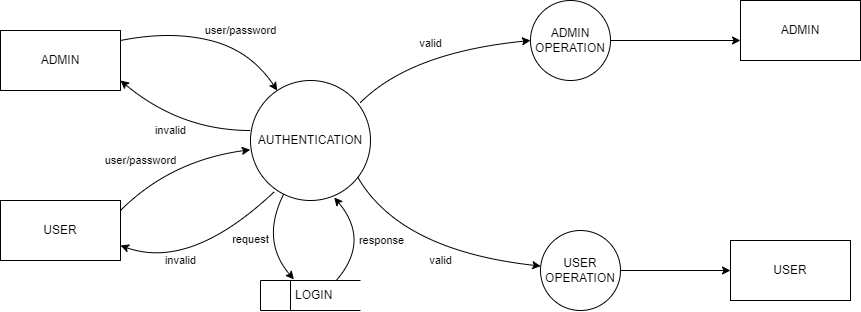
1. **Data Flows:**
   * Data flows are the pathways through which data moves between processes, data stores, and external entities. They show the direction and type of data being transferred.
   * **Symbol:** Arrows.



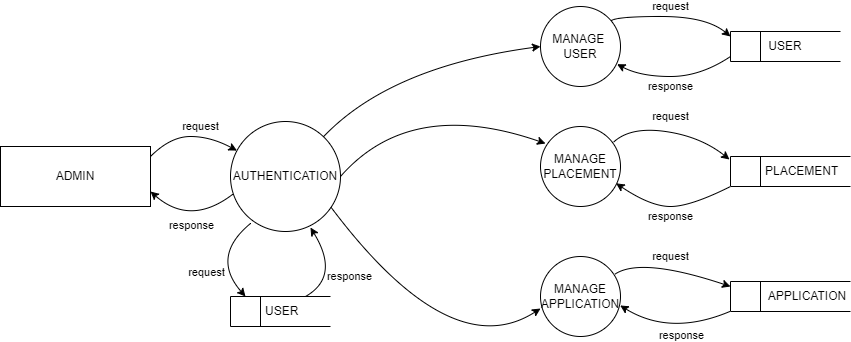
# Level 0



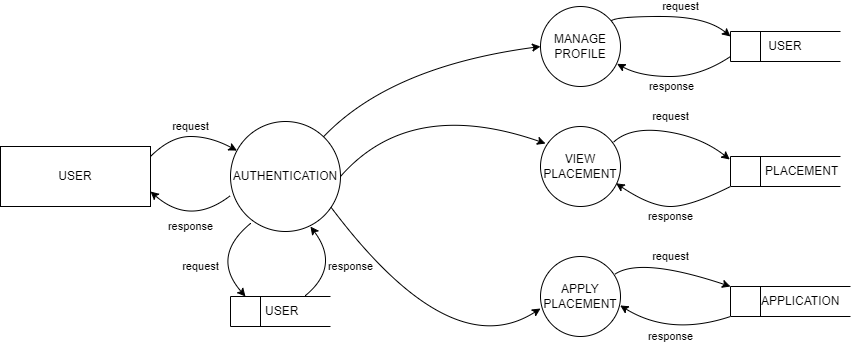
# Level 1



# Level 1.1



# Level 1.2



# **3.2. Database Design**

# Database Name: Placement

Tables: User, Placement Post, Application

# **3.3. Table Design**

\*Primary Key

1) Tbl\_login

Primary key:id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype(Size)** | **Key Constraints** | **Description of the field** |
| 1 | Id | Int(20) | Primary Key | Login id |
| 2 | Email | Varchar(15) | Not Null | Email id |
| 3 | password | Varchar(15) | Not Null | Password |

2) **Tbl\_user\_registration**

Primary key: email

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype(Size)** | **Key Constraints** | **Description of the field** |
| 1 | Id | Int(20) | Not Null | User id |
| 2 | email | Varchar(15) | Primary Key | Email id |
| 3 | password | Varchar(15) | Not Null | Password |
| 4 | Name | Varchar(15) | Not null | Full Name |
| 5 | Age | Int(20) | Not null | Age |
| 6 | Education | Varchar(15) | Not null | Education |
| 7 | Batch | Int(20) | Not null | Batch |
| 8 | Skill | Varchar(15) | Not Null | Interested skill |
| 9 | Role | Boolean(2) | Not Null | Role |
| 10 | date | Date() | Not Null | Date |

**3) Tbl\_placement**

Primary key:id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field Name** | **Datatype(Size)** | **Key Constraints** | **Description of the Field** |
| 1 | Post\_id | Int(20) | Primary Key | Post id |
| 2 | Title | Varchar(15) | Not Null | Post Title |
| 3 | Content | Text(50) | Not null | Content |
| 4 | Image | Image | Not null | Placement Image |
| 5 | Skill | Varchar(15) | Not null | skill |
| 6 | Created\_at | Date() | Not null | Created date |
| 7 | End\_date | Date() | Not null | End date |

