SRINIVAS UNIVERSITY

Skill Bites

Submitted to Srinivas University on partial Completion of Sixth Semester

Bachelor of Computer Application

By

Tanmaya s ail (3SU21SA139)

VI Semester BCA

Under the guidance of

Prof. Jowana Dhrishal

Faculty of

Srinivas University Institute of Computer

And Information Science

Pandeshwar, Mangalore

2021-2024

SRINIVAS UNIVERSITY

Srinivas Nagar, Mukka– 574 146, Mangaluru, Karnataka. Phone : 0824-2477456
(Private University Established by Karnataka Govt. ACT No.42 of 2013, Recognized by UGC, New Delhi & Member of Association of Indian Universities, New Delhi)
Web : www.srinivasuniversity.edu.in, Email: info@srinivasuniversity.edu.in

INSTITUTE OF COMPUTER SCIENCE & INFORMATION SCIENCES

CERTIFICATE

This is to certify that the project work entitled "Skill Bites" carried out by Mr./Ms. Tanmaya S. Ail (Registration No. 39U21SA139), a bonafide student of this institution, in partial fulfillment for the award of Degree **B.C.A/B.Sc/M.C.A** at Srinivas University, City Campus, Pandeshwara, Mangaluru – 575 001, during the academic year 2023-2024. It is also certified that all corrections/suggestions indicated during internal assessment have been incorporated in the Report. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

Name & Signature of the Guide

Name & Signature of the Dean

External Viva :

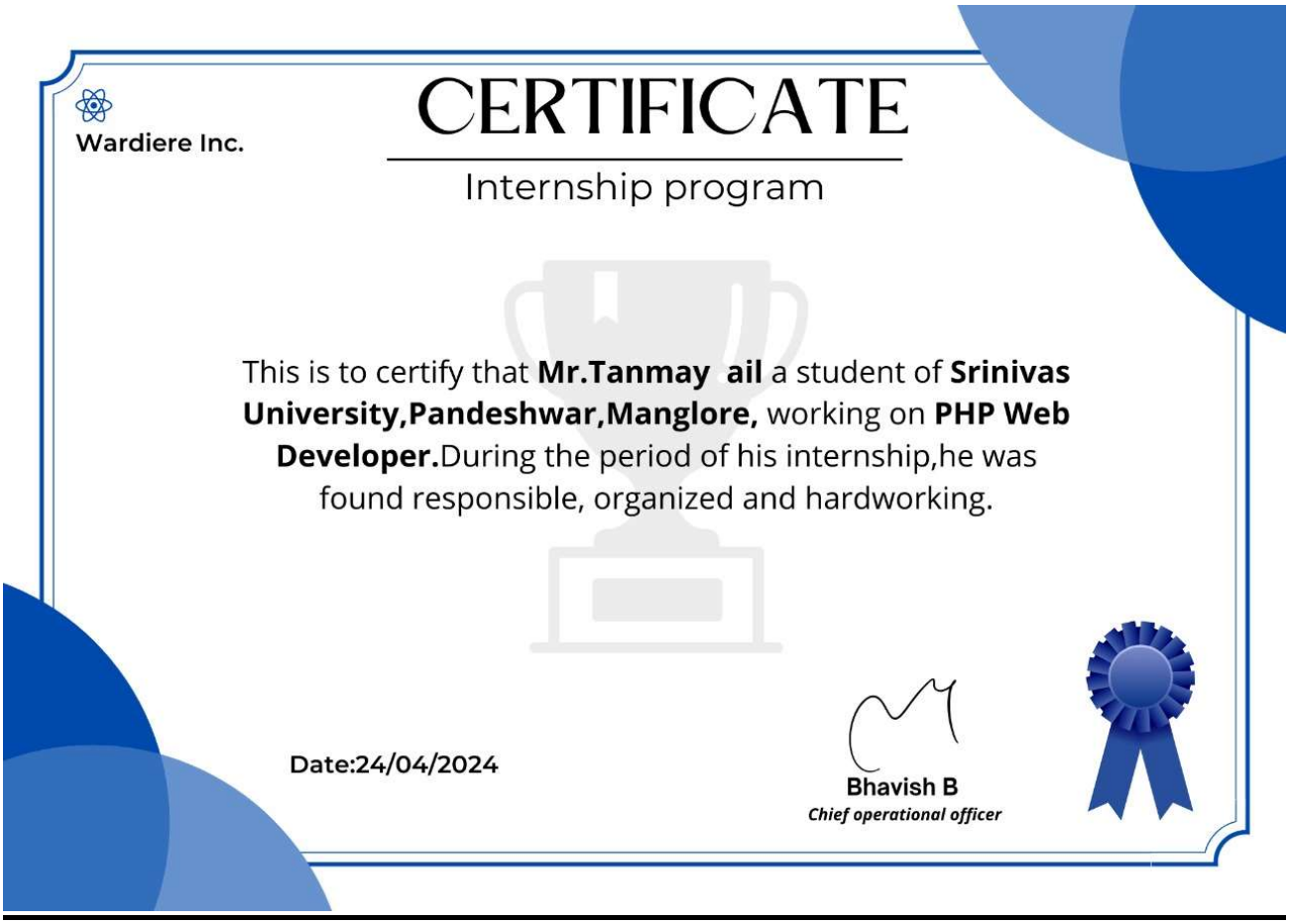
Name of the Examiners

1. R Sridhara Acharya
2 SACHIN.M

Signature with Date

SRINIVAS UNIVERSITY * G.H.S. ROAD * MANGALORE-575001

DEAN
Institute of Computer Science and Information Science
Srinivas University
Mangaluru - 575 001


Wardiere Inc.
CERTIFICATE
Internship program
This is to certify that Mr.Tanmay ail a student of Srinivas University,Pandeshwar,Manglore, working on PHP Web Developer.During the period of his internship,he was found responsible, organized and hardworking.
Date:24/04/2024
Bhavish B
Chief operational officer

DECLARATION

We hereby declare that the project work detail which is being presented in this report is in fulfillment of the requirement for the award of degree of Bachelor of Computer Application.

We hereby declare that we have undertaken our project work on "Skill Bytes" under the guidance of Ms. Jowana Dhirshal, faculty, Department of Computer science and Information Science m Srinivas University, Mangalore.

We hereby declare that this project work report is our own work and the best of our knowledge and belief the matter embedded in report has not been submitted by us for the award of other degree to this or any other university.

Place: Mangalore **Tanmaya S Ail (3SU2LSA139)**

Date: 03-05-2024

ACKNOWLEDGEMENT

Every success, big or small, usually begins with a spark or something that gets it going. It's the continuous help and smart advice from wise people that keep us moving forward. The joy and satisfaction we feel after completing a task wouldn't be complete without acknowledging those who supported us. Their constant guidance and encouragement are what lead us to victory.

We would like to express our sincere and grateful thanks to our **Dr. P. Sridhara Acharya**, Professor and Head, Department of Computer Science, Srinivas University, Mangalore for the valuable guidance, encouragement, technical comments throughout our project work.

It is great pleasure to express our gratitude and indebtedness to our beloved **Dean, Dr. Subrahmanya Bhat** for his continues effort in creating a competitive environment in our college and encouraging throughout this course.

We also wish to thank all the staff members, non-teaching staff members of the Department of Computer Science and Information Science who have helped us directly or indirectly in the completion of our final project successfully.

Finally, we are thankful to our parents, friends and loved ones, who are always our source of inspiration and for their continued moral and material support throughout the course and in helping us to finalize the project.

Mr.Tanmaya s ail

Contents

1. SYNOPSIS

1.1 Title of the project:

SKILL BITES

1.2 Introduction of project:

In today's modern era where technology has evolved and people started to rely more on online courses rather than going to colleges, we have come up with a user-friendly platform that makes online learning easy and engaging. This project focuses on using the latest tech to make personalized learning experiences. We work to make online education simple and accessible for everyone. This project aims to create an engaging and informative online course that helps learners achieve the desired learning outcomes.

1.3 Innovative idea behind the project:

This project aims to create a space where learners and instructors can connect seamlessly. It is created for every instructors who wants to provide knowledge to the students via recordings, courses and assignments and the instructor too get paid.

1.4 Nature of the project:

- Users can access a wide range of courses covering various subjects and skill levels.
- Users can enhance their skills in specific areas through targeted courses and exercises.
- Simplify tasks for instructors and promote efficiency in grading, announcements and enrolment.
- Users can track their progress within courses, set goals, and monitor their achievements over time.
- Incorporate interactive elements, such as assignments, and discussions.
- Save users time and money by eliminating the need for travel and providing cost-effective learning alternatives.

1.5 Literature Survey:

Conducted survey on existing course-based website on the internet to understand the features, functionality and design. Api's related to real time communication, payment gateways.

1.6 Hardware and Software Requirements:

- **Software requirements:**

  Platform: Microsoft windows
  IDE: Visual Studio code
  Frontend Technology: html, css, javascript, ReactJs
  Backend Technology: MongoDB database, Node.js
  Additional libraries and dependencies as per the project requirements.

- **Hardware requirements:**
  Adequate storage for storing course data (MongoDB)
  Cloudinary to store videos and images

This project holds immense potential to become a go-to destination for e learning seeking an ultimate knowledge and education.

1.7 Team Members

Tanmay s ail

Tushan

2. SOFTWARE REQUIREMENT SPECIFICATION

2.1 Introduction:

The Software Requirement Analysis document provides a complete overview description of all the functionalities and specification of the “SkillBites” system. The SRS is a comprehensive document that outlines the functional and non-functional requirements of a software system.

2.2 Purpose:

This document is developed for the project “SkillBites” to provide a platform for the instructors who is willing to start a paid online course to the students who want to learn specific courses. Here the students can interact with the instructors and can go through the posted assignments, study materials, quiz and recorded class. An SRS minimizes the time and effort required by the developers to achieve desired goals and also minimizes the development cost.

2.3 Scope:

The main objective of the project “SkillBites” which is designed to provide a platform to students to buy courses online and study online.

It provides the following features

- The instructors have the capability to create courses with reasonable price, providing a diverse range of learning opportunities. Admin can view Instructer created course
- Students can post queries regarding any assignments.
- Instructer can assign assigment for their students The system captures and maintains records of various aspects of student performance. This includes details about assignment submissions, quiz results, and overall performance. This feature enables both students and instructors to monitor and evaluate progress.
- Students can enroll the courses they want
- Students have the option to post queries related to assignments.
- Upon successful completion of a course, students will receive a certificate of completion.

2.4 Defination, Acronyms,abbreviations

ADMIN - THE ADMINISTRATOR

CPU -CENTRAL PROCESSING UNIT

GUI - GRAPHICAL USER INTERFACE

HTML -HYPER TEXT MARKUP LANGUAGE

CSS -CASCADING STYLE SHEET

DBMS -DATABASE MANAGEMENT SYSTEM

UI -USER INTERFACE

SRS - SOFTWARE REQUIREMENT SPECIFICATION

2.5 Overview:

The project "SkillBites" is a website-based initiative aimed at facilitating instructors in offering paid online courses to students interested in learning specific subjects. The Software Requirements Specification (SRS) document serves as a comprehensive guide outlining all the necessary requirements for the successful development and implementation of this project.

2.6 Overall Description:

This section will give an overview of the whole system. It will also describe the various users who will use the system and what type of functionality is available for those users. The general, functional, performance and security requirements will be presented.

2.7 Product Perspective

The "SkillBites" project envisions the creation and maintenance of an inclusive online learning platform. The users of the system are categorized into three categories which are admin, instructor and student. Every user should be given an online connection. The platform is designed with a user-friendly interface to ensure ease of navigation for both instructors and students, promoting a seamless learning experience.

2.8 Product Functions:

- The project “SkillBites” is developed to provide an online platform for admin, student and Instructors where the admin can manage the other two users, that is the student and instructor.
- The product first asks if the user wants to join in as a student or an instructor.
- Internet connection is mandatory.
- The system enables the instructor to create course and waits for the approval of the admin to add it to the course lists.
- It enables the instructors to post study materials, assignments and quizzes.
- Students can access the study materials and other assessments posted by the instructors of particular course.
- Students can post their queries on recorded videos or to clear any doubts.
- Instructors can get 90% of income while the remaining goes to the admin.

2.9 Browser That Supports:

- Google chrome
- Microsoft edge
- Internet explorer
- Morzilla
- Firefox

2.10 User classes and characteristics:

The system has three levels of users.

Admin:

- Login to admin platform
- Admin can manage the students
- It manages the instructor
- Manage the courses
- Manage feedbacks on the website
- Manage reports on courses students and instructor

Instructor:

- Instructor registration and login
- They can also manage their profile
- Create courses by entering details of the specific course
- Upload assignments
- They can manage attendance of each student
- They can upload video recordings
- Upload study materials
- They can reply to the queries posted by the students
- They can view the student's assignments

Student:

- Student registration and login
- They can manage their profile
- View all courses
- Purchase specified courses
- View instructor created recordings by either downloading or watching online
- They can ask queries to instructor

2.11 General Constraints:

- The web application requires an internet connection
- Users must register themselves
- Nobody has the right to change information of someone else account

2.12 Assumptions and Dependencies:

- It is assumed that there are three levels of users, i.e., admin, student and instructor.
- All the data entered will be correct and up to date.
- The system assumes the implementation of security measures to protect user data, such as login credentials and personal information, from unauthorized access or breaches.

2.13 Specific Requirements:

User Interface:

- Textboxes to enter information.
- Buttons to create, add, delete, update and search
- Labels to display the information
- Checkboxes and radio buttons
- Dropdowns

Software Requirement:

- Operating system - Windows
- Text Editor - Visual studio code
- Language - NodeJS
- UI - HTML, CSS, JS, ReactJS
- Database -MongoDB
- Server -Nodejs server
- Browser -Chrome, Microsoft edge,Mozilla firefox

Hardware Requirements:

- RAM -4gb
- Processor -Intel core i3 or above
- Hard disk -minimum 64gb

2.14 Communication Interface:

This is a web-based system and communication is done through the internet and internet protocols and http protocols.

2.15 Functional Requirements:

Admin:

Login module: Admin can enter the website using his unique email-id and password

Dashboard: Admin can view the number of students and instructors registered in the system.

Manage student: Admin manages the students who have registered to the system.

Manage Faculties: Admin manages the instructors who have registered to the system.

Manage Feedbacks: Admin can manage feedbacks by student or instructors.

Manage Payments: Admin can manage payment related activities.

Student:

Student Registration and login: Students can enroll themselves to the system as a student with username, email-address, password.

Manage Profile: Students can manage their profile within the system. This includes the ability to update and modify their personal information.

View Courses: Students can access a list of available courses in the system.

View bought course: Students can view a list of their acquired courses for reference and tracking.

Assignments: Students can likely access and complete assignments related to their enrolled courses.

Lectures: Students can access a list of recorings provided by the insrtuctor.

Queries: Students can ask questions or seek clarification on course-related matters.

Feedbacks: Students can provide feedback on the courses or the system itself.

Instructors:

Instructor registration and login: Instructors have the ability to register and log in to the system as an instructor.

Manage Course: Instructors can manage details related to the courses they are responsible for tasks such as creating new courses, updating course content, viewing student performance and much more.

Add Assignments: Instructors have the capability to create and add assignments to the courses they are teaching.

Add recordings: They can also add video recordings and post links for more information.

Provide marks: Instructors can provide marks for students who submits the assignments.

Reply to Feedbacks: Instructors can respond to feedback provided by students, addressing comments, concerns, or suggestions submitted by students in the form of feedback.

2.16 Performance Requirement:

The server should be able to support an indefinite number of active students and courses without compromising performance or losing data. Ensures that users, including students and instructors, can quickly access and navigate the platform without extended delays.

2.17 Design constraints:

- All the inputs should be checked and validated.
- Details given by the user during the registration should be stored in the database.
- Details entered by the admin and user should be stored in database.
- Alert the users for incorrect data given to an input field.

2.18 System Attributes:

- Performance: The system executes tasks and processes data, ensuring optimal response times and resource utilization.
- Reliability: A reliable system ensures that users can depend on its functionality, minimizing errors, crashes, and disruptions.
- User-Friendly Interfaces: The design and usability of the system must be intuitive, easy to navigate, and user-friendly.
- Maintainability: The system can be maintained, updated, and modified over time.
- Portability: The system must be easily transferred or adapted to different environments or platforms.
- Flexibility: The system must accommodate changes, updates, or modifications without significant disruption.
- Timeliness: The system must meet the specified deadlines and response time requirements.

2.19 Safety and Security Requirement:

Admin has all rights to the system. Email id should be unique around the system and cannot be used twice for different users. Only the authenticated users can access the system. The server on which the project resides will have security to prevent unauthorized access.

3.SYSTEM DESIGN

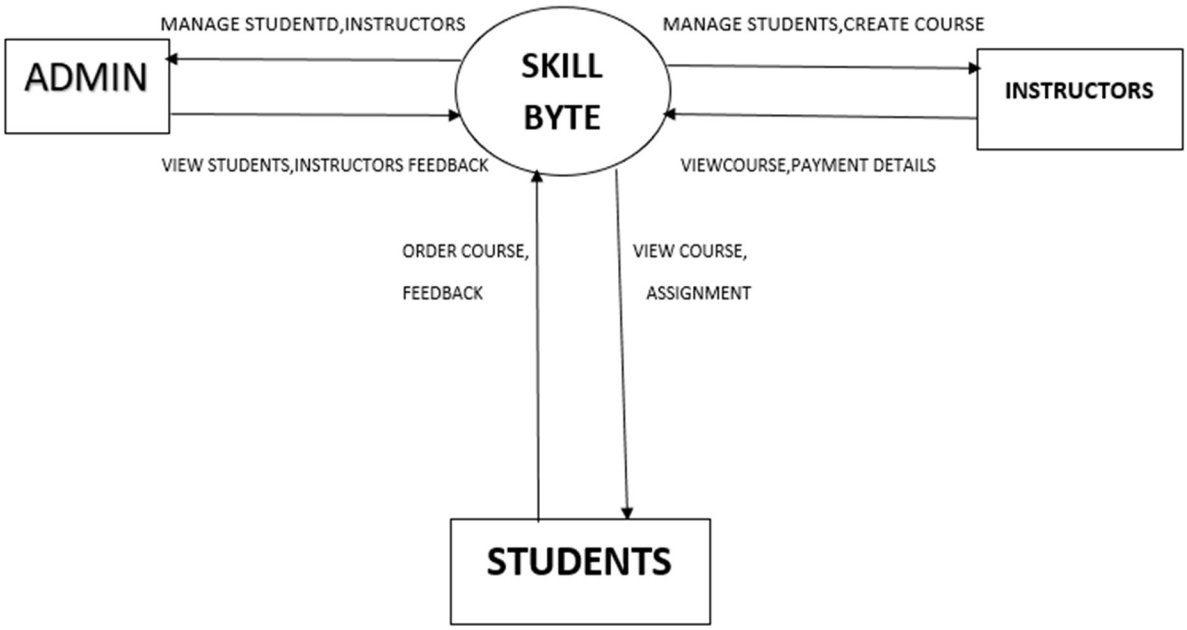
3.1 Introduction

The goal of the design process is to produce a model or representation of a system, which can be used later to build the system. System design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. The focus of system design is on deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected.

This crucial process translates user needs into a blueprint, a model that guides the system's construction. Systems design involves defining the system's architecture, its building blocks (modules), the communication channels (interfaces) between them, and the data it handles. The core focus is on identifying the necessary modules, their detailed functionalities, and how they seamlessly interact to achieve the desired outcome.

3.2 Context Flow Diagram

A context flow diagram is a top-level data flow diagram. It only contains one process node that generalizes the function of the entire system in relationship to external entities. In the context diagram, the entire system is treated as a single process and all its inputs, outputs, sinks, and sources are identified and shown.



3.3 Data flow diagram:

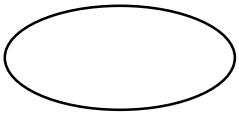
Data Flow Diagrams (DFDs) are like X-rays for systems, revealing the inner workings of data movement. These visual roadmaps depict how information flows through a system, from external sources and destinations (think entry and exit points) to internal processing hubs. Using a set of standardized symbols, DFDs map the transformations of data as it progresses through various stages. This graphical technique, also known as bubble charts or data flow graphs, offers a powerful tool for understanding systems at any level of complexity. By decomposing a system into progressively detailed layers, DFDs provide a clear view of information flow and pinpoint areas for optimization.

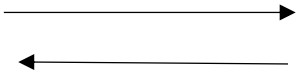
DFD is also known as bubble charts or Data Flow Graphs. DFD may be used to represent the system at any level of abstraction. DFDs may partition into a level that represents increasing information flows and functional details.

