###### Project Report of

**XploreIntern**

**Submitted in partial fulfillment of the requirement for the award of the degree of**

**MASTER of COMPUTER APPLICATION**

**(University of Calicut)**

Submitted by

#### VIDYA V PDAXMCA015

**ASWIN M PDAXMCA016** **KIRAN K PDAXMCA017** **PADMASEKHAR M PDAXMCA019**



**2023-2025 Academic Year**

### C C S I T

**MARUTHAROAD, PALAKKAD**



#### UNIVERSITY OF CALICUT

**C C S I T**

##### MARUTHAROAD, PALAKKAD

**MASTER of COMPUTER APPLICATION**

#### CERTIFICATE

**This is to certify that the project report entitled “XploreIntern“ is a bonafide record**

**done by Ms.VIDYA V, Mr.ASWIN M, Mr.KIRAN K,Mr.PADMASEKHAR M, during**

**the academic year 2022-2024 towards the partial fulfillment of the requirements**

**for the award of MASTER OF COMPUTER APPLICATION of the University of Calicut.**

**................................. .................................**

**Project Guide** **Ms.SHAKEELA H**

**Assistant Professor** **Department of Computer Application**

**Head of Department** **Mrs. LINTA VENUGOPAL**

**Department of Computer**

**Application**

#### DECLARATION

We, **MS.VIDYA V (Register No: PDAXMCA015), Mr. ASWIN M (Register No: PDAXMCA016), Mr. KIRAN K (Register No: PDAXMCA017), Mr. PADMASEKHAR M (Register No: PDAXMCA019) hereby declare that the project work entitled “TRAVELOG” is developed at C C S I T, MARUTHAROAD, PALAKKAD and submitted to the university of Calicut, in partial fulfillment of the requirements for the award of the degree of MASTER of COMPUTERAPPLICATION, is a record of original project work done by us from 2023 to 2025, under the supervision and guidance of Ms. SHAKEELA H, Assistant Professor, Department of Computer Application, C C S I T, MARUTHAROAD, PALAKKAD.**

Place: Marutharoad Signature of the Candidate: Date:

**VIDYA V**

**ASWIN M**

**KIRAN K**

**PDAMASEKHR M**

#### ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to all respected personalities who guided, inspired, and helped us successfully complete this project. First and foremost, we express our thanks to The Lord Almighty for guiding us in this endeavor and making it a success.

We would like to extend our thanks to the Associate Coordinator, Mrs. Linta Venugopal C C S I T Palakkad, for providing us with the best facilities and atmosphere, according to our interests, for the successful completion and presentation of our project.

We would most like to give our sincere gratitude to Mrs.Shakeela guide of our project for all the valuable advice and support given to us. We would also like to extend our sincere thanks to Mrs.Jyothi, and Mrs. Praseeja of the Master of Computer Application department for their support & guidance.

Finally, we express our everlasting thanks to all our friends whose valuable suggestions, criticism, and encouragement helped to make this project a great success. We also thank our parents for their prayers through which our project was completed.

#### ABSTRACT

An internship portal is a comprehensive online platform designed to bridge the gap between students seeking internship opportunities and companies offering them. This portal streamlines the entire internship application process, making it easier for students to discover and apply for positions that match their skills and career goals. It allows companies to post openings, track applications, and connect with prospective interns efficiently. Key features of the portal include personalized dashboards for students and recruiters, search filters for specific industries or skills, application tracking, and automated notifications. Additionally, the portal may provide resources such as resume-building tips, interview preparation, and access to networking events, enhancing the overall internship experience for students and helping companies identify qualified candidates.

An internship portal uses a combination of technologies to provide a smooth and interactive platform for students and recruiters. Django manages the backend, handling data, user authentication, and routing, while HTML structures the content displayed on each page. CSS and Bootstrap add visual styling and responsive design, ensuring the portal looks professional and functions well on all devices. JavaScript adds interactivity, enabling features like real-time form validation, dynamic content loading, and models for a seamless user experience. Together, these technologies create an efficient platform for students to find internships and for recruiters to post opportunities.

**CONTENTS**

1. [INTRODUCTION 9](#_bookmark0)
2. [SYSTEM CONFIGURATION 11](#_bookmark1)
   1. [HARDWARE CONFIGURATION 11](#_bookmark2)
   2. [SOFTWARE CONFIGURATION 11](#_bookmark3)
   3. [USER REQUIREMENTS 11](#_bookmark4)
      1. FRONT END SOFTWARE 11
      2. BACK-END SOFTWARE 13
3. [SYSTEM DEVELOPMENT ANALYSIS 15](#_bookmark5)
   1. OBJECTIVE OF THE SYSTEM 15
   2. [EXISTING SYSTEM 15](#_bookmark6)
   3. [PROPOSED SYSTEM 16](#_bookmark7)
4. [SYSTEM DESIGN 18](#_bookmark8)
   1. [INTRODUCTION 18](#_bookmark9)
   2. [INPUT DESIGN 18](#_bookmark10)
   3. [OUTPUT DESIGN 19](#_bookmark11)
   4. [DATABASE DESIGN 19](#_bookmark12)
   5. [MODULE DESCRIPTION 20](#_bookmark13)
5. [SYSTEM IMPLEMENTATION AND TESTING 22](#_bookmark14)
   1. SYSTEM IMPLEMMENTATION 22
   2. [SYSTEM TESTING](#_bookmark15) 22
      1. [TYPES OF TESTING 23](#_bookmark16)
6. [CONCLUSION 26](#_bookmark17)
7. BIBLOGRAPHY 27

##### APPENDICES 28

##### ER DIAGRAM 29

##### DATA FLOW DIAGRAM 30

##### USE CASE DIAGRAM

##### TABLES USED 34

##### 8.4. SAMPLE CODE 39

##### 8.5. SAMPLE SCREENSHOTS 56

# INTRODUCTION

### INTRODUCTION

An internship portal is an online platform designed to connect students seeking real-world experience with companies offering internship opportunities. It serves as a bridge, making it easier for students to explore, apply to, and track internship positions that align with their skills and career goals. For recruiters, the portal provides an efficient way to post job listings, manage applications, and interact with prospective interns. With features such as search filters, personalized dashboards, application tracking, and notifications, the portal streamlines the entire process. This helps students gain valuable work experience while enabling companies to identify and recruit qualified talent.

In an internship portal, Django, HTML, CSS, Bootstrap, and JavaScript work together to create a smooth and interactive experience for both students and recruiters. Django forms the backend, managing data, user authentication, and routing to ensure secure and efficient handling of applications and listings. HTML provides the structure for each page, while CSS styles the portal with a consistent look and feel. Bootstrap adds responsiveness, allowing the portal to function seamlessly on various devices with a polished, professional design. JavaScript enhances interactivity with real-time validation, dynamic content updates, and models, making the platform intuitive and easy to use. Together, these technologies create a user-friendly interface that enables students to search and apply for internships and helps recruiters manage listings and applications effectively.

# SYSTEM CONFIGURATION

### SYSTEM CONFIGURATION

##### HARDWARE CONFIGURATION

* System : MVC (Model-View-Controller)
* Hard Disk : 500 GB
* Ram : 4GB

##### SOFTWARE CONFIGURATION

* Operating system : Windows 10/11, Linux, or MacOS
* Tools : Visual Studio Code
* Programming Language : Python 3.8 or above
* Framework : Django 3. x or above
* Database : MySQL 8.0
* Version Control : Git
* IDE : Visual Studio Code, PyCharm,

or any compatible IDE

##### USER REQUIREMENTS

* + 1. **FRONT END**

HTML, CSS, JavaScript, Bootstrap.

##### BACK END

Django

##### DATABASE

SQLite

##### HTML

HTML is used to create and save web documents. Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. E.g., Notepad/Notepad++, VS code.

##### CSS

Cascading Style Sheets (CSS) Create an attractive Layout. Cascading Style Sheets **(**CSS**)** is a style sheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, Math ML, or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, and enable multiple web pages to share formatting by specifying the relevant CSS separately. CSS file, and reduce complexity and repetition in the structural content. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

###### JavaScript

It is a programming language, commonly used with web browsers. JavaScript often abbreviated as JS, is an interpreted programming language that conforms to the ECMA Script specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a

Dedicated JavaScript engine to execute it. Originally used only in web browsers, JavaScript engines are also now embedded in server-side website deployments and non-browser applications. Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

###### Bootstrap

It is a responsive site design commonly used with web browsers. Bootstrap is a framework whose primary purpose of adding it to a web project is to apply Bootstrap’s choices of color, size, and font choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables, and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their content. For example, Bootstrap has provisioned light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

* + 1. **BACK END:** DJANGO

###### DJANGO

Django is a high-level Python web framework that enables developers to build robust and scalable web applications quickly and efficiently. It was designed to promote rapid development, clean and pragmatic design, and reusability of components. Django follows the Model-View-Template(MVT) architectural pattern, which is similar to the more commonly known Model-View-Controller(MVC). The framework comes with a built-in admin interface, which is an incredibly powerful feature for managing application data. Django encourages the use of reusable and modular components, offering a wide range of tools such as authentication, URL routing, form handling, and ORM (Object-Relational Mapping) for database management. Its key strengths include its extensive documentation, strong community support, and security features like protection against SQL injection, cross-site scripting(XSS), and cross-site request forgery(CSRF). Additionally, Django follows the “don’t repeat yourself” (DRY) principle, which minimizes code duplication and enhances maintainability.

