CAREERGO
Student Career Guidance and E-learning System

Mini Project Report

Submitted by

Febin Vinod

Reg. No.: AJC19MCA-I026

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, **"CAREERGO"** is the bona fide work of **FEBIN VINOD (Regno: AJC19MCA-I026)** 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 **Ms. Meera Rose Mathew**

Internal Guide **Coordinator**

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

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

Date: 31-10-23 **FEBIN VINOD**

KANJIRAPPALLY **Reg: AJC19MCA-I026**

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 and appreciation 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 her 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 Benjamin** 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.

FEBIN VINOD

ABSTRACT

.

Careergo is a career guidance and elearning website designed specifically to assist students in making informed decisions about their future career paths. The website aims to provide comprehensive resources and support to empower students in exploring various career options, and developing the necessary skills for their desired professions.

It provides interactive questionnaires that help students identify their skills, interests, and values. The website uses the gathered data to generate personalized career recommendations, by providing comprehensive information about educational programs, courses, and institutions.

Students can explore different majors or fields of study, and access different courses that help them make informed decisions about their educational paths. Also provides students with access to internships and Online courses. The platform aggregates opportunities from various sources, making it easier for students to find valuable work experiences and kick-start their careers.

Existing System

There is no unified Online Career Guidance System for career guidance. Students generally have to go through hundreds of colleges sites to choose their alma mater. they often find it difficult to find relevant information about the career they want.

Proposed System

The proposed Online Career Guidance System will make it easy for students to access the information they need. It will guide students by collecting their preferences and provide recommendations about different courses and available colleges according to their skills and interest.

CONTENT

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hypertext Markup Language
CSS	-	Cascading Style Sheets
SQL	-	Structured Query Language
UML	-	Unified Modelling Language
RDBMS	-	Relational Database Management System
ORM	-	Object-Relational Mapping
MTV	-	Model-Template-View
MVC	-	Model-View-Controller
API	-	Application Programming Interface
UI	-	User Interface
BDD	-	Behaviour-Driven Development
1NF	-	First Normal Form
2NF	-	Second Normal Form
3NF	-	Third Normal Form
XSS	-	Cross-Site Scripting
CSRF	-	Cross-Site Request Forgery
CRUD	-	Create, Read, Update, Delete

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The proposed Online Career Guidance System designed to provide students with comprehensive resources and support to make informed decisions about their future career paths. It aims to empower students by helping them explore various career options, identify their skills, interests, and values, and develop the necessary skills for their desired professions. The system consists of various components and functionalities to assist students in their career journey. The website provides personalized career assessments, information on colleges, online courses, internships and other resources related to different career fields. Key features include student registration and accounts, admin management of platform resources, integration with online courses, internships and secure payment gateways. CareerGo aims to be a one-stop guidance system for students to explore careers, education programs, gain experience and get expertise to chart out their career trajectories.

1.2 PROJECT SPECIFICATION

This project aims to bridge the gap in career guidance by offering students a comprehensive and user-friendly platform to make informed decisions about their future. It brings together assessments, online courses, internships and resources in one accessible place, simplifying the process of planning a successful career path.

The system consists of 3 actors. They are:

1. **Admin**:
    - Ability to manage the registrations and login for the website.
    - Ability to add and delete course providers details.
    - Manages student registrations and details.
    - Ability to view courses added by course providers.
    - Sets online assessments and manages them.
    - Views student assessment scores.
2. **Student**:
    - Register and login.
    - Can perform online assessments.
    - View recommended courses from Top listed colleges.
    - Can enroll in online courses.

- Can view and apply for internships.
- Make course payments.
- Achieve course completion certificates.
- Can view personalized student dahboard

3. **Course Provider**:
   - Register and login.
   - Can add, delete, update internships and online course details.
   - Can view and confirm student enrollment details.
   - Can view and manage payment details.
   - Can view personalized dashboard.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System Analysis is a crucial process in the field of information technology and business management. It involves the systematic evaluation of specific systems, whether they are technological or organizational, to understand their functions, identify areas for improvement, and make informed decisions about their development and enhancement.

System Analysis typically follows a structured approach that includes understanding the current state of a system, gathering requirements from stakeholders, defining objectives, and creating a detailed plan for system improvement or development. It is a multidisciplinary field that combines elements of computer science, engineering, and management to optimize systems' efficiency and effectiveness.

The benefits of System Analysis include the ability to streamline processes, enhance productivity, reduce costs, and improve overall functionality. It is widely used in various industries, such as software development, business management, healthcare, and more, to ensure that systems meet the needs of users and the objectives of the organization.

2.2 EXISTING SYSTEM

The current state of career guidance for students is characterized by a lack of a unified and comprehensive online system. As of now, students seeking guidance regarding their career choices and educational paths often face significant challenges. Also students need to scroll through various online course sites to find specific online courses. They typically resort to visiting numerous college websites and various sources to gather information about courses, colleges, and career opportunities. This scattered approach makes it both time-consuming and overwhelming for students to navigate through a myriad of options.

Furthermore, the information available on different websites is often disjointed and lacks consistency, making it difficult for students to make informed decisions. Students struggle to find relevant details about the careers they are interested in, leading to a lack of clarity about their potential career paths. The absence of a structured assessment system results in students being left to make crucial career decisions without a proper understanding of their skills, interests, and values.

2.2.1 NATURAL SYSTEM STUDIED

One of the natural systems studied as part of this project is Coursera, a widely recognized online learning platform. Coursera serves as a prominent example of an existing platform that offers a vast array of online courses from top universities and institutions worldwide. The platform presents a model for effective course delivery and assessment, as well as a user-friendly interface for both students and educators. Coursera's use of technology in delivering high-quality educational content is particularly relevant to our project, as it showcases the potential of online platforms in connecting students with valuable educational resources.

In our analysis of Coursera, we will delve into the structure and design of the platform, its features for course enrollment and completion, as well as the technology it employs for assessments and certification. Furthermore, we will examine how Coursera personalizes the learning experience for students through data-driven recommendations, thereby enhancing user engagement and satisfaction. By studying Coursera, we aim to gain insights into successful practices in online education and leverage these insights to design a more effective career guidance system for our target audience of students.

2.2.2 DESIGNED SYSTEM STUDIED

The core of our project revolves around the development of a comprehensive Online Career Guidance System, tailored to the unique needs of students seeking direction in their educational and career paths. The designed system offers interactive questionnaires to help students identify their skills, interests, and values to personalized assessments often found in e-learning environments. The collected data is then utilized to generate customized career recommendations, providing students with valuable insights into suitable educational programs, courses, and institutions. Much like online learning platforms, our system allows students to explore different online courses and Internships enabling them to make well-informed decisions about their educational paths. In addition, our system aggregates internship opportunities and online courses from various sources, streamlining the process of finding valuable work experiences and course options to kick-start their careers.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Lack of Centralization:** The current career guidance system lacks a centralized platform where students can access comprehensive information about career options and educational paths. Instead, students have to navigate through multiple websites and sources, making the process time-consuming and confusing.
- **Limited Career Assessment:** Existing systems often lack robust career assessment tools. Students may not have access to structured assessments that help them identify their skills, interests, personality traits, and values. Without such assessments, students may struggle to make well-informed career decisions.
- **Inadequate Information:** The information available on various websites is often fragmented and inconsistent. Students face difficulties in finding relevant and detailed information about the careers they are interested in. This lack of clarity can lead to uncertainty and hinder students from pursuing their preferred career paths.
- **No Personalization:** The existing system typically doesn't provide personalized recommendations to students based on their unique characteristics and preferences. Students are left without tailored guidance, which is essential for helping them align their choices with their individual goals and interests.
- **Limited Access to Opportunities:** Students often have difficulty finding information about internships and online courses related to their career goals. The current system lacks a central repository of opportunities, making it challenging for students to access valuable work experiences and additional education options.

2.4 PROPOSED SYSTEM

The proposed Online Career Guidance System is a comprehensive and user-focused solution designed to address the limitations of the existing career guidance system. This innovative platform aims to empower students in making informed decisions about their future career paths by providing them with a centralized and accessible resource. One of the standout features of the proposed system is the use of interactive questionnaires that assist students in identifying their skills, interests, and values, ensuring that their career choices align with their individual strengths and preferences. The data gathered from these assessments is harnessed to generate personalized career recommendations, offering students valuable insights into suitable educational programs, courses, and institutions. The proposed system goes beyond career guidance, offering students access to internships and online courses, effectively aggregating opportunities from various sources and simplifying the process of finding valuable work experiences and furthering their education. The integration of a secure payment gateway ensures seamless and secure transactions, while user accounts and personalization allow students to track their progress. Moreover, the system fosters online courses and internship programs, creating an immersive environment for students to enhance their skills, and knowledge. Overall, the proposed system stands as a promising and user-centric solution that addresses the limitations of the existing system, providing students with the support and resources needed to make well-informed career decisions.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Centralization of Resources:** The proposed system offers a centralized platform where students can access a wide range of career guidance resources, eliminating the need to navigate through multiple websites. This centralization simplifies the process of gathering information about career options and educational paths.
- **Personalized Career Guidance:** Through interactive questionnaires and assessments, the system provides students with personalized career recommendations. This personalization ensures that students receive guidance that aligns with their unique skills, interests, personality traits, and values, helping them make well-informed decisions.
- **Comprehensive Information:** The system offers comprehensive information about educational programs, courses, and institutions, enabling students to explore various career paths. This extensive data empowers students with the knowledge needed to choose the right educational and career options.
- **Access to Opportunities:** In addition to career guidance, the system aggregates internships and online courses from various sources. This accessibility to opportunities simplifies the process of finding internships and online courses, helping students gain valuable work experiences and further their education.
- **User Accounts and Tracking:** Students can create personalized user accounts that allow them to track their progress, view their assessment results, and access enrollment details. This feature helps students stay organized and informed about their educational journey.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study analyses and assesses the viability and practicality of a proposed project or business venture. Its main goal is to establish whether the proposal is feasible, economically viable, and worthwhile of future investigation. Before allocating major resources to a project, the research aids decision-makers and stakeholders in evaluating the potential risks and benefits involved. A feasibility study is conducted to assess the solution's viability and establish whether it is viable and implementable in the software. Information on resource availability, software development costs, the advantages of the software to the company once it is built, and the costs associated with. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards.

The proposed system CareerGo, is a platform designed to empower students in making informed decisions about their future career paths. The system addresses a significant need for students seeking comprehensive career guidance. The user-friendly interface and interactive features contribute to its social feasibility. The system can be integrated into existing operations with relative ease, to ensure smooth adoption by all user groups. The system is feasible due to its relevance, technological availability, user acceptance, operational efficiency, compliance, and achievable project timeline.

3.1.1 Economical Feasibility

It evaluates whether the necessary software may result in financial gains for a company. It includes expenses for the software development team, projected costs for hardware and software, the cost of conducting a feasibility study, and other expenses.
The suggested CareerGo system's economic viability shows that it is commercially viable and has the prospective benefits that outweigh the related costs. The system was entirely developed using open-source software, and all necessary resources were readily available, which further reduced the development cost. This indicates that the system is economically feasible for development. The website can profit from enrolment commissions and ease communication with educational institutions by teaming up with course providers and internship programs.

3.1.2 Technical Feasibility

Technical feasibility, which focuses on determining if the proposed project is technically feasible and realizable. It seeks to ascertain whether the project can be successfully carried out with the available technology, resources, and experience. It evaluates the present technology and resources needed to fulfil user needs in the software within the given constraints of time and money.

The suggested Online Career Guidance System CareerGo is technically feasible since it can be built and operated with current infrastructure and technology. In the proposed system, the frontend is developed using HTML, CSS, Bootstrap, JavaScript and the backend is developed using python programming to create a user-friendly and interactive platform. A robust and scalable database SqLite is used to store user information, course details, internship opportunities, and assessment results. It is possible to create a system that will match user needs and offer a fluid, interactive career exploration and decision-making journey by carefully addressing the technical factors.

3.1.3 Behavioral Feasibility

Behavioral feasibility assesses the likelihood of successful system adoption and its compatibility with the intended users' behaviors. In the context of the proposed Online Career Guidance System, several factors support its behavioral feasibility. First and foremost, the system addresses a critical need for students by providing a user-centric platform that streamlines the process of making informed career decisions.

The system's user-friendly interface, interactive assessments, and personalized recommendations are designed to resonate with the behaviors of students seeking clarity and guidance in their career choices. Furthermore, the incorporation of online courses, and interactive internships aligns with the behavior of students who benefit from expert guidance and interaction in their educational and career journeys. The centralization of resources and the ease of access to internships and online courses are expected to align with the behavioral preferences of students looking for convenience and efficiency in their educational and professional development.

3.1.4 Feasibility Study Questionnaire

A. Project Overview?

The Career guidance website designed specifically to assist students in making informed decisions about their future career paths. The website aims to provide comprehensive resources and support to empower students in exploring various career options, and developing the necessary skills for their desired professions.

Provides interactive questionnaires that help students identify their skills, interests, and values. The website uses the gathered data to generate personalized career recommendations, by providing comprehensive information about educational programs, courses, and institutions. It aims to be a reliable and user-friendly destination for students seeking guidance and direction in shaping their academic and professional journeys. Also provides students with access to internships and Online courses.

B. To what extend the system is proposed for?

The proposed career guidance website is designed to cater specifically to students seeking comprehensive resources and support to make informed decisions about their future career paths. The system's primary objective is to empower students in exploring various career options, identifying their skills, interests, and values through interactive questionnaires, and providing personalized course recommendations based on the gathered data. Also provides students with access to internships and Online courses.

C. Specify the Viewers/Public which is to be involved in the System?

Users (Students), Course providers

D. List the Modules included in your System?

- Registration and Login
- Career Assessment
- Educational Courses
- Internships
- Online courses
- Payment
- Personalized User Accounts

E. Identify the users in your project?
Admin, Students, Course providers

F. Who owns the system?
Administrator

G. System is related to which firm/industry/organization?
Career Consultancy

H. Details of person that you have contacted for data collection?
Justin Sanjay (Student)
St. Thomas H. S. S, Pala
Kerala, India

Questionnaire to collect details about the project?

1. How did you become aware of the career courses available to you?

Consulted school counsellor for guidance and recommendations from friends and family.

2. Have you taken any psychometric tests or career assessment tools to help you decide on your career course?

No, Relied on self-reflection and research.

3. What were the most significant considerations when selecting your career course?

Course content, curriculum, location and future scope of the course were the most important factors.

4. Were there any specific challenges you faced in the decision-making process?

Deciding between two equally interesting courses was difficult.

5. Do you have a clear understanding of the potential career opportunities and job prospects after completing the chosen course?

To some extent, planning to explore more during the course.

6. Have you attended any internships or online courses for your career course selection?

No, Find it difficult to discover effective internships and online courses.

7. Did you face any other difficulties while choosing your career?

Insufficient knowledge about various careers, industries, and educational opportunities may find difficult for decision-making.

8. Are you confident about that your career selection was right ?

Able to gather details about only limited number of courses.

9. How willing are you to adapt to changing career trends and industries?

prepared to be adaptable and continuously learn to stay relevant in the job market.

10. Do you think an online website that guides users to choose their career would be

beneficial in future?

Certainly, an online career guidance website can provide a centralized repository of information about various career options, educational programs, and course recommendations based on various assessments and user interests are beneficial.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel core i5
RAM	-	8GB
Hard disk	-	256 GB

3.2.2 Software Specification

Front End	-	HTML5, CSS, Bootstrap
Back End	-	Python
Database	-	SQLite
Client on PC	-	Windows 7 and above.
Technologies used	-	DJANGO, HTML5, AJAX, Bootstrap, JS, jQuery

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django is a powerful and open-source web framework for Python that simplifies the web development process. It follows the model-view-controller (MVC) architectural pattern, known as the model-view-template (MVT) in Django's terminology. One of Django's standout features is its built-in admin interface, enabling users to manage application data models effortlessly. It includes an Object-Relational Mapping (ORM) system, allowing developers to interact with databases using Python objects instead of complex SQL queries. Additionally, Django offers efficient URL routing, user authentication, and a template system for creating dynamic web content. Security is a top priority, with Django providing safeguards against common web vulnerabilities. The framework also promotes reusability through a wide array of community-contributed apps and packages. All these features make Django a preferred choice for web developers seeking a robust, secure, and efficient solution for web application development

3.3.2 SQLite

SQLite is a self-contained and serverless relational database management system that excels in simplicity, efficiency, and versatility. It is widely known for being lightweight and requires minimal configuration, making it a popular choice for embedded systems, mobile applications, and small to medium-sized projects. SQLite stores data in a single, portable file, eliminating the need for a separate server process and complex setup. Despite its small footprint, it supports SQL syntax, transactions, and various data types, making it suitable for a wide range of applications. Its ease of use and cross-platform compatibility have contributed to SQLite's widespread adoption in the software development world, providing a reliable and efficient means of data storage and retrieval.

- Self-Contained: SQLite is a serverless database system, meaning it doesn't require a separate server process to operate. All data is stored in a single, self-contained file, making it easy to manage and deploy. This simplicity is advantageous for embedded systems and applications where a full-scale database server would be overkill.

- ACID Compliance: SQLite adheres to the principles of ACID (Atomicity, Consistency, Isolation, Durability) to ensure data integrity. It supports transactions, which means that operations can be grouped together and executed as a single unit. If any part of the transaction fails, the entire transaction can be rolled back, maintaining data consistency.

- Cross-Platform Compatibility: SQLite is cross-platform and widely supported across various operating systems and programming languages. This portability makes it an attractive choice for developers working on applications that need to run on different platforms without significant modification.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

System design is a critical phase in the software development process that serves as the bridge between conceptualizing a software solution and bringing it to life. It plays a pivotal role in translating the requirements gathered during the earlier stages into a well-structured and efficient system. At its core, system design involves making key decisions about the system's architecture, components, modules, interfaces, and data flows, creating a blueprint for the entire system. This blueprint not only addresses the technical aspects but also factors like scalability, security, and user experience. System design is instrumental in ensuring that the resulting software system is reliable, maintainable, and performs optimally. It provides a reference point for the development team, guiding them towards the successful creation of the software system.

In this phase, designers and architects collaborate to create a comprehensive system design that offers a clear roadmap for developers to implement. The design document serves as a guiding light throughout the development process, outlining how different parts of the system will interact and function together to achieve the desired objectives. The resulting system design is a crucial foundation for the development team to build upon, ensuring that the software solution meets the needs of its users and stakeholders while adhering to best practices in software engineering.

4.2 UML DIAGRAM

Unified Modeling Language (UML) is a standardized visual language used in software engineering to represent, design, and document software systems. UML diagrams serve as a common communication tool for developers, stakeholders, and designers, offering a clear and structured way to depict various aspects of a software system. UML diagrams encompass a range of types, including class diagrams, sequence diagrams, use case diagrams, and more, each tailored to convey specific information about the system's architecture, behavior, and interactions. UML diagrams play a crucial role in the software development process, aiding in the visualization, analysis, and design of complex systems, ultimately leading to more efficient and effective software development.

Types of UML diagrams

- Use case diagram
- Sequence diagram
- State diagram

- Activity diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a type of Unified Modeling Language (UML) diagram used in software engineering to represent the interactions between various actors (users, systems, or external entities) and a system. It provides a high-level view of the system's functionality by illustrating the different use cases or scenarios in which actors interact with the system.

In a use case diagram, actors are depicted as stick figures, and use cases are represented as ovals. Lines connecting actors to use cases show the interactions or actions they perform within the system. Use case diagrams help in understanding the system's requirements, identifying user needs, and defining the boundaries of the system.

These diagrams are valuable tools for project stakeholders, developers, and designers to visualize the system's behavior and functionalities, ensuring that the software meets the intended user requirements.