3.4 Rules Regarding DFD Construction:

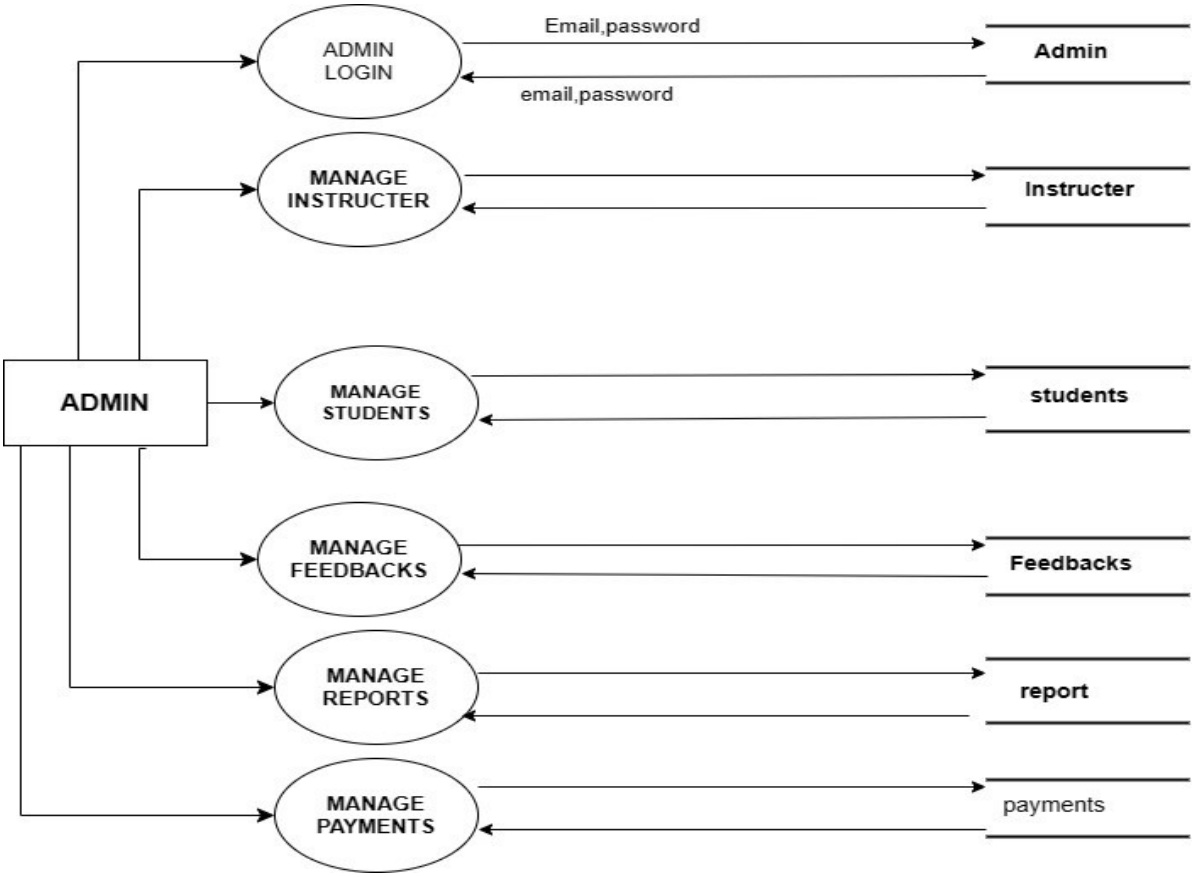
- A process cannot have only outputs.
- A process cannot have only inputs.
- The inputs to a process must be sufficient to produce the outputs from the process.
- All data stores must be connected to at least one process.
- All data stores must be connected to a source or sink.
- A data flow can have only one direction of flow. Multiple data flows to and/or from the same process and the datastore must be shown by separate arrows.
- If the same data flows to two separate arrows, it should be represented by a forked arrow.
- Data cannot flow directly back into the process it has just left. All data flows must be named using a noun phrase.

3.5 DFD Symbols:

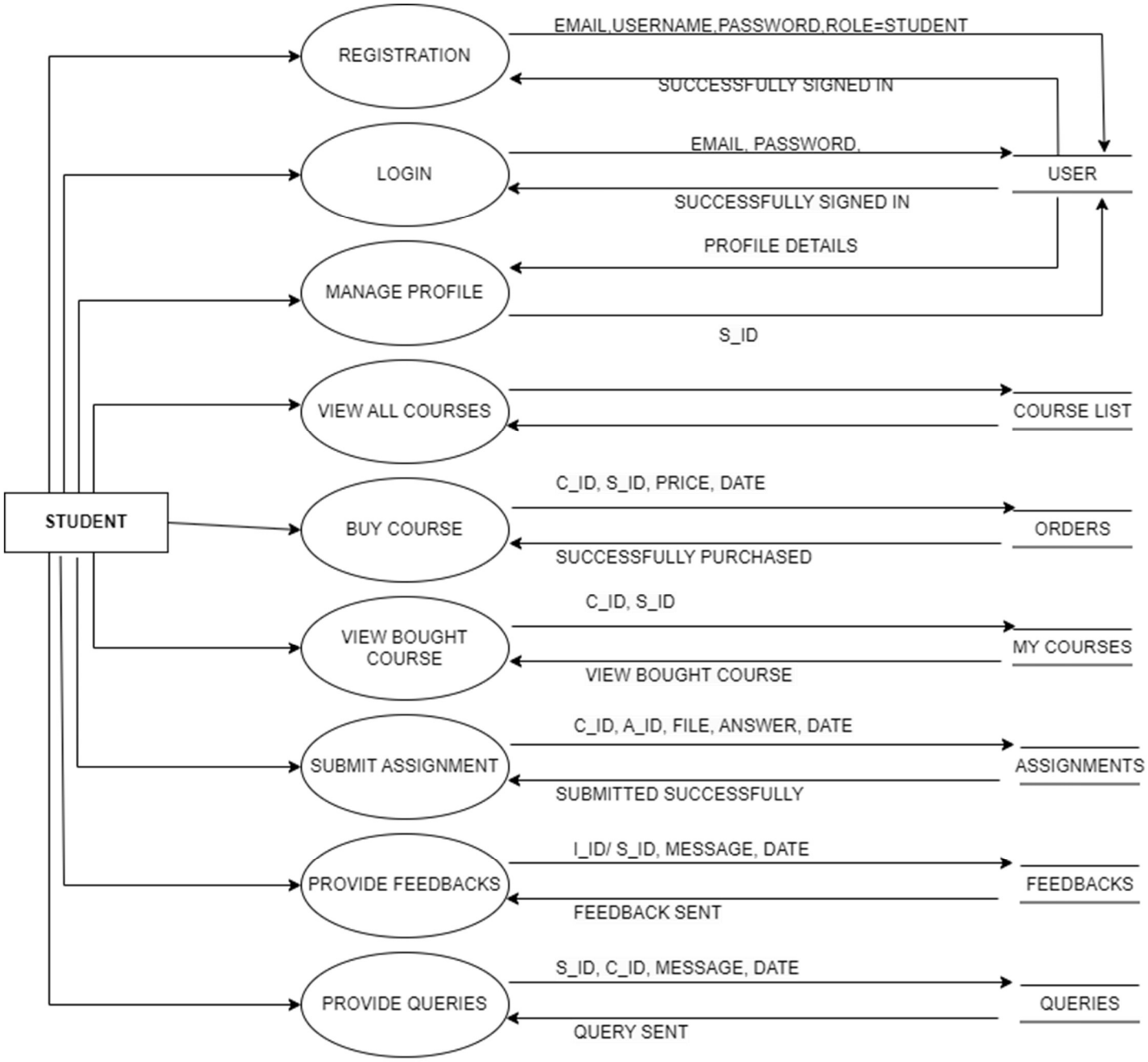
Name	Notation	Description
Process		A process transforms incoming data flow into outgoing data flow. The processes are shown by named circles.
Datastore		Datastores are repositories of data in the system. They are sometimes also referred to as files.

Dataflows		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through them.
External entity		External entities are objects outside the system with which the system communicates. External Entities are sources and destinations of the system's inputs and outputs

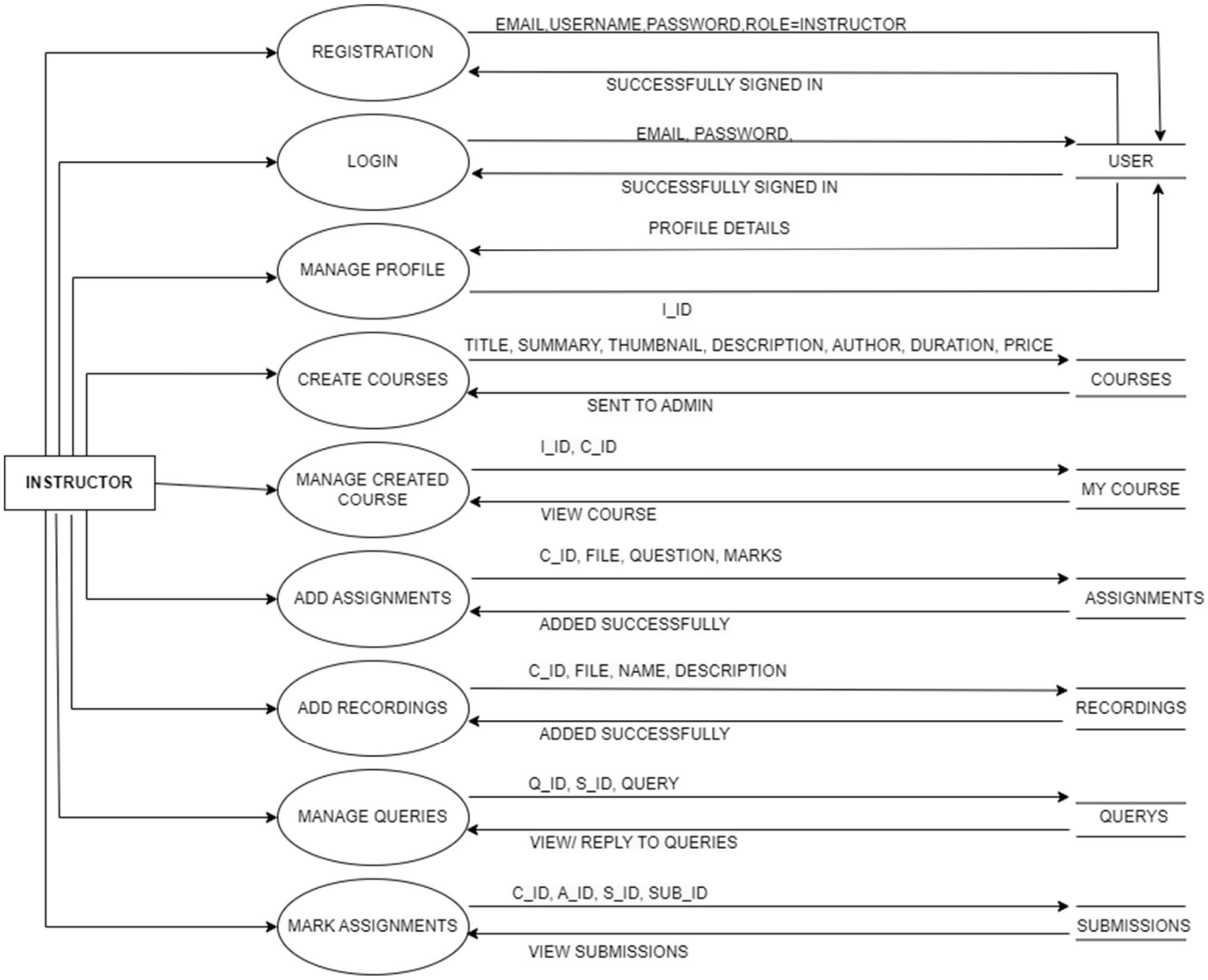
3.6 DFD LEVEL 1 : ADMIN



3.7 DFD LEVEL 1 : STUDENT



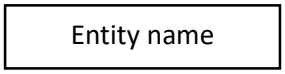
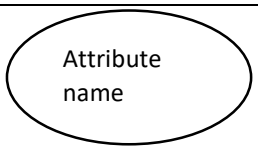
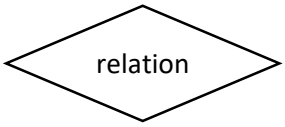
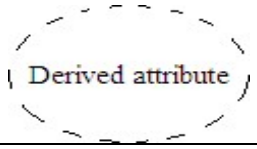
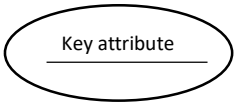
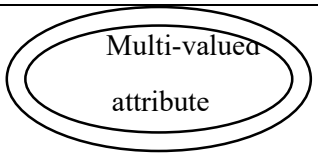
3.8 DFD LEVEL 1 : INSTRUCTER



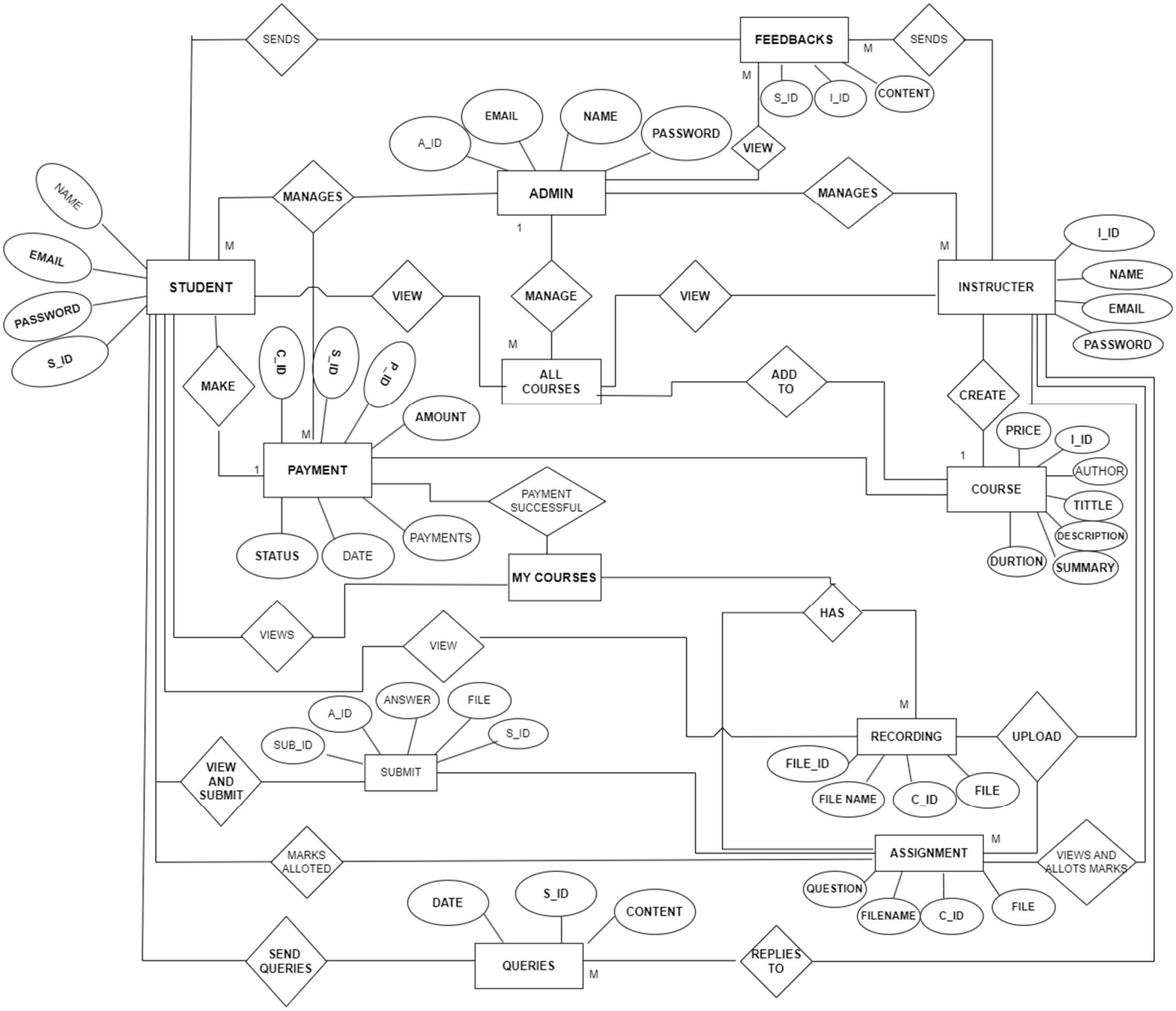
3.9 Entity-Relationship Diagram:

The basic objective of the ER model representation is an entity that is a "thing" in the real world with an independent existence. Entities are physical items or aggregations of data items that are important to the business we analyze or to the system; we intend to build. An entity represents an object defined within the information system about which you want to store information. Entities are named as singular nouns and are shown in rectangles in an ER-Diagram. Each entity is described by several attributes; individual instances of an entity will have different attribute values.

3.10 ER-Diagram Symbols:

Name	Notation	Description
Entity		It may be an object with a physical existence or conceptual existence. It is represented by a Rectangle.
Attribute		The properties of the entity can be an attribute. It is represented by an Ellipse.
Relationship		Whenever an attribute of one entity refers to another entity, some relationship exists. It is represented by a Diamond.
Link		Lines link attributes to entity sets and entity sets to relations.
Derived Attribute		A dashed ellipse denotes derived attributes
Key Attribute		An entity type usually has an attribute whose values are distinct for each entry in the entity set. It is represented by an Underlined word in an ellipse.
Multivalued Attribute		Attributes that have different numbers of values for a particular attribute. It is represented by a Double ellipse that represents multi-valued attributes.
Cardinality Ratio	1. 1:1 2. 1:M 3. M:1 4. M:M	It specifies the maximum number of relationship instances that an entity can participate in. There are four cardinality ratios.

3.11 ER DIAGRAM:



4. DATABASE DESIGN

4.1 Introduction:

The database design is the most crucial part of the SkillBites system. It involves defining the structure of the database, the types of data to be stored. It also establishes relationship between a set of data's. An effective database design ensures storage of data, retrieval and manipulation of data. The SkillBites system use a nosql database which is the MongoDb database where the data is stored in documents.

4.2 The data for "Skill Bites" is organized into 4 documents with sub documents

- Users
- Courses
  - Enrolled
  - Recordings
  - Assignments
    - submit
- Purchases
- Feedbacks

4.3 Database Structure

Structure of Users document

Field name	Type	Constraints	Description
u_id	ObjectId	required	Automatically assigns an object Id
username	String	required	Username of the user
email	String	unique	Email should be unique
password	String	required	Password
role	String	Enum['Admin','Instructor','Student']	Role of users
bio	String		Biography of user
phone	Number		Mobile number of the user
address	String		Address of the user
photo	String		Profile photo
timestamps	Date		Created and updated date

Structure of Course document

Field name	Type	Constraints	Description
c_id	ObjectId		Automatically assigns an objectId
title	String	required	Title of the course
summary	String	required	Brief summary
content	String	required	Description
cover	String	required	Thumbnail
duration	Number	required	Duration in months
price	Number	required	Total Price of the course
author	ObjectId(Users)	required	Refers the instructor who created the course
total	Number	Default:0	Total marks of course including the marks of assignments and recordings
enrolled	Array of embedded objects		List of enrolled students
approved	Boolean	Default:false	Status of approval of course from the admin
recordings	Array of embedded objects		List of recordings
assignments	Array of embedded objects		List of assignments
timestamps	Date		Created and updated date

Embedded Object: enrolled

Field name	Type	Constraints	Description
enroll_id	ObjectId		Automatically assigns an object Id
student	ObjectId (Users)	required	Refers the student id of the enrolled student
totalMarks	Number	Default:0	Total marks of student in a course

Embedded Object: recordings

Field name	Type	Constraints	Description
Rec_id	ObjectId		Automatically assigns an object Id
file	String	required	Lecture file
filename	String	required	Title of the lecture
description	String	required	Lecture details and also provide links
watched	Array of ObjectId(Users)		Contains student id of students who watched the lecture
createdAt	Date	Default:Date.now()	Created date

Embedded Object: Assignments

Field name	Type	Constraints	Description
Assign_id	ObjectId		Automatically assigns an object Id
title	String	required	Title of the assignment
Question	String	required	Question of the assignment
file	String	-	File attached to the assignment
marks	Number		Marks of the assignment
createdAt	Date	Default:Date.now()	Created date
submit	Array of embedded Objects		List of submissions

Embedded Object: submit

Field name	Type	Constraints	Description
Submit_id	ObjectId		Automatically assigns an object Id
student	ObjectId(Users)	required	Refer the student id of the student who submitted the assignment
file	String	-	File attached to the submission
description	String	-	Answer of the student
marks	Number		Marks allotted to the student if status is submitted
status	String	Enum[‘submitted’,’pending’]	Created date
submittedAt	Date	Default:Now	Submitted date

Structure of Purchases document

Field name	Type	Constraints	Description
Purchase_id	ObjectId		Automatically assigns an object Id
userId	ObjectId(Users)	required	Refer the student id of the student who bought the course
courseId	ObjectId(Courses)	required	Refer the course id the particular student bought
paymentId	String	required	Payment id

Structure of Feedbacks document

Field name	Type	Constraints	Description
f_id	ObjectId		Automatically assigns an object Id
name	ObjectId(User)	required	Refer the id of the user who sends the feedback
email	String	required	Email of the user
message	String	required	Message contents
timestamps			Sent date

5. Detailed Design

5.1 Introduction:

The purpose of preparing this document is to explain the complete design in detail of the Online classroom management system. This detailed design report will mainly contain the general definition and features of the project, design constraints, the overall system architecture, and data architecture. Additionally, a brief explanation of our current progress and schedule of the project will be provided in related sections. The design of the system and subsystem modules will be explained both verbally and visually using diagrams to help the programmer to understand all information stated in this document correctly and easily.