Whether or a complex, data-driven application, Django provides an extensive suite of features to handle common web development challenges.

###### SQLite

SQLite is a lightweight, serverless, self-contained relational database engine that is widely used in embedded systems and applications. Unlike other database management systems that require a dedicated server process, SQLite operates directly from disk files. It is a zero-configuration database, meaning that it doesn’t require setup, server management, or installation processes. This makes it ideal for small to medium-sized applications where simplicity and portability are prioritized.

SQLite is particularly useful for applications that need to store local data, such as desktop applications, mobile apps, and even web applications with limited database usage. It supports the majority of SQL standard features, such as transactions, joins, and indexes, but is optimized for simplicity, ease of use, and performance with small to medium datasets.

In Django projects, SQLite is often used as the default database backend because it requires no setup and is suitable for development and testing purposes. It stores all data in a single file, which makes it easy to manage and migrate between environments. For production applications, however, developers typically migrate to more robust database systems like PostgreSQL or MySQL to handle larger datasets and more complex requirements.

# SYSTEM DEVELOPMENT ANALYSIS

### SYSTEM DEVELOPMENT ANALYSIS

##### OBJECTIVES OF XPLOREINTERN

The main objective of an internship portal is to connect students and recent graduates with internship opportunities that align with their career goals, providing a streamlined platform for employers and candidates to interact.

* Facilitate Internship Search: provide students with easy access to a broad array of internships across various fields.
* Efficient Application Management: Enable employers to post internship opportunities, review applications, and update application statuses.
* Dashboard-Based Interaction: Offer customized dashboards for students and employers to manage their respective needs efficiently.
* Automated Notifications: Send updates to students to students regarding application status and upcoming deadlines.

##### EXISTING SYSTEM

Currently, students and employers often rely on separate platforms like job boards, academic portals, or independent career centers, which result in:

* Limited Internship Options: Students have access only to a small pool of internship opportunities.
* Manual Tracking: Employers must manually track applicant details, often using spreadsheets or external tools.
* Communication Delays: Without a unified system, students, and employers face delays in receiving or updating application statuses.

##### PROPOSED SYSTEM

The proposed system for XploreIntern is straightforward in both design and implementation, ensuring ease of use and compatibility with low system resource requirements. XploreIntern is a dedicated internship portal that will streamline the internship application and management process for both students and employers. The system is designed to operate efficiently across most standard system configurations, making it highly accessible and affordable.

It has the following features:

* Data Accuracy: Ensures that all information provided by users is accurate and up to date through validation and integrity checks within the system.
* Efficient Data Management: User and internship records are managed effectively using a database management system (DBMS), ensuring that data is stored, retrieved, and updated efficiently.
* Internship Status Tracking: Students can easily track the status of their internship applications, allowing for clear and timely updates on their application process.
* Application Withdrawal and Management: Students have the option to withdraw applications, and employers can accept or reject applications seamlessly, all in one place.
* Minimal Processing Time: The platform is optimized to reduce the time required for processing various actions, such as applying to internships or updating application statuses.
* Improved Service: By offering specialized functionalities for students, employers, and administrators, XploreIntern provides a user-centered experience, making the process more convenient, transparent, and effective.

# SYSTEM DESIGN

**SYSTEM DESIGN**

##### INTRODUCTION

Design concepts provide the basic criteria for design quality. Design is the meaningful representation of something to build. Design focuses on the three major areas of concern: data, architecture, and interface beginning once the software requirements have been analyzed and specified, software design is the first of three activities - design code generation and test. Each activity transforms information in a manner that ultimately results in validated computer software. Design is the first step in moving from the problem domain towards the solution domain. The detailed design phase. This can be achieved by:

* + - Input design.
    - Output design
    - Database design

##### INPUT DESIGN

Input design is a crucial part of the system design process where user inputs are gathered and converted into a computer-readable format. In the XploreIntern platform, the main objective of input design is to ensure that users can easily and accurately provide the necessary information, such as registration details, job applications, and profile updates. A well-designed input system simplifies the user experience and minimizes errors in data entry.

Since XploreIntern is designed for both employers and students, forms and other input mechanisms are used extensively throughout the system. The input forms are designed in a way that makes it easy for users to interact with the system while providing all the necessary information. The main goal of the input design is to ensure the form is simple, accessible, and user-friendly.

The forms used in XploreIntern serve as the primary interface for users to interact with the system. The design decisions behind the input forms are based on the following user requirements and guidelines:

Specifications and Requirements for XploreIntern**:**

* Interaction window should be user-friendly: The user interface is designed to be intuitive, ensuring that users can navigate through the system effortlessly.
* Forms should be easy to operate: The forms should be simple to fill out with minimal steps and clear labels for each field.
* Proper validations should be in place: To prevent user errors, all inputs must be validated before they are submitted. For example, email fields must follow the correct format, passwords must meet security criteria, and required fields should not be left empty.
* Form design should be clear: Each form must specify the data expected from the user. This includes displaying placeholder text, tooltips, and proper field labeling to guide users during data entry.

##### OUTPUT DESIGN

Output design is a critical aspect of any system, as it determines how data is presented to the user and the form in which it is delivered. In the XploreIntern platform, output design ensures that information is provided in a meaningful and easy-to-understand format. This includes job listings, internship applications, student profiles, and other important system-generated reports. The objective of output design is to ensure that users receive the correct information in a timely and well-organized manner.

Output design involves identifying the type of outputs needed, understanding the user requirements for those outputs, and considering factors such as clarity, format, and mode of delivery. It plays a crucial role in providing relevant and accessible information to end users, whether it's students, employers, or administrators.

Objectives of Output Design:

* + - To develop output design that serves the intended purpose and eliminates the production of unwanted.
    - To develop the output design that meets the end user’s requirements.
    - To deliver the appropriate quantity of output.
    - To form the output in an appropriate format and direct it to the right person.
    - To make the output available on time for making good decisions.

##### DATABASE DESIGN

The data in the system has been stored and retrieved from the database. Designing the database is a part of system design whether it is a collection of interrelated data stored with minimum redundancy to save quickly and efficiently. The main aim is to make database access quick and easy. Database design of the system deals with the relevant data that come into play in the system. According to the relationship of data, tables are designed by allowing the standard database design method. The data type of each data is defined for the optimum design of the database to have better response time, maintain data integrity, avoid redundancy, serve many uses, and be quick and efficient. The general objective is to make information access easy, quick, inexpensive, and flexible for the user. Database design is the most critical path of the design phase- elegant design, A well-defined database is a strong foundation for the whole system files in a relational database are called tables. Columns of the table represent data and rows represent the records in conventional technology.

##### MODULE DESCRIPTION

The XploreIntern system comprises three major modules, each with its sub-modules that address the core functionalities of the platform. These modules are categorized for Employers and Students, with an Administrator module for overall system management.

**Administrator:**

The Administrator is the key entity responsible for managing the entire platform. The administrator has the highest level of access and control over all system functionalities. The admin ensures the smooth operation of the platform by overseeing user registrations, job postings, student applications, and other critical tasks.

**Employer Module:**

The Employer module is designed for companies or businesses that offer internships. Employers can register their organizations, post internships, and manage applications received from students. Employers can register their companies by providing essential details such as company name, location, and type of internships offered. After registration, employers gain access to the platform’s features.

**Student Module:**

The student module is designed to cater to the needs of individuals seeking internships. Students can register on the platform, apply for internships, and track the progress of their applications. Students can register themselves on the platform by providing necessary details such as name, university, and course. They must also upload their resume and other documents as required by the employers. The dashboard provides an overview of the student's profile, the internships they have applied to, and any updates or notifications regarding their application statuses.

# SYSTEM IMPLEMENTATION AND TESTING

**SYSTEM IMPLEMENTATION AND TESTING**

Implementation is a stage of theoretical design that is turned into working the system. The implementation phase is used to test the development package with sample data, correct the error identified, and make the user aware of the various special facilities and features of the computerized system. It also involves user training to minimize resistance to change and give the new system a chance to prove its worth: The successful implementation of the new system depends upon the involvement of the user.

##### SYSTEM IMPLEMENTATION

The implementation phase is the phase, which involves the process of converting a new system design into one operational one. It is the key stage in achieving a successful new system. Implementation is the stage of the project, where the theoretical design is turned into a working system. At this stage, the main workload, the greatest upheaval, and the major impact on existing practices shift to the user department. If the implementation stage is not planned and controlled carefully it can cause chaos. Thus, it can be considered to be the more crucial stage in achieving a successful new stage and in giving the user confidence that the system will work and will be effective.