- **Actor Definition:** Clearly define and label all actors involved in the system. Actors represent external entities interacting with the system.
- **Use Case Naming:** Use descriptive names for use cases to accurately convey the functionality they represent.
- **Association Lines:** Use solid lines to represent associations between actors and use cases. This signifies the interaction between entities.
- **System Boundary:** Draw a box around the system to indicate its scope and boundaries. This defines what is inside the system and what is outside.
- **Include and Extend Relationships:** Use "include" relationships to represent common functionalities shared among multiple use cases. Use "extend" relationships to show optional or extended functionalities.

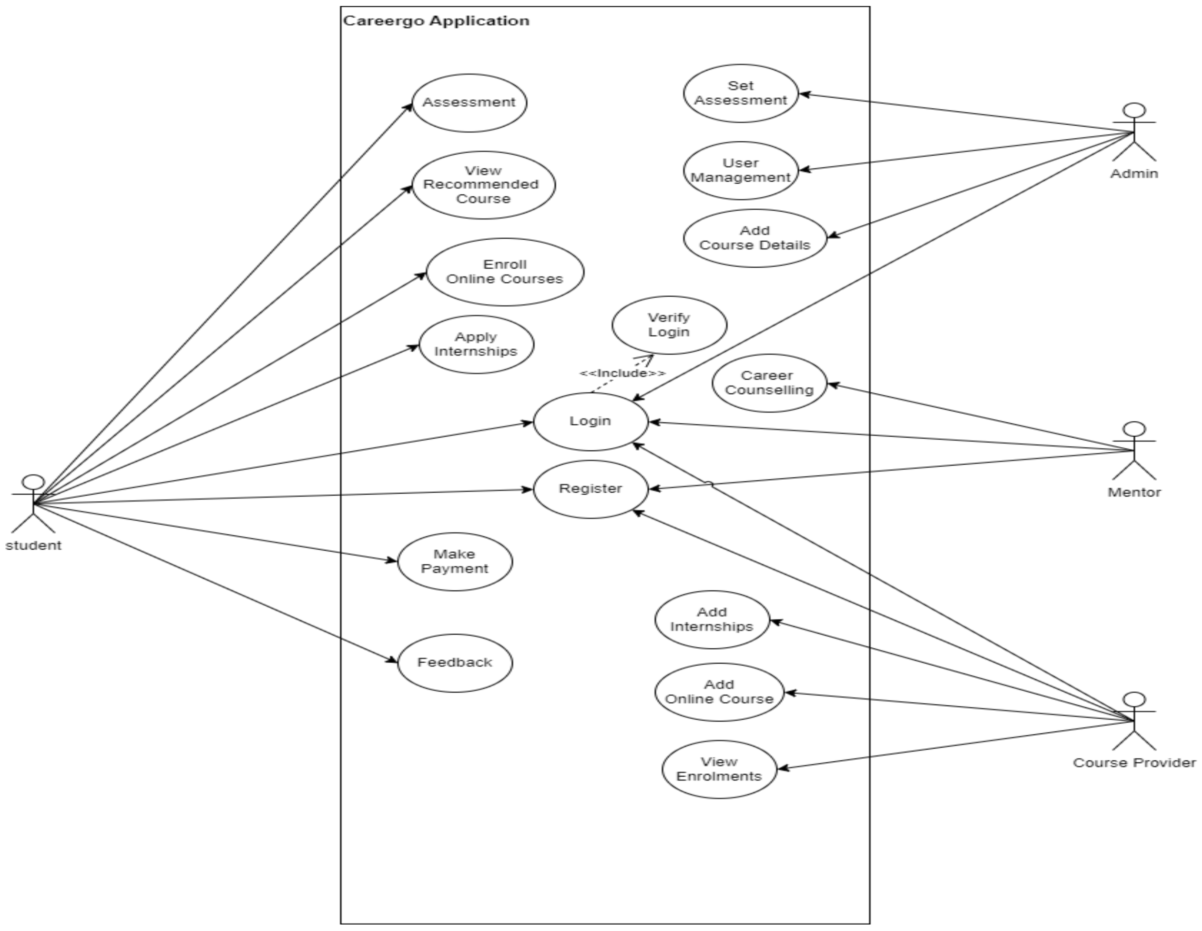


Fig 1: Use Case Diagram of Careergo

4.2.2 SEQUENCE DIAGRAM

Sequence Diagrams stand as dynamic models in software engineering, portraying the chronological flow of interactions between various objects or components within a system. They spotlight the order in which messages are exchanged, revealing the behavior of the system over time. Actors and objects are represented along a vertical axis, with arrows indicating the sequence of messages and their direction. Lifelines, extending vertically from actors or objects, illustrate their existence over the duration of the interaction. These diagrams serve as a vital tool for visualizing system behavior and understanding the temporal aspects of a software process. Through Sequence Diagrams, stakeholders gain valuable insights into how different elements collaborate to achieve specific functionalities, facilitating more effective communication among development teams and stakeholders alike. This detailed representation not only aids in detecting potential bottlenecks or inefficiencies but also provides a foundation for refining system performance in the later stages of software development.

- **Vertical Ordering:** Represent actors and objects along a vertical axis, indicating the order of interactions from top to bottom.

- **Lifelines:** Extend vertical lines from actors or objects to denote their existence and participation in the interaction.
- **Activation Bars:** Use horizontal bars along lifelines to show the period during which an object is active and processing a message.
- **Messages and Arrows:** Use arrows to indicate the flow of messages between objects, specifying the direction of communication.
- **Self-Invocation:** Use a looped arrow to represent self-invocation, when an object sends a message to itself.
- **Return Messages:** Indicate return messages with a dashed arrow, showing the response from the recipient.
- **Focus on Interaction:** Sequence Diagrams focus on the chronological order of interactions, avoiding implementation details.
- **Concise Notation:** Use clear and concise notation to represent messages and interactions, avoiding unnecessary complexity.
- **Consider System Boundaries:** Clearly define the boundaries of the system to indicate what is included in the interaction.
- **Feedback and Validation**: Seek feedback from stakeholders and team members to ensure the diagram accurately represents the system behavior.

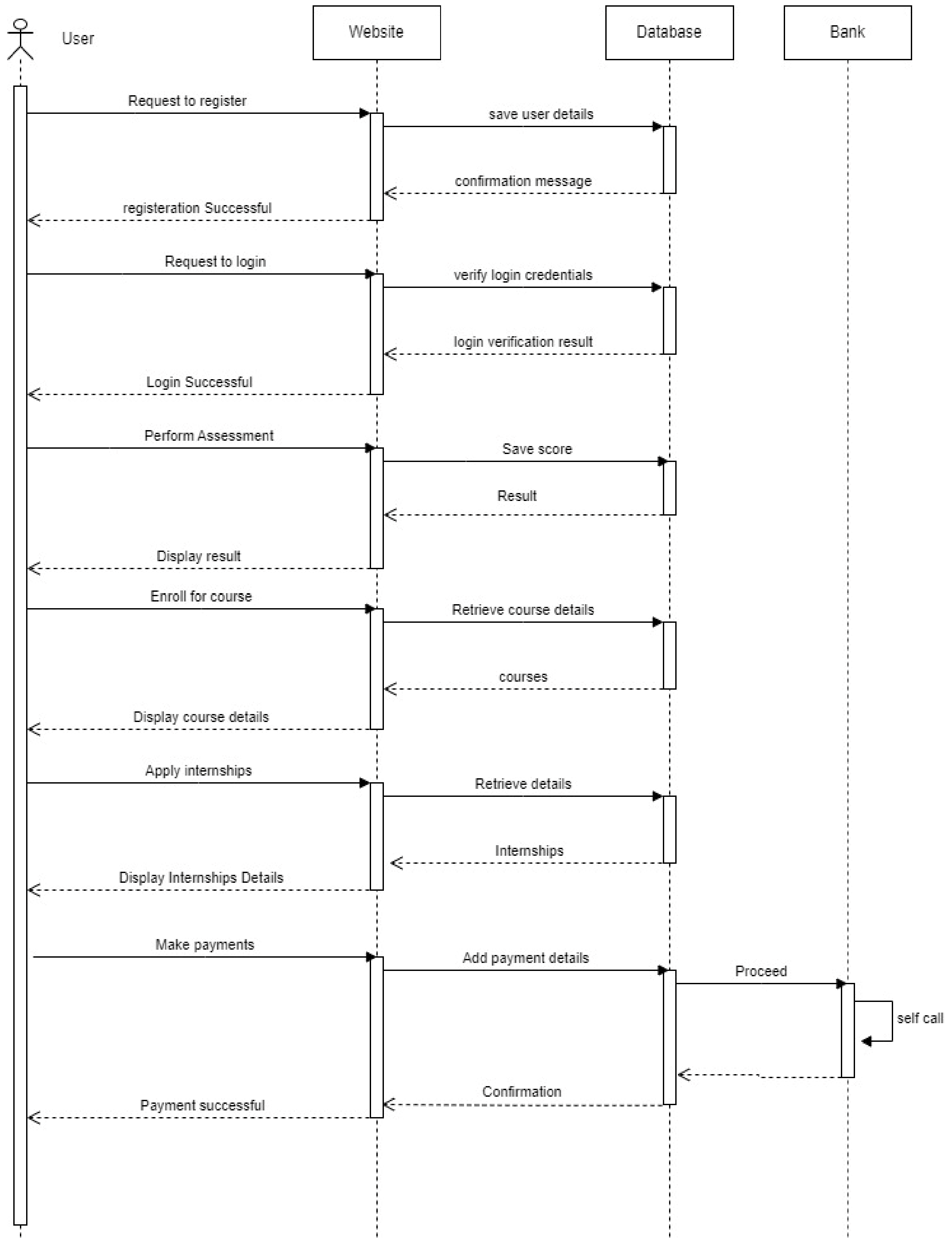


Fig 2: Sequence Diagram of Careergo

4.2.3 State Chart Diagram

A State Chart Diagram, a fundamental component of UML, provides a visual representation of an object's lifecycle states and the transitions between them. It depicts the dynamic behavior of an entity in response to events, showcasing how it transitions from one state to another. Each state represents a distinct phase in the object's existence, while transitions illustrate the conditions triggering state changes. Initial and final states mark the commencement and termination of the object's lifecycle. Orthogonal regions allow for concurrent states, capturing multiple aspects of the object's behavior simultaneously. Hierarchical states enable the representation of complex behaviors in a structured manner. Entry and exit actions depict activities occurring upon entering or leaving a state. Moreover, guard conditions ensure that transitions occur only under specified circumstances. State Chart Diagrams play a crucial role in understanding and designing the dynamic behavior of systems, aiding in the development of robust and responsive software applications.

Key notations for State Chart Diagrams:

- **Initial State:** Represented by a filled circle, it signifies the starting point of the object's lifecycle.
- **State:** Depicted by rounded rectangles, states represent distinct phases in an object's existence.
- **Transition Arrow:**Arrows denote transitions between states, indicating the conditions triggering a change.
- **Event:** Events, triggers for state changes, are labeled on transition arrows.
- **Guard Condition:** Shown in square brackets, guard conditions specify criteria for a transition to occur.
- **Final State:** Represented by a circle within a larger circle, it indicates the end of the object's lifecycle.
- **Concurrent State:**Represented by parallel lines within a state, it signifies concurrent behaviors.
- **Hierarchy:**States can be nested within other states to represent complex behavior.
- **Entry and Exit Actions**: Actions occurring upon entering or leaving a state are labeled within the state.
- **Transition Labels:** Labels on transition arrows may indicate actions or operations that accompany the transition.

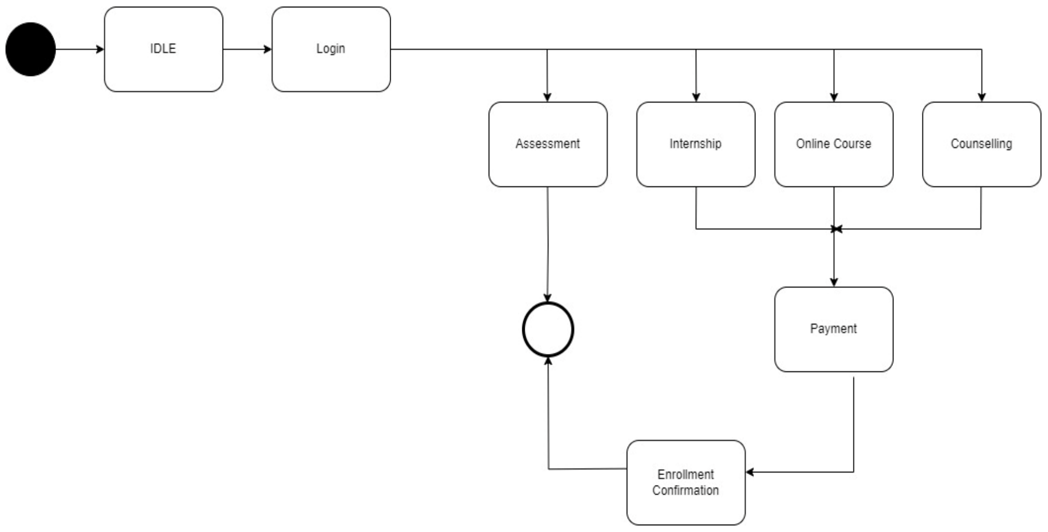


Fig 3: State Chart Diagram of Careergo

4.2.4 Activity Diagram

An Activity Diagram is a visual representation within UML that illustrates the flow of activities and actions in a system or process. It employs various symbols to depict tasks, decision points, concurrency, and control flows. Rectangles signify activities or tasks, while diamonds represent decision points, allowing for conditional branching. Arrows indicate the flow of control from one activity to another. Forks and joins denote concurrency, where multiple activities can occur simultaneously or in parallel. Swimlane segregate activities based on the responsible entity, facilitating clarity in complex processes. Initial and final nodes mark the commencement and completion points of the activity. Decision nodes use guards to determine the path taken based on conditions. Synchronization bars enable the coordination of parallel activities. Control flows direct the sequence of actions, while object flows depict the flow of objects between activities. Activity Diagrams serve as invaluable tools for understanding, modeling, and analyzing complex workflows in systems and processes. They offer a structured visual representation that aids in effective communication and system development.

Key notations for Activity Diagrams:

- **Initial Node:** Represented by a solid circle, it signifies the starting point of the activity.
- **Activity:** Shown as a rounded rectangle, it represents a task or action within the process.

- **Decision Node:** Depicted as a diamond shape, it indicates a point where the process flow can diverge based on a condition.
- **Merge Node:** Represented by a hollow diamond, it signifies a point where multiple flows converge.
- **Fork Node:** Shown as a horizontal bar, it denotes the start of concurrent activities.
- **Join Node:** Depicted as a vertical bar, it marks the point where parallel flows rejoin.
- **Final Node**: Represented by a solid circle with a border, it indicates the end of the activity.
- **Control Flow:** Arrows connecting activities, showing the sequence of actions.
- **Object Flow:** Lines with arrows representing the flow of objects between activities.
- **Swimlane:** A visual container that groups activities based on the responsible entity or system component.
- **Partition**: A horizontal or vertical area within a swimlane, further organizing activities.

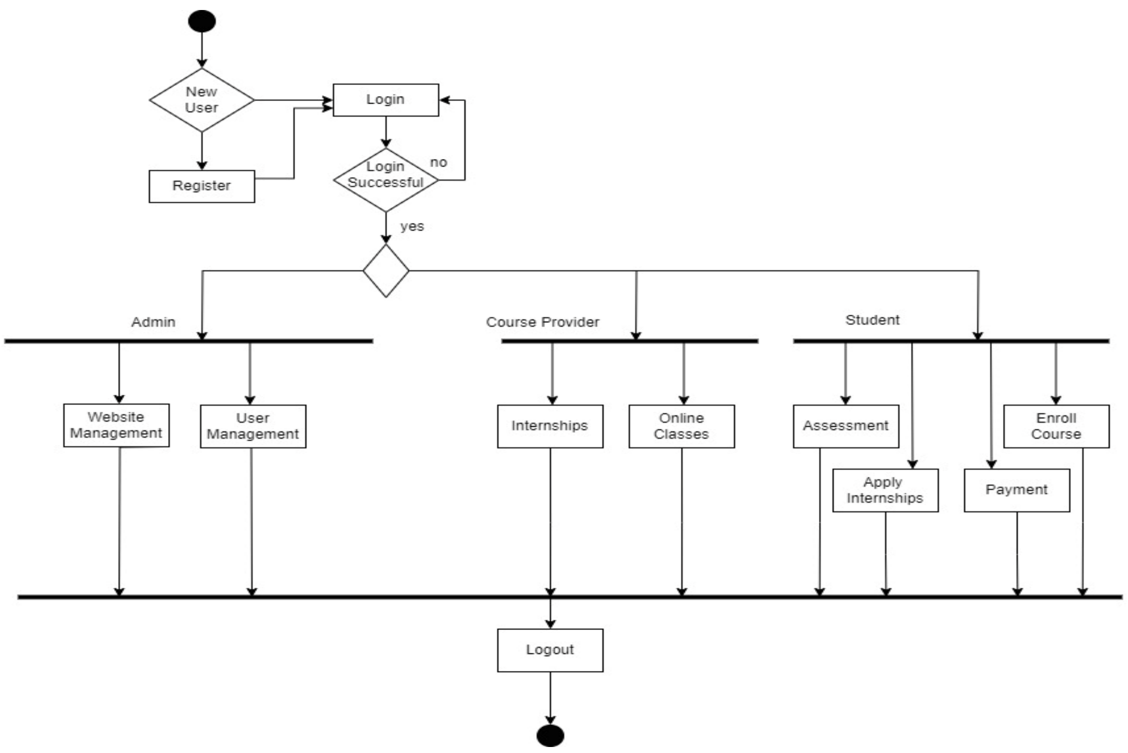


Fig 4: Activity Diagram of Careergo

4.2.5 Class Diagram

A Class Diagram, a fundamental tool in UML, visually represents the structure of a system by illustrating classes, their attributes, methods, and relationships. Classes, depicted as rectangles, encapsulate data and behavior within a system. Associations between classes indicate relationships, showcasing how they interact. Multiplicity notations specify the cardinality of associations. Inheritance is denoted by an arrow indicating the subclass inheriting from a super-class. Aggregation and composition illustrate whole-part relationships between classes. Interfaces, depicted as a circle, outline the contract of behavior a class must implement. Stereotypes provide additional information about a class's role or purpose. Dependencies highlight the reliance of one class on another. Association classes facilitate additional information about associations. Packages group related classes together, aiding in system organization. Class Diagrams play a pivotal role in system design, aiding in conceptualizing and planning software architectures. They serve as a blueprint for the development process, ensuring a clear and structured approach to building robust software systems.

Key notations for Class Diagrams:

- **Class:** Represented as a rectangle, it contains the class name, attributes, and methods.
- **Attributes:** Displayed as a list within the class, they describe the properties or characteristics of the class.
- **Methods:** Also listed within the class, they define the behaviors or operations of the class
- **Associations:** Lines connecting classes, indicating relationships and connections between them.
- **Multiplicity Notation:** Indicates the number of instances one class relates to another.
- **Inheritance:** Shown as an arrow, it signifies that one class inherits properties and behaviors from another.
- **Interfaces:** Represented by a dashed circle, they define a contract of behavior that implementing classes must follow.
- **Stereotypes:** Additional labels or annotations applied to classes to provide more information about their role or purpose.
- **Dependencies:** Shown as a dashed line with an arrow, they indicate that one class relies on another in some way.
- **Association Classes:** Represented as a class connected to an association, they

provide additional information about the relationship.