5.2 Application documents:

The documents used during the detailed design are :

- System Requirements Document
- System Design
- Database Design

5.3 Structure of the software package

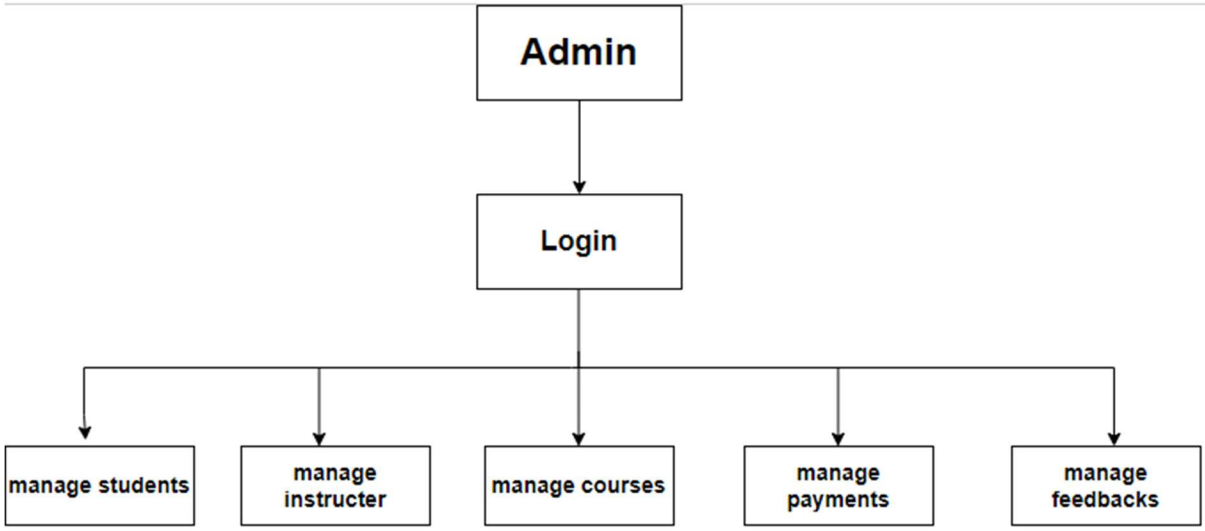
The components are:

- Registration component for Student and Instructer.
- Login component for Admin, Instructer and Student.
- Course component for Instructor, Student and Admin.
- Purchase component for Student.
- Assignment and recording components for Instructor and students.

5.4 Modular Decomposition of Components

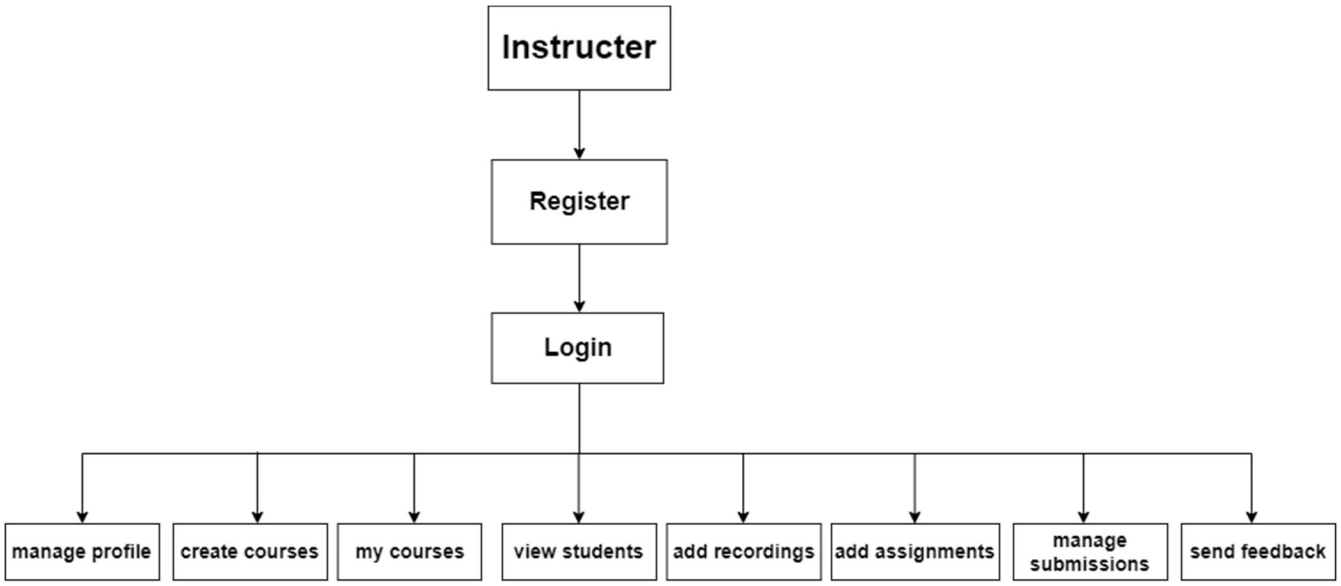
5.4.1 Admin Component:

Structure chart for Admin.



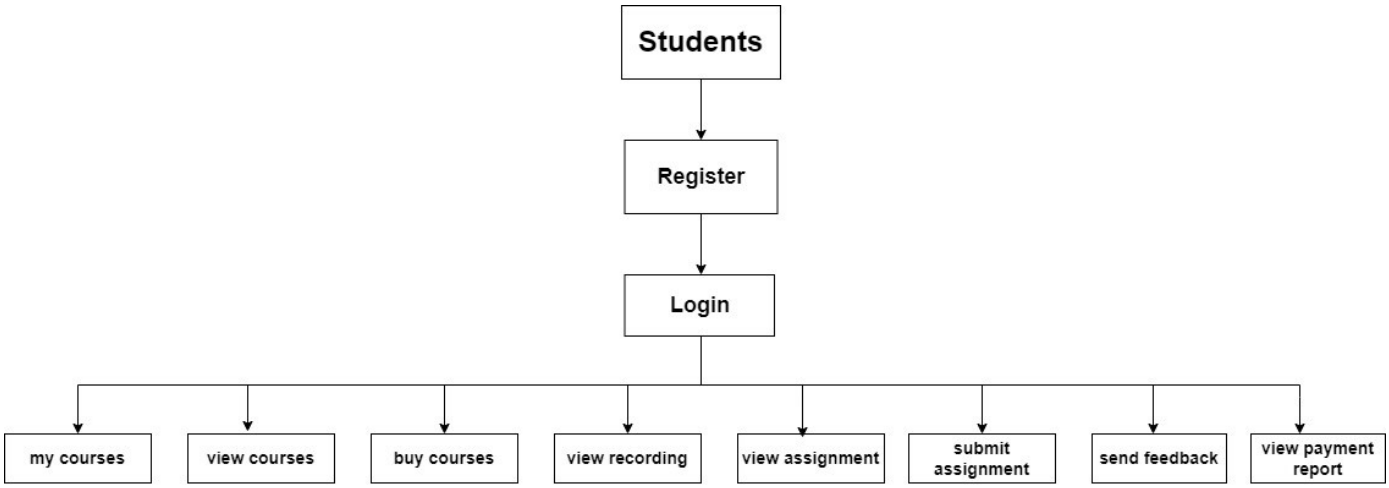
5.4.2 Instructor Component:

Structure chart for Instructor.



5.4.3 Student Component:

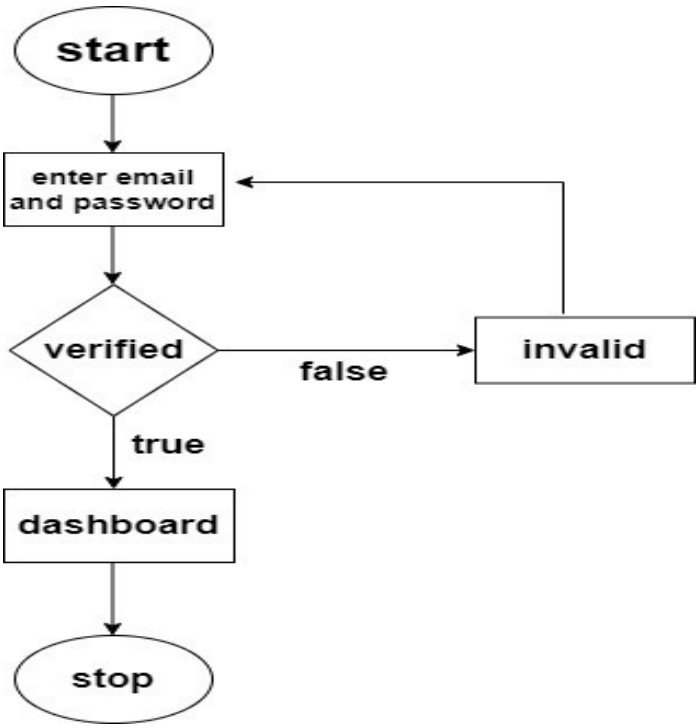
Structure chart for Student.



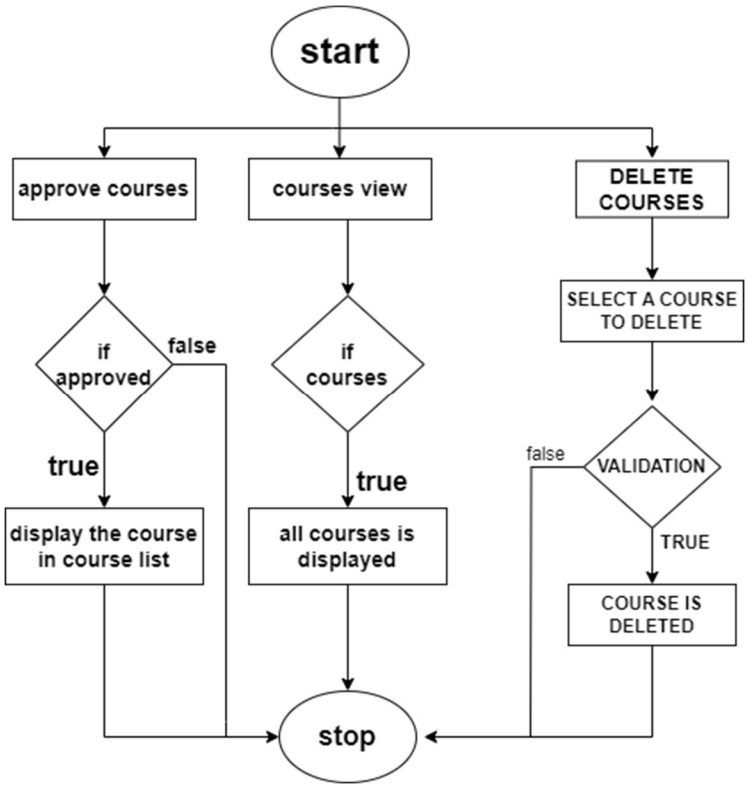
5.4.4 Procedural details for Admin

Login:

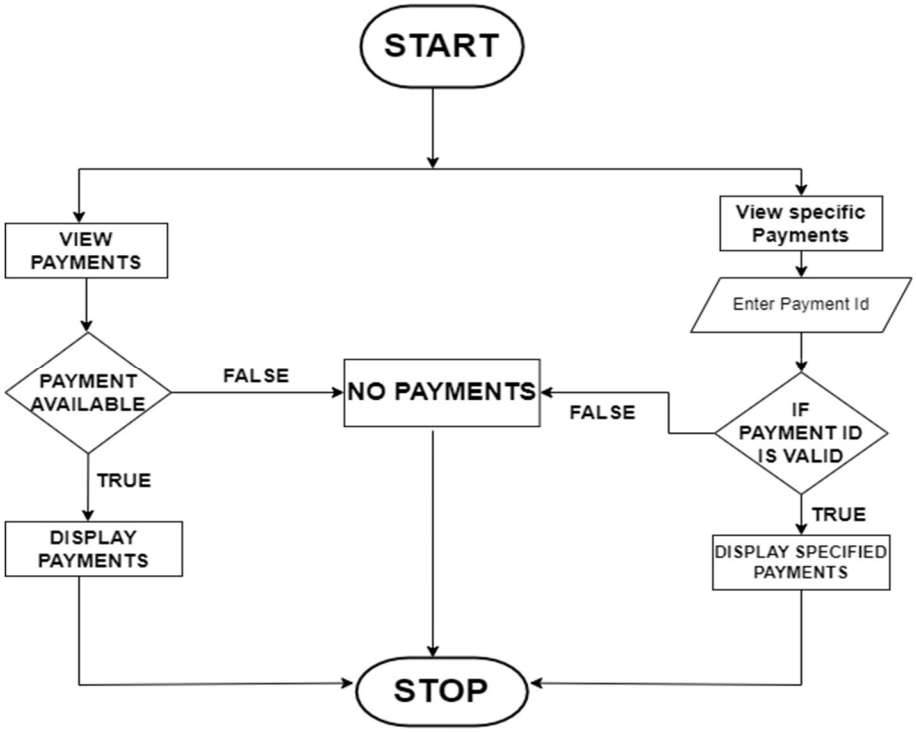
Input:Email,password



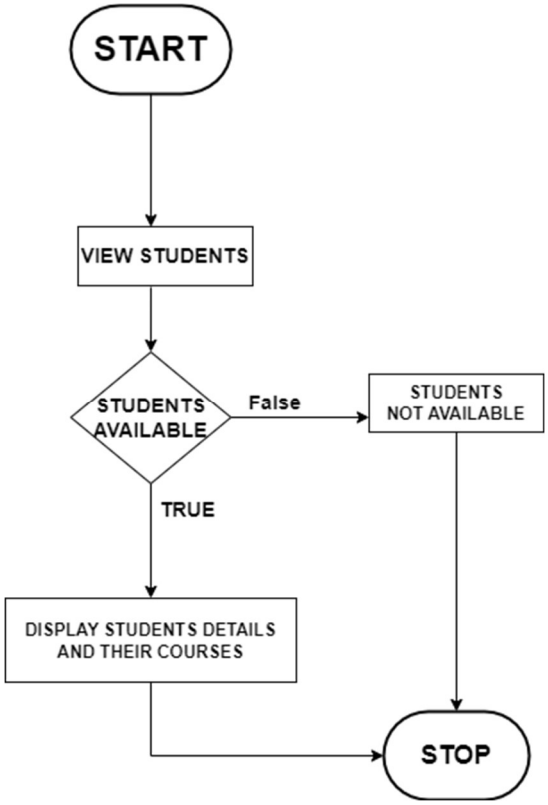
Manage courses:



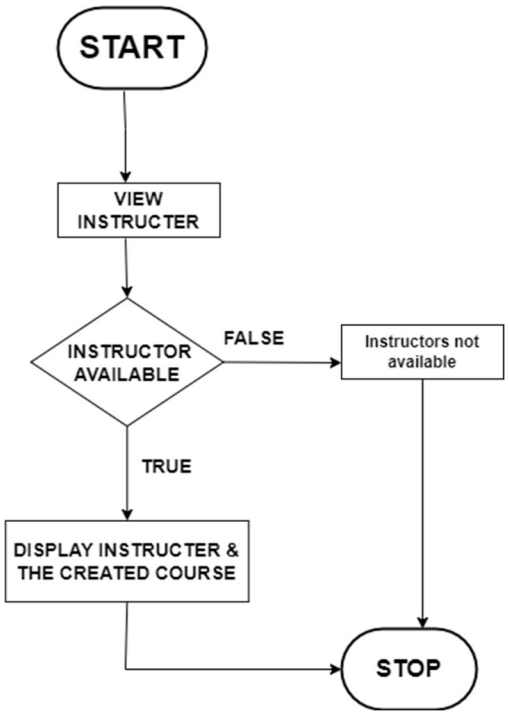
Manage Payments:



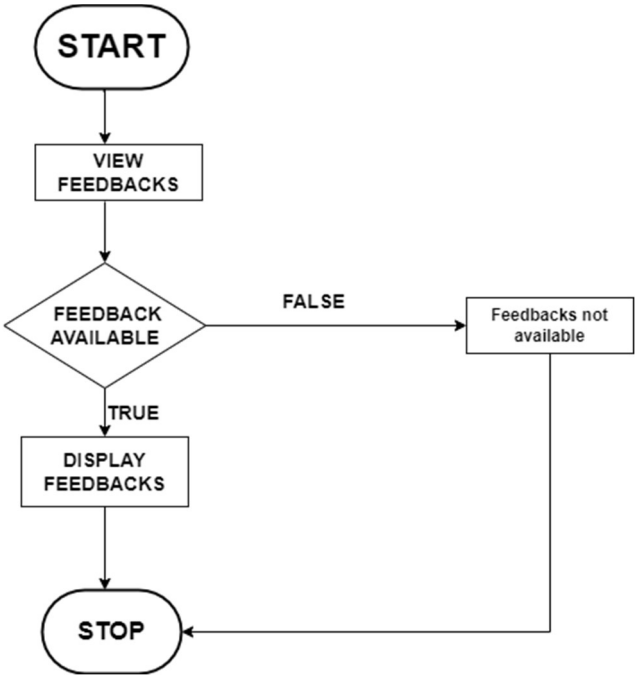
Manage Students:



Manage Instructors:

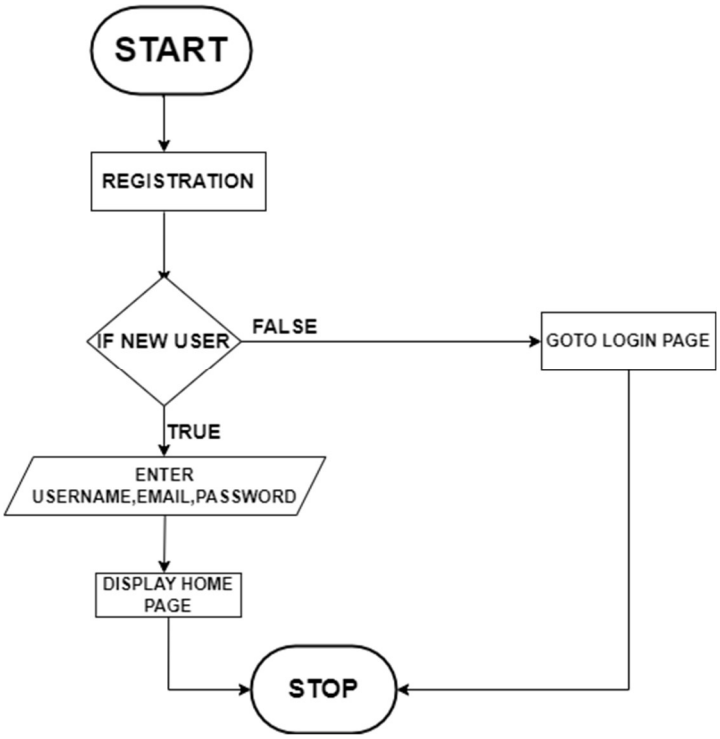


View Feedbacks:

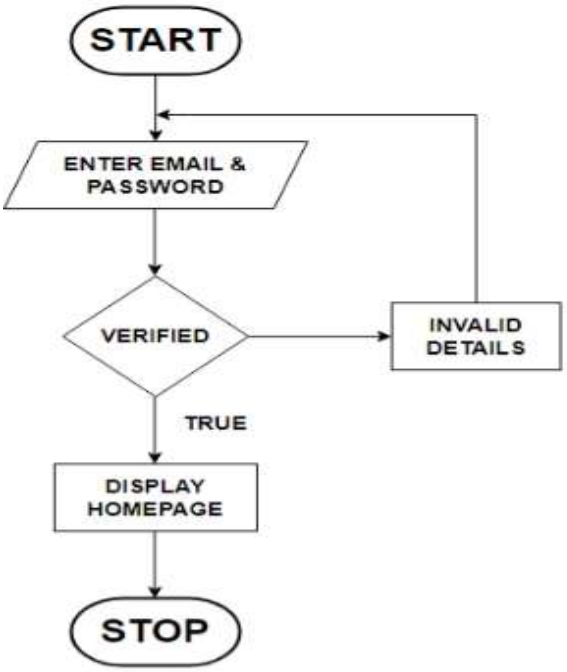


5.4.5 PROCEDURAL DETAILS OF INSTRUCTORS:

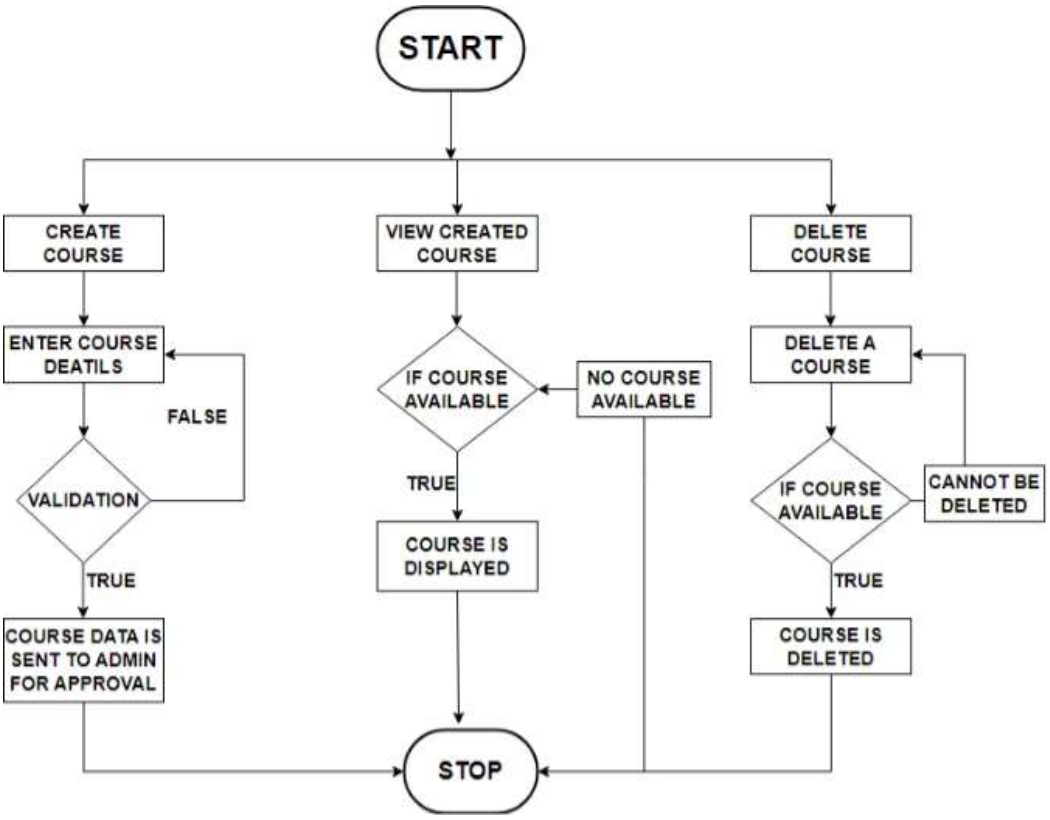
Registration:



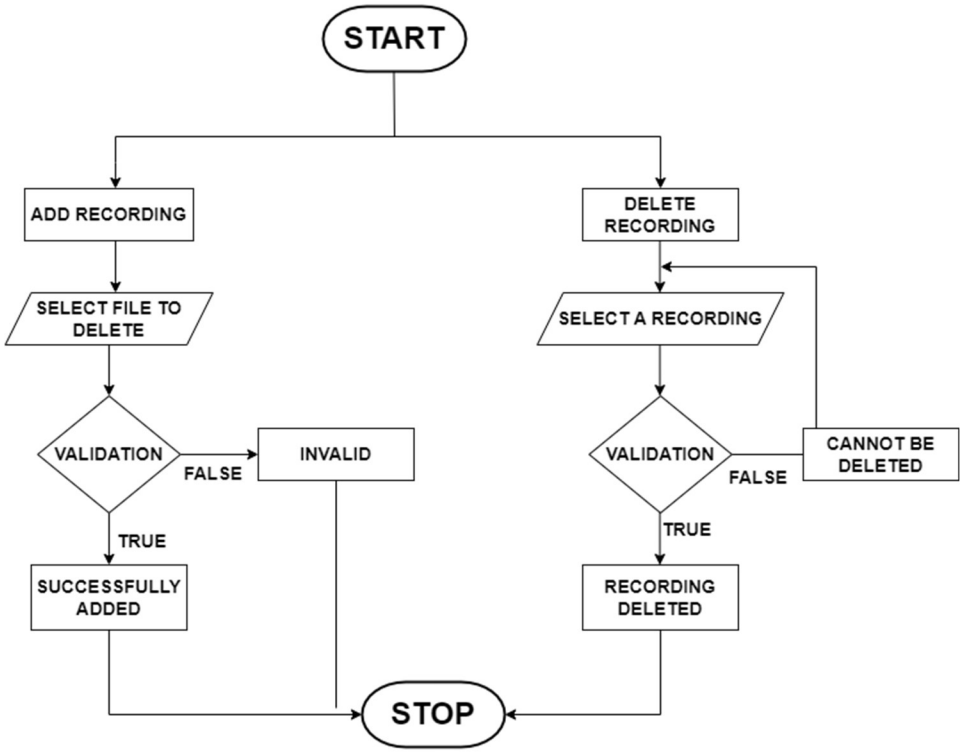
Login:



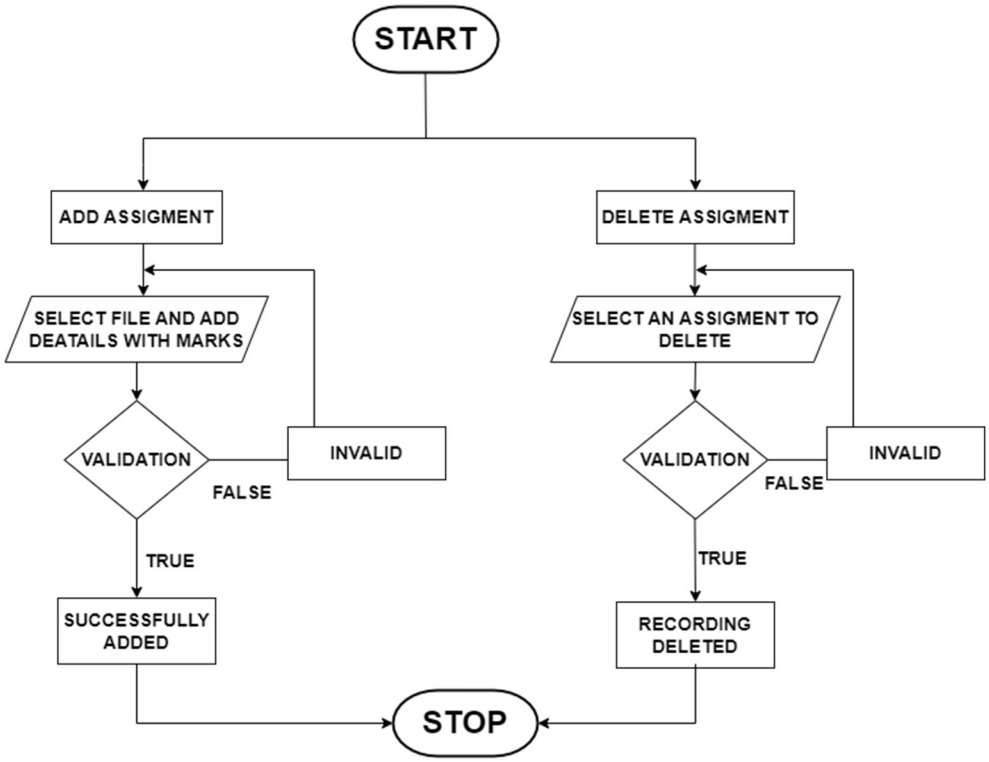
Manage Course:



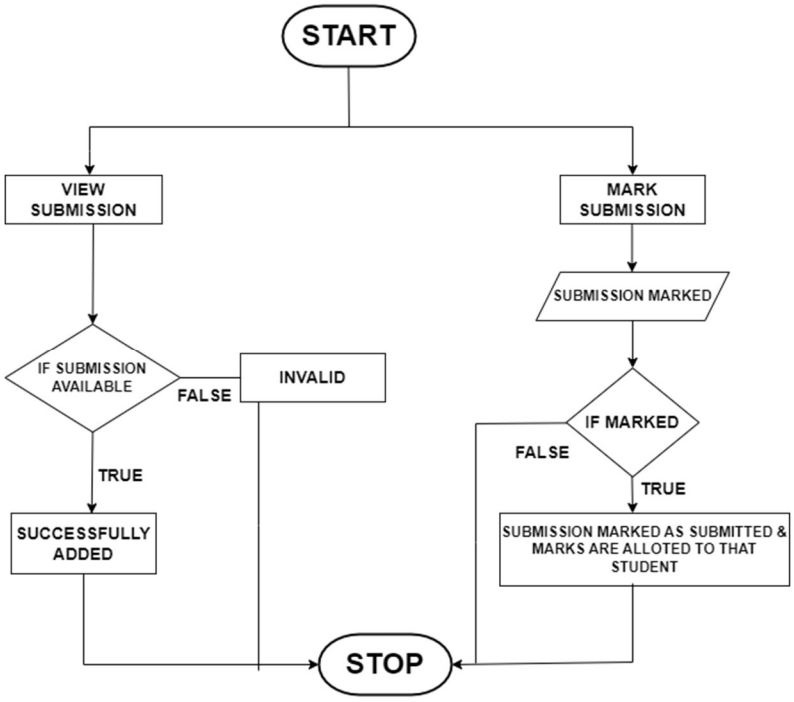
Manage Course Recordings:



Manage Assignments:

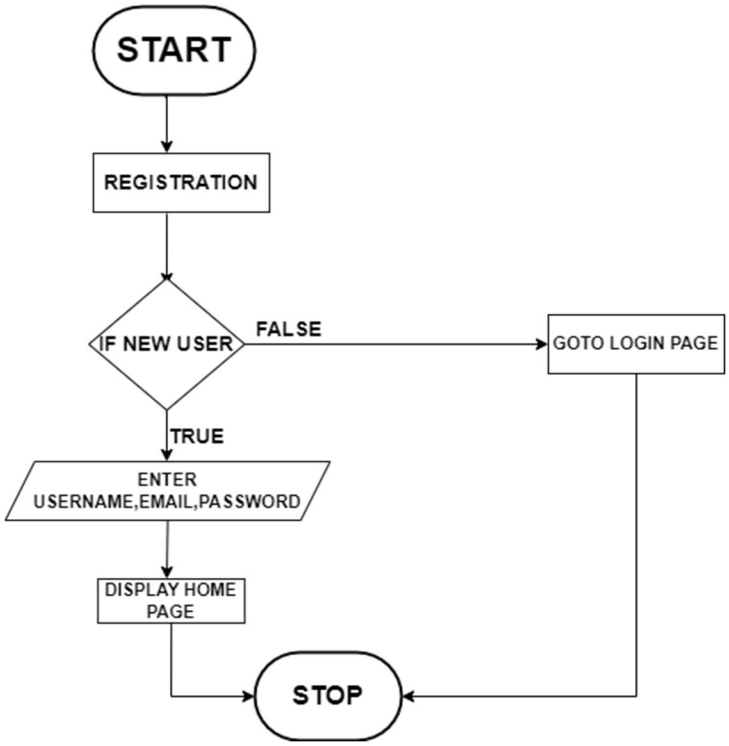


Manage Submissions:

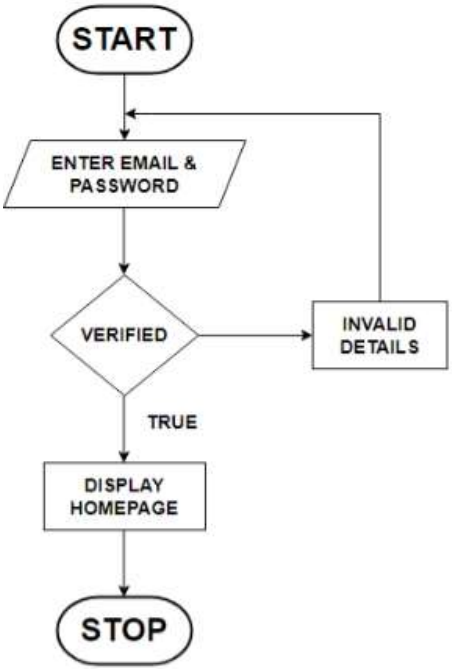


5.4.6 PROCEDURAL DETAILS OF STUDENTS:

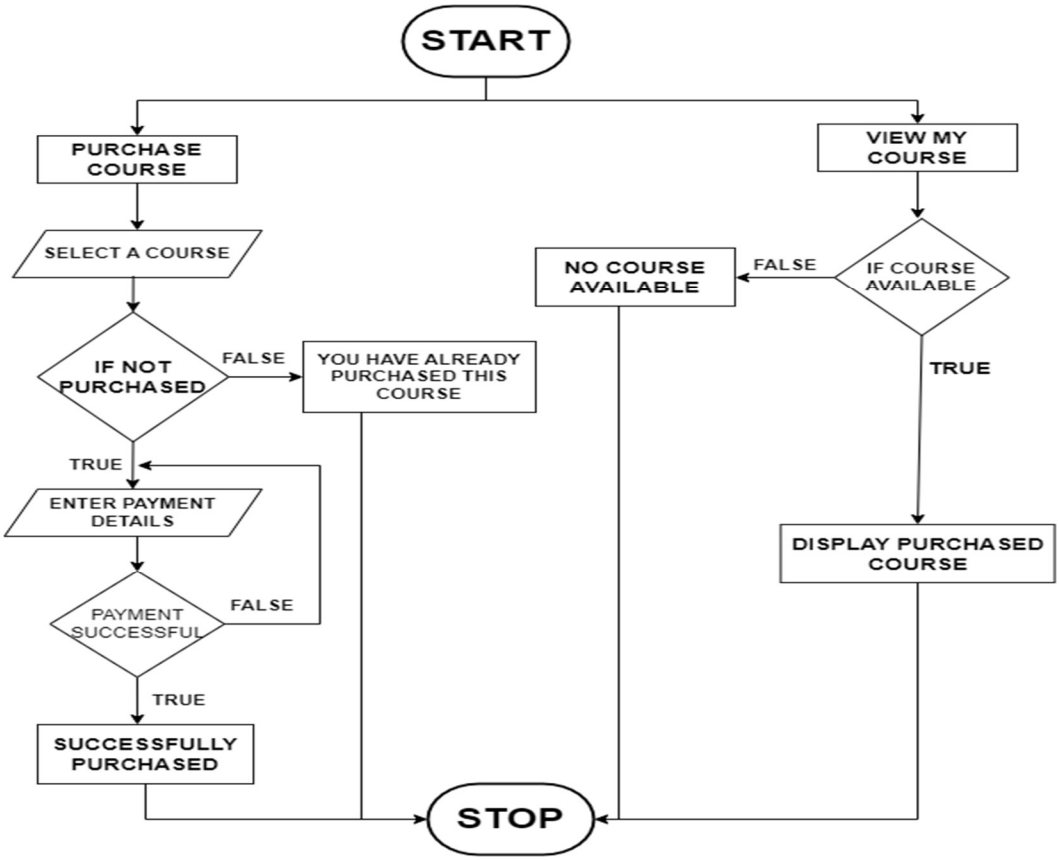
REGISTRATION:



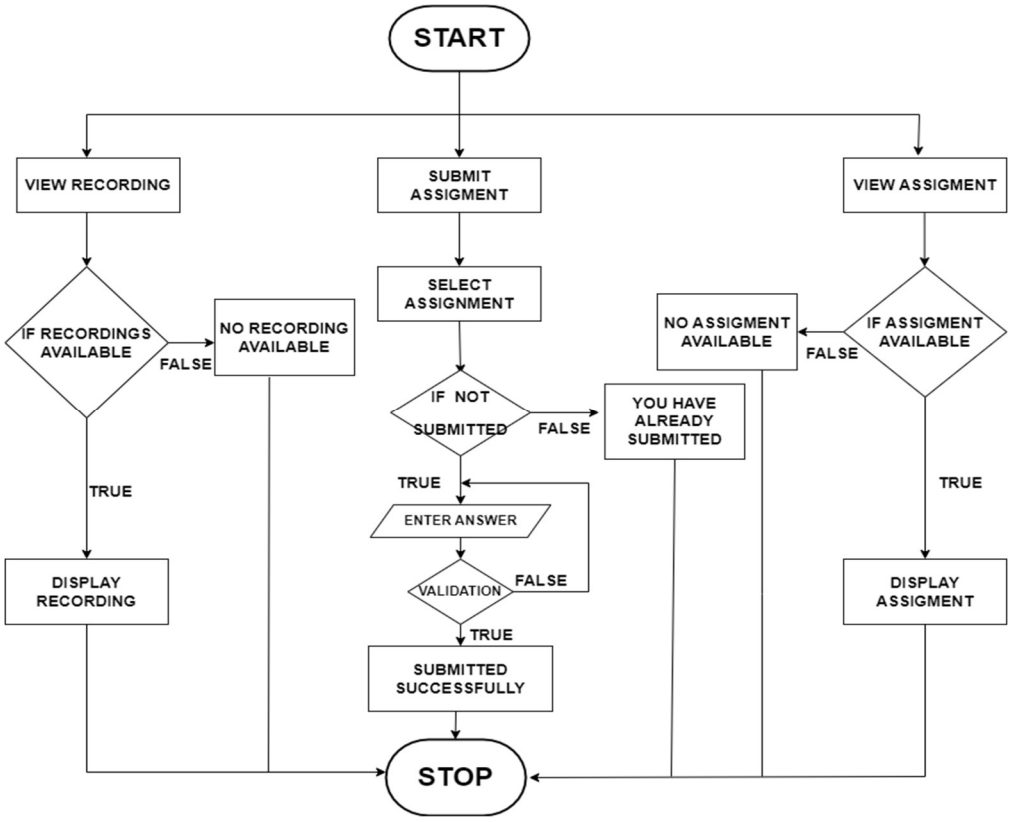
LOGIN:



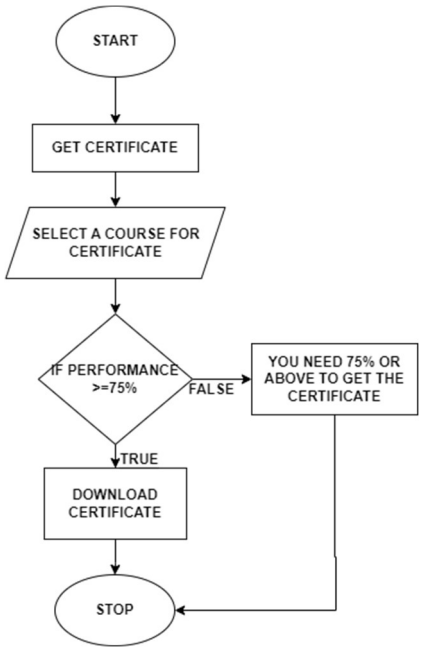
Make Payments and view Purchased Course:



View Recordings, Assignments and Submit Assignment



Get Certificate



6. Coding

Main component

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description"content="Web site created using create-react-app"/>
    <!-- adding logo -->
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!-- bootstrap link -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"integrity="sha384EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3y
D65VohhpuuCOmLASjC" crossorigin="anonymous">
    <!-- google font links -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Angkor&display=swap"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Hind:wght@300;400;500;600;700&display=sw
ap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Revalia&display=swap"
rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Angkor&family=Teko:wght@300..700&display
=swap" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=Gabarito:wght@400..900&display=swap"
rel="stylesheet">
    <title>SkillBites</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
```

```
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
   <!-- font awesome scripts -->
   <script src="https://kit.fontawesome.com/7d95e62f4c.js"crossorigin="anonymous"></script>
   <!-- razorpay checkout link -->
   <script src="https://checkout.razorpay.com/v1/checkout.js"></script>
  </body>
</html>

// import all the components
import { Routes, Route } from "react-router-dom";
import CourseList from "./Pages/CourseList";
import Hero from "./Components/Hero/Hero";
import Login from "./Pages/Login";
import Register from "./Pages/Register";
import { UserContextProvider } from "./UserContext";
import Contact from "./Pages/Admin/Contact";
import CreateCourse from "./Pages/Instructor/CreateCourse";
import CourseDescription from "./Pages/Instructor/CourseDescription";
import Profile from "./Pages/Profile";
import ViewCourse from "./Pages/Student/ViewCourse";
import EditProfile from "./Pages/EditProfile";
import MyCourse from "./Pages/Student/MyCourse";
import AddLectures from "./Pages/Instructor/AddLectures";
import PdfViewer from "./Components/PdfViewer";
import AdminDashboard from "./Pages/Admin/AdminDashboard";
import AdminCourse from "./Pages/Admin/AdminCourse";
import NotFound from './Pages/NotFound';
import Feedbacks from "./Pages/Admin/Feedbacks";
import AdminStudent from "./Pages/Admin/AdminStudent";
import AdminInstructors from "./Pages/Admin/AdminInstructors";
import AdminPayment from "./Pages/Admin/AdminPayment";
import CheckoutSuccess from "./Pages/Student/CheckoutSuccess";
import Orders from "./Pages/Student/Orders";
function App() {
  return (
    <>
    <UserContextProvider>
          //Defining routes
    <Routes>
       <Route path="/" element={<Hero />}>  </Route>
       <Route path="/contact" element={<Contact/>}/>
       <Route path="/course" element={<CourseList />} />
```

```
        <Route path="/Login" element={<Login />}></Route>
        <Route path="/Signup" element={<Register />}></Route>
        <Route path='/profile'element={<Profile/>}></Route>
        <Route path='/profile/edit' element={<EditProfile />}></Route>
        <Route path="/create" element={<CreateCourse />}></Route>
        <Route path="/course/:id" element={<CourseDescription/>}></Route>
        <Route path="/myCourse" element={<MyCourse/>}></Route>
        <Route path="/myCourse/view/:id" element={<ViewCourse/>}></Route>
        <Route path="/file-viewer" element={<PdfViewer/>}></Route>
        <Route path="/Admin/Dashboard" element={<AdminDashboard/>}></Route>
        <Route path="/Admin/courses" element={<AdminCourse/>}></Route>
        <Route path="/Admin/feedbacks" element={<Feedbacks/>}></Route>
        <Route path="/Admin/students" element={<AdminStudent/>}></Route>
        <Route path="/Admin/instructors" element={<AdminInstructors/>}></Route>
        <Route path="/Admin/payments" element={<AdminPayment/>}></Route>
        <Route path="/orders" element={<Orders/>}></Route>
        <Route path="/order-success" element={<CheckoutSuccess/>}></Route>
        <Route path="*" element={<NotFound/>} />
      </Routes>
    </UserContextProvider>
    </>
  )}
export default App;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <StrictMode>
    <BrowserRouter>
    <Provider store={store}>
    <App />
    </Provider>
    <ToastContainer/>
    </BrowserRouter>
  </StrictMode>
);
//Server
const express = require("express");
const cors = require("cors");
const bodyParser = require("body-parser");
const shortid = require("shortid");
const pdfDocument=require('pdfkit');

const mongoose = require("mongoose");
```

```
const User = require("./Models/user.js");
const Course = require("./Models/Course.js");
const Purchase = require("./Models/Orders.js");
const Contact=require("./Models/Contact.js");

const app = express();
const crypto = require("crypto");
const bcrypt = require("bcryptjs");
const cookieParser = require("cookie-parser");
const multer = require("multer");
const uploadFile = multer({ dest: "pdf/" });

const upload = require("./Middleware/multer.js");
const fs = require("fs");
const cloudinary = require("./cloudinary.js");

const { Token, verifyToken } = require("./Middleware/authMid.js");
const path = require("path");
const { config } = require("dotenv");
config({ path: "./.env" });

const razorpay = require("./Razorypay.js");
const AppError = require("./Middleware/error.js");
const ErrorHandler = require("./Middleware/errorHandler.js");
app.use(cors({ credentials: true, origin: "http://localhost:3000" }));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(bodyParser.json({ limit: "50mb" }));
app.use(bodyParser.urlencoded({ limit: "50mb", extended: true }));
app.use("/pdf", express.static(__dirname + "/pdf"));

const connect = mongoose.connect(
  "mongodb://tanmay:azzKA5F0PM0S5rYv@ac-bsnirje-shard-00-
00.9ugir26.mongodb.net:27017,ac-bsnirje-shard-00-01.9ugir26.mongodb.net:27017,ac-bsnirje-
shard-00-02.9ugir26.mongodb.net:27017/?ssl=true&replicaSet=atlas-1gtfzc-shard-
0&authSource=admin&retryWrites=true&w=majority&appName=Cluster0"
);
if (!connect) {
  console.log("Connection failed");
}
```

Layout