**IMPLEMENATATION PROCEDURE**

The development of XploreIntern followed a modular approach, dividing the system into front-end, back-end, and database components to facilitate parallel development. The environment was set up using Django as the framework and MySQL for the database. Responsive web pages were designed with HTML, CSS, and Bootstrap, while the backend was developed to handle core business logic and database interactions through Django views, models, and templates. Automated email notifications were integrated to inform users of registration confirmations and application status updates, ensuring seamless communication and a user-friendly experience.

SYSTEM TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test Software testing also provides an objective, independent view of the software to allow the business to. Appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs.

##### TYPES OF TESTING

System testing is a critical phase in the development of **XploreIntern**, aimed at ensuring that all individual modules and their integrations work as expected. This process helped identify and resolve defects, validate functionality, and ensure the system was ready for deployment. The testing phase involved several methodologies.

**Unit Testing**

Unit testing focused on verifying the functionality of individual components of the system. Each Django module—such as views for handling user requests, forms for input validation, and models for database interaction—was tested in isolation to ensure it performed correctly. For example, the registration form was tested to validate input fields like email and password, and models were tested to confirm accurate data storage and retrieval.

**Integration Testing**

Integration testing evaluated the interactions between different modules to ensure they worked together seamlessly. For instance, the connection between the student application form and the employer dashboard was tested to verify that applications submitted by students appeared correctly on the employer’s dashboard. Additionally, tests ensured that email notifications were triggered appropriately during registration and application status updates.

**Validation Testing**

Validation testing ensured that the system met the specified functional and non-functional requirements. Key features, such as user registration, internship posting, application tracking, and dashboard functionalities, were verified to confirm they aligned with the project’s goals. The system’s responsiveness and compatibility across various devices and screen sizes were also tested to meet user expectations.

**Output testing**

Output testing focused on verifying that the system produced accurate and meaningful outputs. For instance, email notifications were checked to ensure they contained correct information such as the user’s name and application status. Dashboards were tested to confirm that they displayed relevant and up-to-date data, such as internship details, application statuses, and notifications.

**White box testing**

White box testing examined the internal structure, logic, and execution paths of the system. This included testing the functionality of Django views and ensuring algorithms, such as those calculating profile completion percentages or filtering internships, executed correctly. Code efficiency and error-handling mechanisms were also validated to confirm that the system operated without issues under different scenarios.

**Black box testing**

Black box testing focused on testing the functionality of the system without considering its internal code structure. Real-world scenarios were simulated to ensure the system delivered the expected results. For example, users registering, logging in, applying for internships, and receiving error messages for invalid inputs were tested to validate the system’s usability and performance from an end-user perspective.

**User Acceptance Testing (UAT)**

User Acceptance Testing involved testing the system with a group of sample users to validate that the portal met real-world requirements. During UAT, both students and employers navigated the system, performed key tasks (e.g., registering, posting internships, and applying for internships), and provided feedback on usability and functionality. This feedback was used to make necessary improvements, ensuring the platform delivered a user-friendly experience and fulfilled its intended purpose.

# 

# CONCLUSION

### CONCLUSION

The XploreIntern Internship Portal provides a seamless platform that connects students seeking internships with employers offering valuable opportunities. By automating the internship application and management processes, this system significantly enhances the efficiency for both students and employers. Students can easily search for and apply to relevant internship opportunities with just a few clicks. They can track their application status in real time, manage their profiles, and upload necessary documents such as resumes. On the other hand, employers benefit from a streamlined process of posting internship jobs, reviewing applications, and managing candidates—all in one place. This eliminates manual paperwork, reduces administrative overhead, and accelerates the recruitment process.

XploreIntern ensures data security by eliminating the need for paper-based processes and offering real-time updates. Employers can accept, reject, or withdraw applications with ease, making the entire process more organized, transparent, and efficient.

In conclusion, XploreIntern is a comprehensive and effective solution for automating the internship application process, benefiting both students and employers. It not only saves time and effort but also provides a more efficient, user-friendly platform for finding and managing internships. The system reduces reliance on traditional manual processes, leading to increased accuracy and speed in the recruitment process. Through automation, data security, and streamlined workflows, XploreIntern represents a forward-thinking solution to the challenges traditionally faced in internship recruitment. It can potentially be expanded further to include additional features, such as internship reviews, ratings, or integrated feedback, making it a dynamic and adaptable tool for the future of internships.

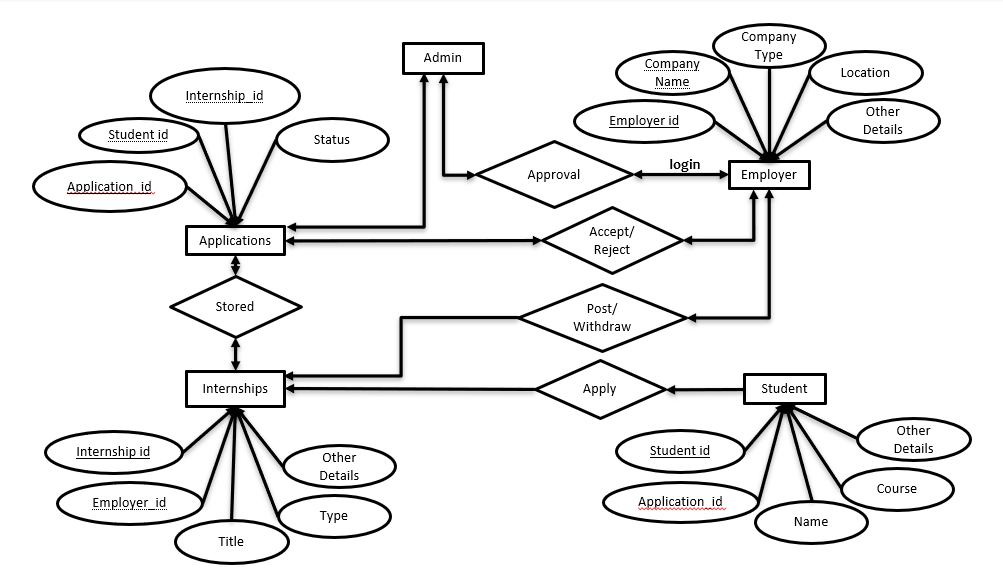
##### 

**7. BIBLIOGRAPHY**

Following websites are referenced to create this project reports:

# APPENDICES

###### 8.2. ER DIAGRAM



###### 8.2. DATA FLOW DIAGRAM

**8.2.1 DATA FLOW DIAGRAM**

**Level 0**

Admin

User

Database

Request

Response

Request

Response

Student

Employer

Response

Request

Database

Request

Response

Request

Response

Response

Request

**8.2.1 DATA FLOW DIAGRAM**

**Level 1**

**8.2.1 DATA FLOW DIAGRAM**

**Level 1.1**

Student

Register

Employer

**Login**

Database

Response

Request

**8.2.1 DATA FLOW DIAGRAM**

**Level 1.1.1**

Register

**Login**

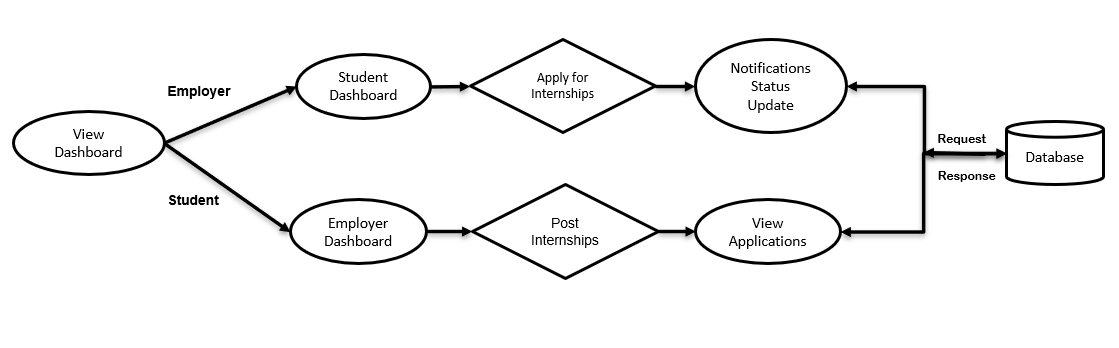
Database

Response

Request

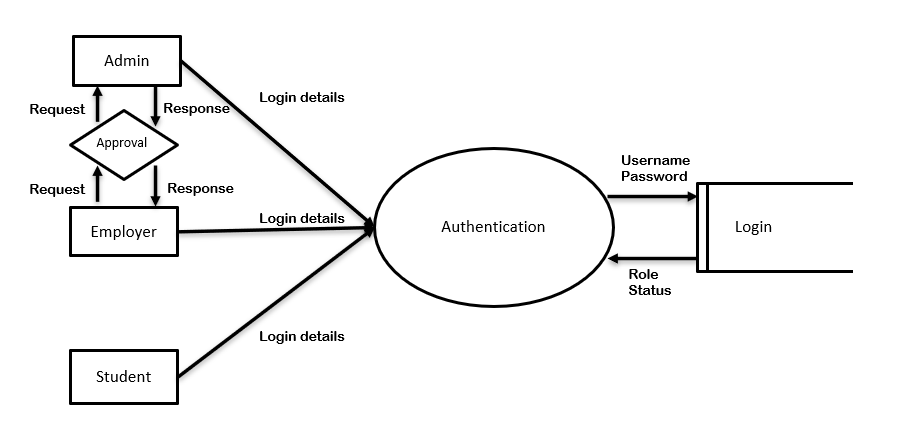
**8.2.1 DATA FLOW DIAGRAM**

**Level 1.1.2**



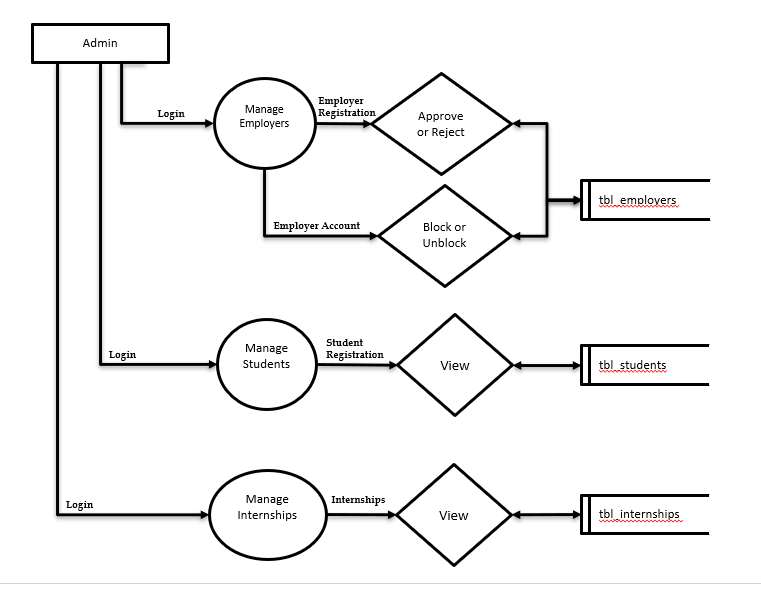
**8.2.1 DATA FLOW DIAGRAM**

**Level 2**



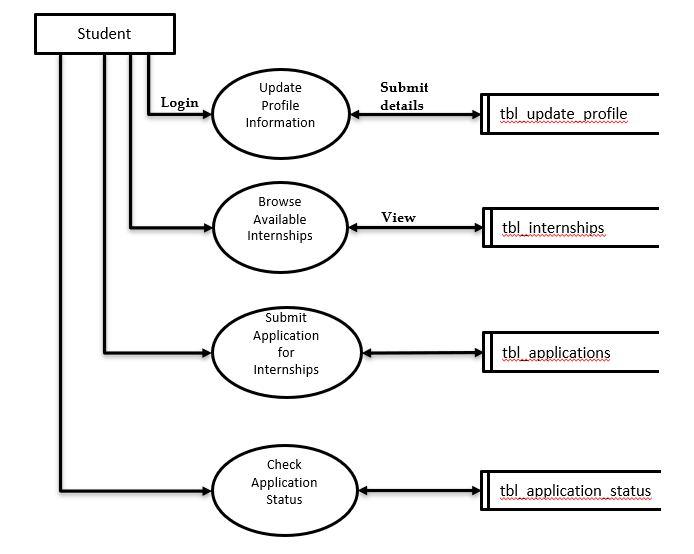
**8.2.1 DATA FLOW DIAGRAM**

**Level 2.1**



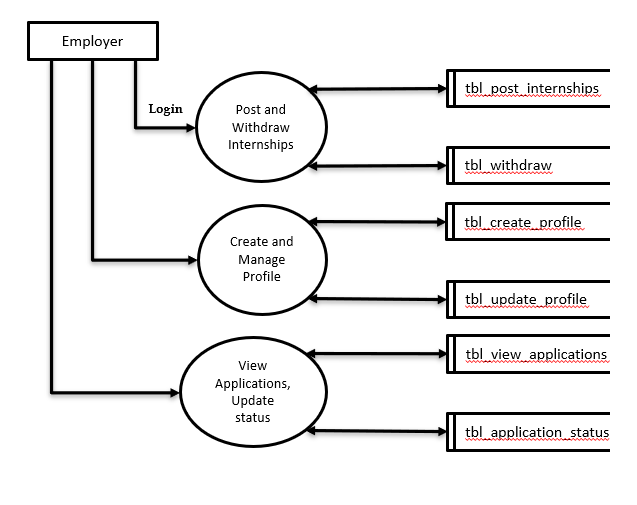
**8.2.1 DATA FLOW DIAGRAM**

**Level 2.2**

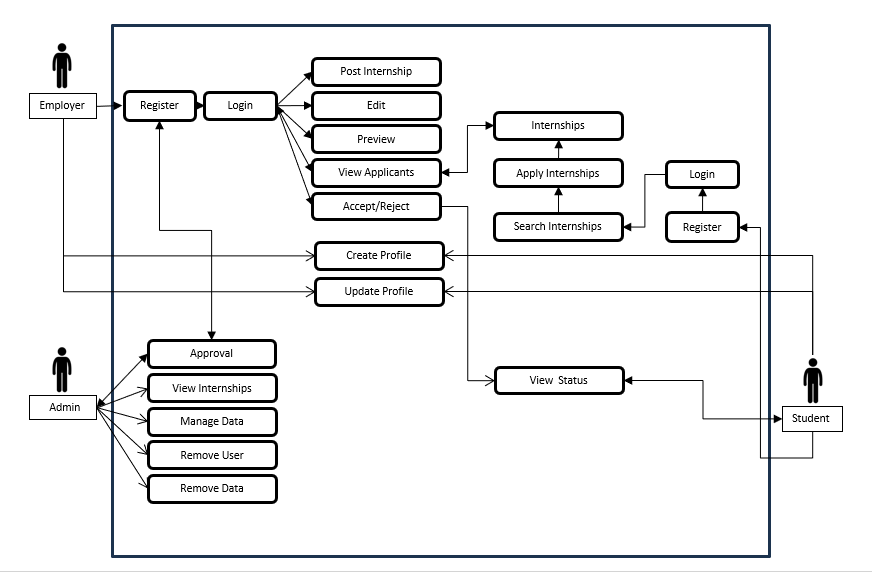


**8.2.1 DATA FLOW DIAGRAM**

**Level 2.3**



##### 8.3 USE CASE DIAGRAM



### 8.3TABLES USED

1. **COMPANY TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slno** | **Field Type** | **Data Type** | **Constraint** | **Description** |
| 1 | Id | Auto field | Primary key | Auto-incrementing primary key |
| 2 | User | One to 0ne field | On delete: CASCADE | Links to the User model (Employer) |
| 3 | Company\_name | Char field | Max Length: 100 | Name of the company |

**2.EMPLOYER TABLE**

**l**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Slno | **Field Type** | **Data Type** | **Constraint** | **Description** |
| 1 | Id | Auto field | Primary key | Auto-incrementing primary key |
| 2 | User | One to one field | On delete: CASCADE | Links to the User model (Employer) |
| 3 | company\_name | Char field | Max Length: 100 | Name of the employer’s company |
| 4 | location | Char field | Max Length: 100 | Location of the employer |
| 5 | Logo | File field | Upload to: logo/ | Company Logo |
| 6 | Updated\_at | Date time field | Auto now: true | Timestamp of the last update |
| 7 | Approved | Boolean field | Default: false | Approval status of the employer |
| 8 | Name | Char field | Max Length: 100 | Employer’s name |
| 9 | Contact\_number | Char field | Max Length: 15 | Employer’s contact number |
| 10 | Description | Text field | Optional | Employer’s description |

**3.STUDENT TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slno** | **Field Type** | **Data Type** | **Constraint** | **Description** |
| 1 | Id | Auto field | Primary key | Auto-incrementing primary key |
| 2 | User | One to one field | On delete: CASCADE | Links to the User model (Student) |
| 3 | University | Char field | Max Length: 100 | University the student is enrolled in |
| 4 | Course | Char field | Max Length: 100 | Course the student is studying |
| 5 | Full\_name | Char field | Max Length: 100 | Full name of the student |
| 6 | Date\_of\_birth | Date field | Optional | Date of birth |
| 7 | Contact\_number | Char field | Max Length: 15 | Student’s contact number |
| 8 | Academic\_status | Char field | Optional | Academic status (e.g.Graduate, undergrad) |
| 9 | Field\_of\_study | Char field | Max Length: 100 | Field of study |
| 10 | Preferred\_location | Char field | Max Length: 100 | Preferred job location |
| 11 | Internship\_duration | Char field | Max Length: 50 | Internship duration |
| 12 | Industry\_interest | Char field | Max Length: 100 | Industry the student is interested in |
| 13 | Logo | File field | Upload to: logo/ | Student logo(optional) |
| 14 | Updated\_at | Date time field | Auto now: true | Timestamp of last profile update |
| 15 | Stipend\_expectation | Decimal field | Max digits: 8, decimal places: 2 | Expected stipend |
| 16 | Skills | Text field | Optional | Skills possessed by the student |
| 17 | Resume | File field | Upload to: resumes/ | Student’s resume |

