ZillionHire
Placement Cell Management System

Mini Project Report

Submitted by

Ashna Karim

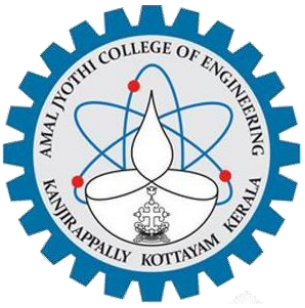
Reg. No.: AJC19MCA-I017

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

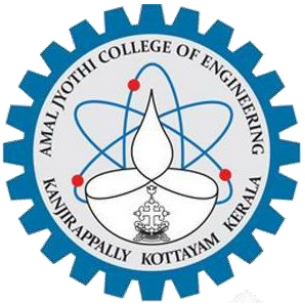
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, "**ZILLIONHIRE"** is the bona fide work of **ASHNA KARIM (Regno: AJC19MCA-I017)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Jetty Benjamin

Internal Guide

Ms. Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **"ZILLIONHIRE"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: 30-10-2023 **ASHNA KARIM**

KANJIRAPPALLY **Reg: AJC19MCA-I017**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Jetty Banjamin** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

ASHNA KARIM

ABSTRACT

ZillionHire is a placement cell management system, consists of an admin, student & company login. The project is beneficial for students and various companies visiting the campus for recruitment for direct updates or information.

The placement cell project proposes an integrated system to revolutionize the functioning of college placement cell, streamlining the process of connecting students with job opportunities. The proposed platform provides a user-friendly interface for students, enabling them to create detailed profiles, upload resumes, and showcase their academic achievements also to browse and apply for job openings which matches with their cgpa, ensuring they have access to a diverse range of opportunities.

For recruiters, the platform offers a streamlined process for posting job vacancies, specifying eligibility criteria. Recruiters can access a talent pool of qualified candidates and efficiently shortlist potential hires. The admin can view student details, company details and can add students, approve companies and jobs. The institution will gain a lot from putting the suggested placement cell initiative into action. The hiring process will be streamlined, and recruiters can view applied students. Additionally, it will provide students more agency by giving them a single platform to research a variety of employment options.

CONTENT

List of Abbreviation

IDE - Integrated Development Environment

HTML - Hyper Text Markup Language.

CSS - Cascading Style Sheet

SQL - Structured Query Language

UML - Unified Modelling Language

JS – JavaScript

AJAX – Asynchronous JavaScript and XML Environment

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

ZillionHire, includes logins for administrators, students, and companies. The project is advantageous to the students since it allows them to directly access updates or information from the numerous companies who visit the campus to hire students.

The project suggests a comprehensive approach to transform how college placement cell operate by expediting the procedure for matching students with employment possibilities. The suggested platform offers students a user-friendly interface that enables them to view complete profiles, upload profile details, and highlight academic accomplishments. Students have access to search and apply for job positions which matches with their cgpa. The platform provides recruiters with a streamlined procedure for advertising job openings that includes eligibility requirements also highlights profile and short list students. Admin add students, approve both company and jobs and checks student and company details. The institution will gain a lot from putting the suggested placement cell initiative into action. The hiring process will be streamlined, and recruiters can view applied students. Additionally, it will provide students more agency by giving them a single platform to research a variety of employment options.

1.2 PROJECT SPECIFICATION

This includes three users,

Admin can check each student details, company details. And can add students. Approves company and jobs and option for view students who are applied for job and short-listed students done by company. Admin has overall right over the system. Manage the website's content, including updating text, images and other relevant information.

Students can create their profiles and upload various data such as personal details, academic details and contact details and can search and view the jobs posted by different companies which matches with their cgpa and apply for job.

Company can view a list of eligible students as per each company's criteria and students applied for the particular job position and also their respective details.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis, which requires gathering and reviewing data to pinpoint concerns and offer solutions, is a crucial phase of system development. Effective communication is vital throughout the entire process. Communication among system users and developers is important. Every system development process must first, undertake a system analysis. The system analyst meticulously plays the part of an investigator evaluating the efficiency of the current system. recognizing the inputs and outputs of the system as Furthermore, a link between its procedures and organizational outcomes is highlighted. Information can be collected using a variety of methods including inquiries and surveys. The objectives are to comprehend how the system operates, find areas of concern, and offer solutions to the problems the company is encountering. The role of the problem-solver is taken on by the designer, and the proposed fixes are thoroughly compared with the existing system. After the best option is selected, the user receives the option to accept or reject the suggestion that was offered. Once the proposal has been assessed in light of their feedback, the process is repeated until the user is delighted. A preliminary study is the procedure of collecting and assessing data for upcoming system studies. To guarantee the success of a system development project, a comprehensive initial inquiry must be conducted.

2.2 EXISTING SYSTEM

In the current system, every process is carried out manually. In the current system, human interaction is used to complete every task. The placement officer had a lot of work to perform because everything was done manually, which also increased the potential of errors. This is extremely time-consuming and slow. The process has gotten more challenging as there are more users. TPO(Training and Placement Officer) manually searches for eligible students based on the company criteria. The sorting issue was caused by the records being saved in modified Excel sheets. Records were frequently duplicated, which led to data redundancy. TPOs must gather all the student data and resumes, manually arrange them, and classify them according to several streams. It is a difficult and time-consuming task to gather the CVs of so many students, and handling too many CVs is a significant burden. Managing, updating, and notifying specific students about company requirements takes too much effort.

2.2.1 NATURAL SYSTEM STUDIED

The current system uses a manual process for every step. The current method requires human intervention for every task that needs to be completed. Due to the heavy strain placed on the placement officer by the manual nature of the work, there are also more opportunities for error. This is extremely time-consuming and slow. All things are done manually here like collecting cv, other placement procedures.

2.2.2 DESIGNED SYSTEM STUDIED

The primary goal is to make it easier Placement coordinators, students and companies to access and edit information. The solution offers a better way to maintain student data in the database and also guarantees data integrity and data correctness. The solution also speeds up information flow between various system components and decreases the amount of time required for paperwork.

2.3 DRAWBACKS OF EXISTING SYSTEM

- The time needed for a placement officer to gather student information and give their approval is considerable.
- Ineffective Integration: The current system may not be properly integrated with other systems or platforms, which might cause data duplication and management issues. These integration challenges can be resolved with a more comprehensive strategy.
- There is a lack of contact between students and the placement officer, making it difficult to recommend new placements.
- Time-consuming and Complex Methods: The placement procedure is frequently inefficient because of the time-consuming and complex nature of the current placement cell methods. This may be a result of the overall system being slowed down by manual procedures, paperwork, and human participation
- Students might not be aware of specifics regarding the business. A difficulty arises from the poor communication.
- Analytical Problems: Human involvement in the hiring process can cause analytical problems, which could result in mistakes when matching applicants with open positions. Automation can lessen this issue by accelerating the placement process' analytical steps.

- If the appropriate training and placement officer is not present, the candidate may not receive the necessary information.
- Lack of Automation: The current method might not be fully automated, which would make it less effective and more prone to mistakes. These issues could be resolved and the placement cell website's general functionality could be enhanced by a more automated and computerized system
- Limited Accessibility: Students and employers may not have simple access to crucial information using the current system. The efficiency of the placement cell website can be increased by ensuring a user-friendly interface and enhanced accessibility.

2.4 PROPOSED SYSTEM

The administration of student placement operations inside a college is made easier by ZillionHire. The suggested solution is web-based, making it simple for administrators and students to access. This makes it handier for users because it can be accessible through a web browser. This system's main objective is to control what happens in the placement cell. It helps to organize and simplify the procedures involved in student placements. The system probably has tools for managing student data. This can include resumes, profiles, and other pertinent information that helps with the hiring process. It could provide businesses a place to list job openings so that students can apply.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Web-Based System: The suggested system is web-based, making it easier for administrators and students to access. This makes it handier for users because it may be reached via a web browser.
- Placement Cell Management: Overseeing the activities of the placement cell is the main objective of this approach. It helps to organize and streamline the procedures involved in student placements.
- Student Data Management: The system probably includes tools for organizing student data. This can include resumes, profiles, and other pertinent information that helps with the hiring process.
- Online Job Postings: It might give businesses a place to advertise job openings that students can apply for.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility study is the degree to which a project can actually be carried out successfully. A feasibility study is conducted to assess the solution's viability, which establishes whether it is viable and implementable in the program. The feasibility study takes into account details like the availability of resources, software development costs, the advantages of the software to the business once it is built, and the costs associated with maintaining it. The outcome of the feasibility study should be a report recommending whether or not the requirements engineering and system development process should be continued.

3.1.1 Economic Feasibility

Assesses whether the necessary software may bring an organization financial benefit. It includes the expenses related to the software development team, the expected cost of the necessary hardware and software, the expense of conducting a feasibility study, and so on. Expenses associated with software development that result in long-term benefits for a business Costs associated with doing a comprehensive software study, including requirements elicitation and requirements analysis cost of the development team, software, hardware, and training.

The suggested system ZillionHire organizes and streamlines the entire placement process, improving its effectiveness. It helps recruiters to publish job vacancies and review resumes digitally, minimizing paperwork and manual labour, and it enables students to view and apply for job possibilities online, individuals from many fields can look into job prospects provided by a wide range of companies, and recruiters from various regions can access a diversified pool of students. The software makes communication between recruiters, the placement cell, and students easy. It makes it possible to schedule interviews quickly and provide feedback, which improves everyone's engagement and experience. The website can gather and analyse data on job placements, student preferences, and employer requirements. This data-driven approach helps in making informed decisions, identifying trends, and improving future placement strategies. Reduces manual intervention and paperwork. The website allows for real-time updates on job openings, interviews, and placement-related events, keeping all stakeholders informed and up-to-date. The website may collect and analyse information on student preferences, employer requirements, and job placements. Making educated decisions, spotting trends, and optimizing future placement tactics are all facilitated by this data-driven approach. All interested parties can be informed and up to date with the website's real-time updates on job vacancies, interviews, and placement-related events.

3.1.2 Technical Feasibility

Technical feasibility studies are essential because they aid stakeholders in understanding whether, from a technological perspective, the project is indeed feasible. It offers useful information that can affect the decision to move on with the project. It evaluates the viability of a given project or business endeavor by technically feasible in their implementation. It concentrates on assessing whether the required infrastructure, resources, skills, and technology are in place or attainable to complete the project successfully. Examines the technical proficiency and qualifications of the members of the software development team evaluates the stability and maturity of the applicable technology, ensures that the software development technology has a sizable user base so that people may be consulted when issues arise or improvements are needed.

ZillionHire uses latest web technologies. Within the allotted time and money, the technologies employed can be modified to meet user requirements in the software, as well as new upgrades.

- For front end: HTML, CSS, JavaScript, Bootstrap
- For the back end: Python- Django

These can be applied to current problems and flexible.

3.1.3 Behavioral Feasibility

Evaluates the degree to which the necessary software executes a sequence of operations to meet user and business criteria. This feasibility involves imagining whether the software will function after it is produced and be functional after it is installed. It is dependent on human resources (the software development team). The following duties are also carried out by behavioral feasibility. Evaluates the priority of the issues raised by the user requirements, determines whether the software development team's solution is appropriate, examines how well people will accept new software. This involves evaluating its compatibility with existing processes and resources within the university or educational institution. By determining whether the website can seamlessly interact with the existing placement methods and databases, the behavioral viability of the website can be proved. The suggested system ZillionHire allows easy registration for students, alumni and companies, which automate job postings& job applying resume submissions furthermore provide administrative tools so that placement coordinators may efficiently manage and monitor the process. Additionally, the website is user-friendly. The feasibility study should look into the possible advantages of increased productivity, less paperwork, and better communication between students and businesses, all of which would help the ZillionHire run smoothly and be used by more people.

3.1.4 Feasibility Study Questionnaire

1. Who are the primary stakeholders involved in the placement process?
   The primary stakeholders are students& alumni students seeking placements, recruiters from various companies, and the college placement team.
2. For which departments placements is conducted?
   CSE, ECE, IT, MCA
3. What features do you envision for the website to support students?
   Job listings from various companies, options for students who can upload details include their resume, notifications about job openings, deadlines, placement process, events, hiring process also tracking the hiring process.
4. What should include in study resources?
   Aptitude tests/ mock tests, mock interviews, other relevant tests and notes.
5. What types of job postings do you expect to have on the website? (Internships, full-time positions, part-time jobs)
   full-time job positions
6. Does your institution conduct any events with students and company and what kind of events are those?
   yes, we conduct students' interaction with company like pre-placement talks.
7. What are the current modes of communication about placement process?
   through email, WhatsApp groups
8. What is the process for reviewing and approving student profiles and job postings by the placement team?
   The placement team will review student profiles and job postings manually to ensure they meet the set criteria and guidelines before they applying to particular job.
9. Are there any specific analytics or reporting features you would like to have on the website?
   analytics on student engagement, job application rates, feedback, and placement success rates to evaluate the website's performance.
10. What functions do you imagine the website having to help recruiters? (For instance, posting jobs, finding students, and scheduling interviews)
    An easy-to-use interface for posting job positions, sophisticated student search filters, and an efficient interview scheduling system are some of the features envisage for recruit.

3.2 SYSTEM SPECIFICATION

3.2.1 Minimum Specification

Processor - Intel core i5

RAM - 8 G B

SSD - 4 7 5 G B

3.2.2 Software Specification

Front End - HTML, CSS

Back End - DJANGO

Database - SQLITE

Client on PC - Windows 7 and above.

Technologies used - JS, CSS, AJAX

3.3 SOFTWARE DESCRIPTION

3.3.1 Python Django

Python A web framework called Django makes it possible to make effective web pages quickly. Django is also known as the "batteries included framework" because to the built-in functionality it offers, like the Django Admin Interface and the SQLite3 default database. A comparable set of elements are usually required when establishing a website: a method for handling user authentication (signing up, signing in, and signing out), a management panel for your website, forms, a method for uploading files, etc. You can use pre-made components that Django provides. Both the documentation and scalability are excellent. Top MNCs and Companies use it, including Instagram, Disqus, Spotify, YouTube, Bitbucket, Dropbox, and a seemingly endless list of others. Fastest development, most comprehensive batteries, and easiest Framework to understand. A rapid web development framework called Django can be used to create fully functional online apps quickly. The third and last justification for learning Django is Python, which offers a sizable library and functions like Web scraping, machine learning, image processing, scientific computing, etc. All of this may be integrated with web applications to perform a ton of complex tasks.

3.3.2 SQLite

A compact, serverless, and lightweight database engine is SQLite. Due to its ease of use, portability, and effectiveness, it is frequently utilized in many applications. When using SQLite, which is frequently used for particular use cases, consider the following:

- SQLite is frequently used in embedded systems and Internet of Things devices. It is appropriate for devices with low resources because to its tiny size and self-contained design.
- Mobile Apps: For storing app-specific data, many mobile apps, including those for Android and iOS, use SQLite as the local database. It provides mobile devices with quick, dependable data storage.
- Development and learning: SQLite is a fantastic tool for learning SQL and database design. It is used by beginners and students to practice SQL queries and comprehend database principles.
- SQLite is frequently used for quick prototyping while designing applications, especially in the beginning phases. Without the need for a full-featured database server, it enables developers to build and test the database structure.
- Data that is regularly accessed or must be accessible offline can be cached using SQLite. For programs that need caching, it offers quick data retrieval.
- Data analysis: SQLite can be used to analyze small to medium-sized datasets. It frequently integrates with Python data analysis frameworks like Pandas.
- Report generation and log data storage are both possible with SQLite. It's a fantastic option if you need to create straightforward reports or keep track of application usage.
- SQLite can be utilized to store and train machine learning

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

The design phase is the first step in the development of any designed system or product. A well-executed design, which is a creative process, is essential to an effective system. It requires using a range of methodologies and ideas to thoroughly characterize a process or system to enable its real execution. Regardless of the development approach selected, the software design step is essential. It aims to create the architectural detail required to construct a system or product and acts as the software engineering process's technical foundation. This program went through a careful design process that maximizes every area of its performance, both effectiveness and precision. A document for users is created from a user-oriented document. During the design phase, programmers or database staff may be involved.

4.2 UML DIAGRAM

Program frameworks are conceptualized, described, planned, and illustrated using a standardized language called Unified Modelling Language (UML). UML was developed by the Question Management Group (OMG), and the first draft of the UML 1.0 definition was published in January 1997. UML is distinct from programming languages like Java, C++, and COBOL. It might be a non-exclusive pictorial language used for program outlines as well as a visual showcasing language for computer program frameworks. UML is typically used to communicate with software frameworks, but it may also be used for non-software frameworks like creating forms.

UML Diagram includes,

- Use case diagram
- Sequence diagram
- State chart diagram
- Activity diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram
- Collaboration diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of how clients and other external onscreen characters are connected to internal system components. Understanding, organizing, and coordinating a system's utilitarian requirements as viewed through the eyes of its users is the fundamental task of a use case diagram. Use case diagrams are typically created using the Unified Modelling Language (UML), a standard language for modeling real-world objects and systems.

Use cases can be used to accomplish a variety of framework goals, such as establishing fundamental requirements, validating equipment plans, testing and researching programs, developing online provide help references, or fulfilling client support duties. Customer service, product sourcing, catalog updating, and payment processing are practically a few examples of use cases in the context of item deals. A use case diagram is made consisting of the system boundaries, actors, use cases, and their linkages. The system boundary determines the boundaries of the system in relation to its surroundings. Actors are frequently categorized based on the roles they take on and how those roles represent the individuals or systems that interact with the system. Use cases are the specific actions or behaviors that actors take when using the technology or while nearby. The image also depicts the relationships between actors and use cases in addition to the use cases themselves.

Use case diagrams are visual representations that are used to record a system's functional needs. It's crucial to adhere to these recommendations while creating a use case diagram to create a successful and efficient diagram.