```
import {  NavLink } from "react-router-dom";
import Navbar from "../Navbar/Navbar";
import Footer from "../Footer";
import "./Layout.css";
import { AiOutlineClose } from "react-icons/ai";
import { IoCreateSharp } from "react-icons/io5";
import { IoMdPerson} from "react-icons/io";
import { useState, useEffect, useContext  } from "react";
import { Data, StudentData, TeachData } from "../../Extras/SidebarData";
import { UserContext } from "../../UserContext";
export default function Layout({ children ,hideFooter,initial=false,index=true}) {
  const [sidebar, setSidebar] = useState(initial);
  const show = () => setSidebar(!sidebar);
  const [img,setImg]=useState();
  const sidebarWidth= sidebar ? "16.4%" : "0"
const { userInfo } = useContext(UserContext);
useEffect(() => {
fetch("http://localhost:4000/profile", {
credentials: "include",
}).then((response) => {
response.json().then((data) => setImg(data));
});},[])
const role = userInfo?.role;
if (!img) return "";
return (
<>
<Navbar onIcon={show} />
<nav className={sidebar ? "nav-menu active" : "nav-menu"}
style={{ top: 0,height:index?'100vh':'68vh'}}>
<ul className="nav-menu-items" onClick={show}>
<NavLink className="menu-bar-close">
<AiOutlineClose />
</NavLink>
{role === "Admin" && (
<>
<h4 className="text-white mt-4 "style={{textAlign:'center'}}>Admin Dashboard</h4>
{Data.map((val, key) => (
<li className= "row1" key={key}>
<NavLink to={val.Link}>
{val.icon}
<span>{val.title}</span>
</NavLink>
```

```
</li>
))}</>)}
{role === "Student" && (
<>
<h4 className="text-white mt-4 "style={{textAlign:'center'}}>Student Dashboard</h4>
<div className="pro">
<img src={img.photo}width="100px" alt='profile' height='100px'style={{borderRadius:'50%',
marginLeft:'40px'}}></img>
<p style={{color:'white',marginTop:'10px',textAlign:'center'}}><IoMdPerson
style={{marginTop:'-5px'}}/>{' '}{img.username}</p>
</div>
<hr/>
{StudentData.map((val, key) => (
<li className= "row1" key={key}>
<NavLink to={val.Link}>
{val.icon}
<span>{val.title}</span>
</NavLink>
</li>
))}</>)}
{role === "Instructor" && (
<>
<h4 className="text-white mt-4 "style={{textAlign:'center'}}>Faculty Dashboard</h4>
<div className="pro">
<img src={img.photo}width="100px" alt='profile' height='100px'style={{borderRadius:'50%',
marginLeft:'40px'}}></img>
<p style={{color:'white',marginTop:'10px',textAlign:'center'}}><IoMdPerson
style={{marginTop:'-5px'}}/>{' '}{img.username}</p>
</div>
{TeachData.map((val, key) => (
<>
<li className="row1" key={key}>
<NavLink to={val.Link}>
{val.icon}
<span>{val.title}</span>
</NavLink>
</li>
</>))}
<li className="row1" >
<NavLink to='/create'>
<IoCreateSharp/>
<span>Create course</span>
</NavLink>
```

```
</li>
<li className="row1">
<NavLink to='/myCourse'>
<IoCreateSharp/>
<span>View Course</span>
</NavLink>
</li>
</>
)}</ul>
</nav>
<div style={{width: `calc(100%-${sidebarWidth})`,marginLeft:index? sidebarWidth:'0',
transition: "350ms",}}>
//render all children components
{children}
</div>
{!hideFooter&&( <Footer
style={{
width: `calc(100%-${sidebarWidth})`,
marginLeft: sidebarWidth,
transition: "350ms",
}}
/>)}</>);
```

HomePage

```
import Layout from "../Layout/Layout";
import "./Hero.css";
import Contact from "../../Pages/Admin/Contact";
import { useContext } from "react";
import { UserContext } from "../../UserContext";
export default function Hero(props) {
const {userInfo}=useContext(UserContext)
const user=userInfo?.username;
return (
//embedding homepage inside the layout component
<Layout>
<section className="Hero" style={props.style}>
<div className=" Content text-white">
<h1 > Transforming Education, One Click at a Time</h1>
<hr className=" my-4" />
<p className="lead">
Join a community of passionate instructors and eager learners on SkillByte. Whether you're here
to teach or to learn, we've got the tools and courses to elevate your skills and knowledge</p>
<div className="btn-content">
```

```
<a className="custom-btn btn btn-primary btn-lg" href= {user ? "/course":'/Login'}
role="button">Explore Courses <i class="fa-solid fa-arrow-right"></i></a>
</div>
</div>
</section>
</Layout>
)}
```

Register page

```
import { useState,useContext } from "react";
import { Link, Navigate, redirect } from "react-router-dom";
import Layout from "../Components/Layout/Layout";
import { UserContext } from "../UserContext";
export default function Register() {
const [redirect,setRedirect]=useState(false);
const{setUserInfo}=useContext(UserContext);
const [formData, setFormData] = useState({
username: "",
password: "",
email: "",});
const changeHandler = (e) => {
setFormData({ ...formData, [e.target.name]: e.target.value }) };
const [selectedRole, setSelectedRole] = useState(null);
const handleRoleSelection = (role) => {setSelectedRole(role);};
async function register(ev) {
ev.preventDefault();
if(!formData.email||!formData.username||!formData.password){
toast.error('Fields cannot be empty');
}
try{
const response= await fetch("http://localhost:4000/Signup",{
method:'Post',
body:JSON.stringify({...formData,role:selectedRole}),
headers:{'Content-Type':'application/json'},
credentials: "include",})
if (!response.ok) {
const error = await response.json();
toast.error(error.message);
}
else {await response.json().then(userInfo=>{
setUserInfo(userInfo.token);
setRedirect(true);
```

```
toast.success('Login Successfull',{theme:'colored',position:'top-center'});
})}
} catch(err){
console.log(err.message)
}}
if(redirect){return(<Navigate to={'/'}/>)}
return (<>
<Layout>
<div className="member">
<form className="Register" onSubmit={register}>
<h1>Register</h1>
{!selectedRole&& <> <div className="role">
<p>Student
<input type="image" src="https://img.freepik.com/free-vector/man-studying-with-book-
illustration-isolated_24911-
115018.jpg?size=626&ext=jpg&ga=GA1.1.1799944211.1707725387&semt=sph"
width='150px' onClick={()=>handleRoleSelection('Student')}/>
</p>  {' '}
<p>Instructor
<input type="image" src="https://img.freepik.com/free-vector/flat-teachers-day-background_23-
2149077610.jpg?t=st=1710238212~exp=1710241812~hmac=bc128e42de8a56142af75f29916ee
1792dad21a838e47be98db0b9ad216f1b1d&w=996"
width='170px' height='150px' onClick={()=>handleRoleSelection("Instructor")}/>
</p></div>
<h5>Sign in as a student or an Intructor?</h5>
</>}
{selectedRole&&<>
<input
className="text" name="username" type="text"
placeholder="Enter name" value={formData.username}onChange={changeHandler}
/><br />
<inpu tclassName="text"name="email" type="text" placeholder="Enter email"
value={formData.email}onChange={changeHandler}/><br />
<input className="text"  name="password" type="password" placeholder="Enter password"
value={formData.password} onChange={changeHandler}/><br /><br />
<button className="Login-button">Register</button><br /><br />
<p>Already have an account? <Link to="/Login"style={{textDecoration:"none" ,
color:"red"}}>Login </Link>
</p></>}
</form>
</div>
</Layout>
</>)}
```

```
//Server
app.post("/Signup", async (req, res, next) => {
let check = await User.findOne({ email: req.body.email });
if (check) {throw new AppError('Email already exists',400}
const { username, email, password, role } = req.body;
try {
const userDoc = await User.create({username,email,password,role,});
const token = Token(userDoc);
res.cookie("token", token, {maxAge: 60 * 60 * 24 * 1000,httpOnly: true,});
res.status(200).json({ success:true, message: "Signup successful", userDoc, token });
} catch (err) {
next(err)
}});
```

Login page

```
import { Link, Navigate } from "react-router-dom";
import { useContext, useState } from "react";
import Layout from "../Components/Layout/Layout";
export default function Login() {
const [email, setEmail] = useState();
const [password, setPassword] = useState();
const [redirect, setRedirect] = useState(false);
const{setUserInfo}=useContext(UserContext);
//posting the data to server
async function login(ev) {
ev.preventDefault();
if(!email||!password){
return toast.error('Fields cannot be empty');
}
try{
const response = await fetch("http://localhost:4000/Login", {
method: "POST",
body: JSON.stringify({ email, password }),
headers: { "Content-Type": "application/json" },
credentials: "include",
});
if (!response.ok){
const error = await response.json();
toast.error(error.message);
}
else{await response.json().then(userInfo=>{
setUserInfo(userInfo);
```

```
setRedirect(true);
toast.success('Login Successfull',{theme:'colored',position:'top-center'});
})}
} catch(err){
console.log(err.message)
}}
//if login success navigate to homepage
if (redirect) {return <Navigate to={"/"} />}
return (
<Layout>
<div className="member">
<form className="Login" onSubmit={login}>
<h1>Login</h1>
<input className="text"type="email"placeholder="Enter email"value={email}
onChange={(ev) => setEmail(ev.target.value)}/><br />
<input className="text"type="password"placeholder="Enter password" value={password}
onChange={(ev) => setPassword(ev.target.value)}/><br />
<button className="Login-button">Login</button>
<br /><br />
<p>Dont have an account? <Link to="/SignUp" style={{textDecoration:"none"
,color:"red"}}>Signup now</Link></p>
</form>
</div>
</Layout>
)}
//Server
app.post("/Login", async (req, res, next) => {
try {
const { email, password } = req.body;
const userDoc = await User.findOne({ email });
if (!userDoc) {throw new AppError('User not found',404)}
const passOk = bcrypt.compareSync(password, userDoc.password);
if (passOk) {
const token = Token(userDoc);
res.cookie("token", token, {maxAge: 60 * 60 * 24 * 1000, httpOnly: true,});
res.json(token);} else {
throw new AppError('Wrong password',400)
}} catch (err) {
next(err)}});
```

Profile

```
import React, { useContext, useEffect, useState } from "react";
```

```
import Layout from "../Components/Layout/Layout";
import EditProfile from "./EditProfile";

const Profile = () => {
const [openModal, setOpenModal] = useState(false);
const [user, setUser] = useState();
const [state, setState] = useState(0);
useEffect(() => {
fetch("http://localhost:4000/profile", {credentials: "include",})
.then((response) => {response.json().then((data) =>
setUser(data));});
}, [state]);
if (!user) return "";
return (
<div>
<Layout>
<div className="pro-container d-flex ">
<div className="pro-divide row gutters-sm" style={{
borderRadius: "20px ",padding: "10px",width:'99%'}}>
<h1 className="profile-head">My Profile</h1>
<div className="col-md-4 mb-3">
<div className="user-card card">
<div className="card-body">
<div className="d-flex flex-column align-items-center text-center">
<div className="image"
style={{ width: "150px", height: "150px" }}>
<img src={user.photo} alt="user" className="rounded-circle" width="160" height="160"
style={{ border: "2px solid green" }}/>
</div>
<div className="mt-3">
<h4>{user?.username}</h4>
<p className="text-secondary mb-1s">{user?.role}</p>
<p className="bio font-size-sm">{user.bio}</p>
</div></div></div>
</div></div>
<div className="col-md-8">
<div className="detail-card card mb-3">
<div className=" card-body">
<div className="row">
<div className="col-sm-3">
<h6 className="mb-0">Full Name</h6>
</div>
<div className="col-sm-9 text-secondary">
```

```
{user?.username}
</div></div><hr />
<div className="row">
<div className="col-sm-3">
<h6 className="mb-0">Email</h6>
</div>
<div className="col-sm-9 text-secondary">{user?.email}</div>
</div><hr />
<div className="row">
<div className="col-sm-3">
<h6 className="mb-0">Gender</h6>
</div>
<div className="col-sm-9 text-secondary">
{user?.gender}
</div></div><hr />
<div className="row">
<div className="col-sm-3">
<h6 className="mb-0">Bio</h6>
</div>
<div className="col-sm-9 text-secondary" style={{ overflowY: "scroll" }}>
{user?.bio}
</div></div><hr />
<div className="row">
<div className="col-sm-3">
<h6 className="mb-0">Mobile</h6>
</div>
<div className="col-sm-9 text-secondary">{user?.phone}</div>
</div><hr />
<div className="row">
<div className="col-sm-3">
<h6 className="mb-0">Address</h6>
</div>
<div className="col-sm-9 text-secondary">{user?.address}</div>
</div<hr />
<div className="row">
<div className="col-sm-12 text-right">
<button onClick={() => setOpenModal(true)} type="button" className="Edit-btn btn btn-md"
style={{marginLeft: "420px",color: "red",fontFamily: "Revalia",}}>Edit</button>
</div></div></div></div></div>
{openModal && (
<EditProfile closeModal={setOpenModal} user={setState} />
)}
</div></div>
```

```
</Layout>
</div>
)};
export default Profile;
//Server
app.get("/profile", verifyToken, async (req, res, next) => {
try {
let userId = req.user.id;
const userDoc = await User.findById(userId).select("-password");
if (!userDoc) {
throw new AppError('User not found',404)
} else {
res.json(userDoc);
}} catch (error) {
next(error)
}});
```

Edit profile page

```
import React, {useEffect, useState } from 'react'
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import { toast } from 'react-toastify';
const EditProfile = ({closeModal,user}) => {
const [avatar,setAvatar]=useState('');
const navigate=useNavigate();
const [loading,isLoading]=useState(false)
const [userdata,setuserData]=useState({
'username':'',
'email':'',
'gender':'',
'bio':'',
'phone':'',
'address':'',
})
function handleInput(e){
setuserData({...userdata, [e.target.name]: e.target.value,
})
}
useEffect(()=>{
fetch('http://localhost:4000/profile',{
credentials:'include',
})
```

```
.then((response)=>{
response.json()
.then ((data)=>
setuserData({
...userdata,
'username':data.username,
'email':data.email,
'gender':data.gender,
'bio':data.bio,
'phone':data.phone,
'address':data.address,
}
))
})
},[])
async function edit(e){
e.preventDefault();
isLoading(true)
const formData = new FormData();
formData.append('avatar', avatar);
Object.entries(userdata).forEach(([key, value]) => {
formData.append(key, value);
});
try{
const res=await axios.put("http://localhost:4000/profile/edit",formData,{
withCredentials:true,
headers:{
'Content-Type':'multipart/form-data',
},
});
user(res)
setuserData({
'username':'',
'email':'',
'gender':'',
'bio':'',
'phone':'',
'address':'',
})
closeModal(false);
}catch(err)
{
console.log(err.message)
```

```
}
finally{
isLoading(false)
}
}
retrn (
<div className='modal1' style={{position:'absolute'}}>
{loading&&(<> <div className="loader">
<div className="spinner-border"
style={{ position: "fixed", left: "50%", top: "50%" }}role="status">
<span className="sr-only">Loading...</span>
</div>
</div></>)}
<div className='modalcontainer'>
<button className='btn rounded-pill btn-danger'
style={{float:'right',marginRight:'13px',marginTop:'10px'}}
onClick={()=>closeModal(false)}>X</button>
<div className='modal-title'>
<h1 style={{marginLeft:'20px',fontFamily:'angkor'}}>Edit Profile</h1>
</div>
<div className='modal-body'>
<div className='row mb-4' >
<div className='col-sm-2'>
<label for="formFile"> Avatar</label>
</div>
<div className='col-sm-6'>
<input type="file" className="edit-text form-control" id="inputGroupFile04"accept='image/*'
name='avatar' onChange={(e) => setAvatar(e.target.files[0])}/>
</div>
</div>
<div className='row mb-4' >
<div className='col-sm-2 '>
<label for='name'>Name</label>{' '}
</div>
<div className='col-sm-6'>
<input type='text' className='edit-text'name='username' value={userdata.username}
onChange={handleInput} id='name' ></input><br/>
</div>
</div>
<div className='row mb-4'>
<div className='col-sm-2'>
<label for='Email'>Email</label>{' '}
</div>
```

```
<div className='col-sm-6'>
<input type='text'className='edit-text'name='email' value={userdata.email}
onChange={handleInput} placeholder='Email'id='Email'></input>
</div>
</div>
<div className='row mb-4'>
<div className='col-sm-2'>
<label for='Email'>Gender</label>{' '}
</div>
<div className='col-sm-6'>
<input type='radio'name='gender'value='Male'checked={userdata.gender==='Male'}
onChange={handleInput}/>Male{' '}
<input type="radio" name='gender' value="Female"checked={userdata.gender==='Female'}
onChange={handleInput}/>Female
</div>
</div>
<div className='row mb-4'>
<div className='col-sm-2'>
<label>Description</label>{' '}
</div>
<div className='col-sm-6'>
<input type='text'className='edit-text'name='bio' value={userdata.bio}
onChange={handleInput} placeholder='Description'></input>
</div>
</div>
<div className='row mb-4'>
<div className='col-sm-2'>
<label>Mobile</label>{' '}
</div>
<div className='col-sm-6'>
<input type='number'className='edit-text'name='phone' value={userdata.phone}
onChange={handleInput} placeholder='Enter your  mobile number'></input>
</div>
</div>
<div className='row mb-4'>
<div className='col-sm-2'>
<label>Address</label>{' '}
</div>
<div className='col-sm-6'>
<input type='text'className='edit-text' name='address' value={userdata.address}
onChange={handleInput}  placeholder='Where are you from?'></input>
</div>
</div>
```

```
<div className='modal-footer'>

<button className='btn btn-md btn-danger'onClick={edit}>Save Changes</button>
</div>
</div>
</div>
</div>
)
}
export default EditProfile
//Server

app.put("/profile/edit",verifyToken,upload.single("avatar"),
  async (req, res, next) => {
    let imageUrl;
    try {
      if (req.file) {
        const { originalname, path } = req.file;
        const parts = originalname.split(".");
        const ext = parts[parts.length - 1];
        const newPath = path + "." + ext;
        fs.renameSync(path, newPath);
        const cloudRes = await cloudinary.uploader.upload(newPath, {
          folder: "avatar",
        });
        fs.unlinkSync(newPath);
        imageUrl = cloudRes.secure_url;
      }

      const updateData = {
        ...(imageUrl && { photo: imageUrl }),
        ...req.body,
      };
      const info = req.user;
      const userDoc = await User.findByIdAndUpdate(info.id, updateData, {
        new: true,runValidators:true,
      });
      res.status(200).json(userDoc);
    } catch (err) {
      next(err)
    }
  }
);
```

```
app.post("/course", verifyToken, upload.single("file"), async (req, res, next) => {
  let coverImageUrl;
  let cloudinaryRes;
  const role=req.user.role
  try {
    if(role!=='Instructor'){
      throw new AppError('You are restricted from creating course',400)
    }
    if (req.file) {
      const { originalname, path } = req.file;
      const parts = originalname.split(".");
      const ext = parts[parts.length - 1];
      const newPath = path + "." + ext;
      fs.renameSync(path, newPath);

      cloudinaryRes = await cloudinary.uploader.upload(newPath, {
        folder: "course",
        width: 320,
        height: 210,
        crop: "fill",
      });
      fs.unlinkSync(newPath);
    }
    coverImageUrl = cloudinaryRes.secure_url;
    const { title, summary, content, duration, price } = req.body;
    const info = req.user;
    const courseDoc = await Course.create({
      title,
      summary,
      content,
      duration,
      price,
      cover: coverImageUrl,
      author: info.id,
    });
    res.json({ courseDoc });
  } catch (error) {
    next(error)
  }
});
```

View all Courses