**4.INTERNSHIP TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slno** | **Field Type** | **Data type** | **Constraint** | **Description** |
| 1 | Id | Auto field | Primary key | Auto-incrementing primary key |
| 2 | Company | Foreign key | On delete: CASCADE | Foreign key to Employer model |
| 3 | Title | Char field | Max Length: 100 | Internship title |
| 4 | Description | Text field |  | Internship description |
| 5 | Location | Char field | Max Length: 100 | Internship location |
| 6 | Type | Char field | Max Length: 50 | Internship type (full-time, part-time) |
| 7 | Deadline | Date field |  | Application deadline |

**5.APPLICATION TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slno** | **Field Type** | **Data type** | **Constraint** | **Description** |
| 1 | Id | Auto field | Primary key | Auto-incrementing primary key |
| 2 | Student | Foreign key | On delete: CASCADE | Foreign key to Student model |
| 3 | Job | Foreign key | On delete: CASCADE | Foreign key to Company model |
| 4 | Applied\_on | Date time field | Auto now add: true | Date and time when application was made |
| 5 | Resume | File field | Upload to: resumes/ | Student’s resume file |
| 6 | Status | Char field | Default: ‘Applied’ | Application status (Applied, Accepted,Rejected) |

**6.JOB TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slno** | **Field Type** | **Data Type** | **Constraint** | **Description** |
| 1 | Id | Auto field | Primary key | Auto-incrementing primary key |
| 2 | Company | Foreign key | On delete: CASCADE | Foreign key to Student model |
| 3 | Internship\_title | Char field | Max Length: 100 | Job title |
| 4 | Company\_name | Char field | Max Length: 100 | Company name |
| 5 | Location | Char field | Max Length: 100 | Job location |
| 6 | Internship\_type | Char field | Max Length: 50 | Types of internship (full-time, part-time) |

###### 8.4 SAMPLE CODE

**Employer\_dashboard.html**

{% load static %}

{% include 'internships/base.html' %}

{% include 'internships/popup\_messages.html' %}

<!DOCTYPE html>

<html>

<head>

<title>Employer Dashboard</title>

<link rel="stylesheet" href="{% static 'css/employer\_dashboard.css' %}">

</head>

<body>

<div class="page-container">

<!-- Content Wrapper -->

<div class="content-wrap">

<h1 style="text-align: center;">Welcome {{ user.username }}</h1>

<div style="text-align: center;"><a href="{% url 'post\_job' %}">Post a New Internship</a></div>

<h2 style="text-align: center; color: white">Your Posted Internships</h2>

<hr style="border-top: 1px solid black; margin: 20px 0;">

<ul>

{% for job in jobs %}

<li>Position: <strong>{{ job.internship\_title }}</strong><br>

<a href="{% url 'job\_preview' job.id %}">View</a> |

<a href="{% url 'edit\_job' job.id %}">Edit</a> <br>

<a class="applicants" href="{% url 'view\_applications' job.id %}">View Applicants</a><br>

{% if job.is\_active %}

<a href="{% url 'withdraw\_job' job.id %}" class="btn btn-danger" onclick="return confirm('Are you sure you want to withdraw this job?');">Withdraw</a>

{% else %}

<span class="badge bg-secondary">Withdrawn</span>

{% endif %}

</li>

{% empty %}

<li>No jobs posted yet.</li>

{% endfor %}

</ul>

</div>

</div>

{% include 'internships/footer.html' %}

</body>

</html>

**Student\_dashboard.html**

{% load static %}

{% include 'internships/base.html' %}

{% include 'internships/popup\_messages.html' %}

<!DOCTYPE html>

<html>

<head>

<title>Student Dashboard</title>

<link rel="stylesheet" href="{% static 'css/student\_dashboard.css' %}">

</head>

<body>

<div class="page-container">

<!-- Content Wrapper -->

<div class="content-wrap">

<h2 style="text-align: center; color: white">Welcome, {{ user.username }}</h2>

<div>

{% if profile\_completion < 100 %}

<div class="circular-progress"

data-inner-circle-color="lightgrey"

data-percentage="{{ profile\_completion }}"

data-progress-color="crimson"

data-bg-color="lightgrey">

<div class="inner-circle"></div>

<p class="percentage">0%</p>

</div>

<div style="text-align: center">

<a href="{% url 'update\_profile' %}">

<button id="update\_profile" type="button">Update profile</button>

</a>

</div>

{% endif %}

</div>

<br>

<div style="text-align: center;"><a class="apply" href="{% url 'apply\_job' %}"> Apply For Internships</a></div>

<h2 style="text-align: center; color: white">Your Applications</h2>

<hr style="border-top: 1px solid black; margin: 20px 0;">

{% if applications %}

<ul>

{% for application in applications %}

<li>

<strong>Internship Title: </strong><b style="color: rgb(179, 226, 230);">

{{ application.job.internship\_title }}</b><br>

<strong>Company: </strong><b style="color: rgb(153, 228, 234);">{{application.job.company\_name}}</b><br>

<strong> Status: </strong> <b style="color: rgb(123, 219, 203);">{{ application.get\_status\_display }}</b><br>

</li>

{% endfor %}

</ul>

{% else %}

<p>You haven't applied to any internships yet.</p>

{% endif %}

</div>

</div>

<script src="{% static 'js/completion.js' %}"></script>

</body>

</html>

{% include 'internships/footer.html' %}

**View\_applications.html**

{% load static %}

{% include 'internships/base.html' %}

<!DOCTYPE html>

<html>

<head>

<title>

Applications

</title>

<link rel="stylesheet" href="{% static 'css/view\_applications.css' %}">

</head>

<body>

<h2 style="text-align: center">Applications for {{ job.internship\_title }}</h2>

{% if applications %}

<ul type="none">

{% for application in applications %}

<li>

<p><strong>Name:</strong> {{ application.student.full\_name }}</p>

<p><strong>Email:</strong> {{ application.student.user.email }}</p>

<p><strong>University:</strong> {{ application.student.university }}</p>

<p><strong>Field of Study:</strong> {{ application.student.field\_of\_study }}</p>

<p><strong>Resume:</strong> <a href="{{ application.resume.url }}" target="\_blank">Download</a></p>

<a href="{% url 'full\_profile' application.id %}">View Profile</a><br>

<div>

{% if application.status == 'applied' %}

<span style="display: flex;">

<form method="post" action="{% url 'accept\_application' application.id %}">

{% csrf\_token %}

<button class="accept" type="submit">Accept</button>

</form>

<form method="post" action="{% url 'reject\_application' application.id %}">

{% csrf\_token %}

<button class="reject" type="submit">Reject</button>

</form>

</span>

{% elif application.status == 'accepted' %}

<button class="accepted" disabled>Accepted</button>

{% elif application.status == 'rejected' %}

<button class="rejected" disabled>Rejected</button>

{% endif %}

</div>

{% endfor %}

</li>

</ul>

{% else %}

<p>No applications for this job yet.</p>

{% endif %}

</ul>

</body>

</html>

{% include 'internships/footer.html' %}

**Admin.py**

from django.contrib import admin

from .models import Employer, Student, Job, Internship, Application

@admin.action(description='Approve selected employers')

def approve\_employers(modeladmin, request, queryset):

queryset.update(approved=True)

@admin.action(description='Disapprove selected employers')

def disapprove\_employers(modeladmin, request, queryset):

queryset.update(approved=False)

class EmployerAdmin(admin.ModelAdmin):

list\_display = ('user', 'company\_name', 'location', 'approved')

actions = [approve\_employers, disapprove\_employers]

list\_filter = ('approved',)

search\_fields = ('user\_\_username', 'company\_name')

class StudentAdmin(admin.ModelAdmin):

list\_display = ('user', 'full\_name', 'university', 'course', 'contact\_number')

search\_fields = ('user\_\_username', 'full\_name', 'university', 'course')

class JobAdmin(admin.ModelAdmin):

list\_display = ('internship\_title', 'company\_name', 'location', 'is\_active')

search\_fields = ('internship\_title', 'company\_name')

list\_filter = ('is\_active', 'experience\_level')

class InternshipAdmin(admin.ModelAdmin):

list\_display = ('title', 'company', 'location', 'type', 'deadline')

search\_fields = ('title', 'company\_\_company\_name')

list\_filter = ('type',)

class ApplicationAdmin(admin.ModelAdmin):

list\_display = ('student', 'job', 'applied\_on', 'status')

list\_filter = ('status',)

search\_fields = ('student\_\_user\_\_username', 'job\_\_internship\_title')

admin.site.register(Employer, EmployerAdmin)

admin.site.register(Student, StudentAdmin)

admin.site.register(Job, JobAdmin)

admin.site.register(Internship, InternshipAdmin)

admin.site.register(Application, ApplicationAdmin)

class UnapprovedEmployerFilter(admin.SimpleListFilter):

title = 'approval status'

parameter\_name = 'approved'

def lookups(self, request, model\_admin):

return (

('approved', 'Approved'),

('not\_approved', 'Not Approved'),

)

def queryset(self, request, queryset):

if self.value() == 'approved':

return queryset.filter(approved=True)

elif self.value() == 'not\_approved':

return queryset.filter(approved=False)

return queryset

class EmployerAdmin(admin.ModelAdmin):

list\_display = ('user', 'company\_name', 'location', 'approved')

actions = [approve\_employers, disapprove\_employers]

list\_filter = ('approved', UnapprovedEmployerFilter)

search\_fields = ('user\_\_username', 'company\_name')

class JobInline(admin.TabularInline):

model = Job

extra = 0 # No extra empty forms in inline display

class ApplicationInline(admin.TabularInline):

model = Application

extra = 0

class EmployerAdmin(admin.ModelAdmin):

list\_display = ('user', 'company\_name', 'location', 'approved')

actions = [approve\_employers, disapprove\_employers]

list\_filter = ('approved', UnapprovedEmployerFilter)

search\_fields = ('user\_\_username', 'company\_name')

inlines = [JobInline]

class StudentAdmin(admin.ModelAdmin):

list\_display = ('user', 'full\_name', 'university', 'course', 'contact\_number')

search\_fields = ('user\_\_username', 'full\_name', 'university', 'course')

inlines = [ApplicationInline]