- Give use cases evocative titles that appropriately describe the functionalities they carry out.
- Give actors proper names to assist users understand their functions within the system.
- Double-check the diagram to make sure all connections and dependencies are shown clearly.
- The key objective is to determine the core criteria; avoid listing every relationship that could possibly exist.
- When required, use notes to clarify crucial points

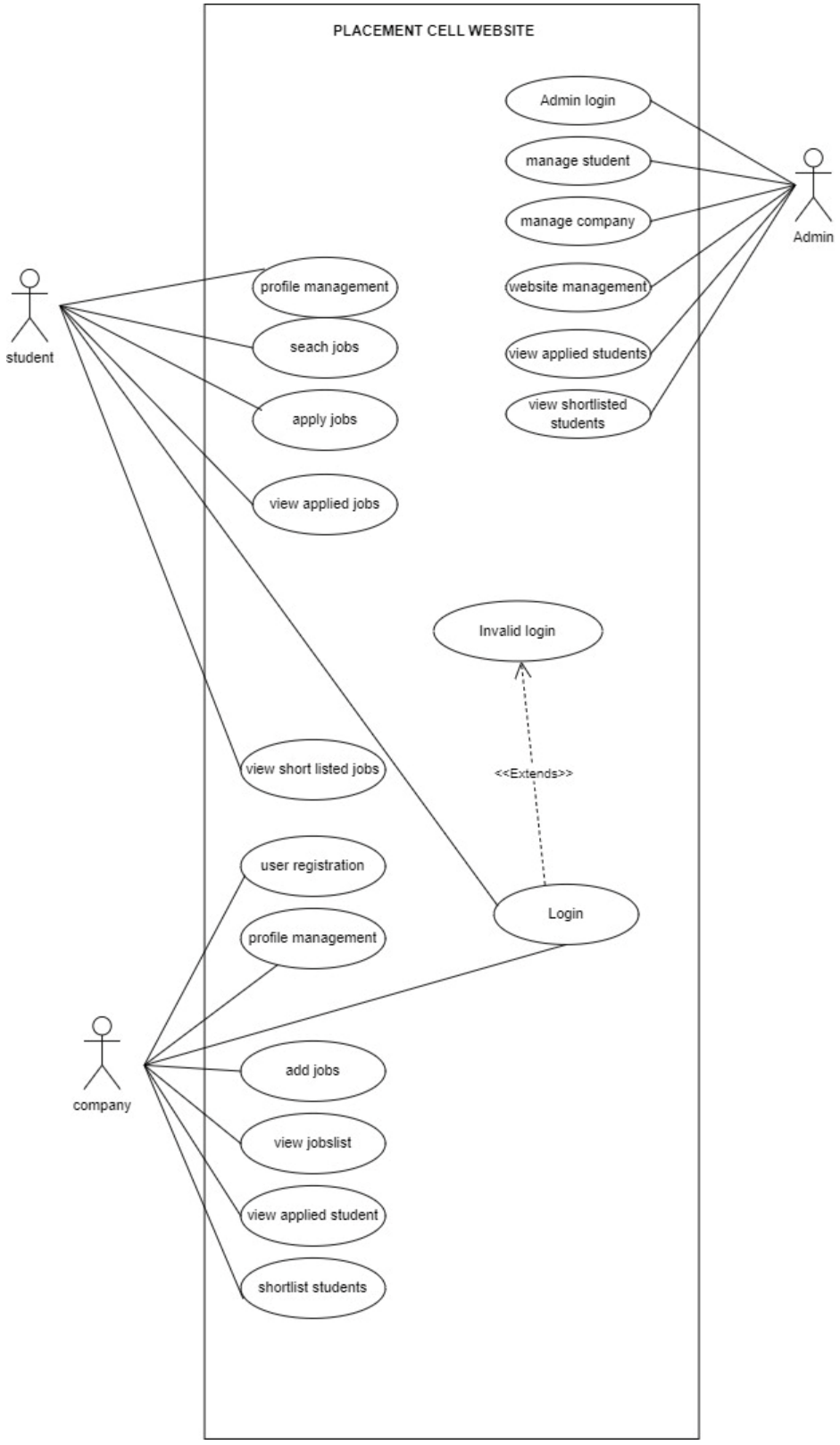


Fig 4.2.1 use case diagram

4.2.2. SEQUENCE DIAGRAM

A sequence diagram, which is a type of interaction diagram, displays the interactions between different system components in chronological order. It shows how various things communicate. Over a series of communications, they communicated with one another. These pictures are occasionally referred to as event scenarios or representations of event scenarios. Sequence diagrams are a common tool in software engineering, utilized regularly to explain and understand the requirements of both new and existing systems. They back the identification of systemic problems and the depiction of object control linkages. This includes,

- Actors - In UML, an actor is a role that interacts with the system and its objects.
  Actors frequently exist outside of the system represented by the UML diagram. A variety of roles, such as those of external subjects or human users, can be played by actors. Actors are depicted in UML diagrams using a stick person notation. There may be more than one actor in a sequence diagram, depending on the circumstance being depicted.
- Lifelines - In a sequence diagram, a lifeline is a vertical dashed line that reflects an object's lifetime during the interaction. Each lifeline is labeled with the participant's name and serves to symbolize a specific participant in the series of events. The lifeline, which is represented as a vertical line running from the participant's activation point to its deactivation point, displays the participant's history of events.
- Messages – A crucial part of sequence diagrams, messages depict the interactions and communication between objects or components in a system. Synchronous and asynchronous communications, create and delete messages, self-messages, reply messages, found messages, and lost messages are some of the categories they fall under. Guards are often used to simulate constraints on message flow.
- Guards – In UML, guards are used to describe conditions and are used to limit the flow of messages when a specific condition is satisfied. This function is crucial for informing software developers of any restrictions or limitations related to a system or specific process.

Sequence diagram uses include:

- Modeling and illuminating the logic of intricate activities, processes, or procedures.
- Displaying information from UML use case diagrams.
- Being aware of the precise workings of present or upcoming systems.

- Visualizing the flow of information between system's various elements or objects.
- In general, sequence diagrams are helpful for depicting the course of interactions between system objects and can assist both businesspeople and software engineers in better understanding and communicating system needs and behavior.

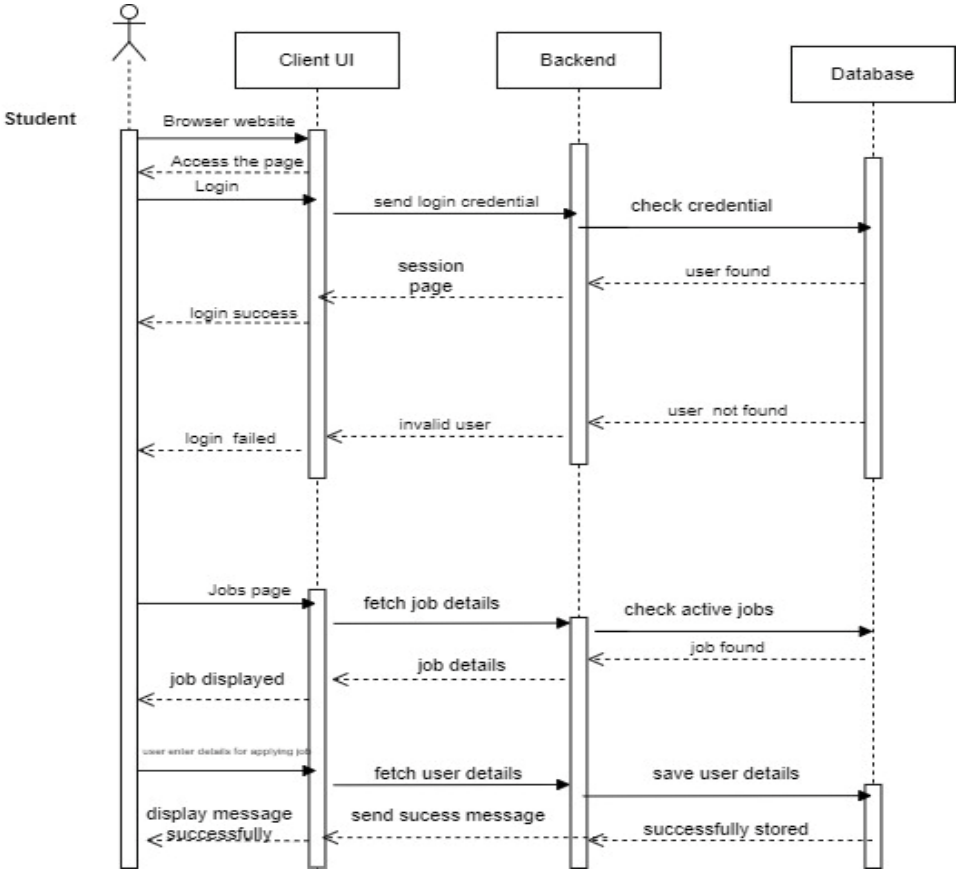


Fig 4.2.2 sequence diagram

4.2.3 State Chart Diagram

The different states and transitions of an object in a system are represented by a State chart Diagram, a form of diagram used in the Unified Modeling Language (UML). It's particularly helpful for simulating the behavior of a system or object that has multiple states and can change between them in response to specific events. Here are some important ideas regarding state chart diagrams:

- States: State chart diagrams list the potential conditions in which an object may be. These states stand for various manifestations or forms of the object. The states in a traffic light system, for instance, might be "Red," "Yellow," and "Green."
- Transitions: Transitions reflect the change in a state of an object. Events or conditions cause these transitions to occur. A timer or a button for pedestrian crossings could cause the traffic light example's "Green" to "Red" transition to occur.
- Events: External happenings or stimuli are what cause states to change. Events include things like human input, sensor readings, and the passage of time. For instance, hitting a floor button in an elevator control system is an event that initiates a transition to move the elevator.
- Actions: When a state or a transition occurs, an action takes place to indicate what happens to an item in that state or at that time. The system's behavior can be defined by these actions. For instance, when the "Dispense" transition takes place in a vending machine, the selected item is dispensed.
- Initial and Final States: Initial and final states are frequently present in state chart diagrams. While the final state reflects the object's finish or termination point, the initial state represents the object's beginning.

State chart diagrams offer a visual way to represent the behavior of complicated systems, making it simpler to comprehend and explain how various system components interact. They are particularly useful in control systems and software engineering

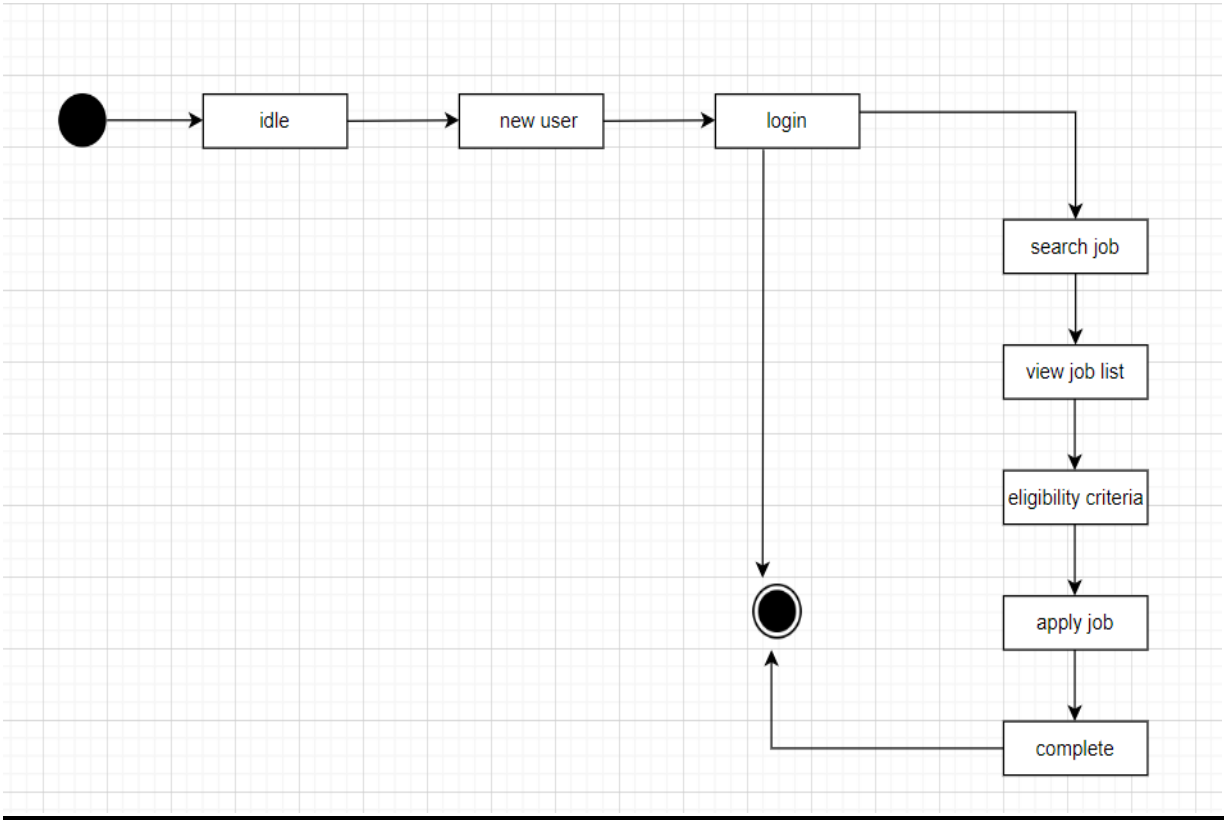


Fig 4.2.3 state chart diagram

4.2.4 Activity Diagram

A sort of Unified Modeling Language (UML) flowchart called a "activity diagram" depicts the progression of one activity across a system or process. It is referred to as a "behavior diagram" because it outlines what ought to occur in the modeled system and is used to depict the various dynamic characteristics of a system.

Activity diagrams can be used to visualize even extremely complicated systems. As a result, activity diagrams are frequently employed in organizational business process modeling or to outline the processes of a use case diagram. They display each step in an activity along with their relative timing. They can also display the data flow between several activities.

The process is depicted in activity diagrams from the beginning (the initial state) to the finish (the final state). There are actions, decision nodes, control flows, start nodes, and end nodes in every activity diagram.

- Initial node - A black circle designates the activity diagram's starting point.
- Activity - An action or task carried out by the entity or system, symbolized by a rectangle with rounded corners.
- Control flow: Depicted by an arrow, it shows the order in which a system or entity performs its various operations.
- choice node - A diamond-shaped node that represents a choice or branching point in the activity flow.
- Merge node - Displayed as a diamond-shaped node with a plus sign inside it, this node is used to combine many activity flow branches into a single flow.
- Fork node - Used to break the activity flow into many parallel flows, represented by a solid black circle with multiple arrows.
- Join node—shown as a solid black circle with numerous arrows pointing at it—is used to combine multiple parallel flows into a single flow.
- Final node - The activity diagram's conclusion, represented by a black circle with a dot inside.
- Object flow - A dashed arrow symbolizes the movement of items or data between activities.

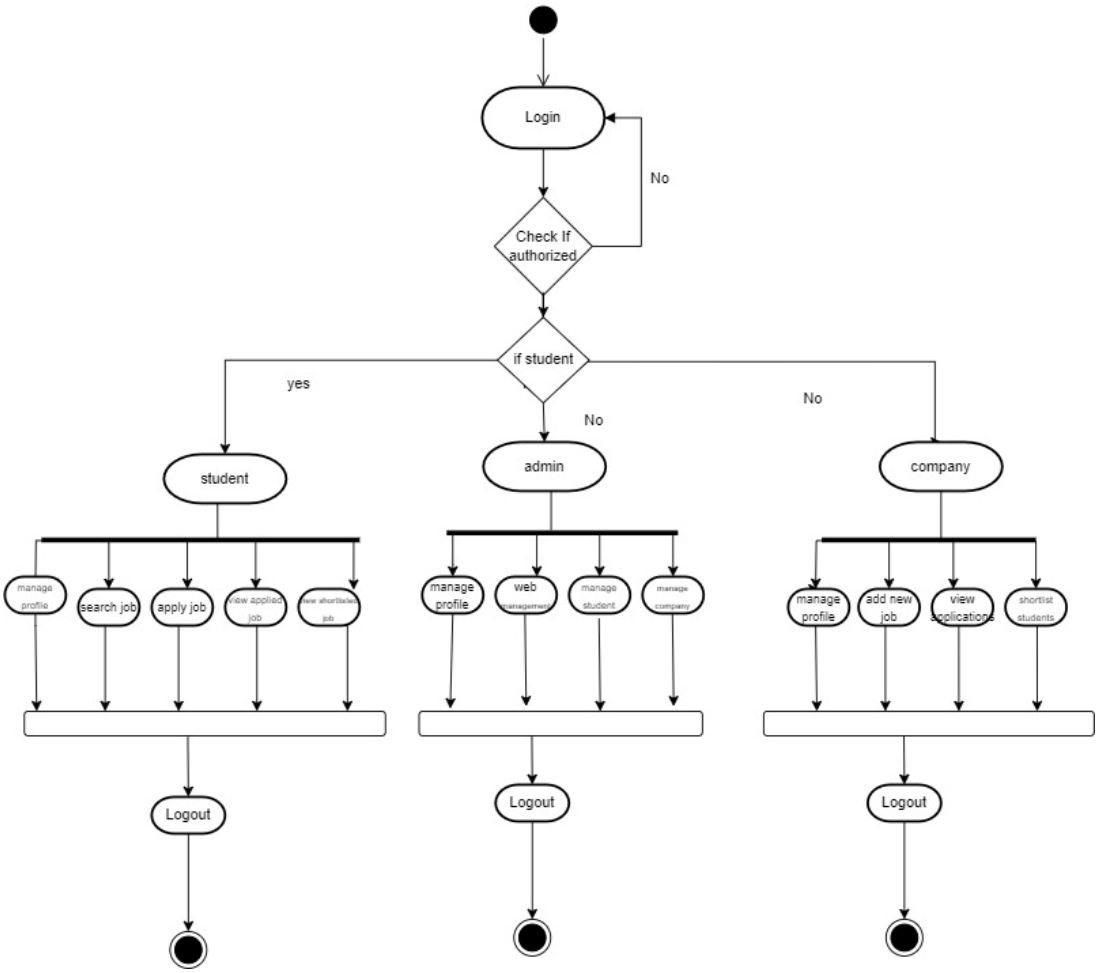


Fig 4.2.4 activity diagram

4.2.5 Class Diagram

A static view of an application is depicted in the class diagram. It depicts the different kinds of objects that are present in the system as well as their interactions. In addition to having its own objects, a class can also inherit from other classes. Various distinct aspects of the system are visualized, described, documented, and executable software code is also created using class diagrams. It displays the classes, relationships, properties, and functions to provide a summary of the software system. In a separate section, it organizes class names, characteristics, and functions to aid in software development. It is known as a structural diagram since it consists of a number of classes, interfaces, affiliations, collaborations, and constraints.