```
import { useEffect, useState } from "react";
import Layout from "../Components/Layout/Layout";
import Navbar from "../Components/Navbar/Navbar";
import Course from "./Course";
import {useDispatch,useSelector}from 'react-redux';
import { fetchCourse } from "../Components/Store/CourseSlice";
export default function CourseList(){
const dispatch=useDispatch();
const state=useSelector((state)=>state)
const courses=state?.course?.courseData.data
useEffect(()=>{dispatch(fetchCourse())},[dispatch])
return(
<>
<Layout>
<div className=" course-heading d-flex"style={{justifyContent:'space-between'}}>
<img src="Assets/course2.png" width='580px'></img>
<img style={{marginTop:'-40px'}}src="Assets/course.png" width='500px'
height='500px'></img></div>
<div style={{width:'100%',background:'white',height:'50px',paddingBottom:'10px',}}>
<h1 style={{textAlign:'center',color:"black",fontFamily:'angkor',padding:''}}>All
Courses</h1></div>
<div className="course">
<div className="row gap-4 m-5">
{courses?.length>0?courses?.map(course=>(
<Course {...course}source='allCourse'/>
)):(<h1>No courses Available</h1>)}
</div></div>
</Layout>
</>
)}
export default function Course({title,summary,cover,price,author,_id,source}){
const dispatch=useDispatch()
const role=useSelector((state)=>state.User.userData.role)
async function handleDelete(courseId){
if(window.confirm('Do u really want to delete this course')){
const res= await dispatch(deleteCourse(courseId))
if(res?.payload?.success){
dispatch(fetchMyCourse())
toast.success(res?.payload?.message,{position:'top-center'})
}}}
useEffect(()=>{dispatch(fetchUser())},[])
const link = source === 'myCourse' ? `/myCourse/view/${_id}` : `/course/${_id}`;
```

```
return(
<>
<div className="course-card card">
<div className="image" style={{width:'320px',height:'210px',borderBottom:'2px
solid',marginLeft:'-13px',marginTop:'-5px',overflow:'hidden',position:'relative',borderRadius:'5px
5px 0 0'}}>
<img src={cover} className="card-img-top" alt="img"
style={{width:'320px',height:'210px',objectFit:'cover'}}/>
</div>
<div className="card-body">
<h4 className="card-title fw-bold "style={{fontFamily:"Teko",
fontStyle:"normal",fontWeight:"400",}}>{title}</h4>
<span className="" style={{fontSize:"12px", color:"grey", marginLeft:"0",
marginTop:""}}><FaUserGraduate/> {author.username}</span>
{source==='myCourse'&&role==="Instructor"&&(
<span style={{color:'red',cursor:'pointer',float:'right'}}><RiDeleteBin2Line
onClick={()=>handleDelete(_id)}/></span>)}
<hr />
<p className="card-text ">{summary}</p><hr/>
<span className="course-price d-flex ">

<p className=" "style={{fontFamily:'Teko', fontSize:"25px", fontWeight:"bold"
,marginTop:'4px'}}>₹{price} </p>
<Link to={link} className="btn btn-primary"style={{width:"110px",
height:"40px",marginTop:'5px'}}>{source==='myCourse'?'Watch':'Explore'}</Link>
</span></div></div>
</>);
//Server
app.get("/course", verifyToken, async (req, res, next) => {
try {
let courses;
const role=req.user.role
if (role === "Admin") {
courses = await Course.find()
.populate("author", ["username"])
.populate("enrolled", ["id", "username"])
.sort({ createdAt: 1 });
} else {
courses = await Course.find({ approved: true })
.populate("author", ["username"])
.populate("enrolled", ["id", "username"])
.sort({ createdAt: 1 });
}
```

```
res.status(200).json({success:true,data:courses})
} catch (error) {
next(error)
}});
```

Course description and Payment page

```
import React, { useContext, useEffect, useState } from "react";
import Layout from "../../Components/Layout/Layout";
import { Link, useParams,useNavigate } from "react-router-dom";
import {useDispatch,useSelector}from 'react-redux';
import { fetchUser} from '../../Components/Store/UserSlice'
const CourseDescription = () => {
const [courseInfo, setCourseInfo] = useState(null);
const { id } = useParams();
const dispatch=useDispatch()
const User=useSelector((state)=>state.User.userData)
const navigate=useNavigate()
useEffect(()=>{dispatch(fetchUser()},[])
useEffect(() => {
fetch(`http://localhost:4000/course/${id}`)
.then((response) => {
response.json().then((course) => {setCourseInfo(course);});
})}, []);
if (!courseInfo&&!User)return " "
const currency='INR';
const amount=courseInfo.price
async function payHandler(e){
try{
const response=await fetch(`http://localhost:4000/order/${courseInfo._id}`,{
method:'POST', credentials:'include',
body:JSON.stringify({amount:amount*100,currency,})
 headers:{
"Content-Type":"application/json",},
})
const order=await response.json();
if (response.status>=400 &&response.status<500) {
toast.info(order.message,{theme:'colored',position:'bottom-right'});
return;
}
//Payment is done through razorpay api
const options = {
key: 'rzp_test_0bzBSxkt9xLCn4',
```

```
amount:order.amount,
currency:order.currency,
name: courseInfo.title,
image:courseInfo.cover,
order_id: order.id,
receipt:order.receipt,
handler: async function (response){
const body={...response,};
const validate=await fetch(`http://localhost:4000/order/validate/${courseInfo._id}`,{
method:'POST', credentials:'include',
body:JSON.stringify(body),
headers:{'Content-Type':'application/json}
});
const res=await validate.json();
if(res.success){
navigate(`/order-success?payment_id=${res.payment_id}`)
}},
prefill: {
name: User.username,
email: User.email,
contact: User.phone},
theme: {color:"#001638"}
};
const rzp1 = new window.Razorpay(options);
rzp1.on('payment.failed', function (response){
toast.error('Try again later')});
rzp1.open();
}catch(err){
console.log(err)
}}
return (
<Layout>
<div className='desc-container container'>
<div className='Head'>
<h2 style={{ letterSpacing:"4px", textAlign: "center", marginTop: "20px",fontFamily:'angkor'
,width:"100%",filter: 'drop-shadow(3px 4px 2px rgb(200, 102, 102))'}}>{courseInfo.title}</h2>
</div>
<div className='content' style={{  display: 'flex', flexDirection: 'row',borderBottom:"1px solid
grey", paddingBottom:"10px" }}>
<div className='image' style={{flex:"1",  padding:"10px",borderRadius:'10px',
height:"270px", objectFit:'fill'}}>
<img src={courseInfo.cover} width="100%" height="100%" alt="
style={{borderRadius:'5px',}}/></div>
```

```
<div className='details' style={{ flex: '1' }}>
<p className=""><span className="detail">Created by:</span>
{courseInfo.author.username}</p>
<p> <span className="detail">Duration:</span> {courseInfo.duration} months</p>
<span className="detail">Summary:</span><p style={{marginLeft:"20px"}}>
{courseInfo.summary}</p>
<p style={{ color: "blue" }}> <span className="detail">Price:
</span><strong>₹{courseInfo.price}</strong></p>
{User.role==='Admin'?(
<Link className="btn btn-md bg-primary text-white"
to={`/myCourse/view/${courseInfo._id}`}>Watch</Link>):
(<button className='btn btn-md bg-primary text-white'onClick={payHandler}
style={{width:'200px'}}><strong>Buy Now</strong></button>)}
</div></div>
<h2 style={{margin:"10px auto"}}>About Course</h2>
<div className='content-description' dangerouslySetInnerHTML={{ __html:
courseInfo.content }} /
{!User.role==='Admin'&&(
<button className='btn btn-md bg-primary text-
white'onClick={payHandler}style={{width:'500px', marginBottom:"10px"}}><strong>Buy
Now</strong></button>)}
</div>
</Layout>
)};
export default CourseDescription;

//Server
//Razorpay Integration
const Razorpay = require("razorpay");

const razorpay=new Razorpay({
   key_id:process.env.RAZORPAY_API_KEY,
   key_secret:process.env.RAZORYPAY_API_SECRET,
});
module.exports=razorpay
//fetch Course description
app.get("/course/:id", async (req, res) => {
const { id } = req.params;
const courseDoc = await Course.findById(id).populate("author", ["username"]);
res.json(courseDoc);
});
//Order page
app.post("/order/:id", verifyToken, async (req, res, next) => {
```

```
try {
const userId = req.user.id;
const info = req.user;
const courseId = req.params.id;
const course = await Course.findById(courseId);
if (!course) {
throw new AppError("Course not found", 404);
}
if (course.author.toString() === userId) {
throw new AppError("You cannot buy your own course", 403);
}
if(info.role!=='Student'){
throw new AppError("You are restricted from buying courses",403)
}
const purchase = await Purchase.findOne({ userId, courseId });
if (purchase) {
throw new AppError("You have already purchased this course",400)
} else {
const options = req.body;
options.receipt = shortid.generate();
const order = await razorpay.orders.create(options);
if (!order) {
throw new AppError("Something went wrong, Try again later",400)
}
res.status(200).json(order);
} } catch (err) {
next(err)
}});

app.post("/order/validate/:id", verifyToken, async (req, res, next) => {
try {
const info = req.user;
const courseId = req.params.id;
const { razorpay_order_id, razorpay_payment_id, razorpay_signature } =
req.body;
const sec = crypto.createHmac("sha256", process.env.RAZORYPAY_API_SECRET);
sec.update(`${razorpay_order_id}|${razorpay_payment_id}`);
const digest = sec.digest("hex");
if (digest === razorpay_signature) {
await Purchase.create({
userId: info.id, courseId, paymentId: razorpay_payment_id,});
const course = await Course.findById(courseId);
course.enrolled.push({student:info.id})
```

```
await course.save()
return res.status(200).json({success:true,payment_id:razorpay_payment_id})
} else {
throw new AppError("Invalid transaction",400)
}} catch (err) {next(err)}});
```

Create course page

```
import { useState } from "react";
import { Navigate } from "react-router-dom";
import Layout from "../../Components/Layout/Layout";
import ReactQuill from "react-quill";
const CreateCourse = () => {
const [Title, setTitle] = useState("");
const [Summary, setSummary] = useState("");
const [Content, setContent] = useState("");
const [file, setFile] = useState("");
const [Duration, setDuration] = useState(0);
const [Price, setPrice] = useState(0);
const [error, setError] = useState("");
const [loading, setLoading] = useState(false);
const [redirect, setRedirect] = useState("");
const data = new FormData();
async function create(e) {
setLoading(true);
data.set("title", Title);
data.set("summary", Summary);
data.set("content", Content);
data.set("duration", Duration);
data.set("price", Price);
data.set("file", file[0]);
e.preventDefault();
try {
const res = await fetch("http://localhost:4000/course", {
method: "POST",body: data,credentials: "include",});
if (res.ok) {
toast.success("The course will be available publicly after admin approval",
{ position: "top-center", autoClose: false });
setRedirect(true);
}
} catch (error) {
console.log(error);
} finally {setLoading(false);}
```

```
}
if (redirect) {return <Navigate to={"/"}></Navigate>;}
return (
<Layout>
//loading spinner start
{loading && (
<div className="loader">
<div className="spinner-border" style={{ position: "fixed", left: "50%", top: "50%"
}role="status">
<span className="sr-only">Loading...</span>
</div></div>)}
//loading spinner end
<div className="createContainer ">
<h1 className="text-center" style={{ fontFamily: "angkor" }}>Create course</h1>
<form className="create" onSubmit={create}>
<input type="title" className="edit-text" style={{ width: "100%" }} placeholder={"Title"}
value={Title} onChange={(ev) => setTitle(ev.target.value)}required/><br />
<input type="summary" className="edit-text" style={{ width: "100%" }}
placeholder={"Summary"} value={Summary} onChange={(ev)
setSummary(ev.target.value)}/><br />
<label for="image">Add Thumbnail:{" "}
<input type="file" className=" form-control" id="inputGroupFile04" accept="image/*"
aria-describedby="inputGroupFileAddon04" aria-label="Upload" onChange={(ev) =>
setFile(ev.target.files)}/>
</label><br />
<div className="row mb-4">
<div className="column sm-2">
<label for="desc">Set Duration(in months):</label>
</div>
<div className="column sm-6">
<input type="number" className="edit-text" style={{ width: "20%" }} value={Duration}
max="12" onBlur={() => { Duration > 12 ? setError("maximum 12 months!") : setError("");
}} onChange={(ev) => setDuration(ev.target.value)} required/>
<span style={{ fontSize: "15px",color: "red",flexDirection: "row",}}>{error}</span>{" "}
<br /></div>
<label for="desc">Set Price(in rupees):</label>
<input type="number" value={Price} className="edit-text" style={{ width: "20%" }}
onChange={(ev) => setPrice(ev.target.value)}required/><br />
<label for="desc">Description:</label>
<ReactQuill placeholder="List every information about the course" style={{ background:
"white", border: "1px solid" }} onvalue={Content} id="desc" onChange={(newValue) =>
setContent(newValue)}/><br />
<button className="btn btn-lg btn-secondary">Create course</button><br />
```

```
</form>
</div></div>
</Layout>);};
export default CreateCourse;
//Server
app.post("/course", verifyToken, upload.single("file"), async (req, res, next) => {
let coverImageUrl;
let cloudinaryRes;
const role=req.user.role
try {
if(role!=='Instructor'){
throw new AppError('You are restricted from creating course',400)
}
if (req.file) {
const { originalname, path } = req.file;
const parts = originalname.split(".");
const ext = parts[parts.length - 1];
const newPath = path + "." + ext;
fs.renameSync(path, newPath);
cloudinaryRes = await cloudinary.uploader.upload(newPath, {
folder: "course",
width: 320,
height: 210,
crop: "fill",
});
fs.unlinkSync(newPath);
}
coverImageUrl = cloudinaryRes.secure_url;
const { title, summary, content, duration, price } = req.body;
const info = req.user;
const courseDoc = await Course.create({
title, summary, content, duration, price, cover: coverImageUrl, author: info.id,});
res.json({ courseDoc });
} catch (error) {
next(error)
}});
```

View course content page

```
import { useContext, useEffect, useRef, useState } from "react";
import Layout from "../../Components/Layout/Layout";
import { UserContext } from "../../UserContext";
import { useParams } from "react-router-dom";
```

```
import {IoIosArrowDown,IoIosArrowUp,IoIosAddCircleOutline} from "react-icons/io";
import { useDispatch, useSelector } from "react-redux";
import { deleteLecture, getLectures, updateLecture } from "../../Components/Store/RecordSlice";
import AddAssignment from "../Instructor/AddAssignment";
import { deleteAssign, submitAssign } from "../../Components/Store/AssignSlice";
import Performance from "./Performance";
import StudentAssigns from "../Instructor/StudentAssigns";
import AddLectures from "../Instructor/AddLectures";
import './viewCourse.css';

function ViewCourse() {
const [videoSrc, setVideoSrc] = useState("");
const [isActive, setIsActive] = useState(false);
const [openModal, setOpenmodal] = useState(false);
const [lectureModal, setLectureModal] = useState(false);
const [assignments, setAssignments] = useState(null);
const [selectedAssign, setSelectedAssign] = useState(null);
const [desc, setDesc] = useState("");
const fileInputRef = useRef("");
const [pdf, setPdf] = useState(false);
const dispatch = useDispatch();
const [selectedLecture, setSelectedLecture] = useState(null);
const lectures = useSelector((state) => state?.Lecture?.lectures?.data);
const params = new URLSearchParams(window.location.search);
const q_id = params.get("assignment_id");
const { userInfo } = useContext(UserContext);
const { id } = useParams();
const role = userInfo?.role;
const user = userInfo?.id;
console.log(lectures,"40")
function fetchAssignments(){
try {
fetch(`http://localhost:4000/getAssignments/${id}`, {
credentials: "include",
}).then((res) => {
res.json().then((data) => setAssignments(data.data));
});
} catch (err) {
console.log(err);}}
useEffect(() => {fetchAssignments() dispatch(getLectures(id));}, [openModal]);

function handleClick(lecture) {
setVideoSrc(lecture.file.secure_url);
```

```
setSelectedLecture(lecture);
setSelectedAssign(null);
params.set("lecture_id", lecture._id);
params.delete("assignment_id");
const newUrl = `${window.location.pathname}?${params.toString()}`;
window.history.replaceState({}, "", newUrl);
}
function clear(e) {
setDesc("");
if (fileInputRef.current) {fileInputRef.current.value = "";}}

function handleAssign(assignment) {
setSelectedAssign(assignment);
setSelectedLecture(null);
clear();
params.delete("lecture_id");
params.set("assignment_id", assignment._id);
const newUrl = `${window.location.pathname}?${params.toString()}`;
window.history.replaceState({}, "", newUrl);
}

useEffect(() => {if (q_id) {
const selectedAssignment =assignments &&
assignments.find((assignment) => assignment._id === q_id);
if (selectedAssignment) {setSelectedAssign(selectedAssignment);}}
}, [q_id, assignments]);

useEffect(() => {const updatedSelectedLecture = lectures?.find((lecture) =>
lecture._id === selectedLecture?._id);
if (updatedSelectedLecture) {setSelectedLecture(updatedSelectedLecture);}
}, [lectures, selectedLecture]);

async function submit(e) {
e.preventDefault();
if (!desc && !fileInputRef.current.files.length) {
toast.error('Please type your answer or attach a file',{position:'bottom-center'});
return;
}
const data = new FormData();
data.append("description", desc);
data.append("submit", fileInputRef.current.files[0]);
try {
const res = await dispatch(submitAssign({ c_id: id, q_id, formData: data }));
```

```
if (res.payload.success) {toast.success(res.payload.data);clear();
} else {
toast.error(res.payload.message);}}
 catch (err) {
toast.error(err);
}}
async function handleLecDelete(l_id){
if(window.confirm('Do u really wish to delete this lecture')){
const res=await dispatch(deleteLecture({courseId:id,lectureId:l_id}))
if(res.payload.success){
toast.success(res.payload.data,{position:'top-center'})
dispatch(getLectures(id))
}}}
async function handleAssignDelete(a_id){
if(window.confirm('Do u really wish to delete this lecture')){
const res=await dispatch(deleteAssign({courseId:id,assignId:a_id}))
if(res.payload.success){
toast.success(res.payload.data,{position:'top-center'})
fetchAssignments()
}}}
async function handleMarked(l_id){
const res=await dispatch(updateLecture({courseId:id,lectureId:l_id}))
if(res.payload.success){
toast.info('Progress updated')
dispatch(getLectures(id))
}}
const marked=selectedLecture?.watched?.includes(user)
const pdfUrl = selectedAssign && selectedAssign.file;
const sub =selectedAssign &&selectedAssign.submit &&
selectedAssign.submit.find((sub) => sub.student === user);
return (
<Layout hideFooter={true} index={false}>
<div className="view-container">
<div className="sidebar">
<div className="bars" style={{ overflowY: "scroll", flex: 1 }}>
<div className="sidebar-heading">
<h1 className="text-center"
style={{lineHeight: "3.9",width: "335px",}}>
Videos{" "}
{role === "Instructor" && (<IoIosAddCircleOutline className="addIcon" onClick={() =>
setLectureModal(true)}/>)}
</h1></div>
<div className="span-files">
```

```
{lectures?.map((lecture) => (
<>
<li className={`file ${selectedLecture === lecture ? "active" : ""}`}
key={lecture._id} onClick={() => handleClick(lecture)}>
<p style={{ fontWeight: "bold" }}>{lecture.filename}</p>
{role==='Instructor'&&(<p onClick={(e)=>e.stopPropagation()}style={{marginTop:'-
40px',float:'right',color:'red'}}><RiDeleteBin4Line
onClick={()=>handleLecDelete(lecture._id)}/></p>)}
<p style={{fontSize: "12px",marginTop: "-15px",color: "#6B6860",}} >
{format(lecture.createdAt).date}
<span style={{ float: "right", color: "#6B6860" }}>{format(lecture.createdAt).time}
</span></p></li></>))}
</div></div>
<div className="bars" style={{ overflowY: "scroll", flex: 1 }}>
<div className="sidebar-heading"
style={{height:'auto',borderTop:'2px solid'}}>
<h1 className="text-center" style={{lineHeight: "",width: "335px", }}>Assignment
{role === "Instructor" &&<IoIosAddCircleOutlineclassName="addIcon"onClick= {()=>
setOpenmodal(true)}/>)}</h1></div>
<div className="span-files" style={{ marginTop: "61px" }}>
{assignments &&assignments.map((assignment) => {
const counting = assignment.submit &&assignment.submit.filter((submission) =>
submission.status === "pending").length;
return (
<>
<li className={`file ${selectedAssign === assignment ? "active" : ""}`}onClick={() =>
handleAssign(assignment)}>
<p style={{ fontWeight: "bold" }}>{assignment.title}
{role === "Instructor" && (<>
<span onClick={(e)=>e.stopPropagation()} style={{float:'right', color:'red'}}>
<RiDeleteBin4Line onClick={()=>handleAssignDelete(assignment._id)}/></span>
<span className="badge rounded-pill bg-danger"style={{float:'right'}}>
{counting > 0 && counting}</span>
</>)}
</p>
<p style={{fontSize: "12px", marginTop: "-15px",color: "#6B6860",}}>
{format(assignment.createdAt).date}
</span>
</p></li></>);})}
</div></div></div>
<div className="view-content d-flex flex-column ">
{!selectedLecture&&!selectedAssign&&(
<div className="performance">
```