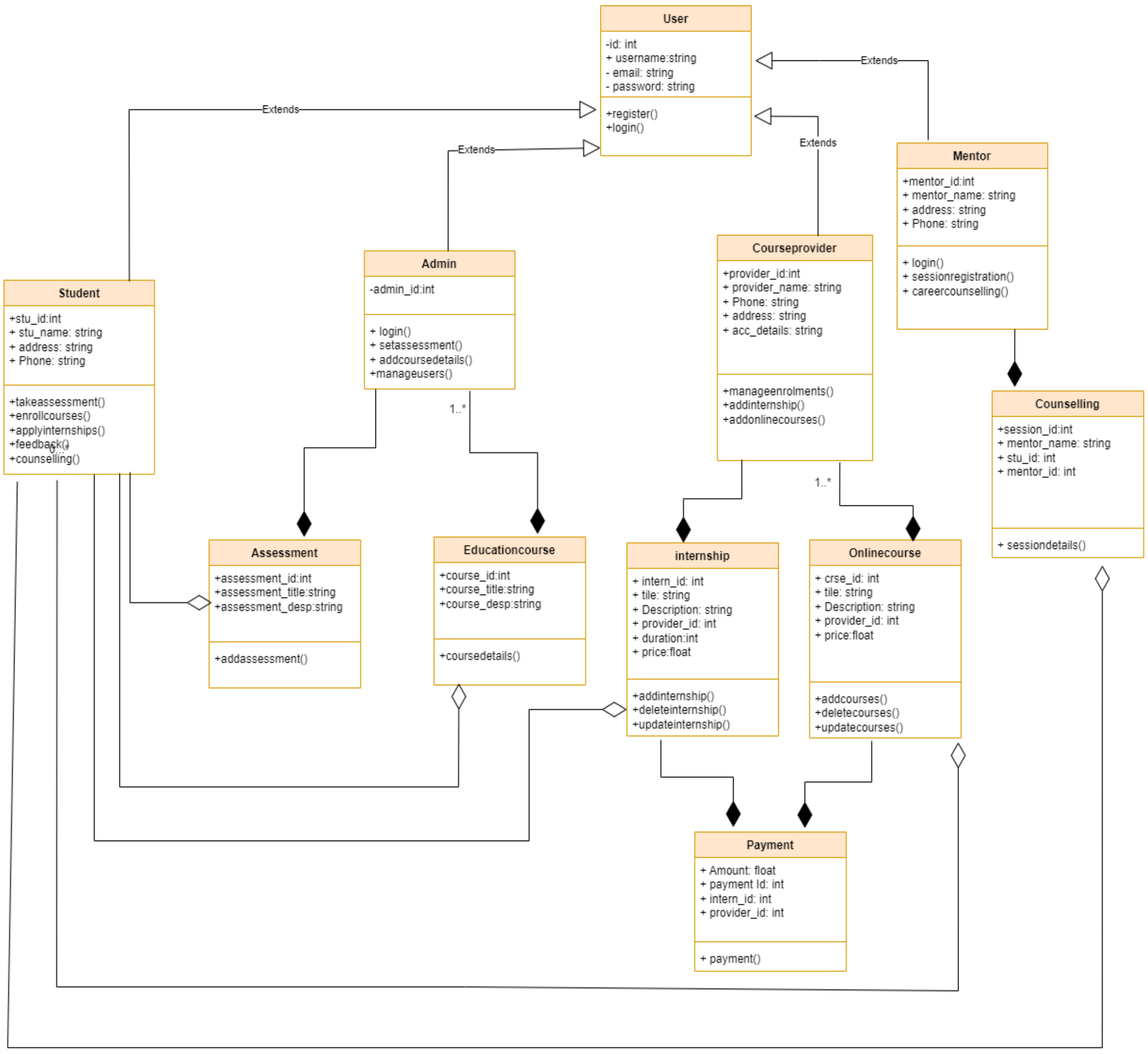


Fig 5: Class Diagram of Careergo

4.2.6 Object Diagram

An Object Diagram in UML provides a snapshot of a system at a specific point in time, displaying the instances of classes and their relationships. Objects, represented as rectangles, showcase the state and behavior of specific instances. Links between objects depict associations, highlighting how they interact. Multiplicity notations indicate the number of instances involved in associations. The object's state is displayed through attributes and their corresponding values. Object Diagrams offer a detailed view of runtime interactions, aiding in system understanding and testing. They focus on real-world instances, providing a tangible representation of class relationships. While similar to Class Diagrams, Object Diagrams emphasize concrete instances rather than class definitions. They serve as valuable tools for validating system design and verifying that classes and associations work as intended in practice. Object Diagrams play a crucial role in system validation, ensuring that the system's components and their interactions align with the intended design and requirements.

Key notations for Object Diagrams:

- **Object**: Represented as a rectangle, it contains the object's name and attributes with their values.
- **Links:** Lines connecting objects, indicating associations or relationships between them.
- **Multiplicity Notation:** Indicates the number of instances involved in associations.
- **Attributes with Values:** Displayed within the object, they represent the state of the object at a specific point in time.
- **Role Names:** Labels applied to associations, providing additional information about the nature of the relationship.
- **Object Name:** Represents the name of the specific instance.
- **Association End:** Indicates the end of an association, often with a role name and multiplicity.
- **Dependency Arrow:** Indicates a dependency relationship, where one object relies on another.
- **Composition Diamond:** Represents a stronger form of ownership, where one object encapsulates another.
- **Aggregation Diamond:** Signifies a whole-part relationship between objects.

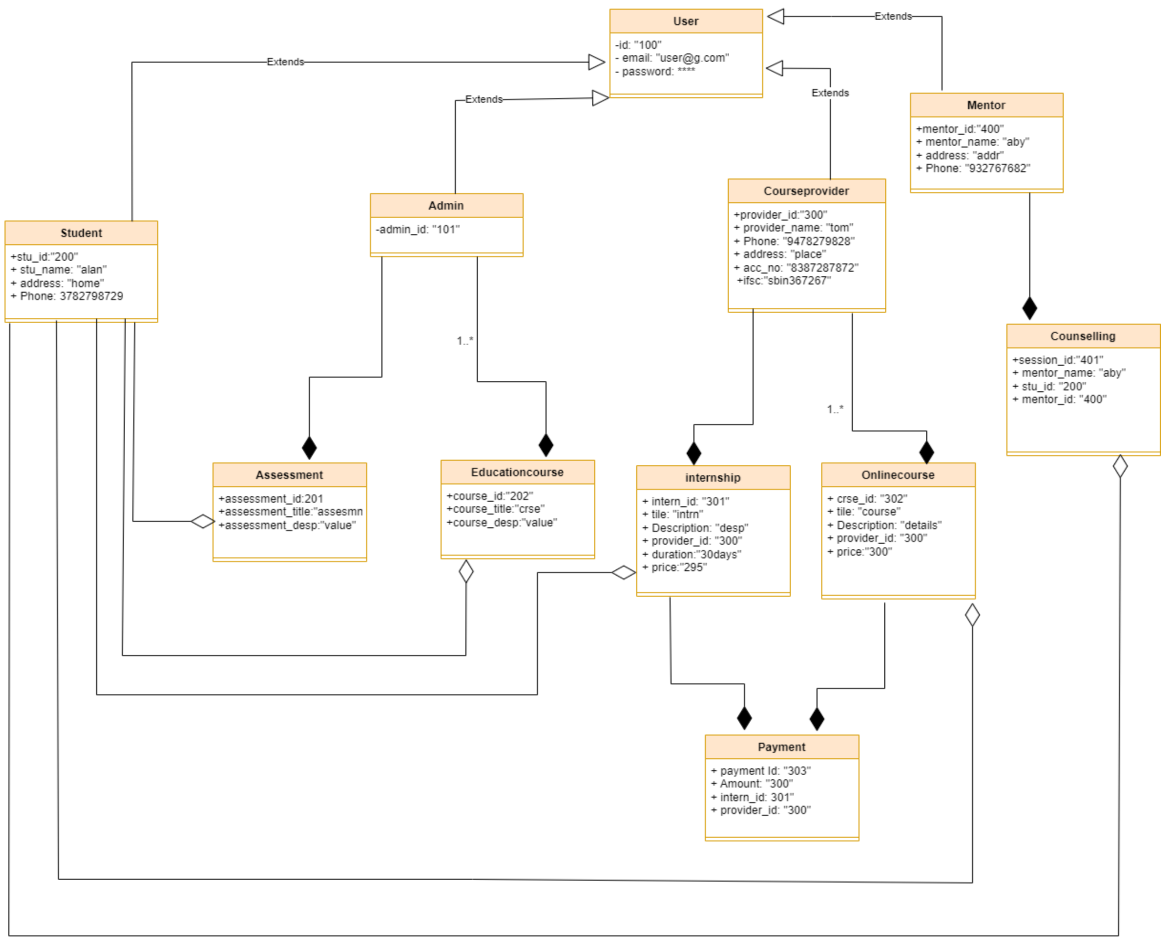


Fig 6: Object Diagram of Careergo

4.2.7 Component Diagram

A Component Diagram, a vital aspect of UML, offers a visual representation of a system's architecture by showcasing the high-level components and their connections. Components, depicted as rectangles, encapsulate modules, classes, or even entire systems. Dependencies between components are displayed through arrows, signifying the reliance of one component on another. Interfaces, represented by a small circle, outline the services a component offers or requires. Connectors link interfaces to denote the required or provided services. Ports, depicted as small squares, serve as connection points between a component and its interfaces. Stereotypes provide additional information about the role or purpose of a component. Deployment nodes indicate the physical location or environment in which components are deployed. Component Diagrams are instrumental in system design, aiding in the organization and visualization of system architecture. They emphasize the modular structure, facilitating ease of development, maintenance, and scalability of complex software systems. Overall, Component Diagrams play a pivotal role in planning and orchestrating the architecture of sophisticated software applications.

Key notations for Component Diagrams:

- **Component:** Represented as a rectangle, it encapsulates a module, class, or system.
- **Dependency Arrow:** Indicates that one component relies on or uses another.
- **Interface:** Depicted as a small circle, it outlines the services a component offers or requires.
- **Provided and Required Interfaces:** Connectors link provided interfaces to required interfaces.
- **Port:** Shown as a small square, it serves as a connection point between a component and its interfaces.
- **Stereotypes:** Additional labels or annotations applied to components to provide more information about their role or purpose.
- **Assembly Connector:** Represents the physical connection between two components.
- **Artifact:** A physical piece of information that is used or produced by a software development process.
- **Deployment Node:** Indicates the physical location or environment in which components are deployed.
- **Manifestation Arrow:** Indicates the implementation of an interface by a component

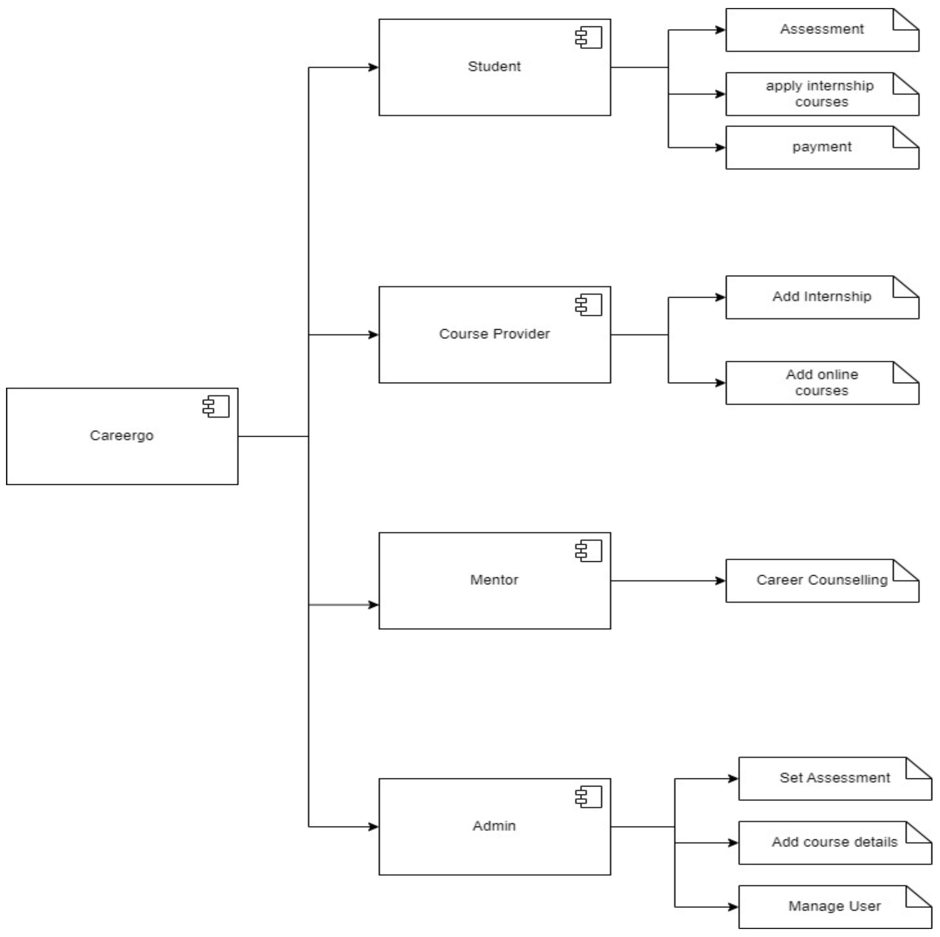


Fig 7: Component Diagram of Careergo

4.2.8 Deployment Diagram

A Deployment Diagram, a crucial facet of UML, provides a visual representation of the physical architecture of a system, showcasing the hardware nodes and software components. Nodes, representing hardware entities like servers or devices, are depicted as rectangles. Artifacts, denoted by rectangles with a folded corner, represent software components or files deployed on nodes. Associations between nodes and artifacts indicate the deployment of software on specific hardware. Dependencies illustrate the reliance of one node on another. Communication paths, shown as dashed lines, represent network connections between nodes. Stereotypes provide additional information about the role or purpose of nodes and artifacts. Deployment Diagrams are instrumental in system planning, aiding in the visualization and organization of hardware and software components. They emphasize the allocation of software modules to specific hardware nodes, ensuring efficient utilization of resources. Overall, Deployment Diagrams play a pivotal role in orchestrating the physical infrastructure of complex software applications.

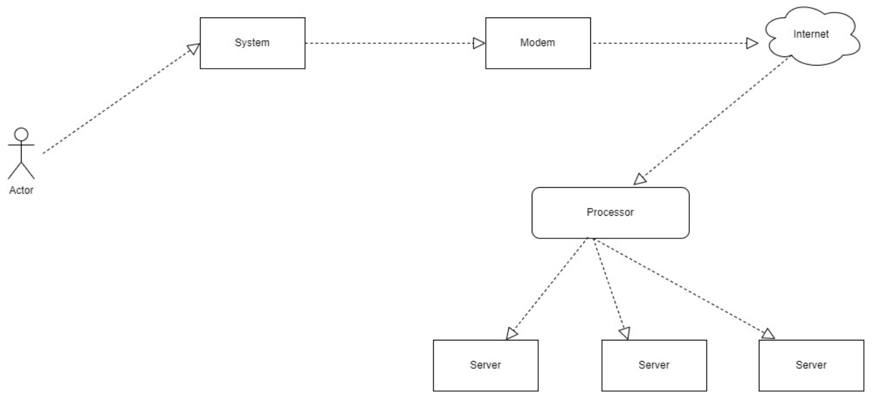
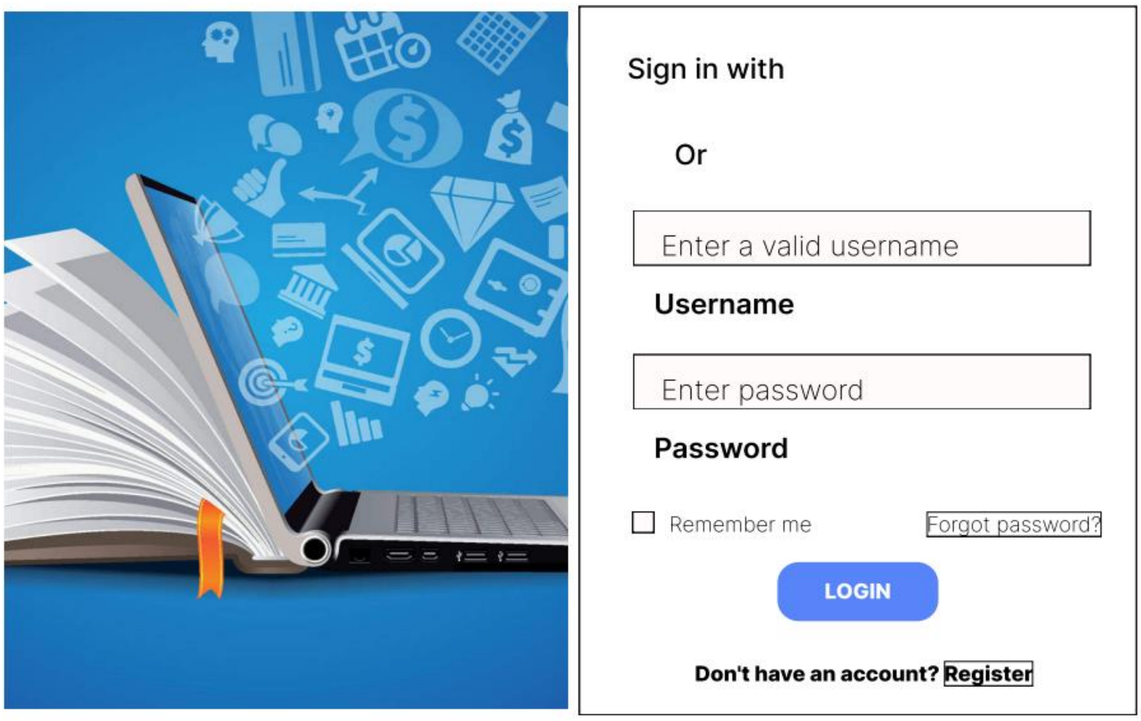


Fig 8: Deployment Diagram of Careergo

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Login



Sign in with

Or

Enter a valid username

Username

Enter password

Password

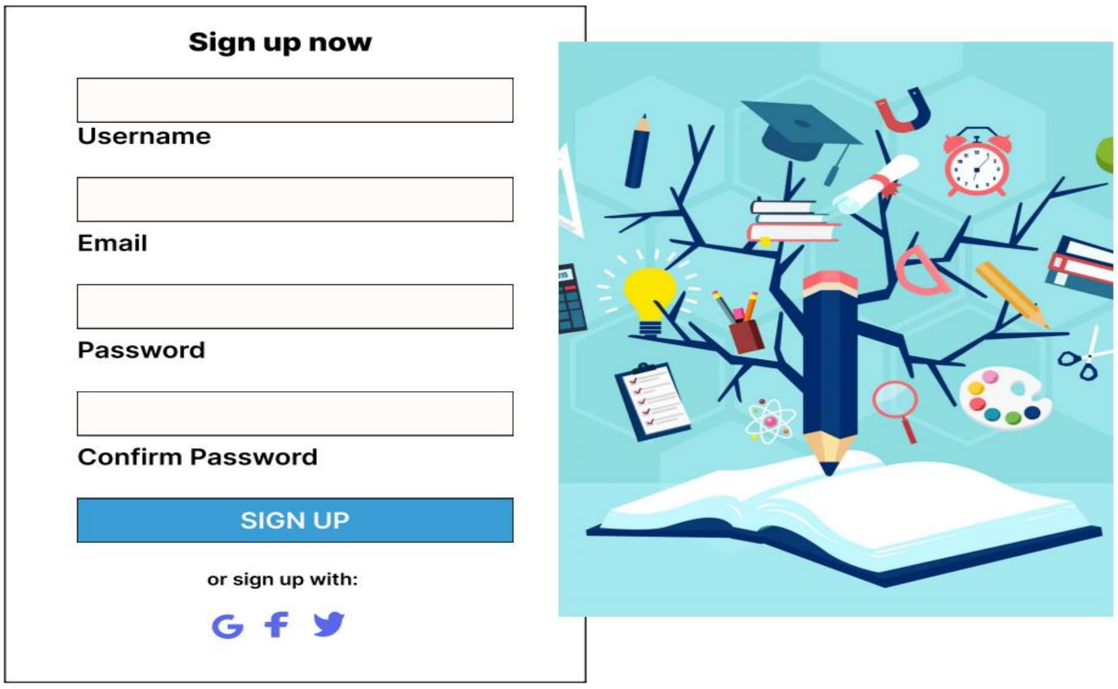
Remember me

Forgot password?