- Class: A rectangle with the class name, attributes, and methods is used as a blueprint or template for constructing objects.
- Interface: An interface is a group of abstract methods that define the terms of the agreement between a class and the outside world. It appears as a circle with the name of the interface inside.
- Object: It is a state- and behavior-filled instance of a class. The object name is enclosed inside a rectangle for representation.
- Association: An association is a connection or link between two classes that is shown as a line with optional directionality, multiplicity, and role names.
- Aggregation: On the aggregator side, the relationship is depicted as a diamond shape, with the whole (aggregator) made up of pieces (aggregates).
- Composition: On the aggregator side, it is shown as a filled diamond shape and is a stronger kind of aggregation where the components cannot exist without the whole.
- Inheritance: This relationship between a superclass and its subclasses is symbolized by a line with an open arrowhead pointing from the subclass to the superclass. It indicates a "is-a" relationship.
- Dependency: This relationship, which is shown as a dashed line with an arrowhead pointing from the dependent class to the independent class, is one in which a change in one class may have an impact on the other class.
- Multiplicity: A range of values close to the association or aggregation line is used to describe the number of instances of a class that can be related with another class.

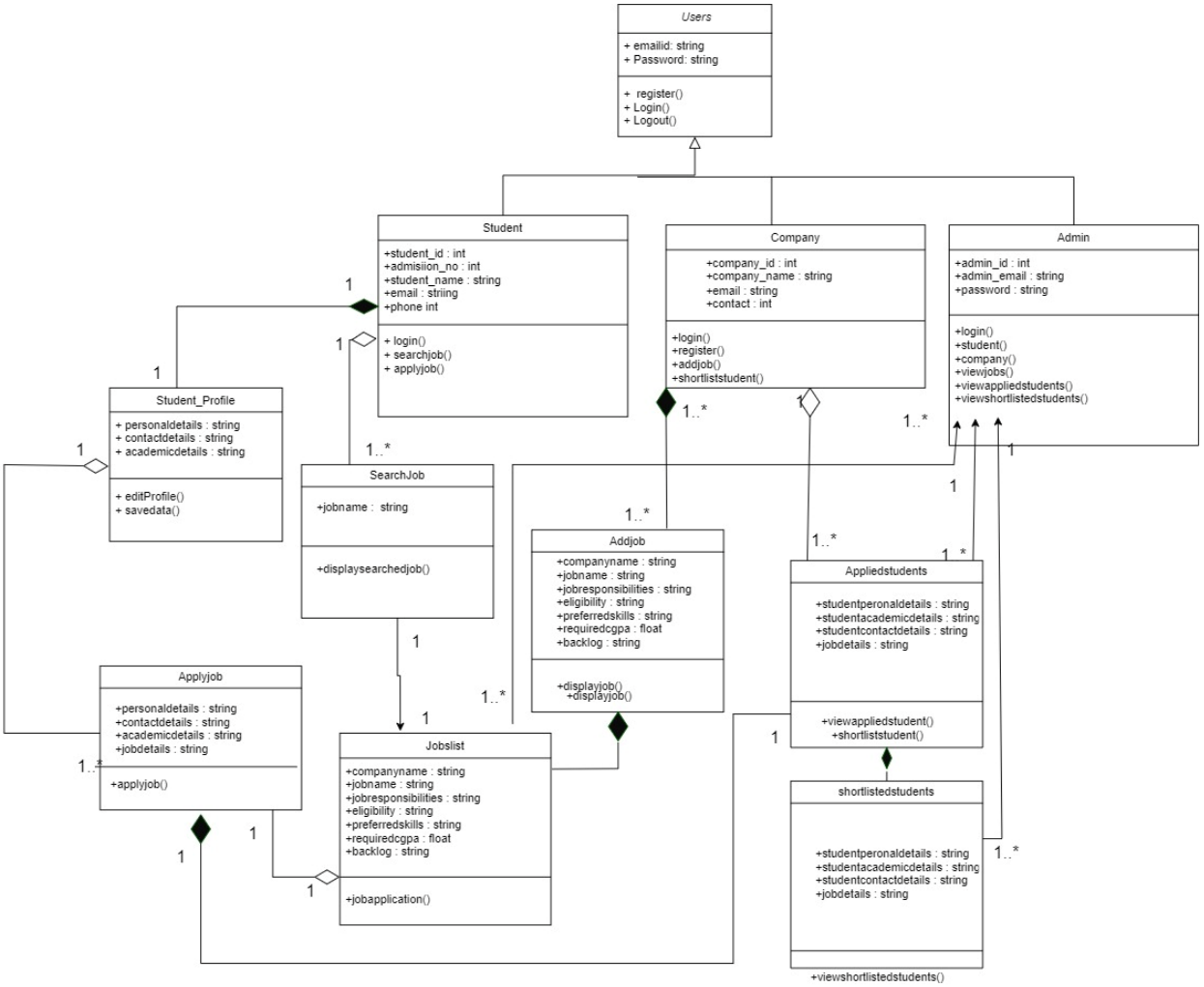


Fig 4.2.5 class diagram

4.2.6 Object Diagram

Object diagrams help to visualize the relationships between objects and their attributes in a system. They are useful for understanding the behavior of a system at a specific point in time and for identifying potential issues or inefficiencies in the system. A screenshot of the instances in a system and the relationships between them can be found in an object diagram. We are able to investigate the behavior of the system at a specific instant since object diagrams show behavior when objects have been instantiated. To represent and comprehend the functional requirements of a system, object diagrams are essential. The definition of an object diagram in the Unified Modeling Language (UML) is, in other words, "a diagram that shows a complete or partial view of the structure of a modeled system at a specific time."

- Object: An object is an instance of a class that stands in for a particular system entity. The object name is enclosed inside a rectangle for representation.
- Class: A class is an outline or model for building objects that specifies the characteristics and methods of the object. The class name, attributes, and methods are displayed in three compartments within a rectangle.
- Link: A link is a connection or association between two objects that exists between them. It appears as a line joining two items with omitted labels.
- Attribute: A property or characteristic of an item that describes its state is called an attribute. It is displayed inside the object rectangle as a name-value pair.
- Value: A value is a particular use or configuration of an attribute. Within the attribute name-value pair, it is shown as a value.
- Operation: An item can engage in a behavior or action known as an operation. Inside the class rectangle, it appears as a method name.
- Multiplicity: The quantity of instances of one class that may be linked to another class is referred to as multiplicity. It is shown as a range of values close to the link between objects (e.g., 0..1, 1..*, etc.).

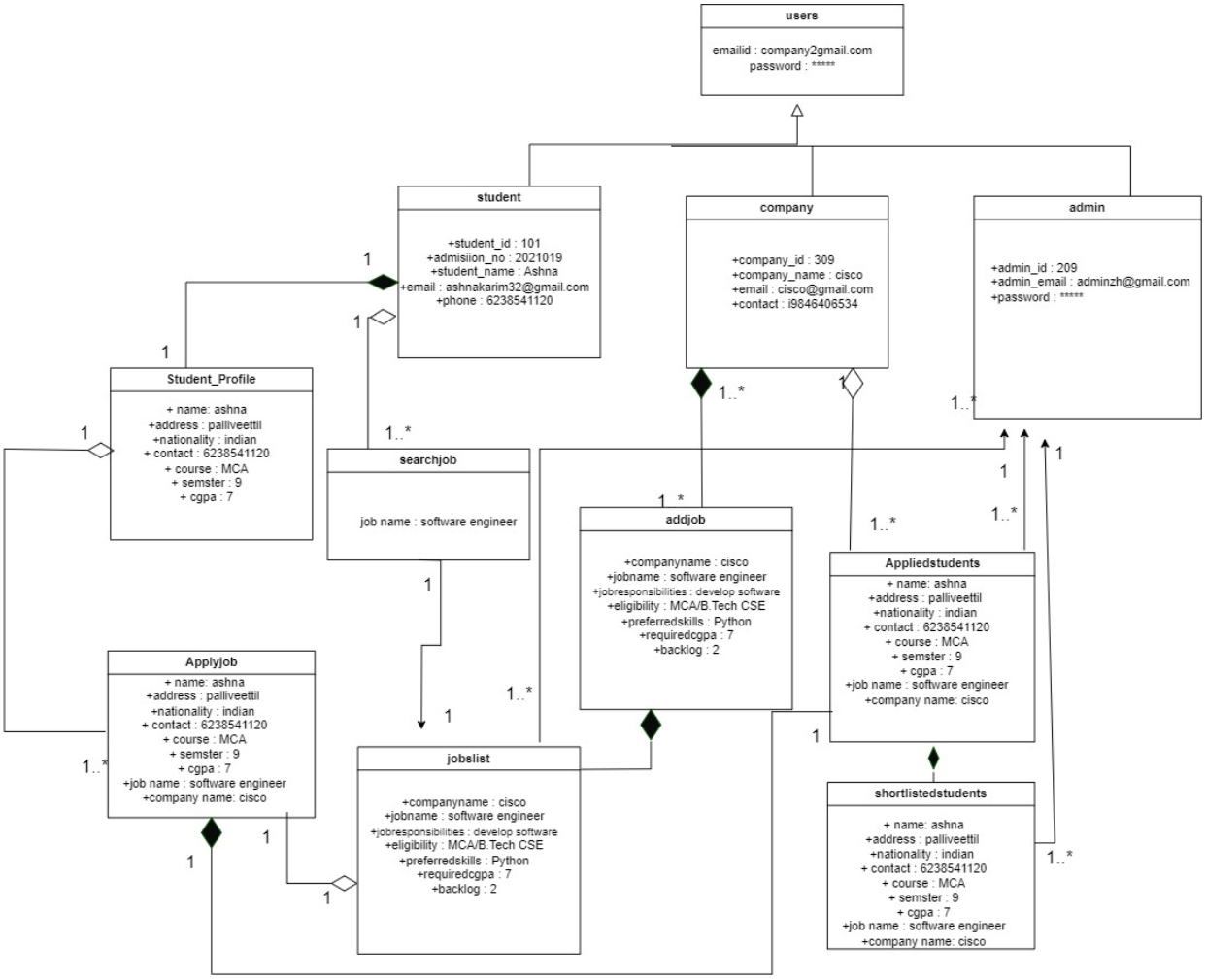


Fig 4.2.6 object diagram

4.2.7 Component Diagram

To make a complex object-oriented system more understandable, the smaller components are separated out using a component diagram. It simulates the physical view of a system, including its internal node's executables, files, libraries, etc. It depicts the connections and hierarchies that exist between the system's components. It aids in creating an operational system. An individual, replaceable, and executable system unit is referred to as a component. A component's implementation details are concealed; therefore, an interface is required to carry out a function. It functions like a "black box," with the provided and necessary interfaces explaining its behavior. When modeling the architecture of a software system, component diagrams are helpful because they can point out potential problems and suggest design changes. They can also be used to explain a system's behavior and structure to stakeholders, including programmers and project managers.

- Component: A module-based, functionally-encapsulated unit of a system that provides interfaces for interacting with other components. It looks like a rectangle with the component name inside of it.
- Interface: An agreement between a component and its surroundings or other components that outlines a group of methods that other components may utilize. It appears as a circle with the name of the interface inside.
- A port is a point where a component interacts with the environment or other components.
- It appears as a little square on a component's edge.
- Connector: An interface that allows communication or data exchange between two components.
- It is shown as a line with optional labels and adornments.
- Dependency is a relationship between two components in which one depends on the other for functionality or implementation. A dashed line with an arrowhead pointing from the dependent component to the independent component serves as its visual representation.
- Association: A connection or link between two elements in a relationship.
- With optional directionality, multiplicity, and role names, it is shown as a line joining two components.
- Supplied/needed Interface: A supplied interface is one that a component provides to other components, whereas a needed interface is one that a component requires from other components in order for it to operate effectively. Half-circles and lollipops, respectively, are used to depict this.

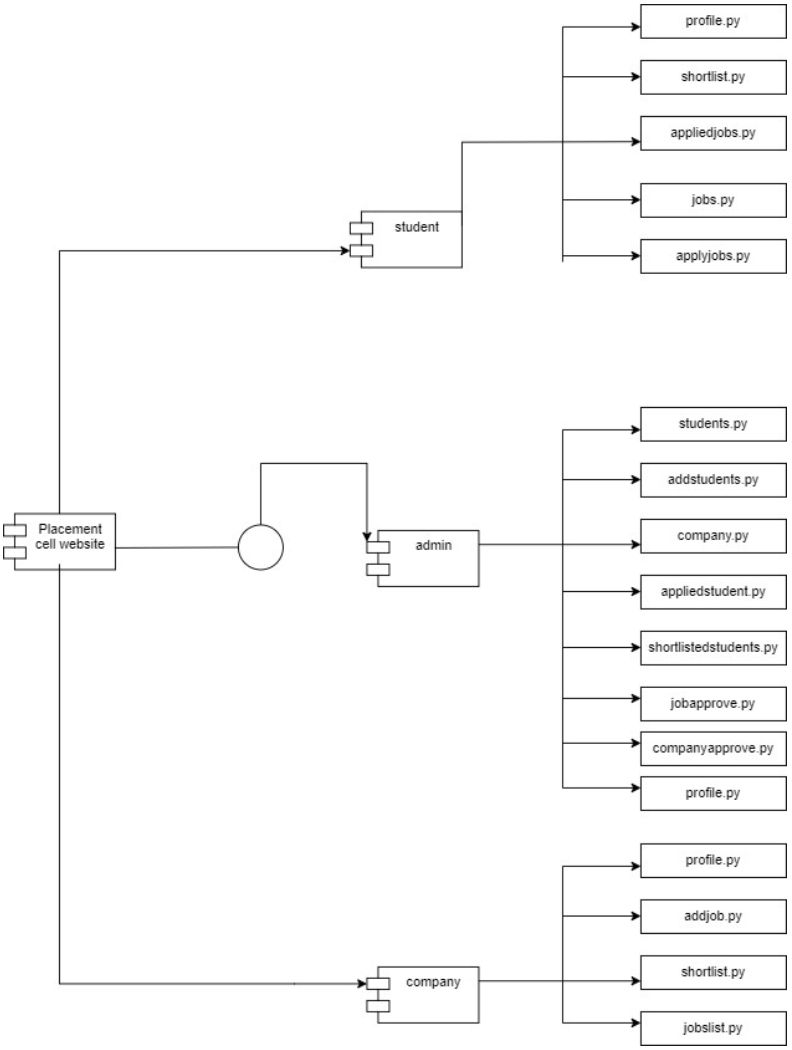


Fig 4.2.7 component diagram

4.2.8 Deployment Diagram

For depicting the physical components where software components are delivered, deployment diagrams are employed. Deployment diagrams and component diagrams have a lot in common. Diagrams of the components are used to describe them, and diagrams of their deployment in hardware are shown. UML is primarily made to concentrate on a system's software artifacts. These two diagrams, however, are unique ones that highlight the hardware and software components. Deployment diagrams are meant to concentrate on the hardware topology of a system, whereas most UML diagrams are used to handle logical components. The system engineers make use of deployment diagrams. The physical architecture of a system can be visualized using deployment diagrams, which also aid in spotting any possible problems or deployment process bottlenecks. They also support deployment planning. A plan and maximizing the utilization of the available hardware resources.

- Node: A node is a deployed component or artifact that can be either a physical or virtual machine.
- It is shown as a box with the name of the node inside.
- Component - A component is a piece of software that carries out a particular task or offers a particular service. It appears as a rectangle with the name of the component inside.
- Artifact - An artifact is a tangible piece of data that a component uses or creates. It is symbolized by a rectangle with the name of the artifact inside.
- Deployment Specification - A deployment specification outlines the steps involved in deploying a component or artifact on a node. It contains details about the component or artifact's location, version, and configuration settings.
- Association - A relationship between a node and a component or artifact that symbolizes a deployment dependence is known as an association. A line joining the two components with possible directionality, multiplicity, and role titles serves as its visual representation.
- Communication Path - The link between nodes, such as a network connection or communication channel, is represented by a communication path. It is symbolized by a line with supplemental labels and decorations.

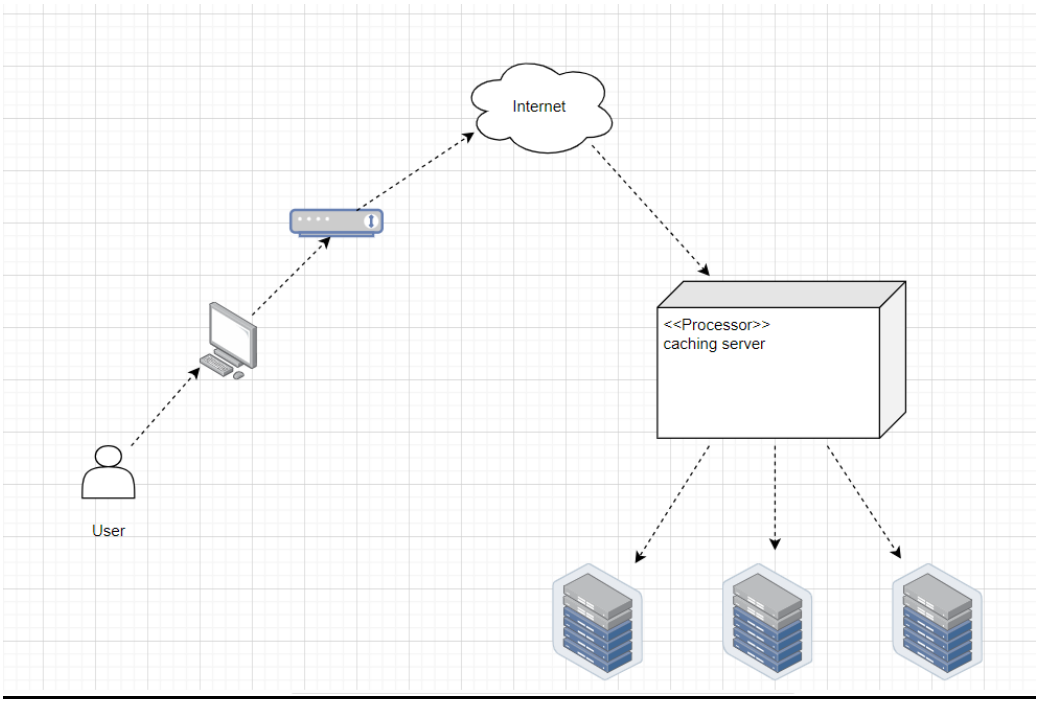


Fig 4.2.8 deployment diagram

4.2.9 Collaboration Diagram

The relationship between the objects in a system is depicted using the cooperation diagram. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object is made up of various features. Multiple objects present in the system are connected to each other. The object's architecture within the system is shown using the collaboration diagram, also referred to as a communication diagram.