**models.py**

from django.db import models

from django.contrib.auth.models import User

from django.utils import timezone

class Company(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE) # Company linked to User (employer)

company\_name = models.CharField(max\_length=100)

class Employer(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

company\_name = models.CharField(max\_length=100)

location = models.CharField(max\_length=100)

logo = models.FileField(upload\_to='logo/', null=True, blank=True)

updated\_at = models.DateTimeField(auto\_now=True)

approved = models.BooleanField(default=False) # Field to track approval status

name= models.CharField(max\_length=100, null=True, blank=True)

contact\_number = models.CharField(max\_length=15, null=True, blank=True)

description = models.TextField(null=True, blank=True)

def \_\_str\_\_(self):

return self.company\_name

class Student(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

university = models.CharField(max\_length=100)

course = models.CharField(max\_length=100)

full\_name = models.CharField(max\_length=100, blank=True)

date\_of\_birth = models.DateField(null=True, blank=True)

contact\_number = models.CharField(max\_length=15, null=True, blank=True)

academic\_status = models.CharField(max\_length=50, null=True, blank=True)

field\_of\_study = models.CharField(max\_length=100, null=True, blank=True)

university = models.CharField(max\_length=100, null=True, blank=True)

preferred\_location = models.CharField(max\_length=100, null=True, blank=True)

internship\_duration = models.CharField(max\_length=50, null=True, blank=True)

industry\_interest = models.CharField(max\_length=100, null=True, blank=True)

logo = models.FileField(upload\_to='logo/', blank=True, null=True)

updated\_at = models.DateTimeField(auto\_now=True)

stipend\_expectation = models.DecimalField(max\_digits=8, decimal\_places=2, null=True, blank=True)

skills = models.TextField(null=True, blank=True)

resume = models.FileField(upload\_to='resumes/', null=True, blank=True)

def profile\_completion(self):

# Check each field's completion

fields = [

self.full\_name, self.date\_of\_birth, self.contact\_number, self.academic\_status,

self.field\_of\_study, self.university, self.preferred\_location, self.internship\_duration,

self.industry\_interest,

self.stipend\_expectation, self.skills, self.resume

]

completed\_fields = sum([1 for field in fields if field])

total\_fields = len(fields)

return round((completed\_fields / total\_fields) \* 100)

class Internship(models.Model):

company = models.ForeignKey(Employer, on\_delete=models.CASCADE)

title = models.CharField(max\_length=100)

description = models.TextField()

location = models.CharField(max\_length=100)

type = models.CharField(max\_length=50, choices=[('full-time', 'Full-Time'), ('part-time', 'Part-Time')])

deadline = models.DateField()

class Application(models.Model):

student = models.ForeignKey('Student', on\_delete=models.CASCADE)

job = models.ForeignKey('Job', on\_delete=models.CASCADE)

applied\_on = models.DateTimeField(auto\_now\_add=True)

resume = models.FileField(upload\_to='resumes/')

status = models.CharField(max\_length=20, choices=[('applied', 'Applied'), ('accepted', 'Accepted'), ('rejected', 'Rejected')],

default='applied')

class Job(models.Model):

company = models.ForeignKey(User, on\_delete=models.CASCADE) # Link to Employer model

internship\_title = models.CharField(max\_length=100)

company\_name = models.CharField(max\_length=100)

location = models.CharField(max\_length=100)

internship\_type = models.CharField(max\_length=50, choices=[('Full-time', 'Full-Time'), ('Part-time', 'Part-Time'), ('Contract', 'Contract'), ('Temporary', 'Temporary')])

experience\_level = models.CharField(max\_length=50, choices=[('Entry-level', 'Entry-Level'), ('Junior', 'Junior'), ('Mid-level', 'Mid-Level'), ('Senior', 'Senior'), ('Executive', 'Executive')])

salary\_range = models.CharField(max\_length=100)

benefits = models.TextField()

summary = models.TextField()

key\_responsibilities = models.TextField()

required\_qualifications = models.TextField()

preferred\_qualifications = models.TextField(blank=True)

skills = models.TextField()

application\_instructions = models.TextField()

application\_deadline = models.DateField()

contact\_information = models.TextField()

company\_overview = models.TextField()

company\_website = models.URLField()

is\_active = models.BooleanField(default=True)

def \_\_str\_\_(self):

return self.internship\_title

**views.py**

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.contrib.auth import authenticate, login, logout

from django.core.mail import send\_mail, EmailMessage

def register(request):

return render(request, 'internships/register.html')

def home(request):

featured\_jobs = Job.objects.filter(is\_active=True)[:5] # Display 5 featured jobs

for job in featured\_jobs:

try:

employer = job.company.employer # Access Employer linked to the company (User)

job.company\_logo\_url = employer.logo.url if employer.logo else None

except Employer.DoesNotExist:

job.company\_logo\_url = None

return render(request, 'internships/home.html', {'featured\_jobs': featured\_jobs})

def employer\_registration(request):

if request.method == 'POST':

form = EmployerRegisterForm(request.POST)

if form.is\_valid():

user = form.save()

Employer.objects.create(

user=user,

company\_name=form.cleaned\_data['company\_name'],

location=form.cleaned\_data['location']

)

messages.success(request, "You have successfully registered.")

# Render email HTML content

email\_content = render\_to\_string('internships/email/registration\_success.html', {

'user': user,

'company\_name': form.cleaned\_data['company\_name']

})

email = EmailMessage(

'Registration Success - Welcome to XploreIntern',

email\_content,

settings.EMAIL\_HOST\_USER,

[user.email]

)

email.content\_subtype = "html" # Set the email to HTML

email.send(fail\_silently=False)

return render(request, 'internships/home.html', {'show\_login\_modal': True})

else:

form = EmployerRegisterForm()

return render(request, 'internships/employer\_register.html', {'form': form})

def student\_registration(request):

if request.method == 'POST':

form = StudentRegisterForm(request.POST)

if form.is\_valid():

user = form.save()

Student.objects.create(user=user, university=form.cleaned\_data['university'], course=form.cleaned\_data['course'])

messages.success(request, "You have successfully registered." )

# Render email HTML content

email\_content = render\_to\_string('internships/email/student\_register\_success.html', {

'user': user,

'university': form.cleaned\_data['university']

})

email = EmailMessage(

'Registration Success - Welcome to XploreIntern',

email\_content,

settings.EMAIL\_HOST\_USER,

[user.email]

)

email.content\_subtype = "html" # Set the email to HTML

email.send(fail\_silently=False)

return render(request, 'internships/home.html', {'show\_login\_modal': True})

else:

form = StudentRegisterForm()

return render(request, 'internships/student\_register.html', {'form': form})

def login\_user(request):

if request.method == 'POST':

username = request.POST.get('username')

password = request.POST.get('password')

user = authenticate(request, username=username, password=password)

if user is not None:

try:

employer = user.employer

if employer.approved:

login(request, user)

return redirect('employer\_dashboard')

else:

messages.error(request, 'Employer account is not approved yet.')

return redirect('login')

except Employer.DoesNotExist:

login(request, user)

return redirect('student\_dashboard')

else:

messages.error(request, 'Invalid credentials.')

return render(request, 'internships/home.html')

@login\_required

def employer\_dashboard(request):

employerr = Employer.objects.get(user=request.user)

employer = Employer.objects.filter(user=request.user).first()

if employer:

jobs = Job.objects.filter(company=request.user)

print(f"Number of jobs found: {len(jobs)}")

else:

jobs = []

return render(request, 'internships/employer\_dashboard.html', {'jobs': jobs , 'employerr':employer})

**forms.py**

from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

from .models import Employer, Student

from .models import Job, Application

class EmployerRegisterForm(UserCreationForm):

company\_name = forms.CharField(max\_length=100)

location = forms.CharField(max\_length=100)