LOGIN

Don't have an account? Register

Form Name: Registration



Sign up now

Username

Email

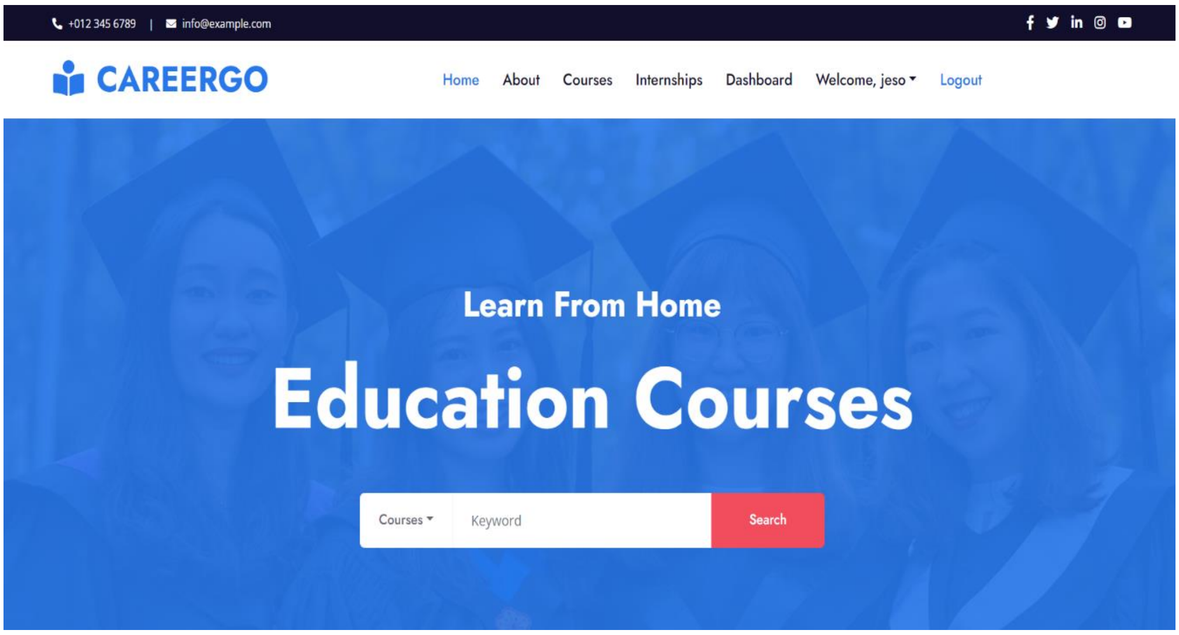
Password

Confirm Password

SIGN UP

or sign up with:

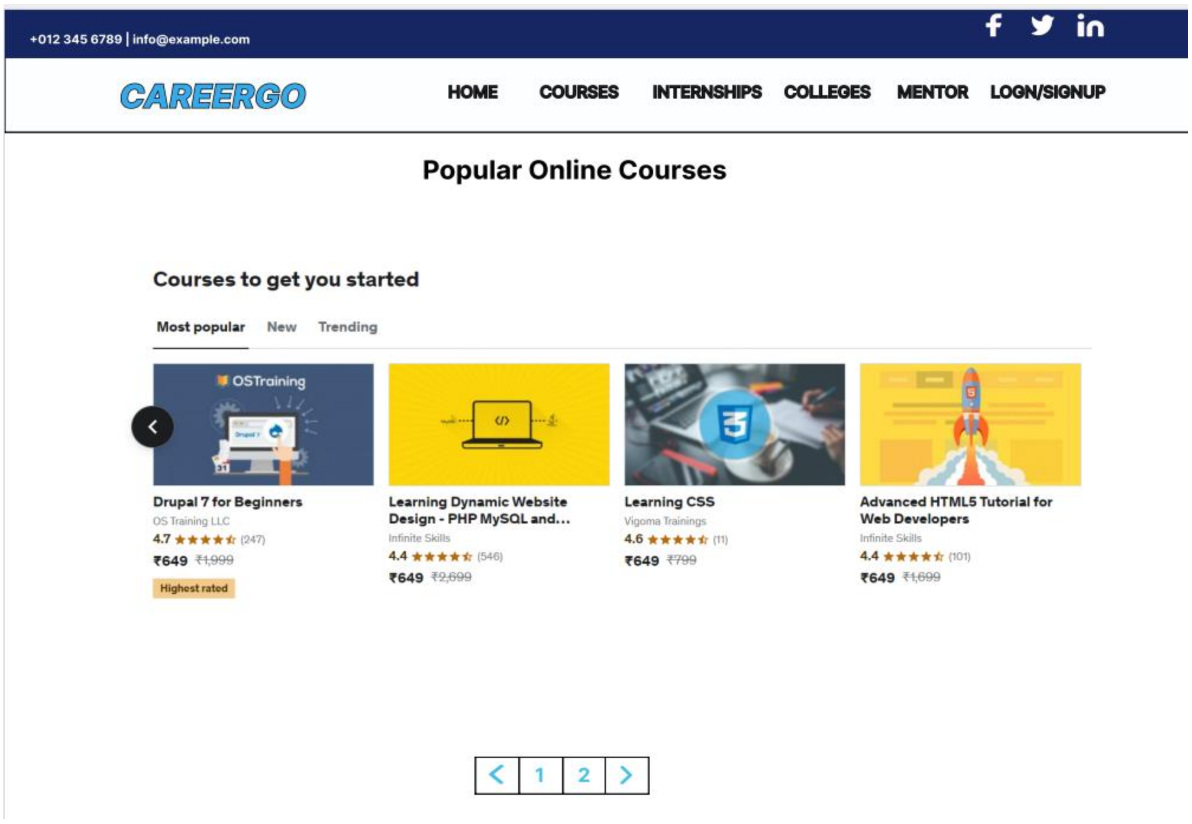
Form Name: Home Page



Learn From Home

Education Courses

Form Name: Course List Page



Popular Online Courses

Courses to get you started

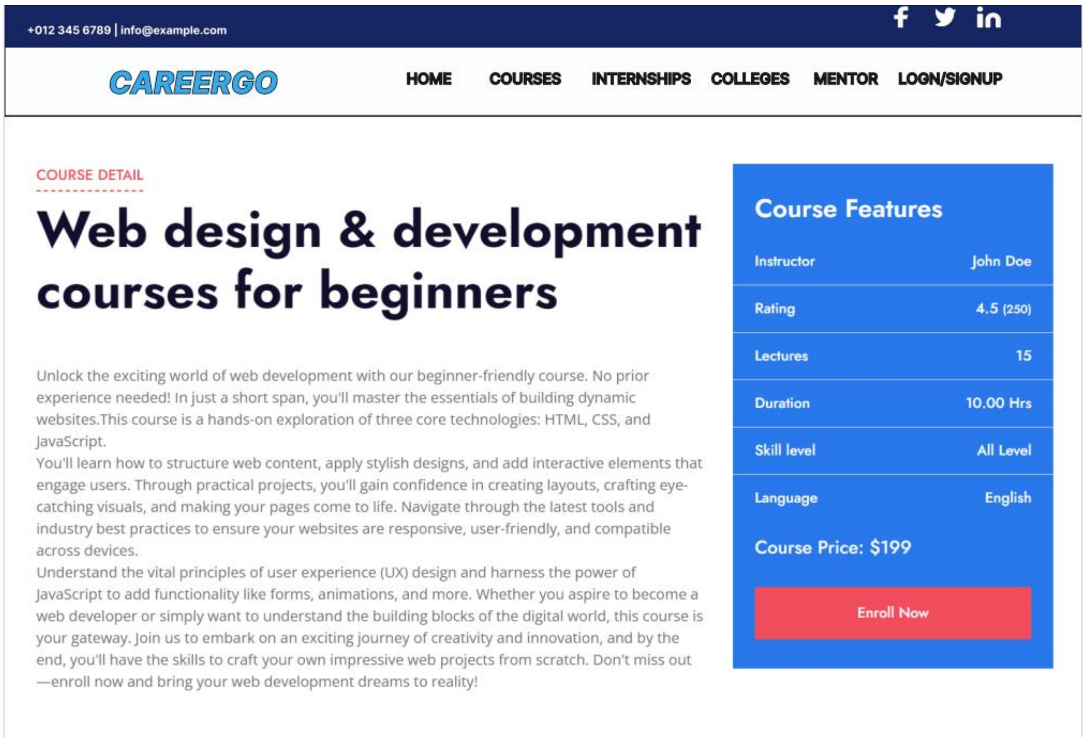
Drupal 7 for Beginners
OS Training LLC
4.7 ★★★★★ (247)
₹649 ~~₹1,999~~
Highest rated

Learning Dynamic Website Design - PHP MySQL and...
Infinite Skills
4.4 ★★★★★ (546)
₹649 ~~₹2,699~~

Learning CSS
Vigoma Trainings
4.6 ★★★★★ (11)
₹649 ~~₹799~~

Advanced HTML5 Tutorial for Web Developers
Infinite Skills
4.4 ★★★★★ (101)
₹649 ~~₹1,699~~

Form Name: Course Detail



COURSE DETAIL

Web design & development courses for beginners

Unlock the exciting world of web development with our beginner-friendly course. No prior experience needed! In just a short span, you'll master the essentials of building dynamic websites.This course is a hands-on exploration of three core technologies: HTML, CSS, and JavaScript.
You'll learn how to structure web content, apply stylish designs, and add interactive elements that engage users. Through practical projects, you'll gain confidence in creating layouts, crafting eye-catching visuals, and making your pages come to life. Navigate through the latest tools and industry best practices to ensure your websites are responsive, user-friendly, and compatible across devices.
Understand the vital principles of user experience (UX) design and harness the power of JavaScript to add functionality like forms, animations, and more. Whether you aspire to become a web developer or simply want to understand the building blocks of the digital world, this course is your gateway. Join us to embark on an exciting journey of creativity and innovation, and by the end, you'll have the skills to craft your own impressive web projects from scratch. Don't miss out—enroll now and bring your web development dreams to reality!

Course Features

Instructor	John Doe
Rating	4.5 (250)
Lectures	15
Duration	10.00 Hrs
Skill level	All Level
Language	English

Course Price: $199

4.4 DATABASE DESIGN

A database is an organized collection of information that's organized to enable easy accessibility, administration, and overhauls. The security of information could be a essential objective of any database. The database design process comprises of two stages. In the first stage, user requirements are gathered to create a database that meets those requirements as clearly as possible. This is known as information-level design and is carried out independently of any DBMS. In the second stage, the design is converted from an information-level design to a specific DBMS design that will be used to construct the system. This stage is known as physical-level design, where the characteristics of the specific DBMS are considered. Alongside system design, there is also database design, which aims to achieve two main goals: data integrity and data independence.

4.4.1 Relational Database Management System (RDBMS)

A relational database management system (RDBMS) is a popular type of database that organizes data into tables to facilitate relationships with other stored data sets. Tables can contain vast amounts of data, ranging from hundreds to millions of rows, each of which are referred to as records. In formal relational model language, a row is called a tuple, a column heading is an attribute, and the table is a relation. A relational database consists of multiple tables, each with its own name. Each row in a table represents a set of related values.

In a relational database, relationships are already established between tables to ensure the integrity of both referential and entity relationships. A domain D is a group of atomic values, and a common way to define a domain is by choosing a data type from which the domain's data values are derived. It is helpful to give the domain a name to make it easier to understand the values it contains. Each value in a relation is atomic and cannot be further divided.

In a relational database, table relationships are established using keys, with primary key and foreign key being the two most important ones. Entity integrity and referential integrity relationships can be established with these keys. Entity integrity ensures that no primary key can have null values, while referential integrity ensures that each distinct foreign key value must have a matching primary key value in the same domain. Additionally, there are other types of keys such as super keys and candidate keys.

4.4.2 Normalization

The simplest possible grouping of data is used to put them together so that future changes can be made with little influence on the data structures. The formal process of normalizing data structures in a way that reduces duplication and fosters integrity. Using the normalization technique, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard form of data modelling. A row in a table is uniquely identified by a key. Primary keys and foreign keys are two different kinds of keys. Primary key is an element, or set of components, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalized.

Normalization is a process in database design that aims to organize data into proper tables and columns, making it easily correlated to the data by the user. This process eliminates data redundancy that can be a burden on computer resources. The main steps involved in normalization
include:

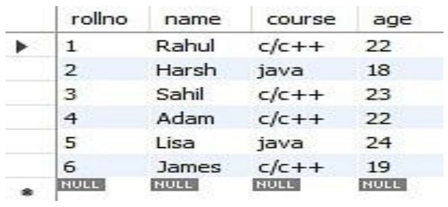
- Normalizing the data
- Choosing appropriate names for tables and columns
- Choosing the correct names for the data

By following these steps, a developer can create a more efficient and organized database that is easier to manage and maintain.

First Normal Form(1NF)

The First Normal Form (1NF) requires that each attribute in a table must contain only atomic or indivisible values. It prohibits the use of nested relations or relations within relations as attribute values within tuples. To satisfy 1NF, data must be moved into separate tables where the data is of similar type in each table, and each table should have a primary key or foreign key as required by the project. This process eliminates repeating groups of data and creates new relations for each non-atomic attribute or nested relation. A relation is considered to be in 1NF only if it satisfies the constraints that contain the primary key only.

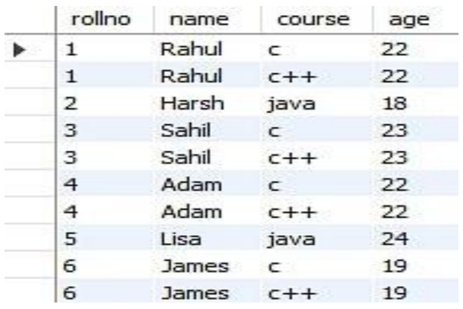
E.g., The table contains information pertaining to students, including their roll number, name, course of study, and age.



rollno	name	course	age
1	Rahul	c/c++	22
2	Harsh	java	18
3	Sahil	c/c++	23
4	Adam	c/c++	22
5	Lisa	java	24
6	James	c/c++	19
NULL	NULL	NULL	NULL

Fig 9 : Table not in 1NF

The table containing the records of students displays a violation of the First Normal Form due to the presence of two distinct values in the course column. To ensure compliance with the First Normal Form, the table has been modified resulting in the formation of a new table.



rollno	name	course	age
1	Rahul	c	22
1	Rahul	c++	22
2	Harsh	java	18
3	Sahil	c	23
3	Sahil	c++	23
4	Adam	c	22
4	Adam	c++	22
5	Lisa	java	24
6	James	c	19
6	James	c++	19

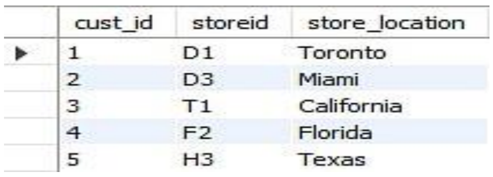
Fig 10 : Table in 1NF

The First Normal Form is applied to achieve atomicity in the system, which ensures that each column has unique values. By following this normalization process, the data is organized into individual atomic values, eliminating any redundancies or repeating groups. As a result, data integrity is maintained, and the system can efficiently handle complex data manipulations and updates.

Second Normal Form(2NF)

To meet requirements of Second Normal Form, a table must satisfy the criteria of First Normal Form as a prerequisite. Furthermore, the table must not exhibit partial dependency, which refers to a scenario where a non-prime attribute is dependent on a proper subset of the candidate key. In other words, the table should have no attributes that are determined by only a portion of the primary key.

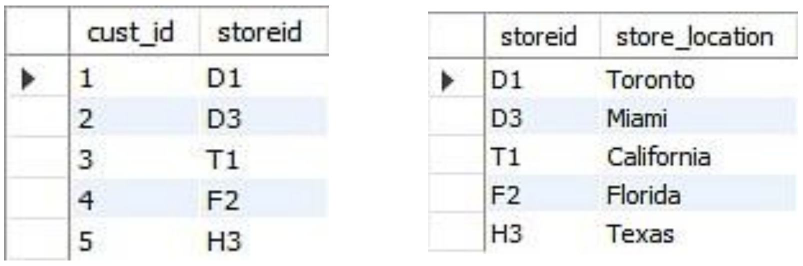
Now understand the Second Normal Form with the help of an example.
Consider the table Location:



cust_id	storeid	store_location
1	D1	Toronto
2	D3	Miami
3	T1	California
4	F2	Florida
5	H3	Texas

Fig 11 : Table not in 2NF

The Location table currently has a composite primary key consisting of cust id and storied, and its non-key attribute is store location. However, the store location attribute is directly determined by the storeid attribute, which is part of the primary key. As a result, this table does not meet the requirements of second normal form. To address this issue and ensure second normal form is met, it is necessary to separate the Location table into two separate tables. This will result in the creation of two distinct tables that accurately represent the relevant data and relationships: one for customer IDs and store IDs, and another for store IDs and their respective locations.:



cust_id	storeid
1	D1
2	D3
3	T1
4	F2
5	H3

storeid	store_location
D1	Toronto
D3	Miami
T1	California
F2	Florida
H3	Texas

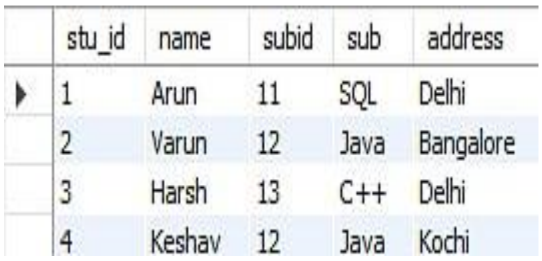
Fig 12 : Table in 2NF

After eliminating the partial functional dependency in the location table, it can be observed that the column storing the location is now entirely dependent on the primary key of the same table. In other words, the location column is fully determined by the primary key, thereby ensuring greater data consistency and integrity in the tabl

Third Normal Form

A table must meet the requirements of Second Normal Form in order to be considered in Third Normal Form, and also fulfill two additional conditions. The second condition states that non-prime attributes should not rely on non-prime characteristics that are not a part of the candidate key within the same table, thus avoiding transitive dependencies. A transitive dependency arises when A → C indirectly, due to A → B and B → C, where B is not functionally dependent on A. The main objective of achieving Third Normal Form is to reduce data redundancy and guarantee data integrity.

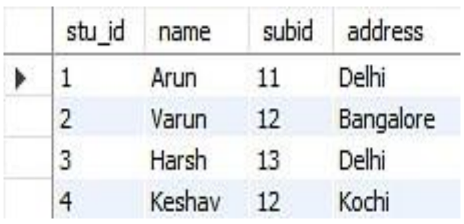
For instance, let's consider a student table that includes columns like student ID, student name, subject ID, subject name, and address of the student. To comply with the requirements for Third Normal Form, this table must first meet the standards of Second Normal Form and then ensure that there are no transitive dependencies between non-prime attributes.