- Objects: Symbols that represent objects include their name and class underlined and separated by a colon. Objects are used to represent a class instance and indicate its name and class in a collaboration diagram. Every class does not necessarily need to have an object. A single class may have several objects. Objects are initially formed, and then their class is determined. It's crucial to give objects names in order to distinguish them from one another.
- Actors: Since they initiate the interaction, actors are important in the cooperation diagram.
- Each actor has a unique name and function. One actor starts the use case in the diagram.
- Links: Actors and objects are linked together by links, which are examples of association. They stand in for the connection between items that conveys signals. Links, which are shown as solid lines, allow items to move between one another.
- Messages: Identified by a sequence number, messages are a representation of communication between objects that contain information. They are represented by labelled arrows that are sent from the sender to the receiver and put close to the link. The receiver must comprehend the message, and the direction must be navigable in that particular direction.

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Index

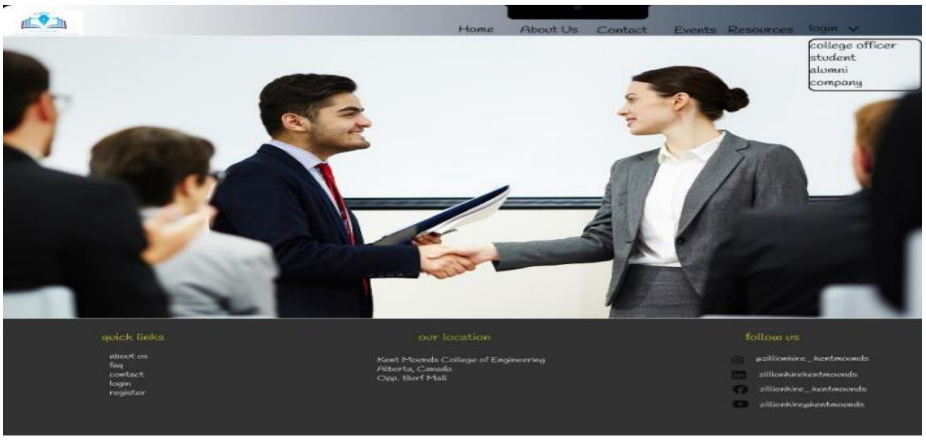


Fig 4.3 figma_index

Form Name: About

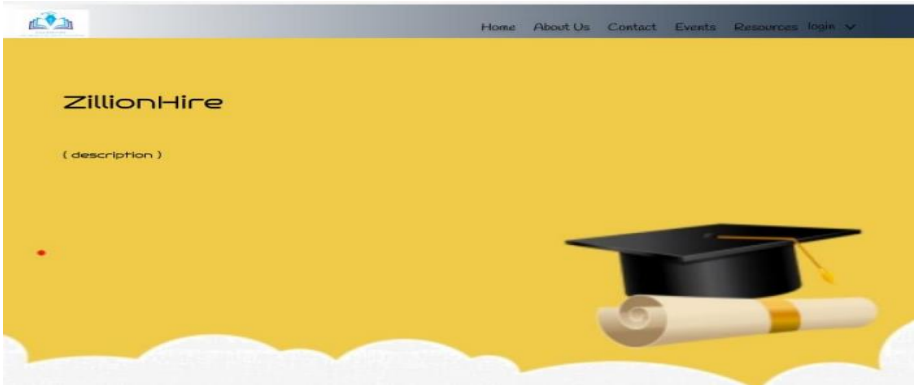


Fig 4.3 figma_about

Form Name: Login

Fig 4.3 figma_login

Form Name: Student index

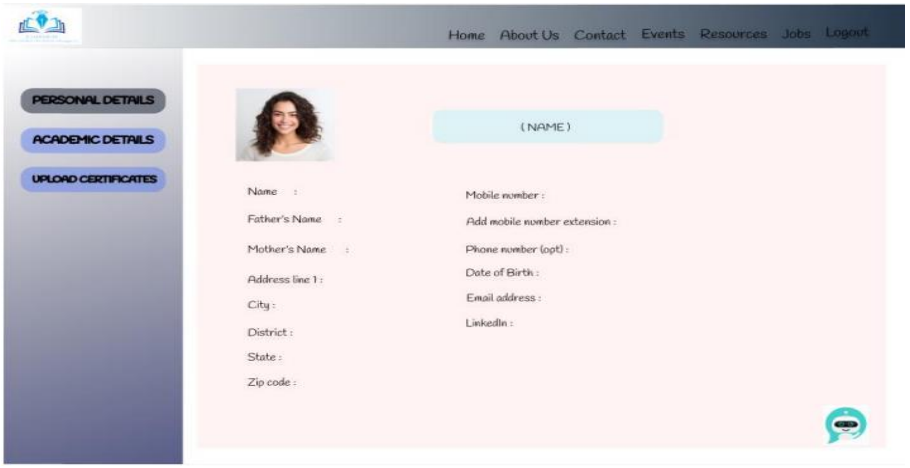


Fig 4.3 figma_studentindex

Form Name: Student Page2

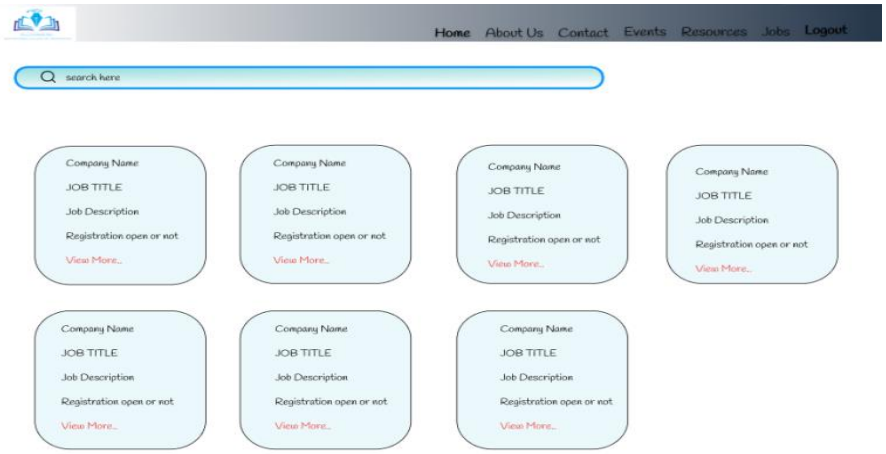


Fig 4.3 figma_studentpage2

Form Name: Admin Index

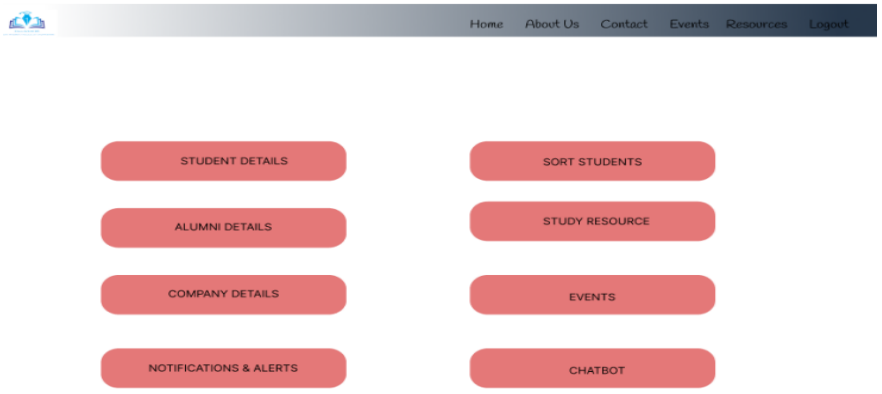


Fig 4.3 figma admin index

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

A common kind of database, known as a relational database management system (RDBMS), arranges data into tables to make connections with other stored data sets easier. The rows in tables, which are each referred to as records, can range in number from a few hundred to millions. A column heading is an attribute, a row is referred to as a tuple, and the table is referred to as a relation in formal relational model language. Multiple tables with unique names make up a relational database. A set of associated values is represented by a row in a table.

Relationships between tables are pre-established in relational databases to guarantee the consistency of both referential and entity relationships. Choosing a data type from which the domain's data values are derived is a popular technique to define a domain, which is a collection of atomic values. Giving the domain a name makes it simpler to comprehend the values it contains. A relation's values are all atomic and cannot be further subdivided.

Keys are used in relational databases to create table associations, with the main key and foreign key being the two most crucial types. These keys can be used to create relationships between entity integrity and referential integrity. Referential integrity, on the other hand, makes sure that every unique foreign key value has a corresponding primary key value in the same domain. In addition, there are many key types, including super keys and candidate keys.

4.4.2 Normalization

A data preprocessing method called normalization is used in statistics and machine learning to scale numerical features to a standard range. It guarantees that the data is scaled consistently, preventing some variables from influencing the modeling process more than others. The following are some essential ideas for data normalization:

The goal of normalization is to give a dataset's features a standard scale, frequently between 0 and 1, by transforming them. This is crucial for machine learning algorithms like gradient-based optimization techniques that are sensitive to the scale of features.

Common Normalization procedures: Min-Max scaling, Z-score (Standardization), and Robust scaling are a few examples of normalization procedures. While Z-score normalization changes features to have a mean of 0 and a standard deviation of 1, min-max scaling maps features to a specific range.

Impact on Random Forest: Random Forest is a decision tree ensemble-building machine learning technique. It is not affected by feature normalization or scaling. Without taking into account the absolute scale of the features, Random Forest makes selections based on feature thresholds.

Therefore, while utilizing a Random Forest technique, you often do not need to normalize your data. The use of normalization is crucial when utilizing machine learning algorithms that are sensitive to feature scales, such as Support Vector Machines and k-Nearest Neighbors. Random Forest does not require normalization.

In database architecture, normalization is a technique that seeks to properly arrange data into tables and columns so that the user may readily connect it to the data. This procedure gets rid of redundant data, which can be a drain on computer resources. The primary phases in normalization are: normalizing the data, selecting acceptable table and column names, and selecting accurate names for the data. A developer can construct a database that is easier to administer and maintain by following these steps to make it more effective and organized.

First Normal Form

According to the First Normal Form (1NF), a table's attributes can only have atomic or indivisible values. The use of nested relations, or relations inside relations, as attribute values in tuples is forbidden. Data must be transferred into distinct tables with data of a comparable type in each table to satisfy 1NF, and each table must have a primary key or foreign key depending on the project's specifications. For each non-atomic property or nested relation, this technique generates new relations, eliminating repetitive sets of data. Only when a relation complies with the constraints requiring the primary key to be present alone is it deemed to be in 1NF.

Second Normal Form

According to the second normal form (2NF) rule of database normalization, non-key properties in a relation with a composite primary key should not be functionally dependent on just one portion of the primary key. To put it another way, every non-key attribute should be dependent on the complete main key, not just a portion of it. To accomplish this, the table must be broken down, and new connections must be created for each subkey and the dependent attributes. The relationship with the original primary key and all properties that are entirely functionally dependent on it must be preserved. Only when a relation meets all of the 1NF requirements for each non-primary key and the primary key is it considered to be in 2NF.

Third Normal Form

A relation must not have a non-key attribute that is functional in order to satisfy the third normal form (3NF) another non-key attribute or group of non-key attributes determines. Therefore, there should there won't be any transitive reliance on the main key. We breakdown the relation and to reach 3NF. Create a new relation with non-key qualities that serve as the functional determinants of

other non-key attributes. This aids in removing any dependencies that include more than simply the primary key. a connection is regarded as a relation in 3NF if it meets the requirements of 2NF as well as the non-key attributes of the relation are not dependent on any other non-key attribute

4.4.3 Sanitization

Data sanitization is the process of removing any illegal characters or values from data. External data from a variety of sources, such as user input from forms, cookies, web services data, server variables, and database query results, is frequently included into online applications. It is crucial to sanitize all external input to guarantee its security and ensure that it is free of any harmful code or values.

4.4.4 Indexing

A database structure called an index accelerates table operations. For streamlined record ordering and speedy lookups, indexes can be built on one or more columns. The columns that will be utilized in SQL queries should be taken into account when establishing an index, and one or more indexes should be created on those columns. The primary key or index field plus a pointer to each item in the real table are stored in a specific sort of table called an index. Users cannot see indexes; the database search engine alone uses them to find records quickly. Table indexes are created using the CREATE INDEX statement. The INSERT and UPDATE operations take longer when a table has indexes because the database also needs to insert or update the index values. However, because the index makes it easier for the database to locate records, the SELECT operations on those tables become faster.

4.5 TABLE DESIGN

1.Tbl_login

Primary key: **login_id**

Foreign key: **login_id** references table **Tbl_student profile ,Tbl_company profile**

No	Field name	Datatype	Key constraint	Description
1	login_id	IntegerField	Primary key	Id of login
2	registration_id	IntegerField	Foreign key	Registration id
3	email	EmailField	Not null	User email

2. Tbl_company registration

Primary key: **registration_id**

Foreign key: **registration_id** references table **Tbl_login, Tbl_companyprofile**

No	Field name	Datatype	Key constraint	Description
1	registration_id	IntegerField	Primary key	Registration Id
2	company_name	CharField(100)	Not null	Name of company
3	contact	IntegerField	Not null	Contact of company
4	email	EmailField	Not null	Email of company

3. Tbl_adminstudent

Primary key: **adminstu_id**

Foreign key: **adminstu_id** references table **Tbl_studentprofile**

No	Field name	Datatype	Key constraint	Description
1	adminstu_Id	IntegerField	Primary key	Id of Student
2	admission_no	IntegerField	Not null	Admission number of student
3	phone	IntegerField	Not null	Contact of student
4	firstname	CharField(100)	Not null	First name of student
5	lastname	CharField(100)	Not null	Last name of student
6	email	EmailField	Not null	Email of student
7	is_active	CharField(100)	Not null	Showing student is existing or not

4. Tbl_companyprofile

Primary key: **company_id**

No	Field name	Datatype	Key constraint	Description
1	company_id	IntegerField	Primary Key	Id of company
2	registration_id	IntegerField	Foreign key	Registration id
3	company name	CharField(100)	Not null	Company name
4	contact	IntegerField	Not null	Contact of company
5	addressline 1	CharField(300)	Not null	Address of company
6	is_active	IntegerField	Not null	Showing is existing or not
7	companydp	CharField(100)	Not null	Company logo
8	is_approved	IntegerField	Not null	Showing company is approved or not

5. Tbl_jobapplication

Primary key: **jobapplication_id**

No	Field name	Datatype	Key constraint	Description
1	jobapplication_id	IntegerField	Primary key	Id of job application2
2	studentprofile_id	IntegerField	Foreign key	Student profile id
3	job_id	IntegerField	Foreign key	Id of job
4	c_institution	CharField(100)	Not null	Current institution
5	c_university	CharField(100)	Not null	Current university
6	crime	CharField(10)	Not null	Student crime data
7	doc	DateField	Not null	Date of crime
8	dtoc	CharField(500)	Not null	Details of crime
9	nature	CharField(100)	Not null	Nature of crime
10	workexperience	CharField(200)	Not null	Showing student has any work experience
11	period	CharField(100)	Not null	Period of previous work
12	companydetails	CharField(100)	Not null	Previous company details
13	skills	CharField(300)	Not null	Student skills
14	newcourse	CharField(100)	Not null	Additional course completed
15	newcert	FileField	Not null	Certificate of course
16	is_approved	IntegerField	Not null	Showing job application is approved or not

6. Tbl_studentprofile

Primary key: **studentprofile_id**

Foreign key: **studentprofile_id** references **Tbl_jobapplication**

No	Field name	Datatype	Key constraint	Description
1	studentprofile_id	IntegerField	Primary key	Admission no. of student
2	adminstu_id	IntegerField	Foreign key	References to adminstu_id
3	first name	CharField(100)	Not null	First name of student
4	last name	CharField(100)	Not null	Last name of student
5	dob	DateField	Not null	Date of birth of student
6	gender	CharField(700)	Not null	Gender of student
7	nationality	CharField(100)	Not null	Nationality of student
8	religion	CharField(100)	Not null	Religion of student
9	profile_photo	FileField	Not null	Profile photo of student
10	email	CharField(100)	Not null	Email of student
11	area_code	CharField(100)	Not null	Area code of contact
12	phone	IntegerField	Not null	Contact number of student
13	present_address	CharField(300)	Not null	Present address of student
14	permanent_address	CharField(300)	Not null	Permanent address
15	academic_year	CharField(100)	Not null	Current academic year
16	tenth_institution	CharField(100)	Not null	Tenth institution
17	tenth_board	CharField(100)	Not null	Tenth board of education
18	tenth_cgpa	FloatField	Not null	Tenth cgpa
19	twelfth_institution	CharField(100)	Not null	Twelfth institution
20	twelfth_board	CharField(100)	Not null	Twelfth education board

21	twelfth_cgpa	FloatField	Not null	Twelfth cgpa
22	twelfth_certtificate	FileField	Not null	Twelfth certificate
23	ug_institution	CharField(100)	Not null	UG institution
24	ug_course	CharField(100)	Not null	UG course
25	ug_board	CharField(100)	Not null	UG university
26	ug_cgpa	FloatField	Not null	UG cgpa
27	ug_certificateupload	FileField	Not null	UG certificate
28	c_cgpa	FloatField	Not null	Current cgpa
29	c_backlog	IntegerField	Not null	Active backlog
30	course	CharField(700)	Not null	Current course
31	department	CharField(700)	Not null	Department
32	c_semester	IntegerField	Not null	Current semester

7. Tbl_jobs

Primary key: **job_id**
Foreign key: **job_id** references **Tbl_jobapplication**

No	Fieldname	Datatype	Keyconstraint	Description
1	id	IntegerField	Primary key	Id of job
2	cname	CharField(100)	Not null	Company name
3	jname	CharField(100)	Not null	Job name
4	salary	FloatField	Not null	Salary of job
5	email	EmailField	Not null	Email of company
6	sdate	DateField	Not null	Application start date
7	edate	DateField	Not null	Application end date
8	link	CharField(100)	Not null	Company website
9	job_description	CharField(500)	Not null	Job description
10	preferred_skills	CharField(500)	Not null	Preferred skills for job
11	qualifications	CharField(500)	Not null	Eligibility for applying job
12	responsibilities	CharField(500)	Not null	Job responsibilities
13	required_current_cgpa	FloatField	Not null	Required cgpa of current course
14	required_tenth_cgpa	FloatField	Not null	Required cgpa of tenth
15	required_twelfth_cgpa	FloatField	Not null	Required cgpa of twelfth
16	required_backlog	IntegerField	Not null	Minimum backlog
17	is_active	IntegerField	Not null	Showing job is existing or not
18	is_approved	IntegerField	Not null	Showing job is approved or not

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is controlled execution of a software program to see if it behaves as intended, frequently utilizing verification and validation techniques. Verification might comprise evaluations, analyses, inspections, and walkthroughs whereas validation entails reviewing a product to confirm that it meets with specifications. While dynamic analysis investigates the software's behavior while it is running to acquire data like execution traces, timing profiles, and test coverage details, static analysis looks at the software's source code to find problems.