```
<Performance id={id}/>
</div>)}
{selectedLecture && (
<>
{role==='Student'&&( <div className=" d-flex gap-2"style={{justifyContent:'flex-
end',marginBottom:'5px',marginRight:'20px'}}>
<input className="marked" type='checkbox'checked={marked}id="mark"name='mark'
onChange={(e)=>handleMarked(selectedLecture._id)}/>
<label htmlFor="mark">Mark as watched</label>
</div>)}
<div className="video-container">
<video src={videoSrc}controls disablePictureInPicture autoPlay controlsList="nodownload"
width="100%" height="100%"></video>
</div>
<div className="accordion">
<div className={`accordion-header ${isActive ? "active" : ""}`} onClick={() =>
setIsActive(!isActive)}>
<h2> {selectedLecture ? (selectedLecture?.filename) : (<span style={{ color: "grey" }}>-
</span>)}
</h2>
<span>{isActive ? ( <IoIosArrowDown className="icon" />  ) :
( <IoIosArrowUp className="icon" />)}</span>
</div>{isActive && (
<div className={`accordion-content ${isActive ? "active" : ""}`}
dangerouslySetInnerHTML={{ __html: selectedLecture?.description.replace/href=
"((?!https?:\/\/)[^"]+)"/g,'href="https://$1"'), }}>
</div>)}
</div>
</>
)}
{selectedAssign && (
<div className="assignment-container">
<div className="assign-heading">
<h1 style={{ textTransform: "uppercase", borderBottom: "2px solid",}}>
{selectedAssign && selectedAssign.title}
</h1>
<span className="q">Question:</span>
<p style={{ textIndent: "60px", color: "var(--txt-clr)" }}>{selectedAssign &&
selectedAssign.description}</p>
{pdfUrl && (<>{" "}<button className={!pdf ? "btn" : "btn"}onClick={() =>
setPdf((prevShowPdf) => !prevShowPdf)}>
{!pdf ? "View question" : "close"}
</button>
```

```
</>)}
<span style={{ float: "right", color: "red" }}>marks: {selectedAssign &&
selectedAssign.marks}</span>
</div>
{pdf && (<div style={{ width: "100%", height: "700px", marginTop: "10px" }}>
<object data={`http://localhost:4000/${pdfUrl}`} type="application/pdf" width="100%"
height="100%">
<p>
Not a valid Pdf<a href={pdfUrl}>to the PDF!</a>
</p>
</object>
</div>)}
{role === "Student" && (<div className="submit">
<h3 style={{ marginLeft: "10px" }}>{!sub ? "Submit Your answer" : ""}</h3>
{sub ? (<div>
<h2 style={{ fontWeight: "bold" }}>Answer Successfully Submitted<FaCheck
className="check" /></h2>
<p>Status:<span className={sub.status === "pending"? "badge bg-warning": "badge bg-
success"}>{sub.status}</span>
</p>{sub.status === "submitted" ? (<p>You have gained<span>{sub.marks}</span> marks
</p>) : ("")}
</div>) : (
<form onSubmit={submit}>
<textarea className="textBox" placeholder="Type your answer" value={desc}
onChange={(e) => setDesc(e.target.value)}/>
<label style={{ marginLeft: "5px" }} htmlFor="formFile">Attach files</label>
<input className="form-control" type="file" id="formFile" ref={fileInputRef}/>
<div className="bt">
<button type="submit">Submit</button>
<button type="reset" value="reset" onClick={clear}>clear</button>
</div></form>)}</div>)}
{role === "Instructor" && (
<div className="students">
<h3>Assignments</h3>
<StudentAssigns format={format} c_id={id} q_id={q_id} />
</div>)}
</div>)}
</div>
{openModal && <AddAssignment closeModal={setOpenmodal} />}
{lectureModal && <AddLectures closeModal={setLectureModal} />}
</div>
</Layout>
)}
```

```
export default ViewCourse;
//Server
//fetch lectures
app.get("/getLectures/:id", verifyToken, async (req, res, next) => {
try{
const { id } = req.params;
const course = await Course.findById(id);
if (!course) { throw new AppError("Course not found", 404);}
course.recordings.sort( (a, b) => new Date(a.createdAt) - new Date(b.createdAt)  );
res.status(200).json({ status: true, data: course.recordings });
}catch(err) next(err)
}});
//deleteLectures:
app.delete('/deleteLecture/:id',verifyToken,async(req,res,next)=>{
try{
const {id}=req.params;
const userId=req.user.id
const role=req.user.role
const lectureId=req.query.lecture_id
const course=await Course.findById(id);
if (!course) {
throw new AppError("Course not found", 404);
}
if (role!=='Instructor') {
throw new AppError("You are forbidden", 403);
}
if (course.author.toString() !== userId.toString()) {
throw new AppError("You cannot make changes to this course", 403);
}
const lectureIndex = course.recordings.findIndex(
(lecture) => lecture._id.toString() === lectureId.toString()
);
if (lectureIndex === -1) {
throw new AppError('Lecture does not exist.', 404);
}
await cloudinary.uploader.destroy(
course.recordings[lectureIndex].file.public_id,
{resource_type: 'video',});
course.total-=1
course.recordings.splice(lectureIndex, 1);
await course.save()
res.status(200).json({success:true,data:'successfully deleted'})
}catch(err){next(err)}})
```

```
app.get("/getAssignments/:id", verifyToken, async (req, res, next) => {
try {
const user=req.user.id;
const { id } = req.params;
const course = await Course.findById(id);
if (!course) {
throw new AppError("Course not found", 404);
}
course.assignments.sort((a, b) => new Date(a.createdAt) - new Date(b.createdAt));
return res.status(200).json({ status: true, data: course.assignments });
} catch (err) {
next(err)
}
});
//delete Assignment
app.delete('/deleteAssignment/:id',verifyToken,async(req,res,next)=>{
try{
const {id}=req.params;
const userId=req.user.id
const role=req.user.role
const assignId=req.query.assignment_id
const course=await Course.findById(id);
if (!course) {
throw new AppError("Course not found", 404);
}
if (role!=='Instructor') {
throw new AppError("You are forbidden", 403);
}
if (course.author.toString() !== userId.toString()) {
throw new AppError("You cannot make changes to this course", 403);
}
const assignIndex = course.assignments.findIndex(
(assign) => assign._id.toString() === assignId.toString()
);
if (assignIndex === -1) {
throw new AppError('Course does not exist.', 404);
}
const assignmentMarks = course.assignments[assignIndex].marks;
course.total-=assignmentMarks
course.assignments.splice(assignIndex, 1);
await course.save()
res.status(200).json({success:true,data:'Deleted successfully'})
}
```

```
catch(err){
next(err)
}
})
```

Add Lectures:

```
import { useState } from "react";
import ReactQuill from "react-quill";import {useDispatch,useSelector} from 'react-redux';
import { addLecture, getLectures } from "../../Components/Store/RecordSlice";
import { Navigate, useParams } from "react-router-dom";
const AddLectures = ({ closeModal }) => {
const [video, setVideo] = useState();
const [title,setTitle]=useState();
const [desc,setDesc]=useState();
const {id}=useParams()
const loading=useSelector((state)=>state?.Lecture?.loading)
const dispatch=useDispatch()
const data = new FormData();
async function add(e){
data.set("video", video);
data.set("title",title);
data.set("description",desc);
e.preventDefault()
if(!video||!title||!desc){
return toast.error('Every field is mandatory')}
try{
const res=await dispatch(addLecture({c_id:id,formData:data}))
if(res?.payload?.success){toast.success(res?.payload?.message)
await dispatch(getLectures(id))
closeModal(false}
else{
toast.error(res?.payload?.message)
}}
catch(err){console.log(err)}}
return (<>
<div className="modal1" style={{ position: "fixed" }}>
<div className="modalcontainer">
{loading && (<div className="loader"style={{zIndex:'2'}}>
<div className="spinner-border"
style={{ position: "fixed", left: "50%", top: "50%" }}role="status">
<span className="sr-only">Loading...</span>
</div></div>)}
```

```
<button className="btn rounded-pill btn-danger" style={{ float: "right", marginRight: "13px",
marginTop: "10px" }}onClick={() => closeModal(false)}>X</button>
<div className="modal-title " style={{ padding: "10px", background: "white", borderRadius:
"20px" }}>
<h1 style={{ marginLeft: "100px", fontFamily: "angkor" }}>
Add a new Lecture
</h1></div>
<form onSubmit={add}>
<div className="modal-body">
<div className="row mb-4">
<div className="col-sm-2">
<label for="formFile"> Video</label>
</div>
<div className="col-sm-6">
<input type="file" className="edit-text form-control"id="inputGroupFile04"accept="video/*"
name="videorequiredonChange={(e)=>setVideo(e.target.files[0])}/>
<label for="name">Title</label>{" "}
</div>
<div className="col-sm-6">
<input type="text"className="edit-text" name="title" value={title}
onChange={(e)=>setTitle(e.target.value)}id="name"required></input><br />
</div></div>
<div className="row mb-4">
<div className="col-sm-2">
<label >Description</label>{" "}
</div>
<div className="col-sm-9">
<ReactQuill placeholder="provide description,links"style={{ background: "white", }}
onvalue={desc} id="desc" onChange={(newValue) => setDesc(newValue)}/>
</div></div></div>
<div className='modal-footer'>
<button className='btn btn-md btn-info'>Add Lecture</button>
</div></form></div></div>
</>};
export default AddLectures;
//Add Lectures
app.post("/addLecture/:id",verifyToken,upload.single("video"),async (req, res, next) => {
let course;
let videoUrl={};
try {
const { id } = req.params;
const info = req.user;
course = await Course.findById(id);
```

```
if (!course) {
throw new AppError("Course not found", 404);
}
if (info.role !== "Instructor") {
throw new AppError("You dont have permission to add lectures",400)
}
if (req.file) {
const cloudinaryRes = await cloudinary.uploader.upload(req.file.path, {folder: "video",
resource_type: "video",});
fs.unlinkSync(req.file.path);
videoUrl.secure_url = cloudinaryRes.secure_url;
videoUrl.public_id=cloudinaryRes.public_id;
} else {
throw new AppError("File not found", 404);
}
const { title, description } = req.body;
course.recordings.push({
file: videoUrl,
filename: title,
description,
});
course.total+=1
await course.save();
res.status(200).json({ success: true, message: "successfully added", course });
} catch (err) {
next(err)
}});
```

Add Assignments:

```
const AddAssignment = ({closeModal}) => {
const [pdf,setPdf]=useState();
const [title,setTitle]=useState();
const [desc,setDesc]=useState();
const [marks,setMarks]=useState();
const {id}=useParams()
const dispatch=useDispatch();
const data=new FormData
async function add(e){
data.append('title',title)
data.append('description',desc)
data.append('marks',marks)
```

```
data.append('pdf',pdf)
e.preventDefault();
try{
const res=await dispatch(addAssignments({c_id:id,formData:data}))
if(res?.payload?.success){toast.success("Submitted Successfully")
closeModal(false)}
else{toast.error(res.payload.message)}
}catch(err){
console.log("24",err)
}}
return (
<div className="modal1" style={{ position: "fixed"}}>
<div className="modalcontainer"style={{height:'500px'}}>
<button className="btn rounded-pill btn-danger"style={{ float: "right", marginRight: "13px",
marginTop: "10px" }}onClick={() => closeModal(false)}>X</button>
<div className="modal-title " style={{ padding: "10px", background: "white", borderRadius:
"20px" }}>
<h1 style={{ marginLeft: "10px", fontFamily: "angkor" }}>
Add a new Assignment
</h1>
</div>
<form onSubmit={add}>
<div className="modal-body">
<label for="name">Question</label>{" "}
</div>
<div className="col-sm-6">
<input type="text" className="edit-text" name="title" value={title} onChange={(e)=>
setTitle(e.target.value)}id="name"required></input><br />
<label >Description</label>{" "}
</div>
<div className="col-sm-9">
<textarea className="edit-text"style={{height:'100px'}}requiredname='description'
value={desc} onChange={(e)=>setDesc(e.target.value)}/>
</div></div>
<div className="row mb-4">
<div className="col-sm-2">
<label >Marks</label>{" "}</div>
<div className="col-sm-9">
<input type='number' className="edit-text"style={{}}required name='description'
value={marks}onChange={(e)=>setMarks(e.target.value)}/></div></div>
<div className="row mb-4">
<div className="col-sm-2">
<label for="formFile"> File</label></div>
```

```
<div className="col-sm-6">
<input type="file" className="edit-text form-control" id="inputGroupFile04"
accept="application/pdf"onChange={(e)=>setPdf(e.target.files[0])}/>
</div></div></div>
<div className='modal-footer'>
<button className='btn btn-md btn-info'>Add Assignment</button>
</div></form></div></div>)}
export default AddAssignment
//Add Assignments
app.post("/addAssignments/:id", verifyToken,uploadFile.single("pdf"),
async (req, res, next) => {
let course;
let newPath;
try {
const { id } = req.params;
const info = req.user;
course = await Course.findById(id);
if (!course) {
throw new AppError("Course not found", 404);
}
if (info.role !== "Instructor") {
throw new AppError("You are restricted from adding assignments", 400);
}
if (req.file) {
const { originalname, path } = req.file;
const parts = originalname.split(".");
const ext = parts[parts.length - 1];
newPath = path + "." + ext;
fs.renameSync(path, newPath);
}
const { title, description, marks } = req.body;
const parseMarks=parseInt(marks)
course.assignments.push({title,description,marks:parseMarks,file: newPath,});
course.total+=parseMarks
await course.save();
return res.status(200).json({ success: true, data: course });
} catch (err) {next(err)}});

//Submit server
app.post("/submitAssign/:id",verifyToken,uploadFile.single("submit"),
async (req, res, next) => {
try {
const info = req.user;
```

```
const { id } = req.params;
const { assignment_id } = req.query;
let newPath;
const course = await Course.findById(id);
if (!course) {
throw new AppError("Course not found",404)
}
const assignment = await course.assignments.id(assignment_id);
if (!assignment) {
throw new AppError("Assignment not found",404)
}
if (info.role !== "Student") {
throw new AppError("Only Student can send assignment",400)
}
if (req.file) {
const { originalname, path } = req.file;
const parts = originalname.split(".");
const ext = parts[parts.length - 1];
newPath = path + "." + ext;
fs.renameSync(path, newPath);
}
const { description } = req.body;
assignment.submit.push({
student: info.id,
description,
file: newPath,
});
await course.save();
return res.status(200).json({ success: true, data: "Submitted successfully" });
} catch (err) {
next(err)
}
}
);

app.get('/assignments/:id',verifyToken,async(req,res,next)=>{
try{
const info=req.user.role
const {id}=req.params
const { assignment_id } = req.query;
const course=await Course.findById(id)
if(!course){
throw new AppError("Course not found",404)
```

```
}
const assignment=await course.assignments.id(assignment_id)

if (!assignment) {
throw new AppError("Assignement not found",404)
}
const submit=assignment.submit
const studentIds = submit.map(subdoc => subdoc.student);
const students = await User.find({ _id: { $in: studentIds } }, 'username');
submit.forEach(subdoc => {
const student = students.find(student => student._id.toString() === subdoc.student.toString());
subdoc.student = student;
});
submit.sort((a, b) => new Date(a.createdAt) - new Date(b.createdAt));
res.status(200).json({success:true,data:submit})
}
catch(err){
next(err)
}
})

app.put('/UpdateSubmission/:id',verifyToken,async(req,res,next)=>{
try{
const info=req.user.role
const {id}=req.params
const { assignment_id } = req.query;
const {status,sub_id}=req.body;

const course=await Course.findById(id)
if(!course){
throw new AppError("Course not found",404)
}
const assignment=await course.assignments.id(assignment_id)
if (!assignment) {
throw new AppError("Assignment not found",404)
}
if(info!=='Instructor'){
throw new AppError("You dont have permission to access this route",403)
}

const submissionIndex = assignment.submit.findIndex((submission) =>
submission._id.toString() === sub_id)
```

```
if (submissionIndex === -1) {
throw new AppError("Submission not found",404)
}
const studentId = String(assignment.submit[submissionIndex].student);
const studentIndex = course.enrolled.findIndex(enroll => String(enroll.student) === studentId);
if (studentIndex === -1) {
throw new AppError("Student not found",404)
}
if(status){
assignment.submit[submissionIndex].status ='submitted';
assignment.submit[submissionIndex].marks=assignment.marks
course.enrolled[studentIndex].totalMarks += assignment.marks;
await course.save();
const updatedSubmission = assignment.submit.id(sub_id);
res.status(200).json({success:true,data:updatedSubmission})
}
else{
res.status(400).json({success:false,data:assignment.submit,message:'You cant change the form
once the it is corrected'})
}
}
catch(err){
next(err)
}
})
```

Contact Form

```
import { useEffect, useState } from "react";
import Layout from "../../Components/Layout/Layout";
import { useDispatch } from "react-redux";
const Contact = () => {
const dispatch=useDispatch()
const [contact,setContact]=useState({'username':','email':','message':',})
function handleInput(e){setContact({...contact, [e.target.name]: e.target.value,})}
async function contactData(e){
e.preventDefault()
const res=await dispatch(AddContact(contact))
if(res.payload.success){
toast.success('Submitted successfully')
setContact({'username':','email':','message':''})}}
return (
<Layout>
```

```
<section className="container">
<div className="Container-contact">
<h1>How can we help you?</h1>
<form className="row g-3" onSubmit={contactData}>
<div className="col-md-6">
<label for="inputName4" className="form-label">Name</label>
<input type="text" className="form-control" id="inputName4" name='username'
value={contact.username} onChange={handleInput} /></div>
<div className="col-md-6">
<label for="inputEmail4" className="form-label">Email</label>
<input type="text" className="form-control" id="inputEmail4"
name='email'value={contact.email}onChange={handleInput} /></div>
<div className="col-12">
<label for="textContent" className="form-label">Message </label>
<textarea  className="form-control" id="textContent"
name='message'value={contact.message}onChange={handleInput} />
</div>
<div className="col-12">
<button type="submit" className="btn btn-primary">Send message</button>
</div></form></div></section></Layout>);};
export default Contact;
//Contact Server
app.post('/contact',async(req,res,next)=>{
try{
const {username,email,message}=req.body;
if(!username||!email||!message){
throw new AppError('All fields are mandatory',400)
}
const contact=await Contact.create({name:username,email,message,})
res.status(200).json({success:true,data:contact})
}catch(err){
next(err)
}
})
```

Performance Page

```
import { useEffect } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { fetchUser } from '../../Components/Store/UserSlice'
import { fetchCourse, fetchPerformance } from '../../Components/Store/CourseSlice'
import { useParams } from 'react-router-dom'
import { CircularProgressbar, buildStyles } from 'react-circular-progressbar';
```

```
import 'react-circular-progressbar/dist/styles.css';
import Certificate from './Certificate'

const Performance = () => {
const dispatch=useDispatch()
const User=useSelector((state)=>state.User.userData)
const performance=useSelector((state)=>state.course.performanceStats)
const percentage=Math.floor(performance.studentPerformance*100)/100
console.log(percentage)
const {id}=useParams()
useEffect(()=>{
dispatch(fetchUser())

dispatch(fetchPerformance(id))
},[])
return (
<div className='performance-container'>
{User.role==='Student'&&( <> <h1>Html/css/Javascript</h1>
<p>No of lectures:<span>{performance.noOfLectures}</span></p>
<p>No of assignments:<span>{performance.noOfAssignments}</span></p>
<div className="">
<Certificate></Certificate>
</div>
<div className='performance-details'>
<h3>Your Performance</h3>
<div style={{width:'100px'}}>
<span><CircularProgressbar styles={
buildStyles({textColor:'green',pathColor:'green'})}
value={percentage} text={`${percentage}%`} /></span>
</div>

</div></>)}

{User.role==='Instructor'&&( <div className='student-performance'>
<table class="table table-bordered table-striped">
<thead className="thead-dark">
<tr>
<th scope="col">SL.No</th>
<th scope="col">Student Name</th>
<th scope="col">Student Email</th>
<th scope="col">Date of join</th>
<th scope="col">Student performance</th>
</tr>
```

```
</thead>
<tbody>
<tr>
<th scope="row">1</th>
<td>Tushan</td>
<td>tushan@gmail.com</td>
<td>20/02/2024</td>
<td>{percentage}%</td>
</tr>
</tbody>

</table>
</div>)}
</div>
)
}

export default Performance
```

Payment success

```
import React, { useEffect } from 'react'
import Layout from '../../Components/Layout/Layout'
import { useDispatch, useSelector } from 'react-redux'
import { getPayment } from '../../Components/Store/PaymentSlice'

const CheckoutSuccess = () => {
const dispatch=useDispatch()
const payment=useSelector((state)=>state.Payments.payments.data)
const loading=useSelector((state)=>state.Payments.isLoading)
const params = new URLSearchParams(window.location.search);
const payment_id=params.get('payment_id')
useEffect(()=>{
dispatch(getPayment(payment_id))
},[])

function format(formatted){
const date=new Date(formatted)
return date.toLocaleDateString('en-IN', {
day: '2-digit',
month: '2-digit',
year: 'numeric',
});
```

```
}
return (
<Layout>
<div className='checkout'>
{loading&&<div className='d-flex justify-content-center'>
<div className='loading-animation'>Wait a moment...</div>
</div>}
{payment&&!loading&&<div className='checkout-details'>
<h1>SkillBites</h1>