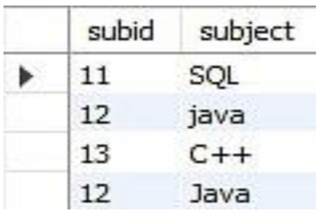
stu_id	name	subid	sub	address
1	Arun	11	SQL	Delhi
2	Varun	12	Java	Bangalore
3	Harsh	13	C++	Delhi
4	Keshav	12	Java	Kochi

Fig 13 : Table not in 3NF

Now to change the table to the third normal form, you need to divide the table as shown below: Based on the given student table, it can be observed that the stu_id attribute determines the sub_id attribute, and the sub_id attribute determines the subject (sub). This implies that there is a transitive functional dependency between stu_id and sub. As a result, the table does not satisfy the criteria for the third normal form. To adhere to the third normal form, the table must be divided into separate tables where each table represents a unique entity or relationship. In this case, the table can be divided into two tables: one for the student-subject relationship (stu_id and sub_id), and another for the subject information (sub_id and sub):



stu_id	name	subid	address
1	Arun	11	Delhi
2	Varun	12	Bangalore
3	Harsh	13	Delhi
4	Keshav	12	Kochi



subid	subject
11	SQL
12	java
13	C++
12	Java

Fig 14: Table in 3NF

The two tables illustrate that the non-key attributes are completely determined by and reliant on the primary key. In the first table, the columns for name, sub_id, and addresses are all exclusively linked to the stu_id. Likewise, the second table demonstrates that the sub column is entirely dependent on the sub_id.

4.4.3 Sanitization

Data sanitization is the process of removing any illegal characters or values from data. In web applications, sanitizing user input is a common task to prevent security vulnerabilities. PHP provides a built-in filter extension that can be used to sanitize and validate various types of external input such as email addresses, URLs, IP addresses, and more. These filters are designed to make data sanitization easier and faster. For example, the PHP filter extension has a function that can remove all characters except letters, digits, and certain special characters (!#$%&'*+- =?_`{|}~@.[]), as specified by a flag.

Web applications often receive external input from various sources, including user input from forms, cookies, web services data, server variables, and database query results. It is important to sanitize all external input to ensure that it is safe and does not contain any malicious code or values.

4.4.4 Indexing

An index is a database structure that enhances the speed of table operations. Indexes can be created on one or more columns to facilitate quick lookups and efficient ordering of records. When creating an index, it's important to consider which columns will be used in SQL queries and to create one or more indexes on those columns. In practice, indexes are a type of table that store a primary key or index field and a pointer to each record in the actual table. Indexes are invisible to users and are only used by the database search engine to quickly locate records. The CREATE INDEX statement is used to create indexes in tables.

When tables have indexes, the INSERT and UPDATE statements take longer because the database needs to insert or update the index values as well. However, the SELECT statements become faster on those tables because the index allows the database to locate records more quickly.

4.5 TABLE DESIGN

1. tbl_login

Primary key: **user_id**

No:	**Field name**	**Datatype**	**Key Constraints**	**Description**
1	user _id	int(11)	Primary Key	Unique User Identifier
2	username	varchar(100)	Not Null	User's Username
3	first_name	varchar(100)	Not Null	User's First Name
4	last_name	varchar(100)	Not Null	User's Last Name
5	email	varchar(100)	Not Null	User's Email Address
6	phone_no	varchar(12)	Not Null	User's Phone Number
7	role	PositiveSmallIntegerField	Not Null	User Type (e.g., Student, Admin,Courseprovider)
9	login_ status	Boolean	Not null	User status
10	login_ last_login	Datetime	Not null	Last Logined

2.tbl_register

Primary key: **reg_id**

Foreign key: **user_id** references table **tbl_login**

No:	**Field name**	**Datatype**	**Key Constraints**	**Description**
1	reg_id	int(11)	Primary Key	User Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	profile_pic	varchar(255)	-	Profile Picture

4	country	varchar(15)	-	User's Country
5	state	varchar(15)	-	User's State
6	city	varchar(15)	-	User's City
7	pin_code	varchar(6)	-	User's Pin Code

3.tbl_assessment

Primary key: **assessment_id**

Foreign key: **user_id** references table **tbl_login**

No:	**Field name**	**Datatype**	**Key Constraints**	**Description**
1	assessment_id	int(11)	Primary Key	Assessment Identifier
2	logical_reasoning_score	int(11)	-	Logical reasoning score
3	communication_skill_score	int(11)	-	communication_skills_score
4	quantitative_aptitude_score	int(11)	-	quantitative_aptitude_score
5	analytical_skills_score	int(11)	-	analytical_skills_score
6	total_score	int(11)	-	total_score
7	stream	varchar(50)	Not Null	Plus two Stream
8	plus_two_cgpa	Decimal(4)	-	plus_two_cgpa
9	assessment_status	boolean	-	status
10	User_id	int(11)	Foreign Key	References table tbl_login

4. tbl_oncourse

Primary key: **course_id**

Foreign key: **user_id** references table **tbl_login**

No:	Field name	Datatype	Key Constraints	Description
1	course_id	int(11)	Primary Key	Unique Course Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	thumbnail	File	Not Null	course Images
4	course_title	Varchar(100)	Not Null	Tiltle of courses
5	description	Varchar(500)	Not Null	Description of course
6	duration	Varchar(100)	Not Null	Course Duration
7	instructor	Varchar(100)	Not Null	Instructor Name
8	price	Int(11)	-	Course Price
9	category	Varchar(100)	Not Null	Course category
10	is_free	boolean	-	Course mode

5. tbl_video

Primary key: **video _id**

Foreign key: **user_id references table tbl_login**

Foreign key: **course_id references table tbl_oncourse**

No:	Field name	Datatype	Key Constraints	Description
1	video_id	int(11)	Primary Key	Unique Video Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	course_id	int(11)	Foreign Key	References table tbl_oncourse
4	video_file	file	Not Null	Course video
5	section_name	varchar(100)	Not Null	Section name

6. tbl_uservideo

Primary key: **uservideo_id**

Foreign key: **user_id references table tbl_login**

Foreign key: **course_id references table tbl_oncourse**

Foreign key: **video_id references table tbl_video**

No:	Field name	Datatype	Key Constraints	Description
1	uservideo_id	int(11)	Primary Key	Unique Video Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	course_id	int(11)	Foreign Key	References table tbl_oncourse
4	Video_id	int(11)	Foreign Key	References table tbl_video
5	checked	boolean	Not Null	Video status

7.tbl_internship

Primary key: **internship_id**

Foreign key: **user_id references table tbl_login**

No:	Field name	Datatype	Key Constraints	Description
1	internship_id	int(11)	Primary Key	Unique Internship Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	internship_title	Varchar(255)	Not Null	Internship Title
4	description	Text	Not Null	Description of Internship
5	duration	int(11)	Not Null	Internship Duration
6	instructor	Varchar(100)	Not Null	Instructor Name
7	price	Decimal(7,2)	Not Null	Internship Price
8	category	Varchar(100)	Not Null	Internship category
9	start_date	datetime	Not Null	start_date
10	end_date	datetime	Not Null	end_date
11	positions	int(11)	Not Null	No.of positions available
12	internship_type	Varchar(20)	Not Null	Type of Internship
13	internship_mode	Varchar(20)	Not Null	Mode of Internship
14	application_ deadline	datetime	Not Null	application_dea dline
15	company_name	Varchar(255)	-	Name of company
16	company_website	URL(200)	-	Company website link

17	thumbnail	ImageField	Not Null	Thumbnail image

8. tbl_certificate

Primary key: **certificate_id**

Foreign key: **user_id references table tbl_login**

Foreign key: **course_id references table tbl_oncourse**

No:	**Field name**	**Datatype**	**Key Constraints**	**Description**
1	certificate_id	int(11)	Primary Key	Unique Certificate Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	course_id	int(11)	Foreign Key	References table tbl_oncourse
4	issued_date	datetime	Not Null	Issued date

9. tbl_payment

Primary key: **payment_id**

Foreign key: **user_id references table tbl_login**

Foreign key: **course_id references table tbl_oncourse**

Foreign key: **internship_id references table tbl_internship**

No:	**Field name**	**Datatype**	**Key Constraints**	**Description**
1	payment_id	int(11)	Primary Key	Unique Payment Identifier
2	user_id	int(11)	Foreign Key	References table tbl_login
3	course_id	int(11)	Foreign Key	References table tbl_oncourse

4	Internship_id	int(11)	Foreign Key	References table tbl_internship
5	razorpay_order_id	varchar(255)	-	Razorpay Order ID
6	payment_id	varchar(255)	-	Razorpay Payment ID
7	amount	decimal(8,2)	-	Payment Amount
8	currency	varchar(3)	-	Currency Code (e.g., "INR")
9	payment_date	datetime	-	Timestamp of the Payment
10	payment_status	varchar(20)	-	Payment Status

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

System Testing is a crucial phase in the software development life cycle, where the entire system is evaluated against specified requirements and functionalities. It is a comprehensive and structured approach to validate that the software meets its intended objectives. This phase involves testing the integrated system as a whole to ensure that all components work together seamlessly. System Testing verifies the system's compliance with both functional and non-functional requirements, including performance, security, and usability. It is conducted in an environment that closely simulates the production environment, providing a real-world scenario for testing. The primary goal of System Testing is to identify and rectify any discrepancies or defects before the software is deployed to end-users. Through rigorous testing processes and thorough documentation, System Testing helps in delivering a reliable and high-quality software product.

Testing is the systematic process of running a program to uncover potential errors or flaws. An effective test case possesses a high likelihood of revealing previously unnoticed issues. A test is considered successful when it reveals a previously unidentified error. If a test functions as intended and aligns with its objectives, it can detect flaws in the software. The test demonstrates that the computer program is operating in accordance with its intended functionality and performing optimally. There are three primary approaches to assessing a computer program: evaluating for accuracy, assessing implementation efficiency, and analyzing computational complexity.

5.2 TEST PLAN

A test plan is a thorough document that delineates the strategy, scope, objectives, resources, schedule, and expected outcomes for a specific testing endeavor. It functions as a guiding framework for carrying out testing activities, guaranteeing that every facet of the testing process is methodically organized and executed. Additionally, the test plan establishes the roles and responsibilities of team members, outlines the required testing environment, and sets forth the criteria for the successful completion of testing activities. This document plays a pivotal role in ensuring that the testing phase is conducted in a structured and effective manner, ultimately contributing to the overall success of the project..The levels of testing include:

- Integration Testing
- Unit testing

- Validation Testing or System Testing
- Output Testing or User Acceptance Testing
- Automation Testing
- Widget Testing

5.2.1 Integration Testing

Integration Testing stands as a pivotal phase in the software testing process, dedicated to scrutinizing the interactions and interfaces among diverse modules or components within a software system. Its primary objective is to ascertain that individual units of code seamlessly converge to create a unified and functional system. In stark contrast to unit testing, which assesses individual units in isolation, integration testing delves into the interplay between these units, with a keen eye for any disparities, communication glitches, or integration hurdles.

By subjecting the integrated components to rigorous testing, development teams aim to affirm that these elements function cohesively, addressing any potential issues before deployment. This systematic evaluation is instrumental in ensuring that the software operates as an integrated whole, free from any unforeseen conflicts or errors that may arise from the convergence of individual modules.

5.2.2 Unit Testing

Unit Testing is not only a meticulous examination of discrete units or components within a software system but also an indispensable quality assurance measure. This phase serves as a crucial foundation for the entire software testing process, where the focus lies on isolating and scrutinizing individual units of code. The objective remains unwavering: to verify that each unit performs its designated function accurately, yielding precise outputs for predefined inputs.

Moreover, Unit Testing operates independently, detached from other components, and any external dependencies are either emulated or replaced by "mock" objects, ensuring controlled evaluation. This meticulous process establishes a robust foundation for the software, confirming that each unit functions reliably and adheres meticulously to its predefined behavior.

The significance of Unit Testing cannot be overstated, as it acts as a vanguard against potential discrepancies or errors early in the development cycle. This proactive approach not only fortifies the integrity and reliability of the software but also lays the groundwork for subsequent testing phases, thereby fostering a robust and dependable software solution.

This meticulous process ensures that each unit functions reliably and adheres precisely to its defined behavior. By subjecting individual code units to rigorous scrutiny, any discrepancies or errors are identified and rectified early in the development cycle, bolstering the overall integrity and reliability of the software.

5.2.3 Validation Testing or System Testing

Validation Testing places the end-users at the forefront of evaluation, ensuring that the software aligns precisely with their anticipated needs and expectations. This phase stands distinct from other testing methodologies, as its primary objective is to authenticate that the software, in its final form, serves its intended purpose seamlessly within the real-world scenarios it was designed for.

As a culmination of the testing process, Validation Testing carries the responsibility of confirming that the software not only meets the defined technical specifications but also delivers genuine value to its users. It does so by scrutinizing the software against the backdrop of actual usage, thereby fortifying its readiness for deployment.

Moreover, in Validation Testing, user stories and acceptance criteria form the cornerstone of assessment. Stakeholders' expectations are meticulously validated, ensuring that every specified requirement is met. Additionally, beta testing, a common practice in this phase, involves a select group of end-users testing the software in a live environment, providing invaluable feedback that can inform potential refinements.

5.2.4 Output Testing or User Acceptance Testing

Output Testing, also known as Results Validation, is a critical phase in the software testing process. Its primary focus is to verify the correctness and accuracy of the output generated by a software application. The goal is to ensure that the system produces the expected results for a given set of inputs and conditions.

Key aspects of Output Testing include:

- **Comparison with Expected Results:** This phase involves comparing the actual output of the software with the expected or predefined results.
- **Test Case Design:** Test cases are designed to cover various scenarios and conditions to thoroughly evaluate the accuracy of the output.
- **Validation Criteria**: The criteria for validating the output are typically defined during the requirements and design phase of the software development process.
- **Regression Testing:** Output Testing often includes regression testing to ensure that changes or updates to the software do not affect the correctness of the output.

- **Data Integrity:** It verifies that data is processed and displayed correctly, without any corruption or loss.
- **Precision and Completeness:** Output Testing assesses not only the precision of the results but also their completeness in addressing the requirements.
- **Error Handling:** It evaluates how the system handles errors or exceptions and ensures that appropriate error messages are displayed.

5.2.5 Automation Testing

Automation Testing stands as a cornerstone in the software testing process, harnessing the power of automated tools and scripts to meticulously execute test cases. In stark contrast to manual testing, which hinges on human intervention, automation testing brings forth a streamlined approach, employing software to conduct repetitive, intricate, and time-consuming tests. This methodology not only heightens operational efficiency but also significantly diminishes the likelihood of human error, ensuring precise and reliable results. Moreover, it empowers thorough testing across a diverse array of scenarios and configurations, from browser compatibility to load and performance assessments.

By automating the testing process, organizations can realize a myriad of benefits. It enables the seamless execution of regression tests, providing confidence that existing functionalities remain intact after each round of enhancements or modifications. Furthermore, automation facilitates the concurrent execution of multiple tests, thereby expediting the overall testing cycle. This approach is particularly invaluable in environments characterized by rapid development and frequent software updates, such as Agile and DevOps setups.

5.2.6 Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors.

In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD).

It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals.

Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

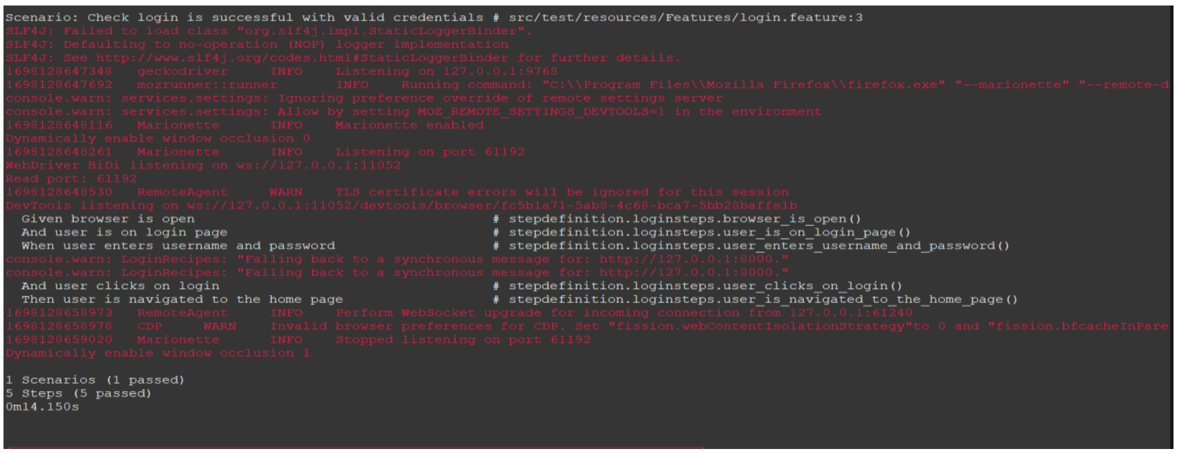
Example:

Test Case 1: User-Login

Code

```
package stepdefinition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.After;
import io.cucumber.java.Before;
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 loginsteps {
    WebDriver driver;
    @Before
    public void setup() {
        System.setProperty("webdriver.gecko.marionette",
"D:\\eclipse\\src\\test\\resources\\drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }
    @After
    public void teardown() {
        if (driver != null) {
            driver.quit();
        }
    }
    @Given("browser is open")
    public void browser_is_open() {
        // No need to do anything here since the setup is handled in @Before
    }
    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {
        driver.get("http://127.0.0.1:8000/login");
        Thread.sleep(2000);
    }
    @When("user enters username and password")
    public void user_enters_username_and_password() {
        WebElement usernameInput = driver.findElement(By.id("usrnme"));
        WebElement passwordInput = driver.findElement(By.id("pswd"));
        usernameInput.sendKeys("jeso");
        passwordInput.sendKeys("Jeso@123");
    }
    @And("user clicks on login")
    public void user_clicks_on_login() {
        WebElement loginButton = driver.findElement(By.id("login-button"));
        loginButton.click();
    }
    @Then("user is navigated to the home page")
    public void user_is_navigated_to_the_home_page() throws Exception {
        // You may want to add some validation logic here to check if you are on the home page.
        Thread.sleep(3000);
    }
}
```

Screenshot



```
Scenario: Check login is successful with valid credentials # src/test/resources/Features/login.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.
1698128647348   geckodriver     INFO    Listening on 127.0.0.1:9768
1698128647692   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
1698128648116   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698128648261   Marionette      INFO    Listening on port 61192
WebDriver BiDi listening on ws://127.0.0.1:11052
Read port: 61192
1698128648530   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:11052/devtools/browser/fc5b1a71-5ab8-4c68-bca7-5bb28baffe1b
  Given browser is open                                    # stepdefinition.loginsteps.browser_is_open()
  And user is on login page                                # stepdefinition.loginsteps.user_is_on_login_page()
  When user enters username and password                   # stepdefinition.loginsteps.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."
  And user clicks on login                                 # stepdefinition.loginsteps.user_clicks_on_login()
  Then user is navigated to the home page                  # stepdefinition.loginsteps.user_is_navigated_to_the_home_page()
1698128658973   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:61240
1698128658978   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInPare
1698128659020   Marionette      INFO    Stopped listening on port 61192
Dynamically enable window occlusion 1

1 Scenarios (1 passed)
5 Steps (5 passed)
0m14.150s
```

Test Report

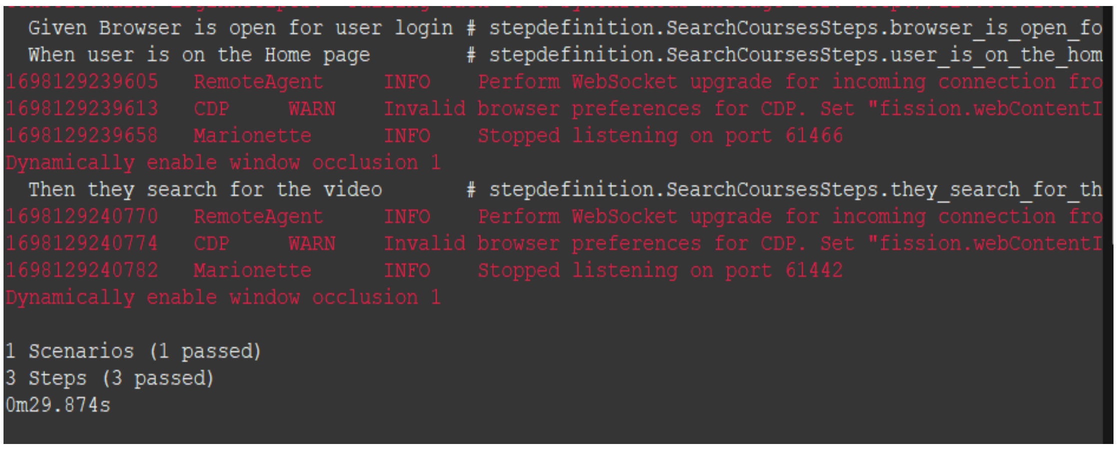
Test Case 1					
Project Name: CareerGo: Career Guidance Website					
Login Test Case					
Test Case ID: Test_1			**Test Designed By: Febin Vinod**		
Test Priority(Low/Medium/High):High			**Test Designed Date:** 10-10-2023		
Module Name: Login Screen			**Test Executed By :** Ms.Jetty Benjamin		
Test Title : User Login			**Test Execution Date:** 12-10-2023		
Description: Verify login with valid email and password					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Login page displayed	Login page displayed	Pass
2	Provide Valid username	username: jeso	User should be able to Login	User Logged in and navigated to home page	Pass
3	Provide Valid Password	Password: Jeso@123			
4	Click on Login button				
5	Logout		Logout from page	Logout from page	Pass
Post-Condition: User is validated with database and successfully login into account.					