The testing process consists of a number of organized and systematic steps that begin with the integration of individual modules and work their way up to the full computer-based system. The goals of testing include finding flaws and problems in the program, ensuring that it operates in accordance with the software's specifications, and confirming that it satisfies performance standards. Testing can be done to evaluate the accuracy, effectiveness, and computational complexity.A successful test is one that finds an error that has not yet been found, and a good test case has a high likelihood of doing so. To achieve system testing objectives, testing is essential and can comprise a variety of methodologies, including functional testing, performance testing, and security testing.

5.2 TEST PLAN

A test plan is a written document that lists the tasks that must be taken to complete different testing procedures. It offers instructions on the tasks that must be carried out during testing. Computer programs, related documentation, and data structures are produced by software developers. They are in charge of evaluating each program component to make sure it serves the desired objective. It is common to create an independent test group (ITG) to solve problems with self-evaluation. The steps that must be done to complete various testing methods are listed in a test plan, which is a written document.

It provides guidelines for the duties that must be performed during testing. Software developers create computer programs, related documentation, and data structures. They are in charge of assessing each component of the program to make sure it achieves the anticipated result. To alleviate issues with self-evaluation, an independent test group (ITG) is frequently formed.

The different levels of testing include:

• Unit testing

• Integration testing

• Data validation testing

• Output testing

5.2.1 Unit Testing

Unit testing is a type of software testing that concentrates on examining specific parts or modules of the program design. Unit testing's goal is to test the smallest possible piece of software code and make sure it functions as intended. Multiple components can be tested at once during unit testing, which is often white-box oriented. During testing, the component-level design description is utilized as a reference to pinpoint crucial control pathways and potential flaws inside the module's boundaries.

Software testing with a focus on evaluating particular sections or modules of the program design is known as unit testing. The purpose of unit testing is to ensure that the smallest possible piece of software code performs as intended. Unit testing, which is frequently focused on the white box, allows for the simultaneous testing of multiple components. During testing, the component-level design description is leveraged as a reference to locate essential control paths and any defects inside the module's bounds.

Data flow across a module interface must be tested before any additional testing can start. All other tests are useless if data cannot correctly enter and depart the system. Selective analysis of execution pathways is a key task during unit testing to identify probable mistakes and make sure that error handling channels are established to redirect or stop activity when an issue occurs. Finally, boundary testing is done to make sure the software still functions properly when it reaches its boundaries.

When unit testing the Sell-Soft System, each module was treated as a separate entity and put through a range of test inputs. Each module was tested and ran independently after coding to address any problems with the internal logic. All unused code was removed, and it was verified that each module worked well and achieved the required result.

5.2.2 Integration Testing

Creating the program structure while using a methodical methodology known as integration testing running tests to find interface problems simultaneously. The goal is to create a program structure based on the design that utilizes components that have undergone unit testing. The program as a

whole is then tested. Due to the scale of the total application, fixing integration testing problems can be difficult. It makes it challenging to identify the faults' root causes. Once one set of errors is corrected. It's possible for new ones to appear, and the cycle to continue indefinitely. All system modules are integrated after unit testing is finished to look for any inconsistent user interfaces. Any differences across program architectures are ironed out, and a distinct program organization is developed.

5.2.3 Validation Testing or System Testing

In the last level of testing, the entire software system—including all forms, code, modules, and class modules—is tested as a whole. System is a popular name for this black box testing, or testing. Black box testing focuses on evaluating the functional requirements of the program. Using this method, a software developer can build input circumstances that will fully run tests for each software need. Black box testing's primary mistake categories include wrong or missing functionalities, interface issues, data structure issues, or failures when accessing external data, errors in performance, startup, and termination.

5.2.4 Output Testing or User Acceptance Testing

To make sure the system satisfies both user needs and business requirements, user acceptability testing is carried out. To make sure that the program meets their needs and expectations, it is crucial to include end users in the development process. The input and output screen designs are tested with various types of test data during user acceptance testing. To ensure thorough testing of the system, test data preparation is essential. Any faults found during testing are addressed, fixed, and the fixes are recorded for future use.

5.2.5 Automation Testing

Automation testing is a method of software testing that involves running a number of test cases through specialized automated testing software tools. Its main objective is to ensure that the hardware or software performs exactly as intended. Automation testing finds errors, faults, and other problems that could occur while developing a product.

Although some testing procedures, such as functional or regression testing, can be carried out manually, automating the procedure has many advantages. Automation testing uses automated routines to assess the software and can be performed at any time of day. The outcomes are given, and this data can be compared to those from earlier test runs. Ruby, JavaScript, and C# are common programming languages used by automation developers.

5.2.6 Selenium Testing

Selenium is an open-source framework for automated testing that is used to validate web applications on various platforms and browsers. Selenium enables for the production of test scripts in many programming languages such as Java, C#, and Python. While working on a web application that required frequent testing in 2004, Jason Huggins, a developer at Thought Works, created Selenium. To automate browser activities and increase testing effectiveness, he developed the JavaScript tool "JavaScriptTestRunner". Since then, a group of collaborators has continued to develop selenium.

Web applications are validated using the open-source Selenium framework for automated testing across a range of platforms and browsers. The creation of test scripts in a variety of computer languages, including Java, C#, and Python, is made possible by Selenium. Jason Huggins, a developer at Thought Works, built Selenium in 2004 while working on a web application that required frequent testing. He created the JavaScript tool "JavaScriptTestRunner" to automate browser functions and boost testing efficiency. Since then, a team of researchers has carried out additional selenium research.

To combine the advantages of both tools, Cucumber and Selenium can be merged. Cucumber offers a formal framework for planning and running tests, while Selenium is used to interface with web browsers and automate browser behaviors. Using a business-readable and maintainable style, this combination enables the implementation of end-to-end tests that confirm the functionality of online applications across various browsers and platforms.

Test Case 1- Company Login

Code

```
package stepdefa3;

import org.openqa.selenium.By;

public class loginstepa3 {

    WebDriver driver = null;

    @Given("browser is open")
    public void browser_is_open() {

        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ashna\\eclipse-workspace\\assignment3\\src\\test\\re
        driver = new FirefoxDriver();
        driver.manage().window().maximize();

    }

    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {
        driver.navigate().to("http://127.0.0.1:8000/loginn");
           Thread.sleep(3000);

    }

    @When("user enters username and password")
    public void user_enters_username_and_password() throws Throwable
    {
        driver.findElement(By.name("email")).sendKeys("cisco@gmail.com");
        driver.findElement(By.name("password")).sendKeys("Ashna@123");
        Thread.sleep(3000);

    }

    @And("User clicks on login")
    public void user_clicks_on_login() {

        driver.findElement(By.id("login")).click();


    }

    @Then("user is navigated to the home page")
    public void user_is_navigated_to_the_home_page() throws Exception {

        driver.findElement(By.id("cindex")).isDisplayed();
        Thread.sleep(4000);
        driver.close();
        driver.quit();

    }


}
```

Screenshot

```
Scenario: Check login is successful with valid credentials # src/test/resources/assignment/loginn.feature:3
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698205356504   geckodriver     INFO    Listening on 127.0.0.1:12844
1698205356891   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-d
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698205357375   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698205357528   Marionette      INFO    Listening on port 59246
WebDriver BiDi listening on ws://127.0.0.1:2856
Read port: 59246
1698205357845   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:2856/devtools/browser/b74b016b-5105-4980-bf6c-f0d046fd243a
  Given browser is open                                 # stepdefa3.loginstepa3.browser_is_open()
  And user is on login page                             # stepdefa3.loginstepa3.user_is_on_login_page()
  When user enters username and password                # stepdefa3.loginstepa3.user_enters_username_and_password()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/static/assets/js/custom.js, line 59: TypeError: $(...).slick is not a function
  And User clicks on login                              # stepdefa3.loginstepa3.user_clicks_on_login()
1698205372897   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:59301
1698205372904   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInPare
1698205372977   Marionette      INFO    Stopped listening on port 59246
Dynamically enable window occlusion 1
  Then user is navigated to the home page               # stepdefa3.loginstepa3.user_is_navigated_to_the_home_page()
```

Test Report

Test Case 1

Project Name: ZillionHire					
Login Test Case					
Test Case ID: Test_1			**Test Designed By:** Ashna Karim		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 25/09/2023		
Module Name: Login Screen			**Test Executed By :** Ms. Jetty Benjamin		
Test Title : Company Login			**Test Execution Date:** 25/09/2023		
Description: Verify login with email and password					
Pre-Condition : Company has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Login page displayed	Pass
2	Provide email and password	Email: cisco@gmail.com	Company should be able to login	Company logged in and navigated to Company dashboard	Pass
3	Provide password	Ashna@123			
4	Click on Login button				
Post-Condition: Company is validated with database and successfully login into account. The account session details are logged in database					

Test Case 2: Admin – Add student

Code

```
package stepdeff;
import org.openqa.selenium.By;

public class stuadd {
    WebDriver driver = null;
    @Given("browser is open")
    public void browser_is_open() {
    System.out.println("Inside Step - Browser Is Open");
    System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ashna\\eclipse-workspace\\add\\src\\test\\resources\\Dri
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
        }

        @And("admin is on the login page")
        public void admin_is_on_the_login_page() throws Exception{
        driver.navigate().to("http://127.0.0.1:8000/loginn");
        Thread.sleep(3000);
       }

        @When("admin enters admin credentials and logs in")
        public void admin_enters_admin_credentials_and_logs_in()throws Throwable {
            driver.findElement(By.name("email")).sendKeys("adminzh@gmail.com");
        driver.findElement(By.name("password")).sendKeys("Admin@123");
        driver.findElement(By.id("login")).click();
        Thread.sleep(3000);
       }

        @And("admin navigates to the admin page")
        public void admin_navigates_to_the_admin_page() throws Exception{

        Thread.sleep(3000);
        driver.findElement(By.id("studentindex")).click();

        }
```

```java
        @And("user enters student details")
        public void user_enters_student_details() throws InterruptedException{
        // Enter category details
        driver.findElement(By.id("first_name")).sendKeys("Aria");
        driver.findElement(By.id("last_name")).sendKeys("Hall");
        driver.findElement(By.id("email")).sendKeys("ariahall3@gmail.com");
        driver.findElement(By.id("admission_no")).sendKeys("2021045");
        driver.findElement(By.id("phone")).sendKeys("9188654964");


        }

        @And("user clicks on the \"Submit\" button")
        public void user_clicks_on_the_add_student_button() {
        driver.findElement(By.id("submitstu")).click();
        }

        @Then("student should be added and displayed on the students page")
        public void student_should_be_added_and_displayed_on_the_students_page()throws InterruptedException {
        driver.findElement(By.id("as")).isDisplayed();

        Thread.sleep(4000);

        driver.close();

            driver.quit();
        }
}
```

Screenshot

```
Scenario: Adding a student as an admin                          # src/test/resources/features/sadd.feature:5
Inside Step - Browser Is Open
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698210056924   geckodriver     INFO    Listening on 127.0.0.1:8618
1698210057280   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionett
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698210057673   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698210057789   Marionette      INFO    Listening on port 61154
WebDriver BiDi listening on ws://127.0.0.1:44298
Read port: 61154
1698210058051   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:44298/devtools/browser/0012db8c-5bd4-483d-9083-3b0831ce6e31
  Given browser is open                                         # stepdeff.stuadd.browser_is_open()
  And admin is on the login page                                # stepdeff.stuadd.admin_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
  When admin enters admin credentials and logs in               # stepdeff.stuadd.admin_enters_admin_credentials_and_logs_in
  And admin navigates to the admin page                         # stepdeff.stuadd.admin_navigates_to_the_admin_page()
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestFirstName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestFirstName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestFirstName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined

JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
  And user enters student details                               # stepdeff.stuadd.user_enters_student_details()
  And user clicks on the "Submit" button                        # stepdeff.stuadd.user_clicks_on_the_add_student_button()
  Then student should be added and displayed on the students page # stepdeff.stuadd.student_should_be_added_and_displayed_on_th

1 Scenarios (1 failed)
7 Steps (1 failed, 6 passed)
0m20.435s
```

Test report

Test Case 2

Project Name: ZillionHire					
Add Student					
Test Case ID: Test_2			**Test Designed By:** Ashna Karim		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 3/10/2023		
Module Name: Add Student			**Test Executed By :** Ms. Jetty Benjamin		
Test Title : Admin Add student			**Test Execution Date:** 25/10/2023		
Description: Create new student and other details by Admin					
Pre-Condition : Admin has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Login page displayed	Pass
2	Provide email and password	Email: adminzh@gmail.com	Admin should be able to login	Admin logged in and navigated to Company dashboard	Pass
3	Provide password	Admin@123			
4	Click on Login button				
5	Click on Add Student		Add Student page should be displayed	Admin navigated to Add student page	Pass
6	Provide new student first name	Student First name : Mia			
7	Provide new student last name	Student last name: Russell			
8	Provide new student Admission number	Student Admission number: 2021025			
9	Provide new student	Student email:miarussell@gmail.co			

	email	m			
10	Provide new student phone	Student phone: 9188654964			
11	Click on Submit button		Admin should be able to add new student	Admin created new student	Pass
Post-Condition: New student is successfully created by Admin					

Test Case 3: Company – Add Job

Code

```
package test3java;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class test3class {
    WebDriver driver = null;
    @Given("browser is open")
    public void browser_is_open() {
    System.out.println("Inside Step - Browser Is Open");
    System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ashna\\eclipse-workspace\\test3\\src\\test\\resources\\
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
        }

        @And("company is on the login page")
        public void company_is_on_the_login_page() throws Exception{
        driver.navigate().to("http://127.0.0.1:8000/loginn");
        Thread.sleep(3000);
       }

        @When("company enters company credentials and logs in")
        public void company_enters_company_credentials_and_logs_in()throws Throwable {
            driver.findElement(By.name("email")).sendKeys("cisco@gmail.com");
        driver.findElement(By.name("password")).sendKeys("Ashna@123");
        driver.findElement(By.id("login")).click();
        Thread.sleep(3000);
       }

        @And("company navigates to the company index")
        public void company_navigates_to_the_company_index() throws Exception{
```

```
        @And("company navigates to the company index")
        public void company_navigates_to_the_company_index() throws Exception{

        Thread.sleep(3000);
        driver.findElement(By.id("postjob")).click();

        }
        @And("company navigates to the addjob page")
        public void company_navigates_to_the_addjob_page() throws Exception{

        Thread.sleep(3000);
        driver.findElement(By.id("jobadd")).click();

        }

        @And("company enters job details")
        public void company_enters_job_details() throws InterruptedException{
        // Enter category details
        driver.findElement(By.id("cname")).sendKeys("CISCO");
        driver.findElement(By.id("jname")).sendKeys("DATA ANALYST");
        driver.findElement(By.id("salary")).sendKeys("20000");
        driver.findElement(By.id("email")).sendKeys("cisco@gmail.com");
        driver.findElement(By.id("sdate")).sendKeys("10/11/23");
        driver.findElement(By.id("edate")).sendKeys("20/11/23");
        driver.findElement(By.id("link")).sendKeys("www.cisco.com");
        driver.findElement(By.id("job_description")).sendKeys("Data analysts gather and scrutinise data using specialist
        driver.findElement(By.id("responsibilities")).sendKeys("Data cleaning and preparation, Data analysis and explorat
        driver.findElement(By.id("preferred_skills")).sendKeys("Python, R");
        driver.findElement(By.id("qualifications")).sendKeys("B.Tech ECE/CSE/IT, MCA");
        driver.findElement(By.name("required_tenth_cgpa")).sendKeys("7");
        driver.findElement(By.name("required_twelfth_cgpa")).sendKeys("7");
        driver.findElement(By.name("required_current_cgpa")).sendKeys("7");
        driver.findElement(By.name("required_backlog")).sendKeys("0");


        }
```

```
@And("user clicks on the \"Post Job\" button")
public void user_clicks_on_the_post_job_button() {
driver.findElement(By.id("postjob")).click();
}

@Then("company should be added and displayed on the company page")
public void company_should_be_added_and_displayed_on_the_company_page()throws InterruptedException {
driver.findElement(By.id("jobtest")).isDisplayed();

Thread.sleep(4000);

driver.close();

    driver.quit();
}
```

Screenshot

```
Scenario: Adding a job as a company                             # src/test/resources/Features/test3feature.feature:5
Inside Step - Browser Is Open
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698216151221   geckodriver     INFO    Listening on 127.0.0.1:14353
1698216151594   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionett
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698216152031   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698216152154   Marionette      INFO    Listening on port 63558
WebDriver BiDi listening on ws://127.0.0.1:40213
Read port: 63558
1698216152440   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:40213/devtools/browser/bf73df53-e630-4819-9ef1-346b9556da5b
  Given browser is open                                         # test3java.test3class.browser_is_open()
  And company is on the login page                              # test3java.test3class.company_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/static/assets/js/custom.js, line 59: TypeError: $(...).slick is not a function
  When company enters company credentials and logs in           # test3java.test3class.company_enters_company_credentials_and_
  And company navigates to the company index                    # test3java.test3class.company_navigates_to_the_company_index
  And company navigates to the addjob page                      # test3java.test3class.company_navigates_to_the_addjob_page()
  And company enters job details                                # test3java.test3class.company_enters_job_details()
  And user clicks on the "Post Job" button                      # test3java.test3class.user_clicks_on_the_post_job_button()
1698216175930   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:63647
1698216175934   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fissior
1698216176079   Marionette      INFO    Stopped listening on port 63558
Dynamically enable window occlusion 1
  Then company should be added and displayed on the company page # test3java.test3class.company_should_be_added_and_displayed_
      org.openqa.selenium.NoSuchSessionException: Tried to run command without establishing a connection
Build info: version: '4.8.3', revision: 'e5e76298c3'
System info: os.name: 'Windows 11', os.arch: 'amd64', os.version: '10.0', java.version: '17.0.6'
Driver info: org.openqa.selenium.firefox.FirefoxDriver
Command: [de48e12e-92c1-471a-9dac-0c042dae6b00, quit {}]
```

```
1 Scenarios (1 failed)
8 Steps (1 failed, 7 passed)
0m28.034s
```