4) **Tbl\_application**

Primary key:id

Foreign key:User\_id

Foreign key: Post\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Datatype(Size)** | **Key Constraints** | **Description of the Field** |
| 1 | Id | Int(20) | Primary key | id |
| 2 | User\_id | Int(20) | Foreign Key | User id |
| 3 | Post\_id | Int(20) | Foreign Key | Post id |
| 4 | Name | Varchar(15) | Not Null | Full Name |
| 5 | Date | Date() | Not Null | Date |
| 6 | Cover letter | Text(100) | Not Null | Cover letter |
| 7 | CV | Varchar(15) | Not Null | CV |

# **3.4. Normalization**

Normalization is a process used in database design to minimize redundancy and dependency by organizing data into tables. The main goal is to divide large tables into smaller, related tables and link them using relationships. This helps to ensure data consistency, reduce data redundancy, and improve data integrity.

**Types of Normalization:**

1. **First Normal Form (1NF):**
   * Ensures that the table has a primary key.
   * All attributes (columns) contain only atomic (indivisible) values.
   * Each record (row) in the table is unique.
2. **Second Normal Form (2NF):**
   * Achieves 1NF.
   * Removes partial dependency, where an attribute depends only on a part of a composite primary key (in cases where the primary key consists of more than one attribute).
3. **Third Normal Form (3NF):**
   * Achieves 2NF.
   * Removes transitive dependency, where non-key attributes depend on other non-key attributes.
4. **Boyce-Codd Normal Form (BCNF):**
   * A stricter version of 3NF.
   * Ensures that for every functional dependency A→BA \ BA→B, AAA is a super key.
5. **Fourth Normal Form (4NF):**
   * Achieves BCNF.
   * Eliminates multi-valued dependencies, where one attribute in a table is allowed to have multiple values independently of other attributes.
6. **Fifth Normal Form (5NF):**
   * Achieves 4NF.
   * Decomposes tables to eliminate any remaining anomalies that could arise from complex dependencies.

# **3.5. Normalization Analysis of Tables**

**1. User Table**

* **1NF:** The User table is in 1NF because all fields are atomic (e.g., email, password, full\_name). Each user has a unique primary key (id), ensuring that each record is unique.
* **2NF:** The User table is in 2NF because it does not have a composite primary key; therefore, there are no partial dependencies. All non-key attributes depend on the primary key (id).
* **3NF:** The User table is in 3NF because all non-key attributes (e.g., email, password, full\_name, age, department, batch, interested\_skill, role) directly depend on the primary key (id), with no transitive dependencies.

**2. Placement Post Table**

* **1NF:** The Placement Post table is in 1NF because all fields are atomic (e.g., title, content, picture). Each post has a unique primary key (id), ensuring that each record is unique.
* **2NF:** The Placement Post table is in 2NF because it does not have a composite primary key; thus, all attributes depend on the primary key (id).
* **3NF:** The Placement Post table is in 3NF because all non-key attributes (e.g., title, content, picture, created\_at) directly depend on the primary key (id), with no transitive dependencies.

**3. Post Table**

* **1NF:** The Application table is in 1NF because all fields are atomic (e.g., user\_id, post\_id, cover\_letter). Each application has a unique primary key (id), ensuring that each record is unique.
* **2NF:** The Application table is in 2NF because it does not have a composite primary key; therefore, all non-key attributes depend on the primary key (id).
* **3NF:** The Application table is in 3NF because all non-key attributes (e.g., user\_id, post\_id, date, cover\_letter, cv) directly depend on the primary key (id), with no transitive dependencies.

## **4. Software Requirements**

**4.2.1 Hardware Specification**

Processor –intel corei3

RAM -4 GB

Hard disk -1 TB

**4.2.2 Software Specification**

Front End - HTML, CSS, JAVASCRIPT

Backend – python Django

Client on PC - WINDOWS 7 & Above

Technologies used- HTML, DJANGO, CSS

# **5. Conclusion**

The College Placement Portal provides an efficient and streamlined solution for managing the placement process, connecting students with valuable career opportunities. By leveraging the Django framework, the portal ensures a secure, scalable, and user-friendly experience, making it an essential tool for both students and administrators. The platform not only simplifies job postings and applications but also enhances transparency and communication throughout the placement process. Ultimately, this project enriches the overall placement experience, supporting students in achieving their career aspirations and helping colleges effectively manage their placement activities.

**6. Reference**

[1] https://www.djangoproject.com/, 03/10/2024,05:30 pm

[2] https://nevonprojects.com.com/, 03/10/2024,05:40 pm

[3] https://www.drawio.com/, 03/10/2024,05:45 pm