Test Case 2: Courses Search

Code

```
package stepdefinition;

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 SearchCoursesSteps{
        WebDriver driver;
        @Given("Browser is open for user login")
        public void browser_is_open_for_user_login() throws Throwable {
            // Write code here that turns the phrase above into concrete actions
                     System.setProperty("webdriver.gecko.marionette",
"D:\\eclipse\\src\\test\\resources\\drivers\\geckodriver.exe");
                driver = new FirefoxDriver();
                driver.manage().window().maximize();
                    Thread.sleep(2000);
                    driver.navigate().to("http://127.0.0.1:8000/login");
                            Thread.sleep(2000);
                            driver.findElement(By.id("usrnme")).sendKeys("jeso");
                            driver.findElement(By.id("pswd")).sendKeys("Jeso@123");
                            driver.findElement(By.id("login-button")).click();
                            Thread.sleep(2000);
            //throw new io.cucumber.java.PendingException();
        }
        @When("user is on the Home page")
        public void user_is_on_the_home_page() throws Throwable {
            // Write code here that turns the phrase above into concrete actions
                    driver.findElement(By.id("course")).click();
            //throw new io.cucumber.java.PendingException();
        }
        @Then("they search for the video")
        public void they_search_for_the_video() throws Throwable {
           // Write code here that turns the phrase above into concrete actions
                driver.findElement(By.id("search-input")).clear();
                driver.findElement(By.id("search-input")).sendKeys("python");
                driver.findElement(By.id("search")).click();
                Thread.sleep(5000);
                driver.close();
           //throw new io.cucumber.java.PendingException();
        }
}
```

Screenshot



```
  Given Browser is open for user login # stepdefinition.SearchCoursesSteps.browser_is_open_fo
  When user is on the Home page        # stepdefinition.SearchCoursesSteps.user_is_on_the_hom
1698129239605   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection fro
1698129239613   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentI
1698129239658   Marionette      INFO    Stopped listening on port 61466
Dynamically enable window occlusion 1
  Then they search for the video       # stepdefinition.SearchCoursesSteps.they_search_for_th
1698129240770   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection fro
1698129240774   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentI
1698129240782   Marionette      INFO    Stopped listening on port 61442
Dynamically enable window occlusion 1

1 Scenarios (1 passed)
3 Steps (3 passed)
0m29.874s
```

Test report

Test Case 2

Project Name: CareerGo: Career Guidance Website					
Search Test Case					
Test Case ID: Test_2			**Test Designed By:** Febin Vinod		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 14-10-2023		
Module Name: Search Courses			**Test Executed By :** Ms.Jetty Benjamin		
Test Title : Search Courses			**Test Execution Date:** 16-10-2023		
Description: Searching Courses by user					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Login page displayed	Login page displayed	Pass
2	Provide Valid username	username: jeso	User should be able to Login	User Logged in and navigated to home page	Pass
3	Provide Valid Password	Password: Jeso@123			
4	Click on Login button				
5	Click on Courses option		Courses list page should be displayed	Navigated to Courses list page	Pass
6	Provide Input on the search field and click search button	Search: python	Search result should be shown	Result is shown to the user	Pass
Post-Condition: User can see the search results					

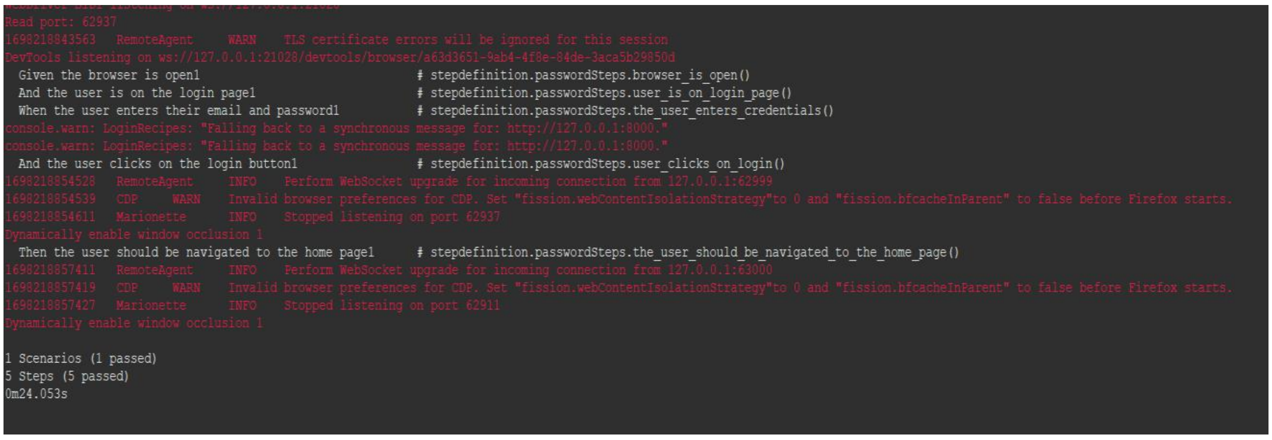
Test Case 3: Change User Password

Code

```
package stepdefinition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import java.time.Duration;
public class passwordSteps {
    WebDriver driver;
    @Given("the browser is open1")
    public void browser_is_open() {
        System.setProperty("webdriver.gecko.marionette",
"D:\\eclipse\\src\\test\\resources\\drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }
    @And("the user is on the login page1")
    public void user_is_on_login_page() throws Exception {
        driver.navigate().to("http://127.0.0.1:8000/login");
        Thread.sleep(2000);
    }
    @When("the user enters their email and password1")
    public void the_user_enters_credentials() {
        driver.findElement(By.id("usrnme")).sendKeys("jeso");
        driver.findElement(By.id("pswd")).sendKeys("Jeso@1234");
    }
    @And("the user clicks on the login button1")
    public void user_clicks_on_login() {
        driver.findElement(By.id("login-button")).click();
    }
    @Then("the user should be navigated to the home page1")
    public void the_user_should_be_navigated_to_the_home_page() {
            driver.findElement(By.id("ProfileDropdown")).click();
            driver.navigate().to("http://127.0.0.1:8000/profile");
//        driver.findElement(By.id("profile")).click();
```

```
        // Scroll to the "twon" element
        WebElement cityField = driver.findElement(By.id("new_password"));
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", cityField);
        // Wait for the element to be visible and clickable
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10)); // Corrected line
        wait.until(ExpectedConditions.visibilityOf(cityField));
        wait.until(ExpectedConditions.elementToBeClickable(cityField));
        // Clear and enter new data
        cityField.sendKeys("Jeso@123");
        // Other form interactions...
        WebElement phone = driver.findElement(By.id("confirm_password"));
        phone.clear();
        phone.sendKeys("Jeso@123");
        // Submit the form to save changes
        WebElement submitButton = driver.findElement(By.id("submit"));
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", submitButton);
        // Close the browser
        driver.quit();
    }
}
```

Screenshot



```
Read port: 62937
1698218843563   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:21028/devtools/browser/a63d3651-9ab4-4f8e-84de-3aca5b29850d
  Given the browser is open1                           # stepdefinition.passwordSteps.browser_is_open()
  And the user is on the login page1                   # stepdefinition.passwordSteps.user_is_on_login_page()
  When the user enters their email and password1       # stepdefinition.passwordSteps.the_user_enters_credentials()
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."
  And the user clicks on the login button1             # stepdefinition.passwordSteps.user_clicks_on_login()
1698218854528   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:62999
1698218854539   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInParent" to false before Firefox starts.
1698218854611   Marionette      INFO    Stopped listening on port 62937
Dynamically enable window occlusion 1
  Then the user should be navigated to the home page1      # stepdefinition.passwordSteps.the_user_should_be_navigated_to_the_home_page()
1698218857411   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:63000
1698218857419   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInParent" to false before Firefox starts.
1698218857427   Marionette      INFO    Stopped listening on port 62911
Dynamically enable window occlusion 1

1 Scenarios (1 passed)
5 Steps (5 passed)
0m24.053s
```

Test report

Test Case 3

Project Name: CareerGo: Career Guidance Website

Change Password Test Case

Test Case ID: Test_3	**Test Designed By:** Febin Vinod
Test Priority(Low/Medium/High): High	**Test Designed Date:** 18-10-2023
Module Name: Change User Password	**Test Executed By :** Ms.Jetty Benjamin
Test Title : Change User Password	**Test Execution Date:** 20-10-2023
Description: Updating new user password	

Pre-Condition : User has valid username and password

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Login page displayed	Login page displayed	Pass
2	Provide Valid username	username: jeso	User should be able to Login	User Logged in and navigated to dashboard	Pass
3	Provide Valid Password	Password: Jeso@123			
4	Click on Login button				
5	From navbar click User Profile option		Profile dashboard should be shown	Profile dashboard is shown to the user	Pass
6	In the form enter the data to be updated and click on submit to apply changes	Password: Jeso@1234 Confirm Password: Jeso@1234	Password should be updated	Password is updated	Pass

Post-Condition: User successfully updated password

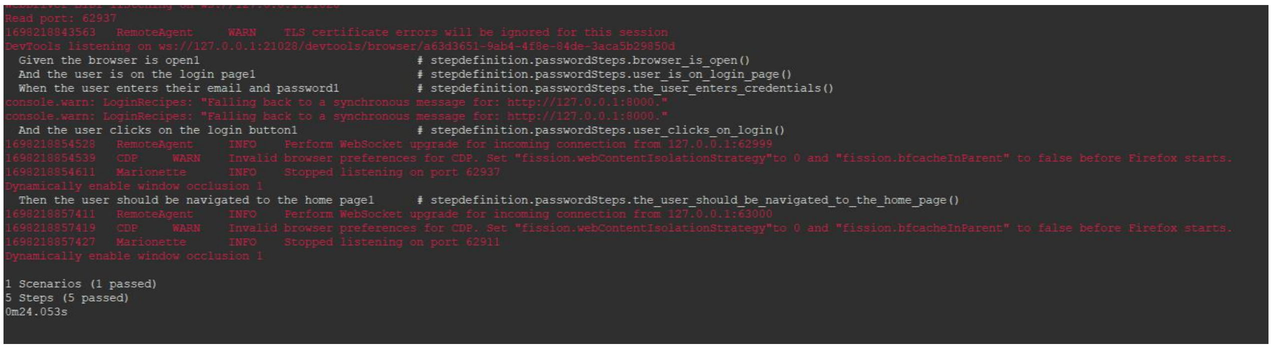
Test Case 4: User Profile Update

Code

```
package stepdefinition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import java.time.Duration;
public class updateSteps {
   WebDriver driver;
   @Given("the browser is open")
   public void browser_is_open() {
      System.setProperty("webdriver.gecko.marionette",
"D:\\eclipse\\src\\test\\resources\\drivers\\geckodriver.exe");
      driver = new FirefoxDriver();
      driver.manage().window().maximize();  }
   @And("the user is on the login page")
   public void user_is_on_login_page() throws Exception {
      driver.navigate().to("http://127.0.0.1:8000/login");
      Thread.sleep(2000); }
   @When("the user enters their email and password")
   public void the_user_enters_credentials() {
      driver.findElement(By.id("usrnme")).sendKeys("jeso");
      driver.findElement(By.id("pswd")).sendKeys("Jeso@123");}
   @And("the user clicks on the login button")
   public void user_clicks_on_login() {
      driver.findElement(By.id("login-button")).click(); }
   @Then("the user should be navigated to the home page")
   public void the_user_should_be_navigated_to_the_home_page() {
          driver.findElement(By.id("ProfileDropdown")).click();
```

```
            driver.navigate().to("http://127.0.0.1:8000/profile");
        driver.findElement(By.id("countryDropdown")).click();
        driver.findElement(By.id("India")).click();
        WebElement cityField = driver.findElement(By.id("twon"));
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", cityField);
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10)); // Corrected line
        wait.until(ExpectedConditions.visibilityOf(cityField));
        wait.until(ExpectedConditions.elementToBeClickable(cityField));
        cityField.clear();
        cityField.sendKeys("Kottayam");
        WebElement phone = driver.findElement(By.id("phone_no"));
        WebElement pincodeField = driver.findElement(By.id("pincode"));
        phone.clear();
        pincodeField.clear();
        phone.sendKeys("9446732818");
        pincodeField.sendKeys("686514");
        WebElement submitButton = driver.findElement(By.id("submit"));
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", submitButton);
        driver.quit();
    }
}
```

Screenshot



```
Read port: 62937
1698218843563   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:21028/devtools/browser/a63d3651-9ab4-4f8e-84de-3aca5b29850d
  Given the browser is open1                              # stepdefinition.passwordSteps.browser_is_open()
  And the user is on the login page1                      # stepdefinition.passwordSteps.user_is_on_login_page()
  When the user enters their email and password1          # stepdefinition.passwordSteps.the_user_enters_credentials()
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."
  And the user clicks on the login button1                # stepdefinition.passwordSteps.user_clicks_on_login()
1698218854528   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:62999
1698218854539   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInParent" to false before Firefox starts.
1698218854611   Marionette      INFO    Stopped listening on port 62937
Dynamically enable window occlusion 1
  Then the user should be navigated to the home page1     # stepdefinition.passwordSteps.the_user_should_be_navigated_to_the_home_page()
1698218857411   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:63000
1698218857419   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInParent" to false before Firefox starts.
1698218857427   Marionette      INFO    Stopped listening on port 62911
Dynamically enable window occlusion 1

1 Scenarios (1 passed)
5 Steps (5 passed)
0m24.053s
```

Test report

Test Case 4

Project Name: CareerGo: Career Guidance Website					
Update Profile Test Case					
Test Case ID: Test_4			**Test Designed By:** Febin Vinod		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 22-10-2023		
Module Name: update Profile			**Test Executed By :** Ms.Jetty Benjamin		
Test Title : update Profile			**Test Execution Date:** 24-10-2023		
Description: Profile Updating by the user					
Pre-Condition : User has valid username 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 Valid username	username: jeso	User should be able to Login	User Logged in and navigated to Home page	Pass
3	Provide Valid Password	Password: Jeso@123			
4	Click on Login button				
5	From navbar click User Profile option		Profile dashboard should be shown	Profile dashboard is shown to the user	Pass
6	In the form enter the data to be updated and click on submit to apply changes	Phone: 9446732818 Country: India City: Kottayam Pincode: 686514	User Profle will be updated	User Profle updated	Pass
Post-Condition: User Successfully update the profile					

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The implementation phase of a project is where the design is transformed into a functional system. It is a crucial stage in ensuring the success of the new system, as it requires gaining user confidence that the system will work effectively and accurately. User training and documentation are key concerns during this phase. Conversion may occur concurrently with user training or at a later stage. Implementation involves the conversion of a newly revised system design into an operational system.

During this stage, the user department bears the primary workload, experiences the most significant upheaval, and has the most substantial impact on the existing system. Poorly planned or controlled implementation can cause confusion and chaos. Whether the new system is entirely new, replaces an existing manual or automated system, or modifies an existing system, proper implementation is essential to meet the organization's needs. System implementation involves all activities required to convert from the old to the new system. The system can only be implemented after thorough testing is done and found to be working according to specifications. System personnel evaluate the feasibility of the system. Implementation requires extensive effort in three main areas: education and training, system testing, and changeover. The implementation phase involves careful planning, investigating system and constraints, and designing methods to achieve changeover.

6.2 IMPLEMENTATION PROCEDURES

Software implementation is the process of installing the software in its actual environment and ensuring that it satisfies the intended use and operates as expected. In some organizations, the software development project may be commissioned by someone who will not be using the software themselves. During the initial stages, there may be doubts about the software, but it's important to ensure that resistance does not build up. This can be achieved by:

•Ensuring that active users are aware of the benefits of the new system, building their confidence in the software.

• Providing proper guidance to the users so that they are comfortable using the application.

Before viewing the system, users should know that the server program must be running on the server. Without the server object up and running, the intended process will not take place.

6.2.1 User Training

User training is all about getting the users ready to test and switch over to the new system. To make sure the computer system works well and brings the expected benefits, it's crucial for the people using it to feel confident about what they need to do. When a system is complex, training becomes even more important. Through user training, people learn how to put in data, handle errors, ask questions to the database, and use routines to generate reports and do other necessary tasks. This training helps everyone know how to use the system effectively and get the most out of it.

6.2.2 Training on the Application Software

Once you've given people the basic training they need to understand computers, it's also important to train them on the new software they'll be using. This training should teach them not just how to use the software, but also the ideas behind it, like how the screens work, how they're designed, the help available, common errors and how to fix them. This training should also cover specific information that each person or group needs to use the software effectively. It's worth noting that this training might be different for various groups of users and at different levels in the organization.

6.2.3 System Maintenance

The maintenance phase is a very important part of making software because this is when the software is used for its real job. Keeping the software working well is crucial to ensure it stays dependable and can handle changes in its environment. Maintenance isn't just about finding and fixing errors or problems in the software. It can also mean updating the software, changing how it works, or making it perform better, among other things. In short, software maintenance is an ongoing process where we keep an eye on the software, check how it's doing, and make it better to meet the needs of the people using it as those needs change

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The development and proposed implementation of the Online Career Guidance System represent a significant step forward in addressing the complex challenges faced by students in making informed career decisions. The existing system's drawbacks, including the lack of centralization, limited career assessments, and the absence of personalized guidance, underscored the need for a more comprehensive and user-centric solution. The proposed system offers a centralized platform where students can access a wealth of career guidance resources. It provides personalized career recommendations based on individual skills, interests, and values, making the decision-making process more informed and tailored to the students' unique needs.

Moreover, the system's provision of comprehensive information about educational programs, courses, internships, and online resources empowers students to explore diverse career paths and gain valuable work experiences. As we move forward with the development of the Online Career Guidance System, we remain committed to the goal of empowering students, simplifying their career decisions, and providing them with the support and resources they need to embark on successful professional journeys.