Test report

Test Case 3

Project Name: ZillionHire					
Add Job					
Test Case ID: Test_3			**Test Designed By:** Ashna Karim		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 9/10/2023		
Module Name: Add Job			**Test Executed By :** Ms. Jetty Benjamin		
Test Title : Company Add job			**Test Execution Date:** 25/10/2023		
Description: Create new student and other details by Admin					
Pre-Condition : Admin has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Login page displayed	Pass
2	Provide email and password	Email: cisco@gmail.com	Company should be able to login	Company logged in and navigated to Company dashboard	Pass
3	Provide password	Ashna@123			
4	Click on Login button				
5	Click on Post Job		Post Job page should be displayed	Company navigated to Post job page	Pass
6	Click on Add Job		Add Job page should be displayed	Company navigated to Add job page	Pass
7	Provide company name	Company name: CISCO			
8	Provide new job name	Job name: DATA ANALYST			
9	Provide new job salary	Salary: 20000			
10	Provide email	Email: cisco@gmail.com			

11	Provide new job application start date	Application start date: 10/11/23			
12	Provide new job application end date	Application end date: 20/11/23			
13	Provide company link	Web: www.cisco.com			
14	Provide new job description	Job description: Data analysts gather..			
15	Provide new job responsibilities	Job responsibilities: Data cleaning and preparation..			
16	Provide new job preferred skills	Preferred skills: Python, R			
17	Provide new job qualification	Qualification: B.Tech ECE/CSE/IT, MCA			
18	Provide new job required tenth cgpa	required tenth cgpa:7			
19	Provide new job required twelfth cgpa	required twelfth cgpa: 7			
20	Provide new job required current cgpa	required current cgpa: 7			
21	Provide new job required backlog	Backlog: 0			
22	Click on Submit button		Company should be able to add new job	Company added new job	Pass
Post-Condition: New Job is successfully added by Company					

Test Case 4: Admin – Search company

Code

```
package stepdeff;
import org.openqa.selenium.By;

public class stuadd {
    WebDriver driver = null;
    @Given("browser is open")
    public void browser_is_open() {
    System.out.println("Inside Step - Browser Is Open");
    System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ashna\\eclipse-workspace\\add\\src\\test\\resources\\Dri
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
        }

        @And("admin is on the login page")
        public void admin_is_on_the_login_page() throws Exception{
        driver.navigate().to("http://127.0.0.1:8000/loginn");
        Thread.sleep(3000);
       }

        @When("admin enters admin credentials and logs in")
        public void admin_enters_admin_credentials_and_logs_in()throws Throwable {
            driver.findElement(By.name("email")).sendKeys("adminzh@gmail.com");
        driver.findElement(By.name("password")).sendKeys("Admin@123");
        driver.findElement(By.id("login")).click();
        Thread.sleep(3000);
       }

        @And("admin navigates to the admin page")
        public void admin_navigates_to_the_admin_page() throws Exception{

        Thread.sleep(3000);
        driver.findElement(By.id("studentindex")).click();

        }

```

```
        @And("user enters student details")
        public void user_enters_student_details() throws InterruptedException{
        // Enter category details
        driver.findElement(By.id("first_name")).sendKeys("Aria");
        driver.findElement(By.id("last_name")).sendKeys("Hall");
        driver.findElement(By.id("email")).sendKeys("ariahall3@gmail.com");
        driver.findElement(By.id("admission_no")).sendKeys("2021045");
        driver.findElement(By.id("phone")).sendKeys("9188654964");


        }

        @And("user clicks on the \"Submit\" button")
        public void user_clicks_on_the_add_student_button() {
        driver.findElement(By.id("submitstu")).click();
        }

        @Then("student should be added and displayed on the students page")
        public void student_should_be_added_and_displayed_on_the_students_page()throws InterruptedException {
        driver.findElement(By.id("as")).isDisplayed();

        Thread.sleep(4000);

        driver.close();

            driver.quit();
        }
}
```

Screenshot

```
Scenario: Adding a student as an admin                                # src/test/resources/features/sadd.feature:5
Inside Step - Browser Is Open
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
1698210056924   geckodriver     INFO    Listening on 127.0.0.1:8618
1698210057280   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionett
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698210057673   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698210057789   Marionette      INFO    Listening on port 61154
WebDriver BiDi listening on ws://127.0.0.1:44298
Read port: 61154
1698210058051   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:44298/devtools/browser/0012db8c-5bd4-483d-9083-3b0831ce6e31
  Given browser is open                                               # stepdeff.stuadd.browser_is_open()
  And admin is on the login page                                      # stepdeff.stuadd.admin_is_on_the_login_page()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
  When admin enters admin credentials and logs in                     # stepdeff.stuadd.admin_enters_admin_credentials_and_logs_in
  And admin navigates to the admin page                               # stepdeff.stuadd.admin_navigates_to_the_admin_page()
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestFirstName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestFirstName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestFirstName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestLastName is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
```

```
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
JavaScript error: http://127.0.0.1:8000/admin_addstudents/, line 1: ReferenceError: suggestAdmissionNumber is not defined
  And user enters student details                                     # stepdeff.stuadd.user_enters_student_details()
  And user clicks on the "Submit" button                              # stepdeff.stuadd.user_clicks_on_the_add_student_button()
  Then student should be added and displayed on the students page     # stepdeff.stuadd.student_should_be_added_and_displayed_on_th

1 Scenarios (1 failed)
7 Steps (1 failed, 6 passed)
0m20.435s
```

Test report

Test Case 4

Project Name: ZillionHire					
Search Company					
Test Case ID: Test_4			**Test Designed By:** Ashna Karim		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 24/10/2023		
Module Name: Add Student			**Test Executed By :** Ms. Jetty Benjamin		
Test Title : Admin Seach company			**Test Execution Date:** 25/10/2023		
Description: Search approved company by Admin					
Pre-Condition : Admin has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to login page		Login page should be displayed	Login page displayed	Pass
2	Provide email and password	Email: adminzh@gmail.com	Admin should be able to login	Admin logged in and navigated to Company dashboard	Pass
3	Provide password	Admin@123			
4	Click on Login button				
5	Click on Company List		Company list page should be displayed	Admin navigated to company list page	Pass
6	Provide Company name	Company name: Cisco			
7	Click on Submit button		Admin should be able to search company by name	Admin searched company by name	Pass
Post-Condition: Company searched by Admin					

CHAPTER 6
IMPLEMENTATION

6.1INTRODUCTION

A project's implementation phase takes place after the design is turned into a workable system. Gaining user trust that the system will function accurately and effectively is essential to the success of the new system at this stage. During this phase, user education and documentation are of utmost importance. Conversion could take place simultaneously with user training or at a later time. The process of implementation entails transforming a recently revised design for the system into an operating system.

The user department carries the most of the workload, goes through the most turmoil, and has the biggest impact on the current system during this phase. Poorly thought out or managed implementation might lead to confusion and anarchy. It doesn't matter if the new system is fully brand-new, replaces an old manual or automated system, or alters an existing system—the needs of the business must be met through proper implementation. The conversion from the old to the new system requires a number of steps, which are all included in system implementation. Only when extensive testing has been completed and it has been determined that the system is operating as intended can it be put into use. System personnel assess the system's viability. Extensive work is needed for implementation in three key areas: system testing, changeover, and teaching & training. In the implementation phase, meticulous planning, system and constraint research, and method development for changeover are all required.

6.2 IMPLEMENTATION PROCEDURES

The process of installing the program in its actual context and confirming that it satisfies the intended usage and performs as expected is known as software implementation. In some companies, the project to develop the software could be ordered by someone who won't be utilizing it themself. Although there might be some early skepticism against the program, it's crucial to prevent resistance from mounting. This can be done by:

- Ensuring that active users are aware of the new system's advantages and fostering their trust in the program.
- Giving consumers the right direction so they feel at ease using the program.
- Users should be aware that the server program needs to be running on the server in order to view the system. The intended process won't happen until the server object is active.

6.2.1 User Training

The purpose of user training is to get the user ready to test and modify the system. The people who will be involved must have faith in their ability to contribute to the goal and benefits anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to enter data, handle error messages, query the database, call up routines to generate reports, and execute other important tasks through user training.

6.2.2 Training on the Application Software

The user must receive training on the recently released application software after receiving the requisite foundational instruction on computer awareness. This instruction should cover the basic principles of using the latest system, as well as the way the screens work together, how they are designed, what kind of help is available, the kinds of errors that might occur when entering data, the checks that should be made to validate each entry, and how to modify the data after it has been entered. Additionally, to efficiently use the system or a component of the system, the training ought to cover knowledge that is particular to the user or group. It is crucial to remember that such instruction may vary depending on the user groups and hierarchical levels.

6.2.3 System Maintenance

As the time when the program is really used and carries out its intended functions, maintenance is an important part of the process of creating software. To maintain the system's functionality, dependability, and adaptability to changes in the system environment, proper maintenance is crucial. Maintenance tasks involve more than just finding and repairing flaws or errors in the system. Updates to the program, adjustments to its features, and speed improvements are just a few examples of what it may entail. Software maintenance is essentially a continuous process that calls for constant monitoring, assessment, and modification of the system to satisfy altering user needs and requirements.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

For educational institutions to efficiently handle the placement process and connect students and potential recruiters, ZillionHire is an essential tool. The existing system has many drawbacks such as lack of contact between students and the placement officer, making it difficult to recommend new placements, limited accessibility, time-consuming. As a solution ZillionHire provides management of numerous placement operations, such as job postings, student registrations. Here student can keep a better profile, browse jobs and apply job which matches with their cgpa and have a record of applied jobs and shortlisted jobs. Company/ recruiters can add many job posts and shortlist students by criteria. Admin have record of student& company also approves company and jobs and can add students. The proposed system provides students more influence by making placement-related information easily accessible. Students who take an active role in their career development can post their resumes, seek for jobs, highlight their profile and monitor their progress. Students, placement coordinators, and recruiters can communicate effectively. This platform makes it easier to collect and manage data in a systematic way. In summary, this placement cell management system streamlines the hiring process by making it more effective, open, and simple to use. It is crucial in bridging the employment gap between students and companies, thereby improving career options for students and achieving educational institutions' placement goals.

7.2 FUTURE SCOPE

The Placement Cell Management System there is scope for improvement of the system.

Alumni Engagement: Including an alumni network might provide opportunities for current students to get mentoring. Alumni can aid in the placement process by providing advice, knowledge of the business, and even job leads also they can browse jobs and apply jobs. This helps pass out students also get employed.

Students can improve their knowledge by online tests and study materials. This can contain modules for developing soft skills, technical training, and interviewing techniques. Push notifications for job postings and interview updates. Interview process also can be done by this placement cell management system. Individualized job recommendations: The website may offer individualized job recommendations based on a student's skills using machine learning algorithms

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

• IEEE Std 1016 Recommended Practice for Software Design Description

• Ken Schwaber, Mike Beedle, Agile Software Development with Scrum, Pearson (2008)

• Roger S Pressman, “Software Engineering”

• PankajJalote, “Software engineering: a precise approach”

WEBSITES:

- https://www.ajce.in/home/placementcell.html
- www.djangoprojects.com
- https://bootstrapmade.com/
- https://www.tutorialspoint.com/

CHAPTER 9

APPENDIX

9.1 Sample Code

1.Login

```
{% load static %}
{% comment %} {% load crispy_forms_tags %}  {% endcomment %}
{% load socialaccount %}
{% block content %}
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
  <!-- Add Bootstrap CSS link -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <style>
    /* Your existing styles */
    .image-container {
      background-size: cover;
      min-height: 70vh;
      display: flex;
      align-items: center;
      justify-content: center;
    }
    .fp {
      text-align: center;
    }
    .form-container {
      background-color: rgba(225, 241, 245, 0.557);
      padding: 70px;
      border-radius: 5px;
      box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
    }
    .image-container img {
      max-width: 100%;
      height: auto;
```

```
    display: block;
   }
   .h3 {
    text-align: center;
   }
   .row{
      margin-left: 200px;
      margin-top: 65px;
   }
  </style>
</head>

<body>
  <div class="container-fluid">
   <div class="row">
    <div class="col-md-6">
     <div class="container form-container mt-5">
      <h3 class="h3"> LOGIN</h3>
      {% for messages in messages %}
      <h3 style="color:blue">{{ messages }}</h3>
      {% endfor %}
      <form id="registrationForm" method="POST">
       {% csrf_token %}
       <div class="form-group">
          <label for="username">email</label>
          <input type="username" class="form-control" id="username" name="email" required>
          <small id="usernameError" class="form-text text-danger"></small>
          {% comment %} {{ form|crispy }} {% endcomment %}
        </div>

        <div class="form-group">
          <label for="password">password</label>
          <input type="password" class="form-control" id="password" name="password"
required>
```

```
            <small id="passwordError" class="form-text text-danger"></small>
          </div>

          <button type="submit" id="login" class="btn btn-primary btn-block">Login</button><br>

          <a href="{% provider_login_url 'google'%}?next=/"><br>
            {% comment %} <a href="#" class="btn btn-danger btn-block mt-3"
onclick="signInWithGoogle()"> {% endcomment %}
            <span class="align-middle">
              <img
src="https://static.vecteezy.com/system/resources/previews/022/484/503/original/google-lens-icon-
logo-symbol-free-png.png" alt="Google Icon" width="24" height="24">
            </span>
            <span class="align-middle ml-2">Sign in with Google</span><br>
            </a><br>
            <pre>New here? <a href="{% url 'reg' %}" style="color: rgb(201, 7, 7); font-size:
14px;">Register</a></pre>

            {% comment %} <h1>Google Login</h1>
            <a href="{% provider_login_url 'google'%}?next=/">Login with Google</a> {%
endcomment %}

        </form>
        <form action="{% url 'password_reset' %}" method="post">
          {% csrf_token %}
          <div class="form-group">
          <a href="{% url 'password_reset' %}" class="forgot-password">Forgot Password?</a>
          </div>
          </form>
      </div>
    </div>
    <div class="col-md-6">
      <div class="row justify-content-center image-container">
        <img src="https://img.freepik.com/free-vector/devops-team-abstract-concept-vector-
```

```
illustration-software-development-team-member-agile-workflow-devops-team-model-it-teamwork-project-management-integrated-practice-abstract-metaphor_335657-2299.jpg?w=740&t=st=1694952991~exp=1694953591~hmac=c103a241c0189fbb2c3cef6541fe8cbf237a25e1fd299ae4e9d5d0f5534731cc" alt="Image">
        </div>
      </div>
    </div>
  </div>
{% endblock %}
  <!--  Bootstrap JS and jQuery scripts -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

  <script>
    const registrationForm = document.getElementById('registrationForm');
    const emailInput = document.getElementById('username');
    const passwordInput = document.getElementById('password');

    function validateEmail() {
      const emailValue = emailInput.value;
      const emailError = document.getElementById('usernameError');

      if (!/^\S+@\S+\.\S+$/.test(emailValue)) {
        emailError.textContent = 'Enter a valid email address.';
        return false;
      } else {
        emailError.textContent = '';
        return true;
      }
    }

    function validatePassword() {
```

```
    const passwordValue = passwordInput.value;
    const passwordError = document.getElementById('passwordError');

    if (!/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])[a-zA-Z0-9!@#$%^&*()_+,.?]{8,}$/.test(passwordValue))
{
      passwordError.textContent = 'Password should contain at least 8 characters, one uppercase
letter, one lowercase letter, one number, and one special character.';
      return false;
    } else {
      passwordError.textContent = '';
      return true;
    }
  }
  emailInput.addEventListener('keyup', validateEmail);
  passwordInput.addEventListener('keyup', validatePassword);
  registrationForm.addEventListener('submit', function(event) {
    const isEmailValid = validateEmail();
    const isPasswordValid = validatePassword();

    if (!isEmailValid || !isPasswordValid) {
      event.preventDefault();
    }
  });
  function signInWithGoogle(){

  }
 </script>
</body>
</html>
def loginn(request):
  if request.method == "POST":
    username=request.POST['email']
    # email = request.POST['email']
    password=request.POST['password']
```

```
        user = authenticate(username=username, password=password)
        if user is not None:
            login(request, user)
            if user.is_superuser:
                return redirect('admin_index2')
            elif user.is_staff:
                return redirect('sindex')
            else:
                return redirect('cindex')
        else:
            messages.info(request, "Invalid Login")
            return redirect('loginn')
    else:
        return render(request, 'login.html')
```