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

class StudentRegisterForm(UserCreationForm):

university = forms.CharField(max\_length=100)

course = forms.CharField(max\_length=100)

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

class JobForm(forms.ModelForm):

class Meta:

model = Job

fields = [

'internship\_title', 'company\_name', 'location', 'internship\_type',

'experience\_level', 'salary\_range', 'benefits', 'summary',

'key\_responsibilities', 'required\_qualifications', 'preferred\_qualifications',

'skills', 'application\_instructions', 'application\_deadline',

'contact\_information', 'company\_overview', 'company\_website'

]

widgets = {

'application\_deadline': forms.DateInput(attrs={'type': 'date'})

}

class ApplicationForm(forms.ModelForm):

class Meta:

model = Application

fields = ['resume']

resume = forms.FileField(

required=False,

widget=forms.ClearableFileInput(attrs={'class': 'file-input'}),

)

class StudentProfileForm(forms.ModelForm):

class Meta:

model = Student

fields = [

'logo', 'full\_name', 'date\_of\_birth', 'contact\_number', 'academic\_status',

'field\_of\_study', 'university', 'preferred\_location', 'internship\_duration',

'industry\_interest',

'stipend\_expectation', 'skills', 'resume'

]

widgets = {

'date\_of\_birth': forms.DateInput(attrs={'type': 'date'})

}

class EmployerProfileForm(forms.ModelForm):

class Meta:

model = Employer

fields = [

'logo', 'name', 'contact\_number', 'location', 'company\_name', 'description'

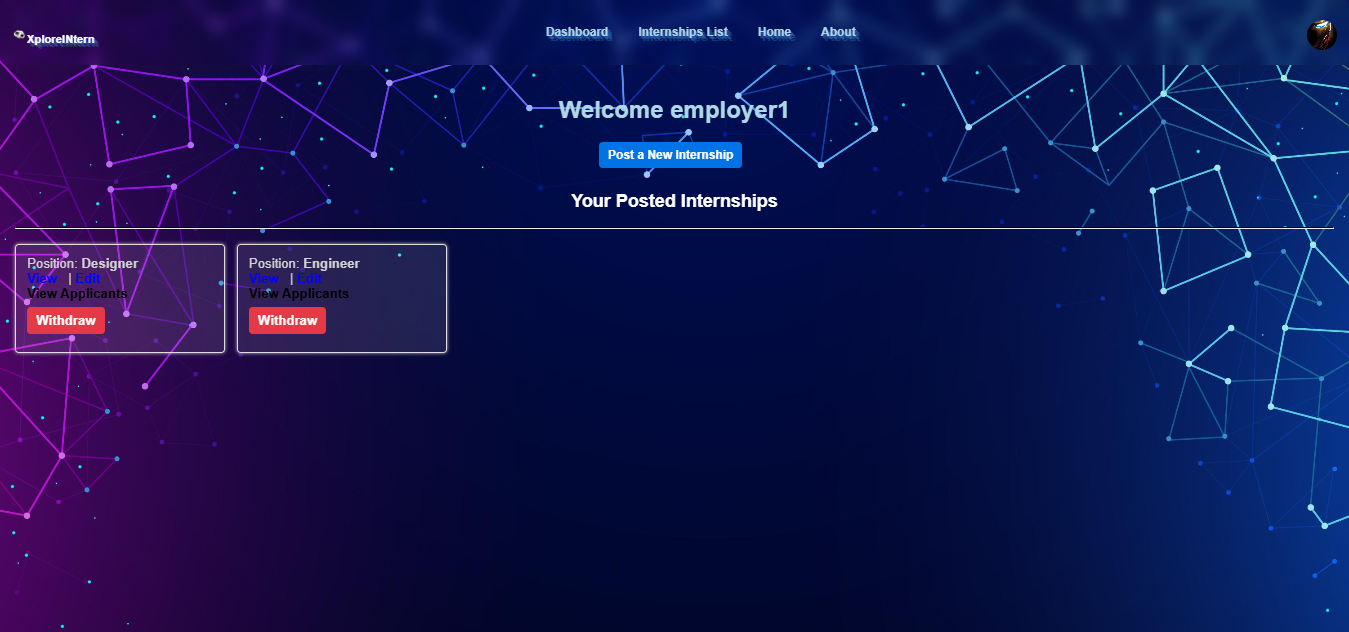
]