In conclusion, the Online Career Guidance System represents a promising and user-centric solution to the challenges faced by students in their pursuit of educational and career excellence. It stands as a testament to our dedication to enhancing the career guidance process for students, ultimately contributing to their personal and professional success.

7.2 FUTURE SCOPE

The Online Career Guidance System, with its innovative features and user-centric approach, opens up numerous opportunities for future enhancements and expansions. One potential avenue for growth is the inclusion of additional resources and tools to cater to specific educational and career needs. This may involve incorporating more specialized assessments, expanding the range of available courses and internships, or providing advanced tools for skill development.

Furthermore, there is room for the system to evolve with the changing landscape of education and careers. As emerging fields and industries develop, the system can adapt to provide up-to-date information on the latest career opportunities and educational programs. Integration with emerging technologies, such as augmented reality or virtual reality, could also provide immersive experiences for students to explore their chosen fields. Collaborations with educational institutions, industry experts, and mentors could also be explored in the future. Such partnerships could enhance the quality of educational content, provide real-world insights, and offer students valuable networking opportunities.

In conclusion, the future scope of the Online Career Guidance System is rich with possibilities for expansion, enhancement, and adaptation to the evolving landscape of education and careers. As it continues to evolve, the system will remain a vital resource for students seeking guidance and support in their educational and career journeys.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- PankajJalote, “Software engineering: a precise approach”
- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009
- Ken Schwaber, Mike Beedle, Āgile Software Development with Scrum, Pearson (2008)
- Roger S Pressman, “Software Engineering”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- www.careerguide.com
- www.w3schools.com
- https:/stackoverflow.com/
- www.djangoproject.com

CHAPTER 9
APPENDIX

9.1 Sample Code

Views.py

LOGIN

```
def login(request):
    if request.method == 'POST':


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


         if username  and password:
              user = authenticate(request, username=username,
password=password)
              if user is not None:
                   auth_login(request, user)
                   if request.user.role == CustomUser.ADMIN:

                        return redirect(reverse('adminpage'))

                   elif request.user.role == CustomUser.STUDENT:

                        return redirect(reverse('index'))
                   elif request.user.role == CustomUser.COURSE_PROVIDER:

                        return redirect(reverse('providerpage'))
              else:
                   messages.info(request,"Invalid User")
                   return redirect('login')
                #    error_message = "Invalid login credentials."
                #    return render(request, 'login.html', {'error_message':
error_message})

    return render(request,'login.html')
```

SIGNUP

```
def signup(request):
    if request.method == 'POST':
        username = request.POST.get('username', None)
        email = request.POST.get('email', None)
        first_name = request.POST.get('first_name', None)
        last_name = request.POST.get('last_name', None)
        phone = request.POST.get('phone', None)
        password = request.POST.get('password', None)
        role = User.STUDENT
        if username and first_name and last_name and email and phone and
password and role:
            if User.objects.filter(username=username).exists():
                messages.info(request, "Username already taken")
                return redirect('/signup')
            elif User.objects.filter(email=email).exists():
                messages.info(request, "Email already taken")
                return redirect('/signup')
            elif User.objects.filter(phone_no=phone).exists():
                messages.info(request, "Phone number already taken")
                return redirect('/signup')
            else:
                user = User(username=username, first_name=first_name,
last_name=last_name, email=email, phone_no=phone, role=role)
                user.set_password(password)
                user.save()

                # Create a user profile associated with the registered user
                user_profile = UserProfile(user=user)
                user_profile.save()

                messages.info(request, "Registered")
                return redirect('login')

    return render(request, 'signup.html')


def profile(request):
    user = request.user
    user_profile = UserProfile.objects.get(user=user)

    if request.method == 'POST':
        # Update user fields
        user.first_name = request.POST.get('first_name')
        user.last_name = request.POST.get('last_name')
        user.phone_no = request.POST.get('phone_no')
        #user.password = request.POST.get('password')

        new_password = request.POST.get('new_password')
        confirm_password = request.POST.get('confirm_password')

        # Check if the old password is correct
        if new_password and new_password == confirm_password:
            user.set_password(new_password)
            update_session_auth_hash(request, user)  # Update session to
prevent logout
        elif new_password:
            messages.error(request, "Passwords do not match")
        user.save()


        # Update user profile fields
        new_profile_pic = request.FILES.get('profile_pic')
        if new_profile_pic:
            user_profile.profile = new_profile_pic
        user_profile.country = request.POST.get('selected_country')
        user_profile.state = request.POST.get('selected_state')
        user_profile.city = request.POST.get('city')
        user_profile.pin_code = request.POST.get('pin_code')
        user_profile.save()

        print("Selected State:", request.POST.get('selected_state'))
        print("Selected counter:", request.POST.get('selected_country'))

        return redirect('profile')
    context = {
        'user': user,
        'user_profile': user_profile,
    }

    return render(request, 'profile.html',context)
```

ASSESSMENT

```
def assessment(request):
    #predict_and_recommend = joblib.load('model/knn.pkl')
    user = request.user
    exist = StudentAssessment.objects.filter(student_id=user.id,
assessment_status=True).exists()

    if request.method == 'POST':
        logical_reasoning_score =
int(request.POST.get('category_logical_reasoning_score', 0))
        communication_skills_score =
int(request.POST.get('category_communication_skills_score', 0))
        quantitative_aptitude_score =
int(request.POST.get('category_quantitative_aptitude_score', 0))
        analytical_skills_score =
int(request.POST.get('category_analytical_skills_score', 0))
        # Retrieve the stream and Plus Two CGPA from the request
        stream = request.POST.get('stream', None)
        plus_two_cgpa = float(request.POST.get('plus-two-cgpa', 0))
        total_score = int(request.POST.get('total_score', 0))

        print(logical_reasoning_score)
        print(total_score)

        # Map the stream to a numeric value
        stream_mapping = {
            'Science': 8,
            'Commerce': 6,
            'Humanities': 5,
        }
        stream_numeric = stream_mapping.get(stream, 0)


        input_data = [[logical_reasoning_score, communication_skills_score, 
quantitative_aptitude_score, analytical_skills_score, plus_two_cgpa, 
stream_numeric]]

        recommended_courses,  sorted_recommended_colleges = 
predict_and_recommend(input_data)


        top_five_colleges = {}
        for i, course in enumerate(recommended_courses):
            top_five_colleges[course] = 
sorted_recommended_colleges[i][['College', 
'NIRF_Ranking_2023']].head(5).to_dict(orient='records')

        print(recommended_courses)


        assessment, created = 
StudentAssessment.objects.get_or_create(student=user)
        assessment.logical_reasoning_score = logical_reasoning_score
        assessment.communication_skills_score = communication_skills_score
        assessment.quantitative_aptitude_score = quantitative_aptitude_score
        assessment.analytical_skills_score = analytical_skills_score
        assessment.stream = stream
        assessment.plus_two_cgpa = plus_two_cgpa
        assessment.total_score = total_score
        assessment.assessment_status = True
        assessment.save()

        context = {
        'assessment_status': exist, 'recommended_course': 
recommended_courses, 'top_five_colleges':  top_five_colleges,'assessment': 
assessment,
    }


        return render(request, 'result.html', context)
    return render(request, 'assesment.html', {'assessment_status': exist})
```

COURSEPROVIDER

```
def addCourseprovider(request):
    if request.method == 'POST':
        user_form = CustomUserForm(request.POST)

        if user_form.is_valid():
            user = user_form.save(commit=False)
            password = user_form.cleaned_data['password']

            # Send welcome email
            send_welcome_email(user.username, password,
user.first_name,user.email)

            user.set_password(password)
            user.is_active = True

            user.role = CustomUser.COURSE_PROVIDER
            user.save()

            if user.role == CustomUser.COURSE_PROVIDER:
                Courseprovider = Courseprovider_profile(user=user)
                Courseprovider.save()

            user_profile = UserProfile(user=user)
            user_profile.save()

            return redirect('index')

    else:
        user_form = CustomUserForm()

    context = {
        'user_form': user_form,
    }

    return render(request, 'addprovider.html', context)

def send_welcome_email(username, password, name,email):

    login_url = 'http://127.0.0.1:8000/login'
    login_button = f'Click here to log in: {login_url}\n'


    subject = 'CareeGo - Course Provider Registration'
    message = f"Hello {name},\n\n"
    message += f"Welcome to Careergo, We are thrilled to have you on board as
a part of our dedicated team of providers.\n\n"
    message += f"Your registration is complete, and we're excited to have you
join us. Here are your login credentials:\n\n"
    message += f"Username: {username}\nPassword: {password}\n\n"
    message += "Please take a moment to log in to your account using the
provided credentials. Once you've logged in, we encourage you to reset your
password to something more secure and memorable.\n\n"
    message += login_button
    message += "Thank you for joining the CareerGo community. We look forward
to your contributions and the positive energy you'll bring to our
platform.\n\n"
    message += "Warm regards,\nThe CareerGo Team\n\n"



    from_email='amalraj89903@gmail.com'
      # Replace with your actual email
    recipient_list = [email]

    send_mail(subject, message, from_email, recipient_list)
```

<u>COURSES</u>

```
def addcourses(request):

    category = Oncourse.CATEGORY_CHOICES
    context = {'category': category}
    if request.method == 'POST':

        new_course = Oncourse()
        user = request.user
        course_title = request.POST['course_title']
        description = request.POST['description']
        duration = request.POST['duration']
        price_str = request.POST.get('price', '')
        price = int(price_str) if price_str else 0
        instructor= request.POST.get('instructor')
        thumbnail = request.FILES['thumbnail']
        category = request.POST['category']
        is_free = request.POST.get('is_free') == 'on'


              if category == 'other':
            category = request.POST.get('other_category')

print({price})

        print({is_free})
        new_course = Oncourse(user=user, course_title = course_title
,price=price,
description=description,duration=duration, thumbnail=thumbnail
,category=category, instructor=instructor,is_free=is_free,)


        new_course.save()

        sections = request.POST.getlist('section_name[]')

        videos = request.FILES.getlist('videos[]')
        for section, video in zip(sections, videos):
            Video.objects.create(course=new_course, video_file=video,
section_name=section,user_id=request.user.id)

        return redirect('courseprovider:courselist')

    return render(request, 'addcourse.html',context)
```

ADMIN DASHBOARD

```
def admindash(request):
    users = CustomUser.objects.filter(role__in=[CustomUser.STUDENT,
CustomUser.COURSE_PROVIDER, CustomUser.MENTOR])

    course_providers = users.filter(role=CustomUser.COURSE_PROVIDER)
    courses_by_provider = {}
    for provider in course_providers:
        courses = Oncourse.objects.filter(user=provider)
        courses_by_provider[provider] = courses

    internships_by_provider = {}
    for provider in course_providers:
        internships = Internship.objects.filter(user=provider)
        internships_by_provider[provider] = internships
    assessments = StudentAssessment.objects.filter(student__in=users)
    return render(request, 'admindash.html', {'users': users,'assessments':
assessments,'courses_by_provider': courses_by_provider,
        'internships_by_provider': internships_by_provider,})
```

STUDENT DASHBOARD

```
def studash(request):
    user = request.user
    enrolled_courses =
Payment.objects.filter(user=request.user,course__isnull=False,payment_status=
Payment.PaymentStatusChoices.SUCCESSFUL)
```

```
    enrolled_internships = Payment.objects.filter(user=request.user,
internship__isnull=False,payment_status=Payment.PaymentStatusChoices.SUCCESSF
UL)
    certificates = Certificate.objects.filter(user=user)
    if user.role == CustomUser.STUDENT:
        assessments = StudentAssessment.objects.filter(student=user)
        return render(request, 'studash.html', {'user': user, 'assessments':
assessments, 'enrolled_courses': enrolled_courses, 'enrolled_internships':
enrolled_internships,'certificates': certificates,})

    return render(request, 'studash')
```

INTERNSHIP

```
def intern(request):
    category = Internship.CATEGORY_CHOICES
    context = {'category': category}

    if request.method == 'POST':

      new_intern = Internship()
      user = request.user
      internship_title = request.POST['course_title']
      description = request.POST['description']
      duration = int(request.POST['duration'])
      price = request.POST.get('price','')
      instructor= request.POST.get('instructor')
      start_date = request.POST['start_date']
      end_date = request.POST['end_date']
      positions = int(request.POST['positions'])
      internship_type = request.POST['internship_type']
      internship_mode = request.POST['internship_mode']
      application_deadline = request.POST['application_deadline']
      company_name = request.POST['company_name']
      company_website = request.POST['company_website']

      thumbnail = request.FILES['thumbnail']
      category = request.POST['category']

      if category == 'other':

            category = request.POST.get('other_category')



      new_intern = Internship(user=user, internship_title = internship_title
,price=price, description=description,duration=duration, thumbnail=thumbnail
,category=category,instructor=instructor,start_date=start_date,end_date=end_d
ate,positions=positions,internship_type=internship_type,internship_mode=inter
nship_mode,application_deadline=application_deadline,company_name=company_nam
e,company_website=company_website)

      new_intern.save()
      return redirect('courseprovider:internlist')

    return render(request, 'addintern.html',context)
```

PAYMENT

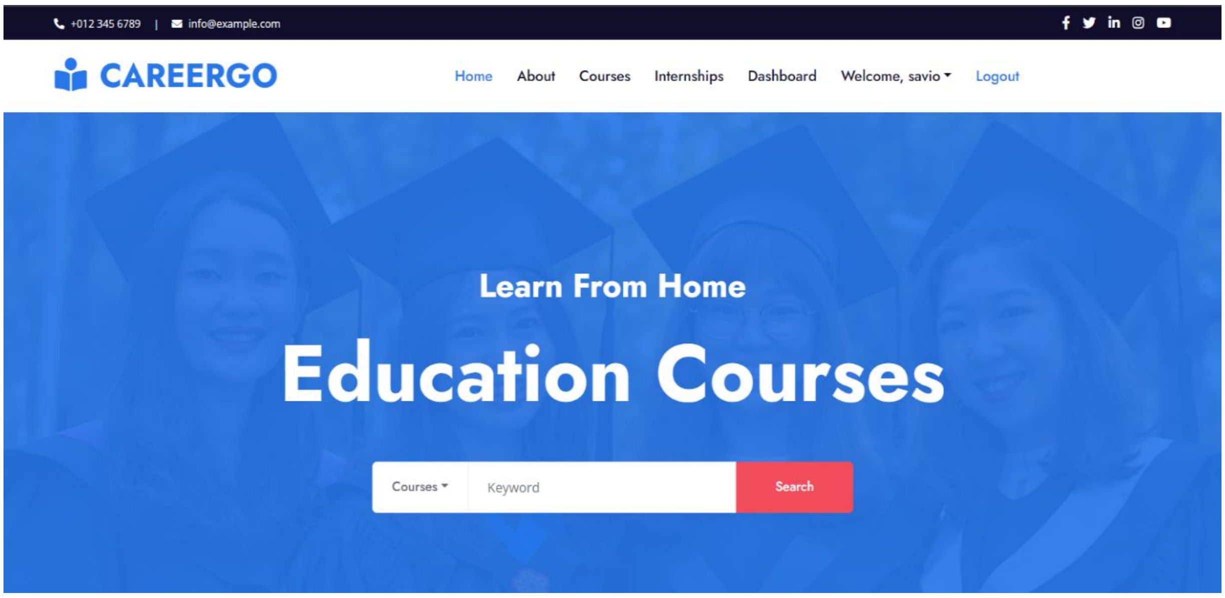
```
def confirm(request, course_id):
    course = get_object_or_404(Oncourse, pk=course_id)
    user = request.user
    existing_payment = Payment.objects.filter(user=request.user,
course=course,
payment_status=Payment.PaymentStatusChoices.SUCCESSFUL).first()

    if existing_payment:
        return render(request, 'already_enrolled.html')

    if course.is_free:
        Payment.objects.create(
            user=request.user,
            course=course,
            razorpay_order_id="",
            payment_id="",
            amount=0,
            currency="INR",
            payment_status=Payment.PaymentStatusChoices.SUCCESSFUL
        )

        return redirect('courseprovider:success')
    currency = 'INR'
    amount = course.price
    amount_in_paise = int(amount * 100)
    razorpay_order = razorpay_client.order.create(dict(
        amount=amount_in_paise,
        currency=currency,
        payment_capture='0'
    ))
    razorpay_order_id = razorpay_order['id']
    callback_url = '/courseprovider/paymenthandler/'

    payment = Payment.objects.create(
        user=request.user,
        course=course,
        razorpay_order_id=razorpay_order_id,
        payment_id="",
        amount=amount,
        currency=currency,
        payment_status=Payment.PaymentStatusChoices.PENDING,
    )
    payment.save()

    context = {
        'user': request.user,
        'oncourse': course,
        'razorpay_order_id': razorpay_order_id,
        'razorpay_merchant_key': settings.RAZOR_KEY_ID,
        'razorpay_amount': amount_in_paise,
        'currency': currency,
        'amount': amount_in_paise / 100,
        'callback_url': callback_url,
    }

    return render(request, 'confirm.html', context)
```

CERTIFICATE

```
def generate_certificate(request, course_id):
    course = Oncourse.objects.get(pk=course_id)
    user = request.user
    existing_certificate = Certificate.objects.filter(user=user,
course=course).first()

    if existing_certificate:
        return existing_certificate

    current_date = datetime.datetime.now().strftime("%Y-%m-%d")
    # Create and save the certificate in the database
    certificate = Certificate(user=user, course=course,
issued_date=current_date)
    certificate.save()

    return certificate

def view_certificate(request, course_id):
    course = Oncourse.objects.get(pk=course_id)
    certificate = generate_certificate(request, course_id)
    if isinstance(certificate, HttpResponse):
        return certificate
    context = {
        'course': certificate.course,
        'current_date': certificate.issued_date.strftime("%Y-%m-%d"),
        'user': certificate.user,
        'certificate_id': certificate.id ,
    }

    return render(request, 'download_certificate.html', context)
```