2. Add Student

```
def addstudents(request):
    if request.method == 'POST':
        sname = request.POST.get('sname')
        email = request.POST.get('email')
        # passw = request.POST.get('passw')
        # password=request.POST.get('password')
        course = request.POST.get('course')
        department = request.POST.get('department')
        semester = request.POST.get('semester')
        obj = Students()
        obj.sname = sname
        obj.email = email
        # obj.passw = passw
        # obj.password = password
        obj.course = course
        obj.department = department
        obj.semester = semester
        obj.save()
```

```
        messages.success(request, 'Student added successfully!')
        return redirect('admin_poststudent')
    return render(request, 'admin/student_add.html')

{% load static %}
<script>
    {% comment %} // Function to validate first and last name
    function validateName(inputField) {
        var nameRegex = /^[a-zA-Z]+$/;
        var fieldValue = inputField.value;
        var feedback = inputField.nextElementSibling; // Get the sibling for feedback

        if (!fieldValue.trim()) {
            feedback.innerHTML = "Please fill out this field.";
            feedback.style.color = "red";
        } else if (!nameRegex.test(fieldValue)) {
            feedback.innerHTML = "Only alphabets are allowed.";
            feedback.style.color = "red";
        } else {
            feedback.innerHTML = "";
        }
    } {% endcomment %}
    // ... Other functions ...

// Function to validate first and last name
function validateName(inputField) {
    var nameRegex = /^[a-zA-Z]+$/;
    var firstNameValue = document.getElementById('first_name').value.toLowerCase();
    var lastNameValue = document.getElementById('last_name').value.toLowerCase();
    var fieldValue = inputField.value.toLowerCase();
    var feedback = inputField.nextElementSibling; // Get the sibling for feedback
    var admissionNumberField = document.getElementById('admission_no');

    if (!fieldValue.trim()) {
```

```
        feedback.innerHTML = "Please fill out this field.";
        feedback.style.color = "red";
    } else if (!nameRegex.test(fieldValue)) {
        feedback.innerHTML = "Only alphabets are allowed.";
        feedback.style.color = "red";
    } else {
        feedback.innerHTML = "";
        // Check if both first name and last name match
        if (fieldValue === firstNameValue && fieldValue === lastNameValue) {
            admissionNumberField.value = getAdmissionNumber(inputField.value);
        }
    }
}

// Function to get admission number based on first name
function getAdmissionNumber(firstName) {
    var matchingEntry = admissionNumberSuggestions.find(function(entry) {
        return entry.toLowerCase().includes(firstName.toLowerCase());
    });

    if (matchingEntry) {
        return matchingEntry;
    } else {
        return "";
    }
}
// ... Other functions ...
    // Function to validate email
    function validateEmail(inputField) {
        var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        var fieldValue = inputField.value;
        var feedback = inputField.nextElementSibling; // Get the sibling for feedback

        if (!fieldValue.trim()) {
```

```
        feedback.innerHTML = "Please fill out this field.";
        feedback.style.color = "red";
    } else if (!emailRegex.test(fieldValue)) {
        feedback.innerHTML = "Invalid email address.";
        feedback.style.color = "red";
    } else {
        feedback.innerHTML = "";
    }
}
// Function to validate license number
{% comment %} function validateLicenseNumber(inputField) {
    var licenseRegex = /^MDIN\d+$/;
    var fieldValue = inputField.value;
    var feedback = inputField.nextElementSibling; // Get the sibling for feedback

    if (!fieldValue.trim()) {
        feedback.innerHTML = "Please fill out this field.";
        feedback.style.color = "red";
    } else if (!licenseRegex.test(fieldValue)) {
        feedback.innerHTML = "License number should start with 'MDIN' followed by digits.";
        feedback.style.color = "red";
    } else {
        feedback.innerHTML = "";
    }
} {% endcomment %}

// Function to validate phone number
function validatePhoneNumber(inputField) {
    var phoneRegex = /^\d{10}$/;
    var fieldValue = inputField.value;
    var feedback = inputField.nextElementSibling; // Get the sibling for feedback

    if (!fieldValue.trim()) {
        feedback.innerHTML = "Please fill out this field.";
```

```
            feedback.style.color = "red";
        } else if (!phoneRegex.test(fieldValue)) {
            feedback.innerHTML = "Invalid Phone number";
            feedback.style.color = "red";
        } else {
            feedback.innerHTML = "";
        }
    }
</script>
<div id="content-wrapper">
    <div class="container-fluid">
        <!-- Icon Cards-->
        <div class="row">
            <div class="container-fluid">
                <h2 class="mt-3 text-center">Add Student</h2>
                <div class="row">
                    <div class="col-md-6 offset-md-3">
                        <form method="POST" action="{% url 'add_student' %}" enctype="multipart/form-
data">
                            {% csrf_token %}

                            {% comment %} <div class="form-group">
                                <label for="first_name">First Name</label>
                                <input type="text" class="form-control" id="first_name" name="first_name"
placeholder="Your first name" oninput="validateName(this)">
                                <div class="feedback" style="color: red;"></div>
                            </div> {% endcomment %}
                            <div class="form-group">
                                <label for="first_name">First Name</label>
                                <input type="text" class="form-control" id="first_name" name="first_name"
placeholder="Your first name" oninput="suggestFirstName(this)">
                                <ul id="first_name_suggestions" class="suggestions"></ul>
                                <div class="feedback" style="color: red;"></div>
                            </div>
```

```
                {% comment %} <div class="form-group">
                    <label for="last_name">Last Name</label>
                    <input type="text" class="form-control" id="last_name" name="last_name"
placeholder="Your last name" oninput="validateName(this)">
                    <div class="feedback" style="color: red;"></div>
                </div> {% endcomment %}
                <div class="form-group">
                    <label for="last_name">Last Name</label>
                    <input type="text" class="form-control" id="last_name" name="last_name"
placeholder="Your last name" oninput="suggestLastName(this)">
                    <ul id="last_name_suggestions" class="suggestions"></ul>
                    <div class="feedback" style="color: red;"></div>
                </div>
                <div class="form-group">
                    <label for="email">Email Address</label>
                    <input type="email" class="form-control" id="email" name="email"
placeholder="Your email address" oninput="validateEmail(this)">
                    <div class="feedback" style="color: red;"></div>
                </div>

                {% comment %} <div class="form-group">
                  <label for="license_no">Admission Number</label>
                    <input type="text" class="form-control" id="license_no"
name="Admission_no" placeholder="Your License Number"
oninput="validateLicenseNumber(this)">
                    <div class="feedback" style="color: red;"></div>
                </div> {% endcomment %}
                <div class="form-group">
                    <label for="admission_no">Admission Number</label>
                    <input type="text" class="form-control" id="admission_no"
name="admission_no" placeholder="Your Admission Number"
oninput="suggestAdmissionNumber(this)">
                    <ul id="admission_no_suggestions" class="suggestions"></ul>
```

```
                    <div class="feedback" style="color: red;"></div>
                </div>
                <div class="form-group">
                    <label for="phone">Phone</label>
                    <input type="text" class="form-control" id="phone" name="phone"
placeholder="Your Phone Number" oninput="validatePhoneNumber(this)">
                    <div class="feedback" style="color: red;"></div>
                </div>

                {% if messages %}
                    <ul class="messages">
                        {% for message in messages %}
                            {% if message.tags == 'error' %}
                                <div class="alert alert-danger">
                                    {{ message }}
                                </div>
                            {% elif message.tags == 'success' %}
                                <div class="alert alert-success">
                                    {{ message }}
                                </div>
                            {% endif %}
                        {% endfor %}
                    </ul>
                {% endif %}

                <div class="text-center mt-4">
                    <button id="submitstu" type="submit" class="btn btn-warning" style="min-
width: 15em;">
                        Submit
                    </button>
                </div>
            </form>
        </div>
    </div>
```

```
            </div>
        </div>
    </div>
</div>
<style>
    .feedback {
        font-size: 14px;
        margin-top: 5px;
    }

    body {
        font-family: Arial, sans-serif;
        background-color: #f0f4f7;
    }

    #content-wrapper {
        padding: 20px;
    }
    .container-fluid {
        padding: 20px;
        background-color: #fff;
        border: 1px solid #d1d8e0;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        border-radius: 10px;
        max-width: 500px;
        margin: 0 auto;
    }

    h2 {
        color: #3498db;
        text-align: center;
    }

    .form-group label {
```

```
    color: #3498db;
    font-weight: bold;
}
.form-group input.form-control {
    border: 1px solid #3498db;
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
}
.form-group input.form-control:focus {
    border-color: #2980b9;
}
.btn.btn-warning {
    background-color: #3498db;
    border: none;
    border-radius: 5px;
    color: #fff;
    font-weight: bold;
    transition: background-color 0.3s ease-in-out;
    width: 100%;
    padding: 10px;
    cursor: pointer;
}
.btn.btn-warning:hover {
    background-color: #2980b9;
}

.alerts {
    margin-top: 10px;
}

.alerts .alert {
    border-radius: 5px;
    margin-bottom: 10px;
```

```
        padding: 10px;
        font-weight: bold;
    }
    .alerts .alert-danger {
        background-color: #e74c3c;
        color: #fff;
        border: 1px solid #c0392b;
    }
    .alerts .alert-success {
        background-color: #2ecc71;
        color: #fff;
        border: 1px solid #27ae60;
    }
</style>
```

3.Add Job

```
def addjob(request):


        # obj=CompanyProfile.objects.get(id=obj_id)
        # cmp = get_object_or_404(CompanyProfile, user=request.user)
        # print(cmp)
    # except CompanyProfile.DoesNotExist
            user = request.use

            if request.method == 'POST':
                cname = request.POST.get('cname')
                jname = request.POST.get('jname')
                salary = request.POST.get('salary')
                email = request.POST.get('email')
                sdate = request.POST.get('sdate')
                edate = request.POST.get('edate')
                link = request.POST.get('link')
                job_descriptions = request.POST.get('job_descriptions')
                qualifications = request.POST.get('qualifications')
                preferred_skills = request.POST.get('preferred_skills')
```

```
        responsibilities = request.POST.get('responsibilities')
        required_current_cgpa = request.POST.get('required_current_cgpa')
        required_tenth_cgpa = request.POST.get('required_tenth_cgpa')
        required_twelfth_cgpa = request.POST.get('required_twelfth_cgpa')
        required_backlog = request.POST.get('required_backlog')
        # criteria=request.FILES['criteria'] if 'criteria' in request.FILES else None

        # if criteria and not criteria.name.endswith('.pdf'):
        #     messages.error(request, 'Please upload a PDF file for the criteria.')
        #     return redirect('postjob')
        obj = Jobs()
        obj.user = request.user
        obj.cname = cname
        obj.jname = jname
        obj.salary = salary
        obj.email = email
        obj.sdate = sdate
        obj.edate = edate
        obj.link = link
        obj.job_descriptions = job_descriptions
        obj.qualifications = qualifications
        obj.preferred_skills = preferred_skills
        obj.responsibilities = responsibilities
        obj.required_current_cgpa = required_current_cgpa
        obj.required_tenth_cgpa = required_tenth_cgpa
        obj.required_twelfth_cgpa = required_twelfth_cgpa
        obj.required_backlog = required_backlog

        # obj.criteria=criteria

        obj.save()
        messages.success(request, 'Job added successfully!')

    # Redirect to the doctors page
```

```
            return redirect('postjob')  # Redirect to the doctors page URL name

        return render(request, 'addjob.html',{'user':user})

{% load static %}
<!DOCTYPE html>
<html lang="en">
<!-- add-doctor24:06-->
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=0">
   <link rel="shortcut icon" type="image/x-icon" href="{% static 'assets2/img/logo.png' %}">
   <title>ZillionHire - Company - Add Job</title>
   <link rel="stylesheet" type="text/css" href="{% static 'assets2/css/bootstrap.min.css'%}">
   <link rel="stylesheet" type="text/css" href="{% static 'assets2/css/font-awesome.min.css'%}">
   <link rel="stylesheet" type="text/css" href="{% static 'assets2/css/select2.min.css'%}">
   <link rel="stylesheet" type="text/css" href="{% static 'assets2/css/bootstrap-
datetimepicker.min.css'%}">
   <link rel="stylesheet" type="text/css" href="{% static 'assets2/css/style.css'%}">

      <script src="{% static 'assets2/js/html5shiv.min.js' %}"></script>
      <script src="{% static 'assets2/js/respond.min.js' %}"></script>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

</head>

<body>
   <header class="">
      <nav class="navbar navbar-expand-lg">
        <div class="container">
          {% comment %} <a class="navbar-brand" href="{% url 'index' %}"><h2>ZillionHire <em>
Website</em></h2></a> {% endcomment %}
          <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-
```

```
label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarResponsive">
          <ul class="navbar-nav ml-auto">
            <li class="nav-item ">
              <a class="nav-link" href="{% url 'cindex' %}">Home
                {% comment %} <span class="sr-only">(current)</span> {% endcomment %}
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{% url 'aboutuser' %}">About Us</a>
            </li>
            {% comment %} <li>
              <a class="dropdown-toggle nav-link" data-toggle="dropdown" href="#" role="button"
aria-haspopup="true" aria-expanded="false">Others</a>

              <div class="dropdown-menu">
                <a class="dropdown-item" href="{% url 'cprofile' %}">Profile</a>
                <a class="dropdown-item" href="{% url 'loginn' %}">Jobs</a>
                <a class="dropdown-item" href="{% url 'loginn' %}">Candidate List</a>

              </div>
            </li> {% endcomment %}
            <li class="nav-item dropdown">
              {% if user.is_authenticated %}
                <a class="nav-link" href="{% url 'loggout' %}">Log Out</a>
              {% else %}
                <a class="dropdown-toggle nav-link" data-toggle="dropdown" href="#" role="button"
aria-haspopup="true" aria-expanded="false">Login</a>

              <div class="dropdown-menu">
                <a class="dropdown-item" href="{% url 'loginn' %}">Student</a>
                <a class="dropdown-item" href="{% url 'loginn' %}">Company</a>
```

```
            <a class="dropdown-item" href="{% url 'loginn' %}">College Officer</a>
            {% comment %} <a class="dropdown-item" href="{% url 'blog' %}">Blog</a>
            <a class="dropdown-item" href="testimonials.html">Testimonials</a>
            <a class="dropdown-item" href="terms.html">Terms</a> {% endcomment %}
          </div>
          {% endif %}
        </li>

        <li class="nav-item">
          <a class="nav-link" href="{% url 'contactuser' %}">Contact Us</a>
        </li>
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'postjob' %}">Post Job</a>
          <span class="sr-only">(current)</span>
        </li>
      </ul>
    </div>
  </div>
 </nav>
</header>
<div class="main-wrapper">

 <div class="page-wrapper">
   <div class="content">
     <div class="row">
       <div class="col-lg-8 offset-lg-2">
         <h4 class="page-title">Add New Job</h4>
       </div>
     </div>
     <br><br>
     <div class="row">
       <div class="col-lg-8 offset-lg-2">
         <form method="POST" action="" enctype="multipart/form-data">
           {% csrf_token %}
```

```
                    <div class="row">
                        <div class="col-sm-6">
                            <div class="form-group">
                                <label>Company Name <span class="text-danger">*</span></label>
                                <input class="form-control" id="cname" name="cname" type="text"
value="{{ user.companyprofile.email }}" onkeyup="">
                            </div>
                        </div>
                        <div class="col-sm-6">
                            <div class="form-group">
                                <label>Job Name <span class="text-danger">*</span></label>
                                <input class="form-control" id="jname" name="jname" type="text"
onkeyup="validateJname()">
                                <span id="jnameError" class="text-danger"></span>
                            </div>
                        </div>
                        <div class="col-sm-6">
                            <div class="form-group">
                                <label>Salary <span class="text-danger">*</span></label>
                                <input class="form-control" id="salary" name="salary" type="number"
onkeyup="validateSalary()">
                                <span id="salaryError" class="text-danger"></span>
                            </div>
                        </div>
                        <div class="col-sm-6">
                            <div class="form-group">
                                <label>Email <span class="text-danger">*</span></label>
                                <input class="form-control" id="email" name="email" type="email">
                            </div>
                        </div>
                        <div class="col-sm-6">
                            <div class="form-group">
                                <label>Application Start Date <span class="text-
danger">*</span></label>
```

```
                        <div class="cal-icon">
                            <input type="text" id="sdate" name="sdate" class="form-control
datetimepicker" onblur="validateStartDate()">
                            <span id="sdateError" class="text-danger"></span>
                        </div>
                    </div>
                </div>
                <div class="col-sm-6">
                    <div class="form-group">
                        <label>Application End Date</label>
                        <div class="cal-icon">
                            <input type="text" id="edate" name="edate" class="form-control
datetimepicker" onkeyup="validateStartDate()">
                            <span id="sdateError" class="text-danger"></span>
                        </div>
                    </div>
                </div>
                <div class="col-sm-6">
                    <div class="form-group">
                        <label>Web Link<span class="text-danger">*</span></label>
                        <input class="form-control" id="link" name="link" type="text"
onkeyup="validateLink()">
                        <span id="linkError" class="text-danger"></span>
                    </div>
                </div>
                {% comment %} <div class="col-sm-6">
                    <div class="form-group">
                        <label>PDF/img</label>
                        <div class="profile-upload">
                            <div class="upload-img">
                                <i class="fa fa-upload" style="font-size: 28px; color: #a4c639"></i>
                            </div>
                            <div class="upload-input">
                                <input type="file" class="form-control" name="criteria" id="criteria"
```

```
accept=".pdf">
                                </div>
                            </div>
                        </div>
                    </div> {% endcomment %}
                    <!-- Job Description -->
                    <div class="col-12">
                        <div class="form-group">
                            <label>Job Description <span class="text-danger">*</span></label>
                            <textarea class="form-control" id="job_description"
name="job_description" rows="5" placeholder="Enter job description..."></textarea>
                            <small class="form-text text-muted">Separate responsibilities with line
breaks.</small>
                        </div>
                        {% comment %} <div class="form-check">
                            <input class="form-check-input" type="checkbox"
id="bullet_point_checkbox" onclick="toggleBulletPoints()">
                            <label class="form-check-label" for="bullet_point_checkbox">Enable
Bullet Points</label>
                        </div> {% endcomment %}
                    </div>


                    <div class="col-12">
                        <div class="form-group">
                            <label>Responsibilities <span class="text-danger">*</span></label>
                            <textarea class="form-control" id="responsibilities"
name="responsibilities" rows="5" required></textarea>
                            <small class="form-text text-muted">Separate responsibilities with line
breaks.</small>
                        </div>
                    </div>
                    <div class="col-12">
                        <div class="form-group">
```

```
                        <label>Preferred Skills <span class="text-danger">*</span></label>
                        <textarea class="form-control" id="preferred_skills"
name="preferred_skills" rows="5"></textarea>
                        <small class="form-text text-muted">Separate preferred skills with line
breaks.</small>
                    </div>
                </div>
                <!-- Qualifications -->
                <div class="col-12">
                    <div class="form-group">
                        <label>Qualifications <span class="text-danger">*</span></label>
                        <textarea class="form-control" id="qualifications" name="qualifications"
rows="5"></textarea>
                        <small class="form-text text-muted">Separate qualifications with line
breaks.</small>
                    </div>
                </div>
                <div class="col-12">
                    <div class="form-group">
                        <label>Required 10th CGPA</label>
                        <input class="form-control" name="required_tenth_cgpa" type="number"
step="0.01" min="0">
                    </div>
                </div>
                <div class="col-12">
                    <div class="form-group">
                        <label>Required 12th CGPA</label>
                        <input class="form-control" name="required_twelfth_cgpa"
type="number" step="0.01" min="0">
                    </div>
                </div>
                <div class="col-12">
                    <div class="form-group">
                        <label>Required Current CGPA</label>
```

```
                        <input class="form-control" name="required_current_cgpa"
type="number" step="0.01" min="0">
                    </div>
                </div>
                <div class="col-12">
                    <div class="form-group">
                        <label>Backlogs</label>
                        <input class="form-control" name="required_backlog" type="number"
step="0.01" min="0">
                    </div>
                </div>
            </div>
            {% comment %} <div class="m-t-20 text-center"> {% endcomment %}
                {% comment %} <button class="btn btn-primary submit-btn">Post
Job</button> {% endcomment %}
                {% comment %} <a href="{% url 'postjob' %}">Post Job</a>

            </div> {% endcomment %}
            <div class="m-t-20 text-center">
                <button id="postjob" class="btn btn-primary submit-btn">Post Job</button>
            </div>
            </div>

        </form>
      </div>
    </div>
  </div>

    </div>

  </div>
</div>
<div class="sidebar-overlay" data-reff=""></div>
```

```
<script src="{% static 'assets2/js/jquery-3.2.1.min.js' %}"></script>
<script src="{% static 'assets2/js/popper.min.js' %}"></script>
<script src="{% static 'assets2/js/bootstrap.min.js' %}"></script>
<script src="{% static 'assets2/js/jquery.slimscroll.js' %}"></script>
<script src="{% static 'assets2/js/select2.min.js' %}"></script>
<script src="{% static 'assets2/js/moment.min.js' %}"></script>
<script src="{% static 'assets2/js/bootstrap-datetimepicker.min.js' %}"></script>
<script src="{% static 'assets2/js/app.js' %}"></script>
{% comment %} <script src="{% static 'assets2/js/validation.js' %}"></script> {% endcomment %}
<script>
    {% comment %} function validateCname() {
        var cname = document.getElementById("cname").value;
        var cnameError = document.getElementById("cnameError");
        if (!cname.match(/^[A-Z]+$/)) {
            cnameError.textContent = "Company name should be in capital letters.";
        } else {
            cnameError.textContent = "";
        }
    } {% endcomment %}
    let bulletPointsEnabled = false;

    function toggleBulletPoints() {
        bulletPointsEnabled = !bulletPointsEnabled;
        const jobDescriptionField = document.getElementById('job_description');
        if (!bulletPointsEnabled) {
            jobDescriptionField.value = jobDescriptionField.value.replace(/• /g, '');
        }
    function handleEnter(event) {
        if (bulletPointsEnabled && event.key === "Enter") {
            event.preventDefault();
            const jobDescriptionField = document.getElementById('job_description');
            const currentPosition = jobDescriptionField.selectionStart;
            const text = jobDescriptionField.value;
```

```
            const newText = text.slice(0, currentPosition) + '\n• ' + text.slice(currentPosition);
            jobDescriptionField.value = newText;
            jobDescriptionField.selectionStart = jobDescriptionField.selectionEnd = currentPosition +
3;
        }
    }

    const jobDescriptionField = document.getElementById('job_description');
    jobDescriptionField.addEventListener('input', handleEnter);
    function validateJname() {
        var jname = document.getElementById("jname").value;
        var jnameError = document.getElementById("jnameError");

        // Regular expression to allow capital letters and spaces
        if (!jname.match(/^[A-Z ]+$/)) {
            jnameError.textContent = "Job name should be in capital letters.";
        } else {
            jnameError.textContent = "";
        }
    function validateSalary() {
        var salary = document.getElementById("salary").value;
        var salaryError = document.getElementById("salaryError");
        if (isNaN(salary) || parseInt(salary) <= 1000) {
            salaryError.textContent = "Salary should be a number greater than 1000.";
        } else {
            salaryError.textContent = "";
        }
    }

    function validateLink() {
        var link = document.getElementById("link").value;
        var linkError = document.getElementById("linkError");
        if (!link.startsWith("www.")) {
            linkError.textContent = "Website should start with 'www.'";
```

```
        } else {
            linkError.textContent = "";
        }
    }
    function validateStartDate() {
        var sdate = document.getElementById("sdate").value;
        var sdateError = document.getElementById("sdateError");

        // Get today's date in the format "YYYY-MM-DD"
        var today = new Date().toISOString().split("T")[0];

        if (sdate <= today) {
            sdateError.textContent = "Start date must be today or after today.";
        } else {
            sdateError.textContent = "";
        }
    }
    $(document).ready(function() {
        // Listen for keyup events on the file input
        $('#criteria').on('keyup', function() {
          var file = this.files[0];

          // Check if a file was provided and if it's a PDF
          if (file && !file.name.endsWith('.pdf')) {
            alert('Please upload a PDF file for the criteria.');
            // Clear the file input
            $('#criteria').val('');
          }
        });
      });
    </script>
</body>
</html>
```