### 8.5 SAMPLE SCREENSHOTS

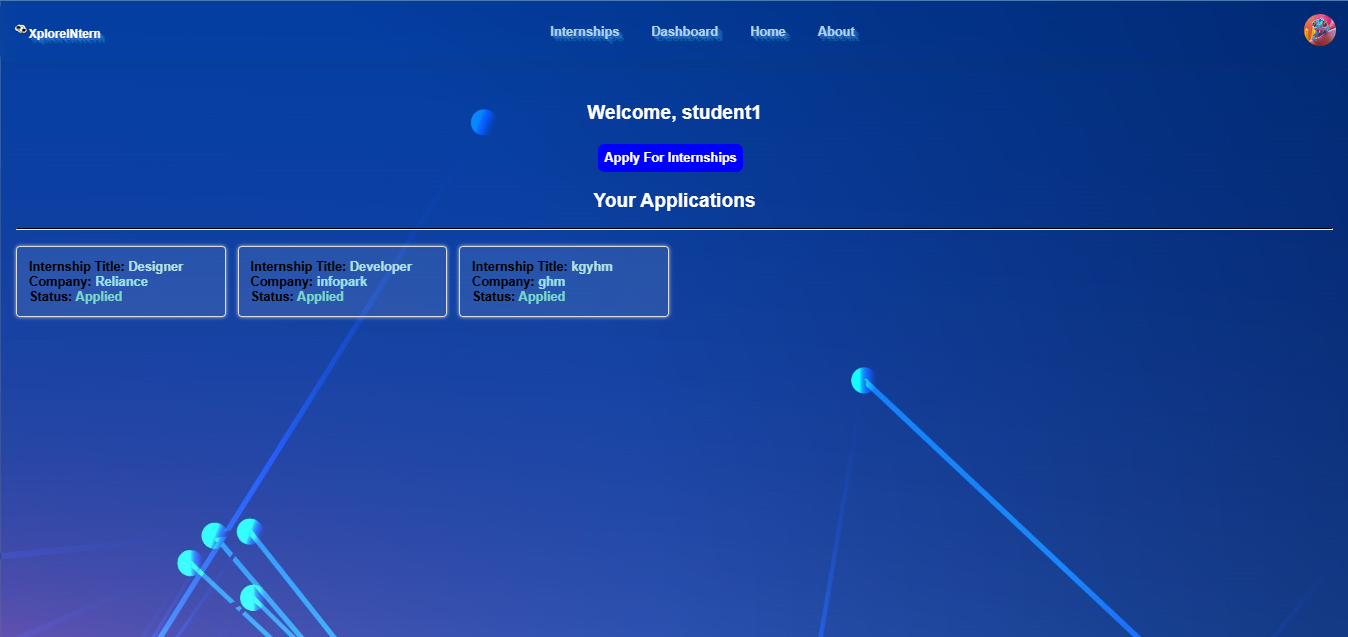
**Home page**



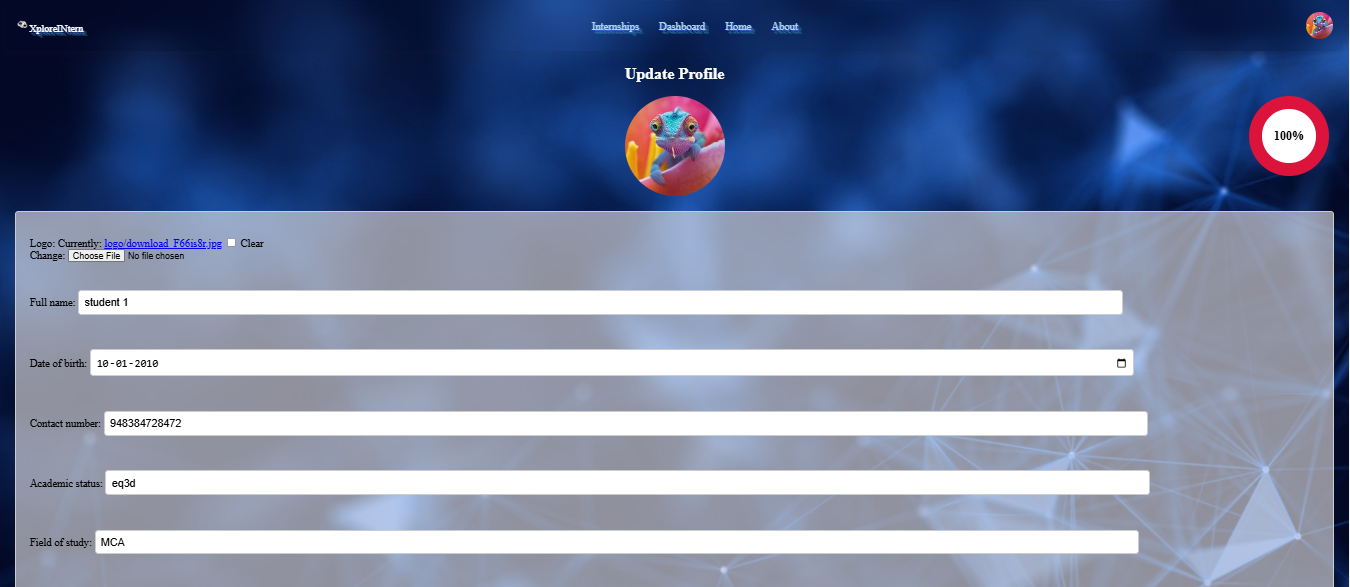
**Employer Dashboard**



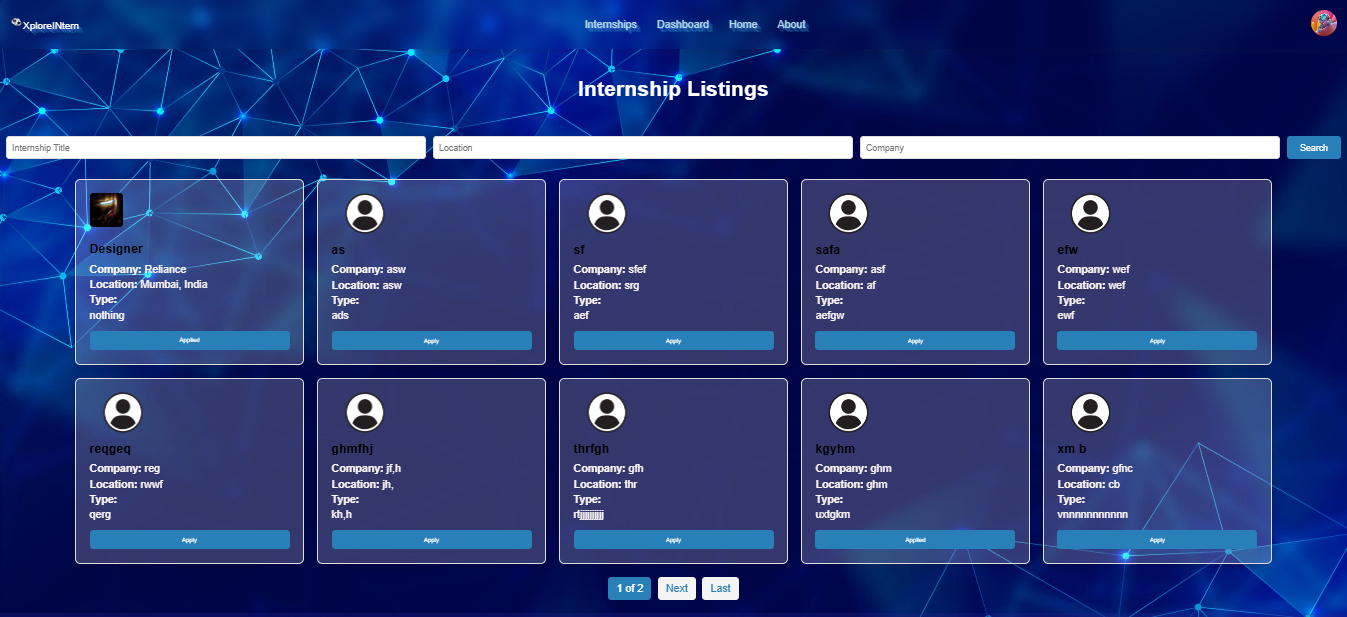
**Student Dashboard**



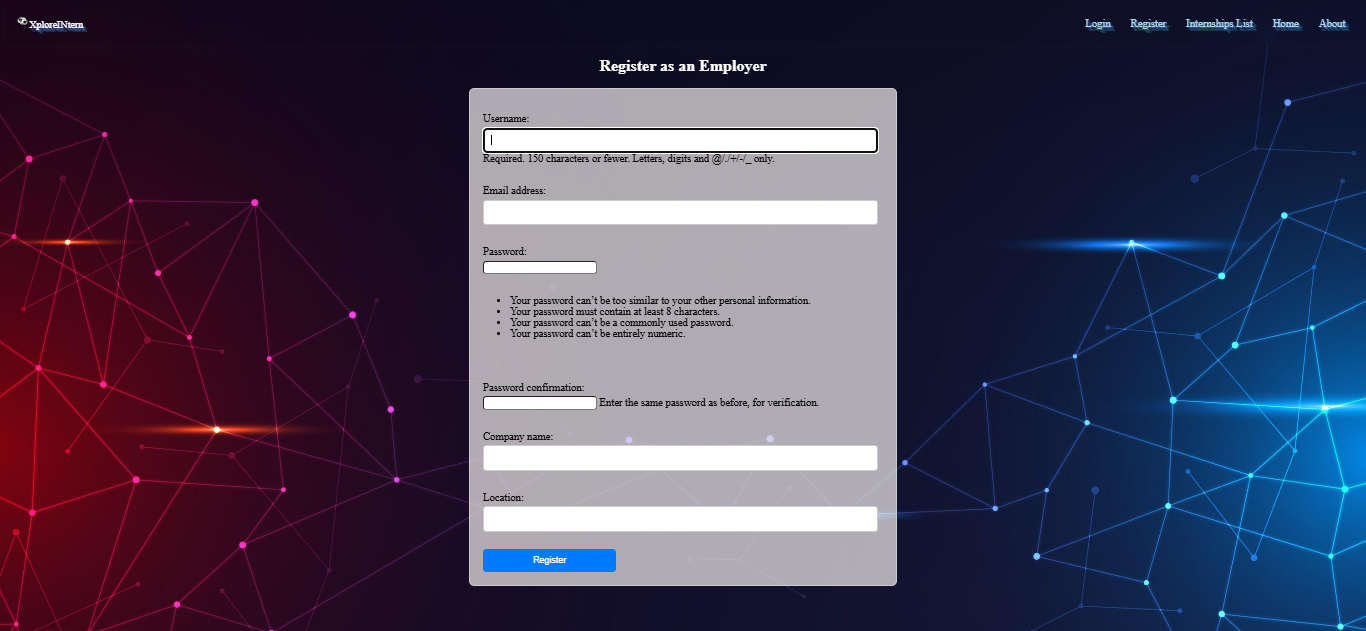
**Update Profile**



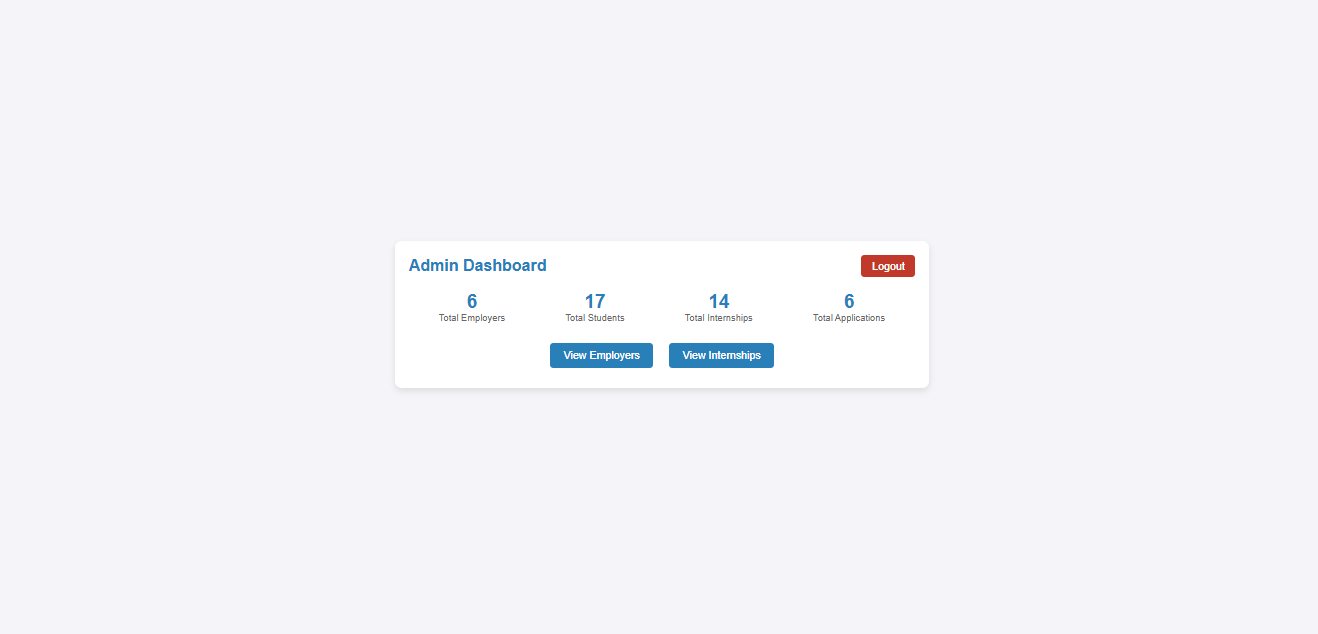
**View Internships**



**Registration**



**Admin Panel**



**Admin Approval Section**