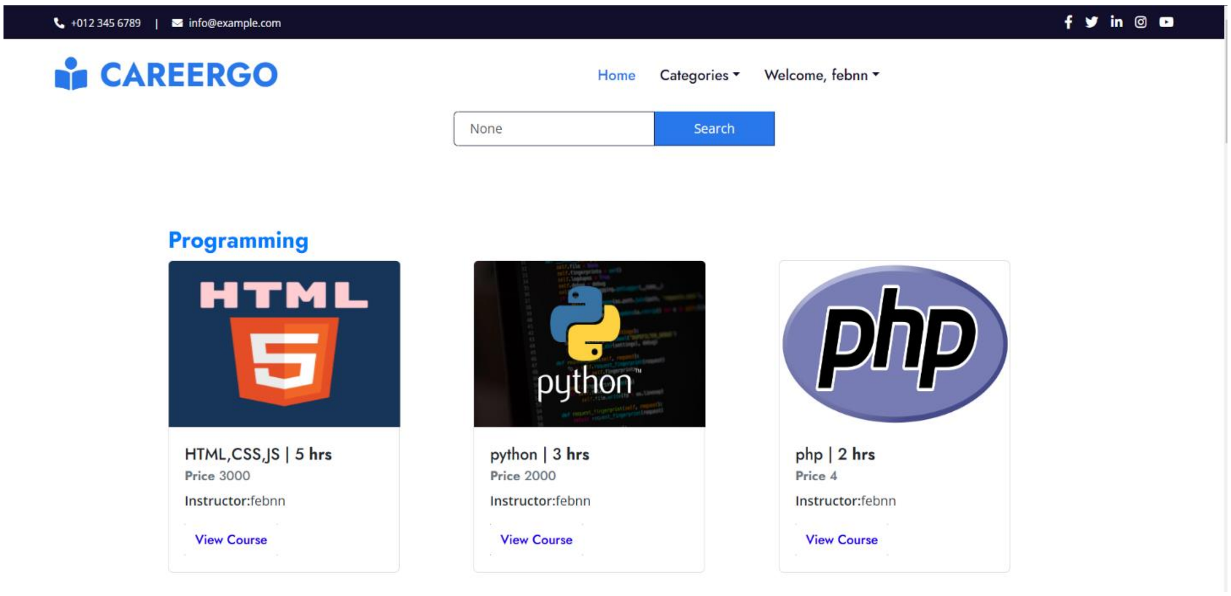
9.2 Screen Shots

Home Page



Home Page

Course List Page



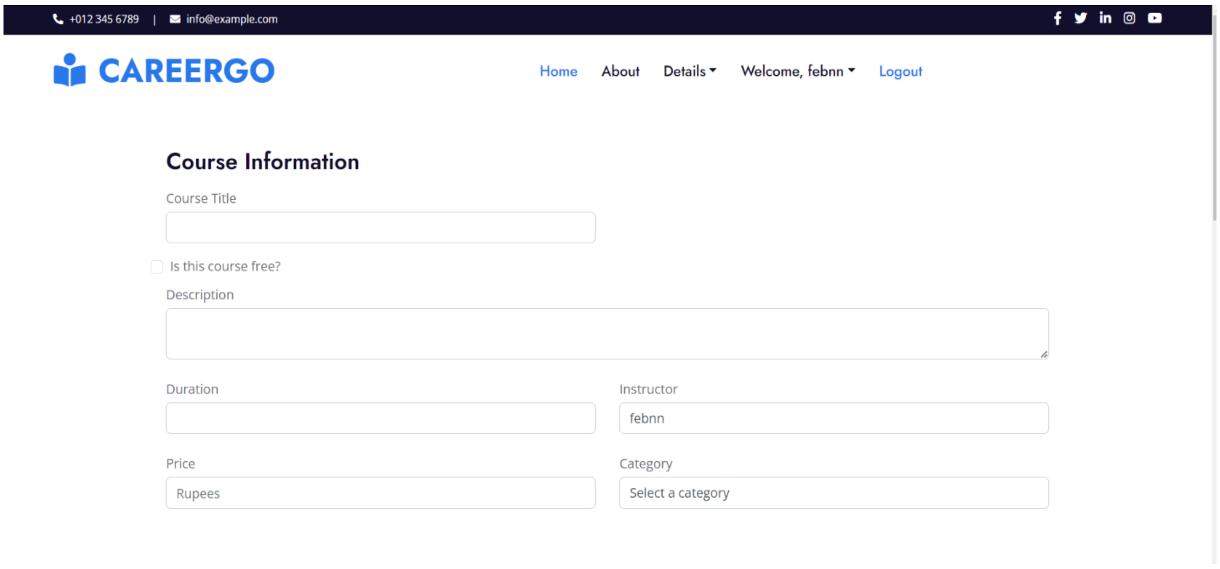
HTML,CSS,JS | 5 hrs
Price 3000
Instructor:febnn

python | 3 hrs
Price 2000
Instructor:febnn

php | 2 hrs
Price 4
Instructor:febnn

Course List Page

Course Add Page


+012 345 6789 | info@example.com

CAREERGO

Home About Details Welcome, febnn Logout

Course Information

Course Title

Is this course free?

Description

Duration

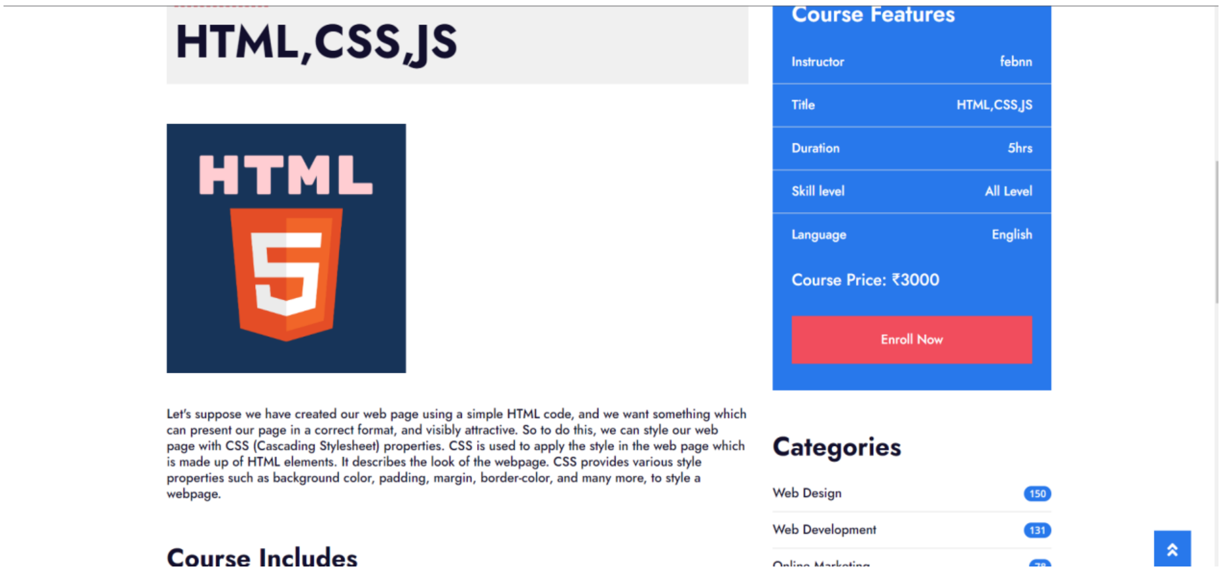
Instructor: febnn

Price: Rupees

Category: Select a category

Course Add Page

Course Details Page


HTML,CSS,JS

Course Features

Instructor	febnn
Title	HTML,CSS,JS
Duration	5hrs
Skill level	All Level
Language	English

Course Price: ₹3000

Enroll Now

Let's suppose we have created our web page using a simple HTML code, and we want something which can present our page in a correct format, and visibly attractive. So to do this, we can style our web page with CSS (Cascading Stylesheet) properties. CSS is used to apply the style in the web page which is made up of HTML elements. It describes the look of the webpage. CSS provides various style properties such as background color, padding, margin, border-color, and many more, to style a webpage.

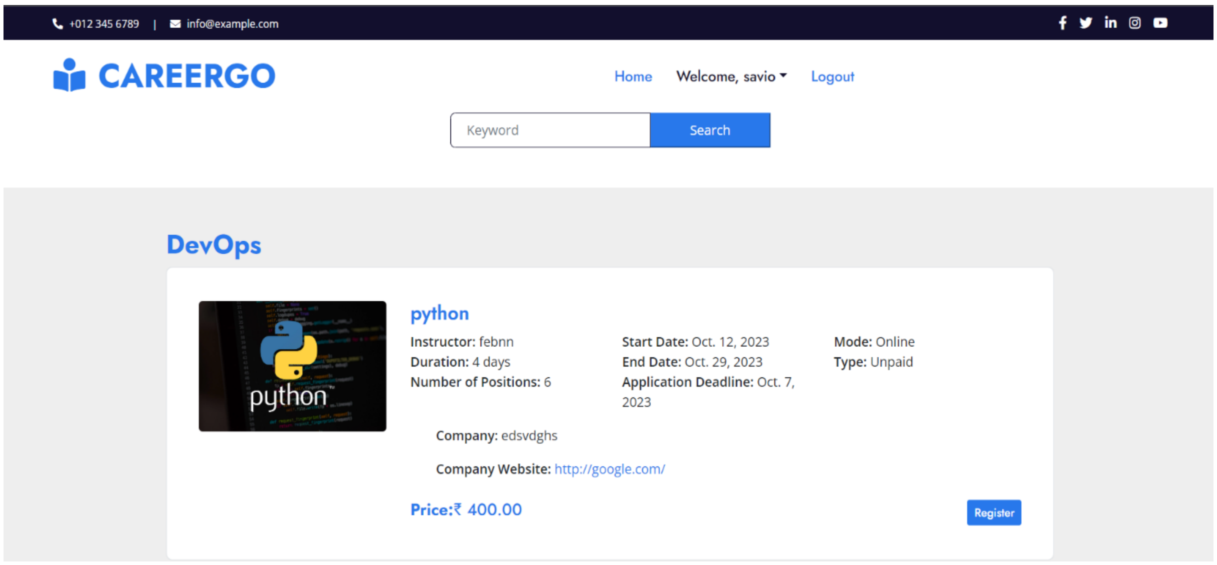
Course Includes

Categories

Web Design	150
Web Development	131

Course Detail Page

Internship List Page



DevOps

python

Instructor: febnn
Duration: 4 days
Number of Positions: 6
Start Date: Oct. 12, 2023
End Date: Oct. 29, 2023
Application Deadline: Oct. 7, 2023
Mode: Online
Type: Unpaid

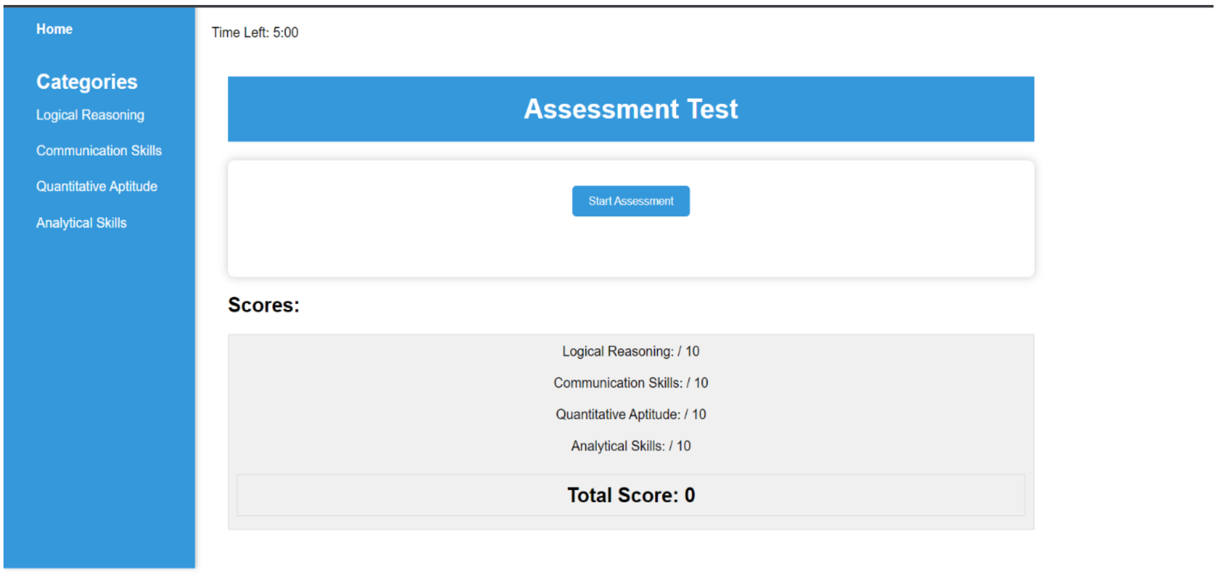
Company: edsvdghs

Company Website: http://google.com/

Price:₹ 400.00

Internship List Page

Assessment Page



Time Left: 5:00

Assessment Test

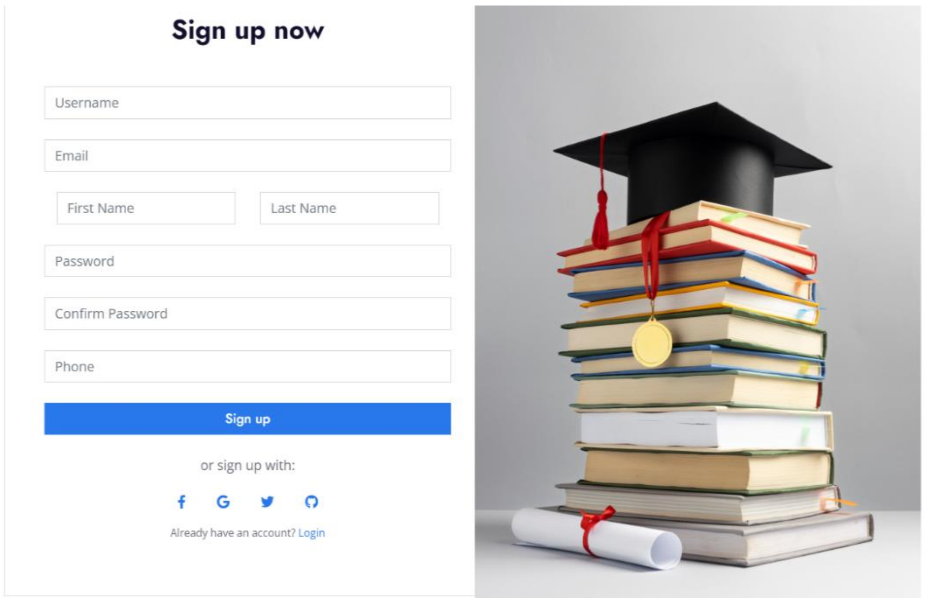
Scores:

Logical Reasoning: / 10
Communication Skills: / 10
Quantitative Aptitude: / 10
Analytical Skills: / 10

Total Score: 0

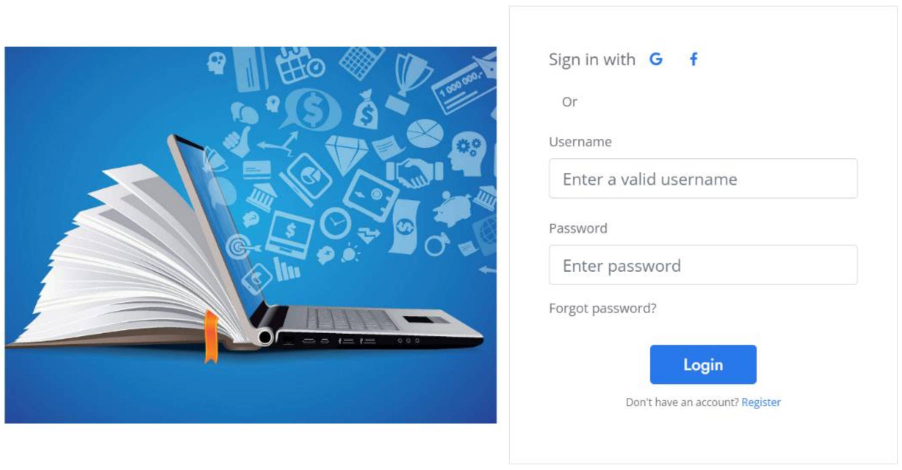
Assessment Page

Registration Page



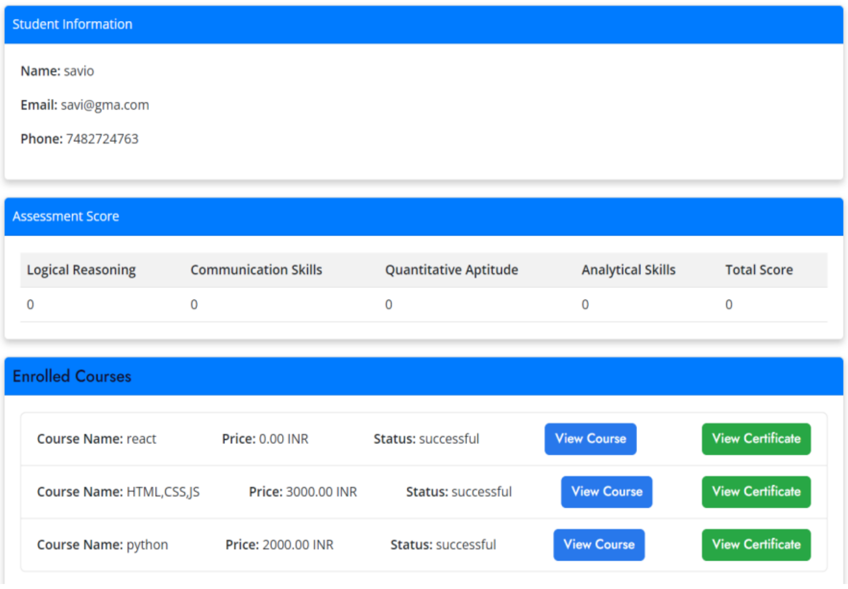
Registration Page

Login Page



Login Page

Student Dashboard



Student Information

Name: savio

Email: savi@gma.com

Phone: 7482724763

Assessment Score

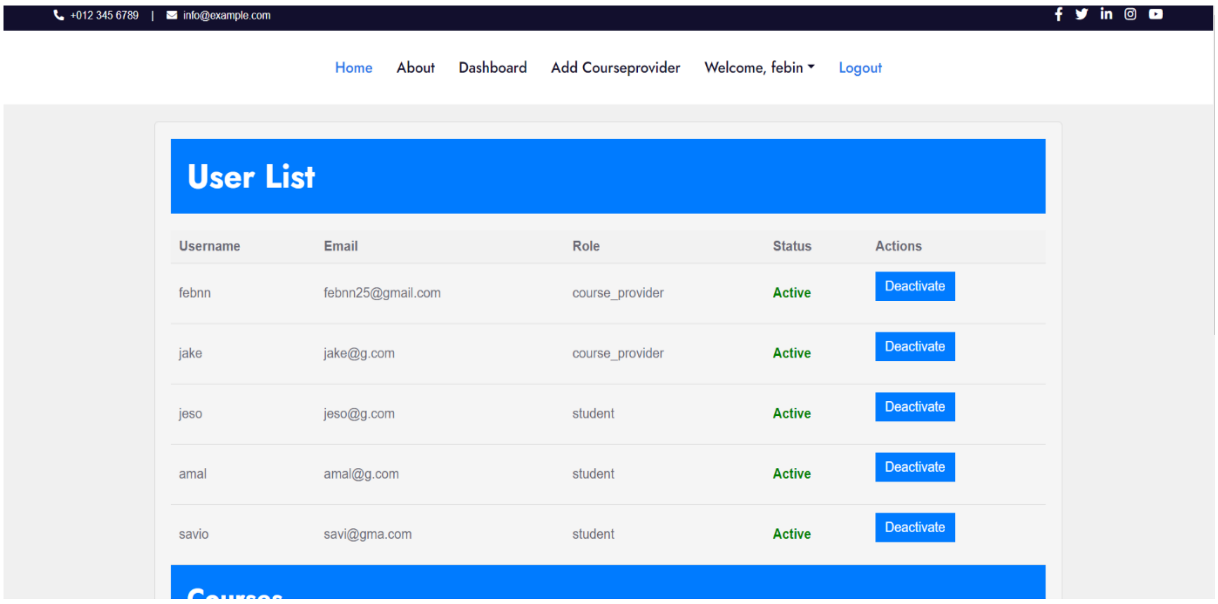
Logical Reasoning	Communication Skills	Quantitative Aptitude	Analytical Skills	Total Score
0	0	0	0	0

Enrolled Courses

Course Name: react	**Price:** 0.00 INR	**Status:** successful	View Course	View Certificate
Course Name: HTML,CSS,JS	**Price:** 3000.00 INR	**Status:** successful	View Course	View Certificate
Course Name: python	**Price:** 2000.00 INR	**Status:** successful	View Course	View Certificate

Student Dashboard

Admin Dashoard



User List

Username	Email	Role	Status	Actions
febnn	febnn25@gmail.com	course_provider	Active	Deactivate
jake	jake@g.com	course_provider	Active	Deactivate
jeso	jeso@g.com	student	Active	Deactivate
amal	amal@g.com	student	Active	Deactivate
savio	savi@gma.com	student	Active	Deactivate

Admin Dashboard