9.2 Screen Shots

1.Index Page

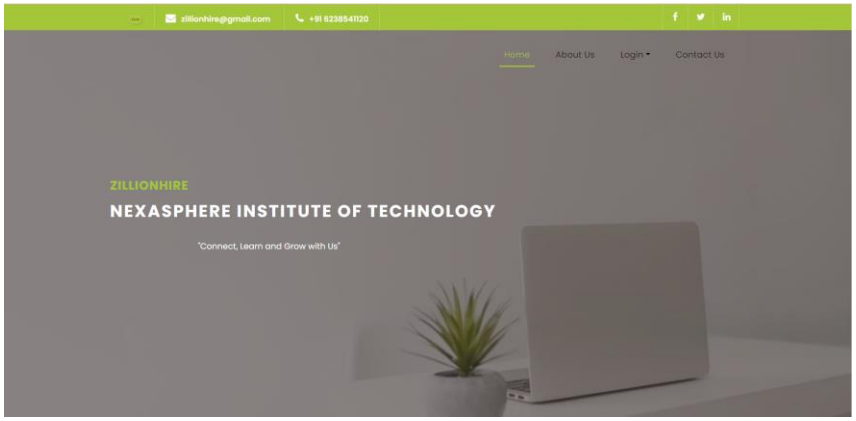


Fig 9.2 index

2.Company Registration Page

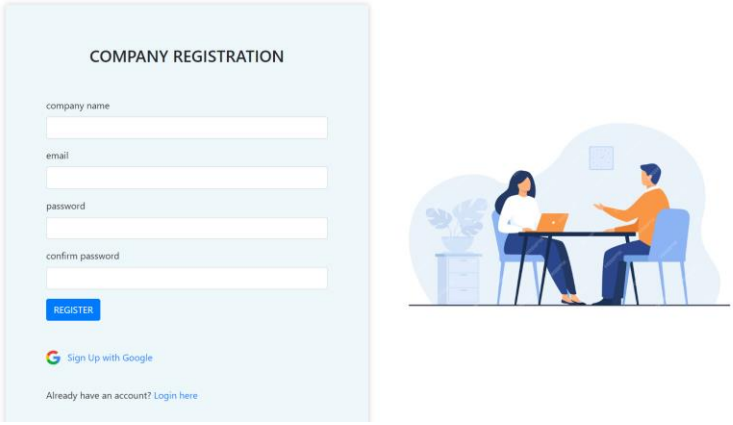


Fig 9.2 company registration

3.User Login

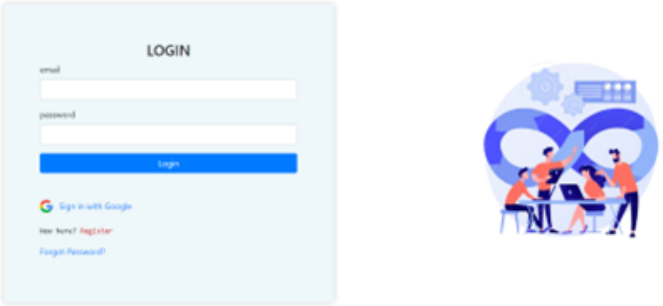


Fig 9.2 user login

4.Student Dashboard- profile

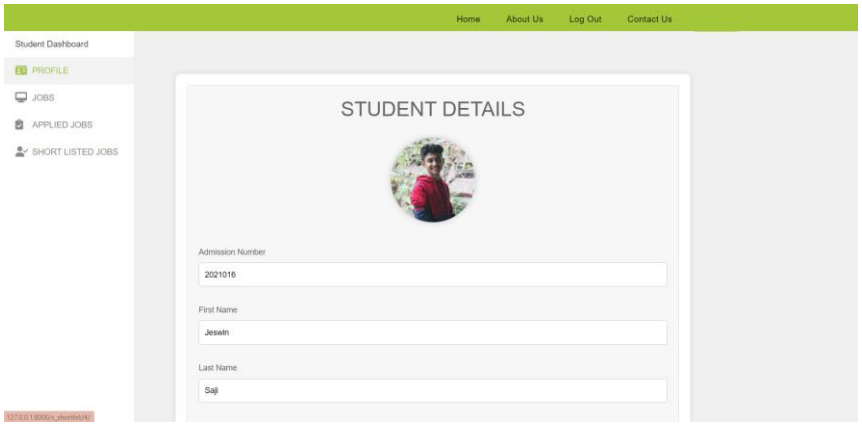


Fig 9.2 student profile

5. Student Dashboard- Jobs

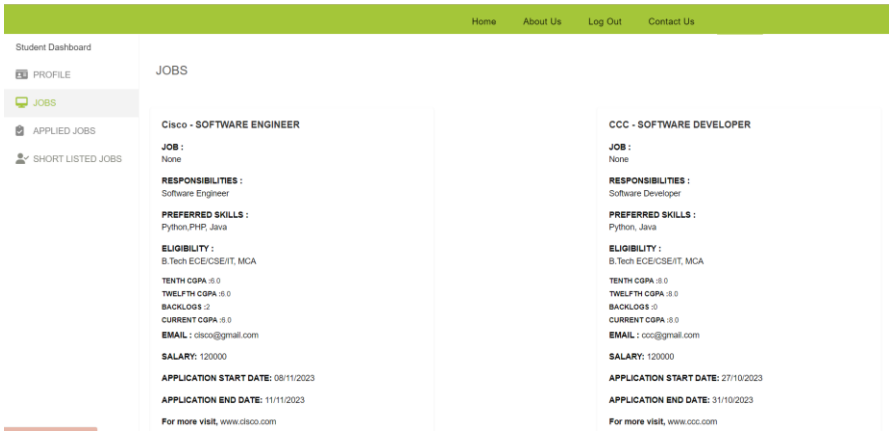


Fig 9.2 student dashboard jobs

6. Student Dashboard- Applied Jobs

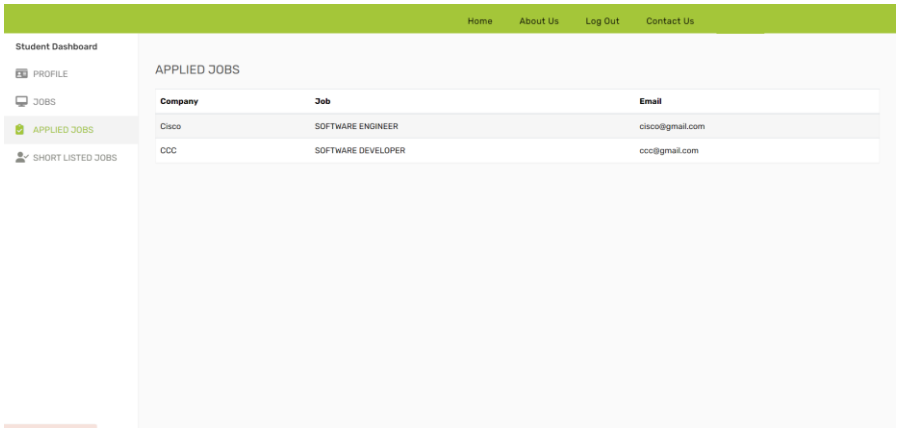


Fig 9.2 student appliedjobs

7. Student Dashboard- Short Listed Jobs

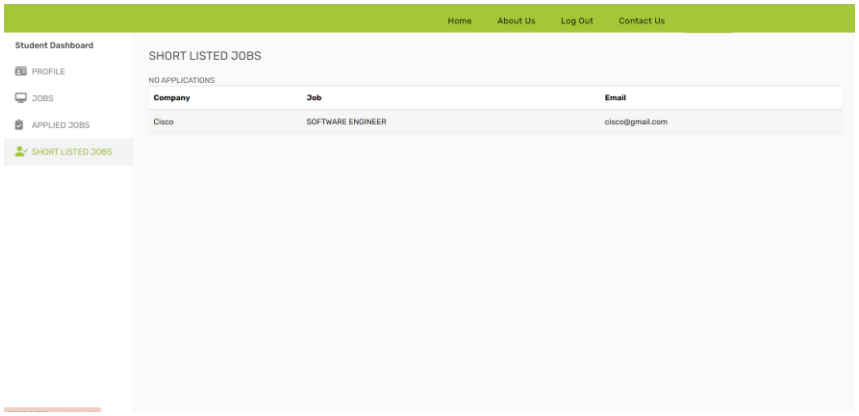


Fig 9.2 student dashboard shortlisted jobs

8. Company profile

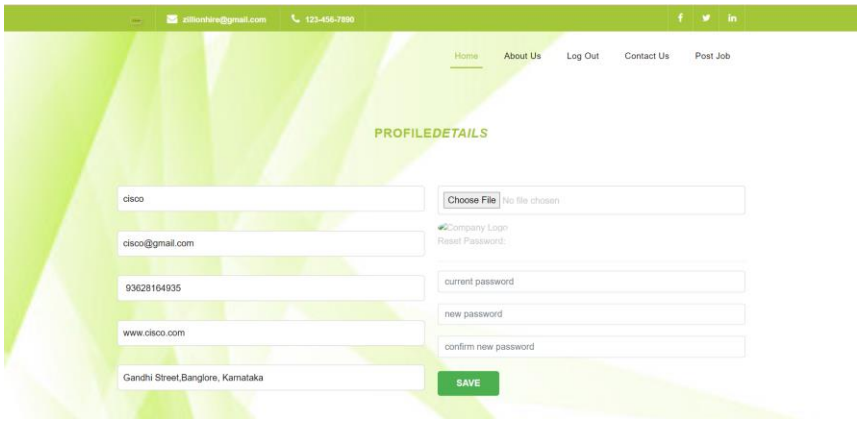


Fig 9.2 company profile

9. Company dashboard jobs page

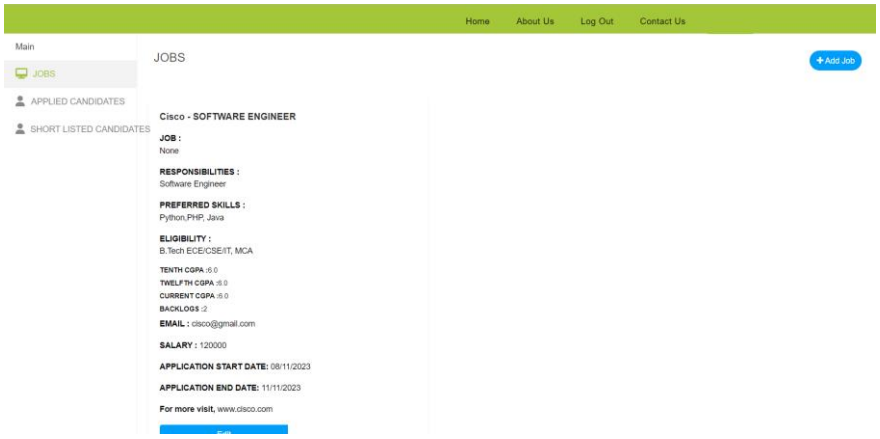


Fig 9.2 company dashboard jobs page

10. Admin Dashboard

Fig 9.2 admin dashboard

11. Admin – Student List

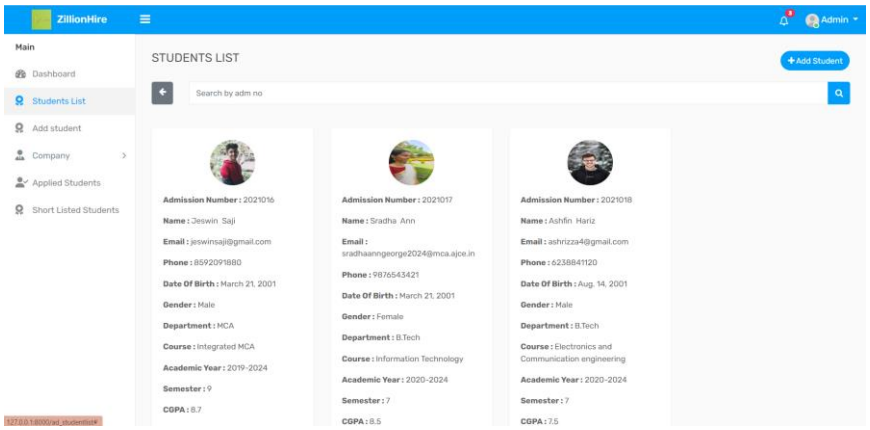


Fig 9.2 admin-student list

12. Admin – Short listed student list

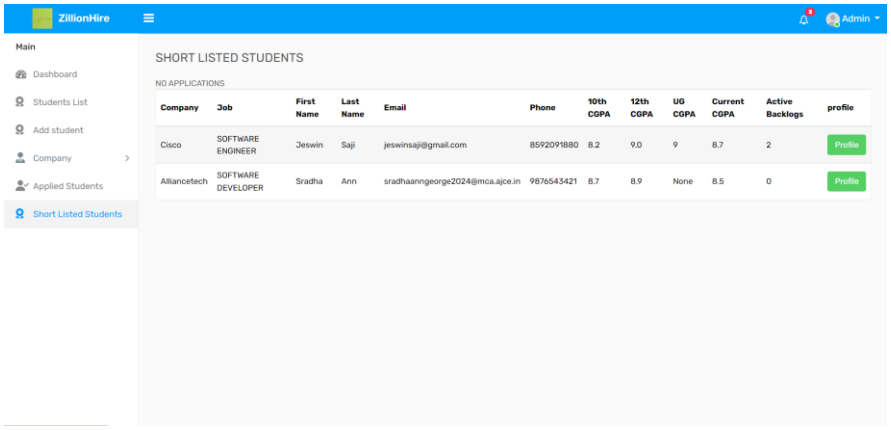


Fig 9.2 admin- shortlisted students