<div className='sub'>
{payment?.map(pay=>(<>
<p style={{float:'right'}}>OrderID:<strong> {pay.orders.order_id}</strong></p>
<p>Payment_id: <strong>{pay.orders.id}</strong></p>
<div className='sub1'>
<p>Name: <strong>{pay?.courses[0]?.userId?.username}</strong></p>
<p>Mobile no:<strong> {pay.orders.contact}</strong></p>
<p>Email:<strong> {pay.orders.email}</strong></p>
<p>Method:<strong> {pay.orders.method}</strong></p>
<p>Date of purchase: <strong> {format(pay.orders.created_at*1000)}</strong></p>
</div>
<table className='table'>
<thead className='table-dark'>
<tr>
<th>#</th>
<th>Product</th>
<th>Amount</th>
</tr>
</thead>
<tbody className='table-secondary'>
<tr>
<td>1</td>
<td>{pay.courses[0].courseId.title}</td>
<td>{pay.orders.amount/100}/-</td>
</tr>
<tr>
<th colSpan='2'style={{textAlign:'right'}}> Grand Total:</th>
<th> {pay.orders.amount/100}/-</th>
</tr>
</tbody>
</table>
<p className='thanks text-center'>Thank You For The Purchase</p>
</>
```

```
))}
</div>
</div>}
</div>
</Layout>
)
}
export default CheckoutSuccess
```

Admin Dashboard

```
import {Chart as ChartJS,ArcElement,Tooltip,LegendCategoryScale, LinearScale, BarElement,
Title,} from 'chart.js/auto'
import { Doughnut } from "react-chartjs-2";
import { useDispatch, useSelector } from 'react-redux';
const AdminDashboard = () => {
const dispatch=useDispatch()
const stats=useSelector((state)=>state.Extras.stats)
const loading=useSelector((state)=>state.Extras.loading)
ChartJS.register(ArcElement,Tooltip,Legend,CategoryScale,LinearScale,BarElement,Title);
const userData={
labels:["Students","Instructors"],
data:[40],
datasets:[{label:"Users",
data:[stats?.studentCount,stats.instructorCount]
}],}
useEffect(()=>{dispatch(fetchStats())},[])
return (
<Layout hideFooter={true} initial={true}>
{stats&& <div className='dashboard'>
<div className='heading'>
<h1>Admin Dashboard</h1>
</div>
<div className='cards'>
<div className='dashboard-card'>
<div className='card-head'>
<h2>Total Users</h2>
</div><div className={loading?'skeleton':'card-content'}>
<span>{!loading?stats.instructorCount+stats.studentCount:""}</span></div></div>
<div className='dashboard-card'>
<div className='card-head'>
<h2>Courses</h2>
</div>
```

```
<div className={loading?'skeleton':'card-content'}>
<span >{!loading?stats.courseCount:''}</span>
</div></div>
<div className='dashboard-card'>
<div className='card-head'>
<h2>Total Income</h2>
</div>
<div className={loading?'skeleton':'card-content'}>
<span style={{whiteSpace:'nowrap'}}>
{!loading?'₹'+stats.totalRevenue+'.00':''}
</span></div></div></div></div>
<div className={!loading?'pie':'pie-skeleton'}>
<h3 className='text-center'>Users</h3>
{!loading&&<Doughnut data={userData}/>}
</div></div>}</Layout>
)}
export default AdminDashboard
```

Payments page

```
import React, { useEffect, useState } from 'react'
import Layout from '../../Components/Layout/Layout'
import { useDispatch, useSelector } from 'react-redux'
import { getAllPayments } from '../../Components/Store/PaymentSlice'
import { toast } from 'react-toastify'
const AdminPayment = () => {
const [payment,setPayment]=useState()
const dispatch=useDispatch()
const payments=useSelector((state)=>state?.Payments?.payments?.data)
const isLoading=useSelector((state)=>state?.Payments?.isLoading)
useEffect(()=>{dispatch(getAllPayments())},[dispatch])
const handleSubmit = async(e) => {
try{
e.preventDefault();
if(payment){await dispatch(getAllPayments(payment));}
else{
await dispatch(getAllPayments())
}
}catch(err){
toast.error(err)}};
return (
<Layout hideFooter={true} initial={true}>
<div className='Payment-Dashboard'>
```

```
{isLoading&&(<> <div className="d-flex justify-content-center loader">
<div className='loading-animation'>Wait a moment...</div>
</div></>)}
<h1 className='text-center'> Payments</h1>
{payments&&!isLoading&&<> <input type='text'value={payment} placeholder='Enter
Payment_id'onChange={(e)=>setPayment(e.target.value)}/>
<button onClick={handleSubmit}>Filter</button>
<table className="table table-hover table-striped">
<thead className='table-dark'>
<tr>
<th scope="col">Payment_id</th>
<th scope="col">Username</th>
<th scope="col">Email</th>
<th scope="col">Course Purchased</th>
<th scope="col">Amount</th>
<th scope="col">Status</th>
<th scope="col">Purchased on</th>
</tr>
</thead>
<tbody>
{payments?.map(payment=>(
<tr key={payment.orders.id} >
<th scope="row">{payment.orders.id}</th>
<td>{payment?.courses[0]?.userId?.username}</td>
<td>{payment.orders.email}</td>
<td>{payment?.courses[0]?.courseId?.title}</td>
<td>{payment.orders.amount/100+'.00'}</td>
<td><span  className={payment.orders.status==='captured'?'badge bg-success':'badge bg-
danger'}>{payment.orders.status}</span></td>
<td>{format(new Date(payment.orders.created_at*1000)).date  +' at '+format(new
Date(payment.orders.created_at*1000)).time}</td>
</tr>))}</tbody></table></>}
</div>
</Layout>
)}
export default AdminPayment
```

Manage Student

```
import  { useEffect, useState } from 'react'
import Layout from '../../Components/Layout/Layout'
import { useDispatch, useSelector } from 'react-redux'
import { fetchAllUsers, fetchUser } from '../../Components/Store/UserSlice'
```

```
import { useNavigate } from 'react-router-dom'
const AdminStudent = () => {
const [userId,setUserId]=useState()
const dispatch=useDispatch()
const navigate=useNavigate()
const userData=useSelector((state)=>state.User.userData.data)
useEffect(()=>{dispatch(fetchAllUsers())},[])
return (
<Layout hideFooter={true}initial={true} >
<div className='Course-Dashboard'>
<h1 className='text-center'>Students</h1>
<table className="table table-hover table-striped">
<thead className='table-dark'>
<tr>
<th scope="col">Student_id</th>
<th scope="col">Username</th>
<th scope="col">Email</th>
<th scope="col">Course Purchased</th>
<th scope="col">Joined on</th>
</tr>
</thead>
<tbody>
{userData&&userData.map(user=>(
<tr key={user.user._id} onClick={()=>setUserId(user.user._id)}>
<th scope="row">{user.user._id}</th>
<td>{user.user.username}</td>
<td>{user.user.email}</td>
<td>
{user.purchases &&user.purchases.length>0? (
<ol>
{user.purchases.map((course, index) => (
<li key={index}>{course?.courseId?.title}</li>
))}
</ol>
):(<span style={{marginLeft:'50px'}}>-</span>)}
</td>
<td>{format(user?.user?.createdAt).date +' at '+format(user.user.createdAt).time}</td>
</tr>))}
</tbody></table></div></Layout>
)}
export default AdminStudent
```

Manage Instructor

```
import React, { useEffect } from 'react'
import Layout from '../../Components/Layout/Layout'
import { useDispatch, useSelector } from 'react-redux'
import { fetchAllUsers } from '../../Components/Store/UserSlice'
const AdminInstructors = () => {
const dispatch=useDispatch()
const instructors=useSelector((state)=>state.User.userData.data2)
useEffect(()=>{
dispatch(fetchAllUsers())
},[])
function format(formatted){
const date=new Date(formatted)
const dateType= date.toLocaleDateString('en-IN', {
day: '2-digit',
month: '2-digit',
year: 'numeric',
});
const timeType= date.toLocaleTimeString('en-IN', {
hour:'2-digit',
minute:'2-digit'
});
return {date:dateType,
time:timeType}
}
return (
<Layout hideFooter={true}initial={true}>
<div className='Course-Dashboard'>
<h1 className='text-center'>Instructors</h1>
<table className="table table-hover table-striped">
<thead className='table-dark'>
<tr>
<th scope="col">Instructor_id</th>
<th scope="col">Instructor Name</th>
<th scope="col">Instructor Email</th>
<th scope="col">Created Courses</th>
<th scope="col">Joined on</th>
</tr>
</thead>
<tbody>
{instructors&&instructors.map(user=>(
<tr key={user.instructor._id}>
<th scope="row">{user.instructor._id}</th>
<td>{user.instructor.username}</td>
```

```
<td>{user.instructor.email}</td>
<td>
{user.courses &&user.courses.length>0? (
<ol>
{user.courses.map((course, index) => (
<li key={index}>{course.title}</li>
))}
</ol>
):(<span style={{marginLeft:'50px'}}>-</span>)}
</td>
<td>{format(user.instructor.createdAt).date +' at '+format(user.instructor.createdAt).time}</td>
</tr>))}
</tbody>
</table>
</div>
</Layout>
)}
export default AdminInstructors
```

Manage Feedbacks

```
import React, { useEffect } from 'react'
import Layout from '../../Components/Layout/Layout'
import { useDispatch, useSelector } from 'react-redux'
import { fetchContact } from '../../Components/Store/ExtraSlice'

const Feedbacks = () => {
const dispatch=useDispatch()
const contact=useSelector((state)=>state.Extras.contact.data)
useEffect(()=>{
dispatch(fetchContact())
},[])
console.log(contact)
let count=0
function format(formatted){
const date=new Date(formatted)
return date.toLocaleDateString('en-IN', {
day: '2-digit',
month: '2-digit',
year: 'numeric',
});
}
return (
```

```
<Layout hideFooter={true}initial={true}>
<div className='Course-Dashboard'>
<h1 className='text-center'>Feedbacks</h1>
<table className="table table-striped">
<thead className='table-dark'>
<tr>
<th scope="col">SL_NO</th>
<th scope="col">Name</th>
<th scope="col">Email</th>
<th scope="col">Sent on</th>
<th scope="col">Message</th>
</tr>
</thead>
<tbody>
{contact&&contact.map(contacts=>(
<tr key={contacts._id}>
<th scope="row">{count=count+1}</th>
<td>{contacts.name}</td>
<td>{contacts.email}</td>
<td>{format(contacts?.createdAt)}</td>
<td style={{width:'40%'}}>{contacts.message}</td>
</tr>))}
</tbody>
</table>
</div>
</Layout>
)
}
export default Feedbacks
```

View student assignment

```
import React, { useState,useEffect } from 'react'
import { FaCheck } from "react-icons/fa";
import {useDispatch,useSelector}from 'react-redux'
import { UpdateSubmission, getSubmission } from '../../Components/Store/SubmitSlice';
import {Link, useNavigate,useLocation} from 'react-router-dom'

const StudentAssigns = ({format,c_id,q_id}) => {
let count=0;
const location=useLocation()
const dispatch=useDispatch()
const submissions=useSelector((state)=>state?.Submissions?.submissions?.data)
```

```
const isLoading=useSelector((state)=>state?.Submissions?.isLoading)
console.log(submissions)
function click(submit,checked){
dispatch(UpdateSubmission({sub_id:submit._id,status:checked,c_id,q_id}))
}
useEffect(()=>{
dispatch(getSubmission({c_id,q_id}))
},[c_id,q_id])

const isPdfViewerPage = location.pathname === '/file-viewer';
return(<>
{isLoading&&(<> <div className="loader">
<div className="spinner-border"
style={{ position: "fixed", left: "50%", top: "50%" }}role="status">
<span className="sr-only">Loading...</span>
</div>
</div></>)}
{!isLoading && !isPdfViewerPage &&(
<table class="table table-bordered tanle-striped">
<thead className="thead-dark">
<tr>
<th scope="col">SL.No</th>
<th scope="col">Student Name</th>
<th scope="col"style={{width:'400px'}}>Answers</th>
<th scope="col">Files</th>
<th scope="col">Submitted on</th>
<th scope="col">mark as submitted</th>
</tr>
</thead>
<tbody>
{submissions&&submissions.map(submit=>(<>
<tr key={submit._id}>
<th scope="row">{count=count+1}</th>
<td>{submit.student.username}</td>
<td>{submit.description}</td>
<td>{submit.file?(<Link to={`/file-viewer?source=${submit.file}`}>View</Link>):'No
file'}</td>
<td>{format(submit.submittedAt).date +' at '+format(submit.submittedAt).time}</td>
<td style={{ textAlign:'center'}}><input
type='checkbox'checked={submit?.status==='submitted'}
onChange={(e)=>click(submit,e.target.checked)}/></td>
</tr>
</>
```

```
))}
</tbody>
</table>)}
</>
```

Purchases Page:

```
import React, { useEffect } from 'react'
import Layout from '../../Components/Layout/Layout'
import { useDispatch, useSelector } from 'react-redux'
import { fetchMyCourse } from '../../Components/Store/CourseSlice'
import { Link, useNavigate } from 'react-router-dom'
const Orders = () => {
const navigate=useNavigate()
const dispatch=useDispatch()
const orders=useSelector((state)=>state.course.MycourseData.data)
useEffect(()=>{
dispatch(fetchMyCourse())
},[])
console.log(orders)
return (
<Layout>
<div className='orders'>
<h1>Purchases</h1>
{orders&&orders.length>0?( <div className='order-details'>
<div className='order-details-row'>
<div className='row'>
<div className='col-md-8'>
<h4>Orders</h4>
</div>
<div className='col'>
<h4>Purchase details</h4>
</div>
</div>
</div>
{orders.map(order=>(
<div className='row mb-2'>
<div className='col-md-8'>
<div className='card d-flex flex-row align-items-center p-3'style={{}}>
<div className='image 'style={{objectFit:'fill'}}>
<img src={order.cover}width='70px'height='40px'className='me-3 rounded'></img>
</div>
<p style={{}}>{order.title}</p>
```

```
</div>
</div>
<div className='col d-flex align-items-center justify-content-center'>
<Link className='btn btn-secondary' to={`/order-
success?payment_id=${order.paymentId}`}>view details</Link>
</div>
</div>))}
</div>):(
<div className='no-Orders'>
<h4 className='mb-3'>You Dont have any purchases</h4>
<p className='d-flex '><Link to='/course' className='btn btn-primary'>Browse
Courses</Link></p>
</div>
)}
</div>
</Layout>
)
}
export default Orders
```

Stats

```
//Server

app.post('/certificate/:id',verifyToken,async(req,res,next)=>{

try{
const user=req.user
const {id}=req.params
const course=await Course.findById(id)
if(!course){
throw new AppError('Course not found',404)
}
const studentIndex = course.enrolled.findIndex(enroll => String(enroll.student) === user.id);
if (studentIndex === -1) {
throw new AppError("Student not found",404)
}
const marks=course.total
const studentMarks=course.enrolled[studentIndex].totalMarks
const studentPerc=studentMarks*100/marks
if (studentPerc < 75) {
throw new AppError("You need 75% or above to get the certificate", 400);
}
const doc=new pdfDocument({layout: 'landscape', size: 'A4'})
```

```
const fileName = `certificate_${user.username}_${course.title}.pdf`;
res.setHeader('Content-Disposition', `attachment; filename="${fileName}"`);
doc.pipe(res)

doc.rect(0, 0, doc.page.width, doc.page.height).fill('#fff');
const distanceMargin = 20;
doc.fillAndStroke('#5a77d8')
.lineWidth(20)
.lineJoin('square')
.rect(distanceMargin,distanceMargin,
doc.page.width - distanceMargin * 2,
doc.page.height - distanceMargin * 2,
)
.stroke();

const logoWidth = 100; // Adjust the width of the logo as needed
const logoX = (doc.page.width - logoWidth) / 2;
doc.image('./uploads/logo.png',logoX,25 ,{
fit: [logoWidth, 100],
align:'center'
}) .moveDown(0.5)
.fontSize(15).text('SKILL BITES',{align:'center'});

doc.moveDown(3)
doc.font('Helvetica-Bold').fontSize(40).fill('#000').text(`CERTIFICATE OF COMPLETION`, {
align: 'center' });
doc.moveDown()
doc.fontSize(14).fill('#000').text(`This is to certify that`, { align: 'center' });

doc.font('fonts/Satisfy-Regular.ttf').fontSize(35).fill('#000').text(`${user.username}`, { align:
'center' });

doc.font('Times-Roman').fontSize(14).fill('#000').text(`has successfully completed the course`, {
align: 'center' });
doc.moveDown()
doc.font('Times-Roman').fontSize(26).fill('#000').text(`${course.title}`, { align: 'center' });
doc.moveDown(0.5)
doc.font('Times-Roman').fontSize(14).fill('#000').text(`with an estimated`, { align: 'center' })
doc.font('Times-
Roman').fontSize(14).fill('#A4AD32').text(`${studentPerc.toFixed(2)+'%'}`,{align:'center',underl
ine:true});
doc.moveDown(2)
```

```
doc.font('Times-Roman').fontSize(14).fill('#000').text(`Date: ${new
Date().toLocaleDateString()}`, { align: 'center' });
doc.end();
}
catch(err){
next(err)
}
})

app.get('/stats',verifyToken,async(req,res,next)=>{
try{
const {role}=req.user;
const courseId=req.query.courseId
if(role!=='Admin'){
throw new AppError('You are restricted from accessing this route',400)
}

const studentCount = await User.countDocuments({ role: 'Student' });
const instructorCount = await User.countDocuments({ role: 'Instructor' });
const courseCount=await Course.countDocuments({approved:true})
const course=await Course.findById(courseId)
const purchases=await Purchase.find().populate('courseId',['title'])
const paymentIds = purchases.map(purchase => purchase.paymentId);
const payments = await razorpay.payments.all({count:100});
let totalRevenue = 0;

for (const payment of payments.items) {
if (paymentIds.includes(payment.id) && payment.status === 'captured')
totalRevenue+=payment.amount/100
}

res.status(200).json({success:true,studentCount,instructorCount,courseCount,totalRevenue})
}
catch(err){
next(err)
}
})
app.get('/performanceStats/:id',verifyToken,async(req,res,next)=>{
try{
const info=req.user
const {id}=req.params
let s_id=req.query.s_id
if(!s_id&&info.role==='Student'){
```

```
s_id=info.id
}
const course=await Course.findById(id)
if(!course){
throw new AppError("Course not found",404)
}
const marks=course.total
const studentIndex = course.enrolled.findIndex(enroll => String(enroll.student) ===s_id);
if (studentIndex === -1) {
throw new AppError("Student not found",404)
}
const noOfLectures=course.recordings.length;
const noOfAssignments=course.assignments.length;
const studentMarks=course.enrolled[studentIndex].totalMarks
const studentPerformance=parseFloat(studentMarks*100/marks)
res.status(200).json({succes:true,studentPerformance,noOfLectures,noOfAssignments})
}
catch(err){
next(err)
}
})

app.post("/logout", (req, res) => {
res.cookie("token", "").json("ok");
res.send("logged out");
});

//Creating databases
app.listen(process.env.PORT, () => {
console.log(`Server is running in port ${process.env.PORT}`);
});
app.use(ErrorHandler)


const mongoose = require("mongoose");
const { Schema, model } = mongoose;
const bcrypt=require('bcryptjs');
const salt = bcrypt.genSaltSync(10);


const UserSchema = new Schema({
username: { type: String, required: [true, "Username is required "] },
email: {
unique: true,
```

```
type: String,
required: true,
lowercase: true,
match: [
/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/,
"Please fill a valid email address",
],
},
role: {
type:String,
enum:['Admin','Student','Instructor'],
},
password: {
type: String,
required: [true,"Please enter a password"],
minlength: 8
},
photo:{
type:String,
default:'https://res.cloudinary.com/djp33bnwu/image/upload/v1710756211/avatar/bmhghzxpe78
gdntcl4zg.jpg',
},
gender:{
type:String,
default:'Male',
},
bio:{
type:String,
default:'N/a',
},
address:{
type:String,
default:'N/a'
},
phone:{
type:String,
default:'N/a'
},
},
{
timestamps:true
});
UserSchema.pre ('save',async function(next){
```

```
if(this.isModified('password')||this.isNew){
// if(this.password!==this.ConfirmPassword){
//   return next(new Error('password does not match'));
// }
this.password= await bcrypt.hashSync(this.password, salt),
// this.ConfirmPassword=undefined;
next();
}
}
const UserModel = model("User", UserSchema);
module.exports = UserModel;


//Course

const mongoose = require("mongoose");
const { Schema, model } = mongoose;

const CourseSchema = new Schema(
{
title: {
type: String,
required: true,
},
summary: {
type: String,
required: true,
},
content: {
type: String,
required: true,
},
cover: {
type: String,
required: true,
},
duration: {
type: Number,
required: true,
},
price: {
type: Number,
required: true,
},
```

```
author: {
type: Schema.Types.ObjectId,
ref: "User",
},
total:{
type:Number,
default:0,
},
enrolled:[{
student:{
type:Schema.Types.ObjectId,
ref:'User',
},
totalMarks:{
type:Number,
default:0,
},
}],
approved:{
type:Boolean,
default:false,
},
recordings: [
{
file: {
public_id: {
type: String,
required: true,
},
secure_url: {
type: String,
required: true,
},
},
filename: {
type: String,
required: true,
},
description: {
type: String,
required: true,
},
watched:[{
```

```
type:Schema.Types.ObjectId,
ref:'User',
}],
createdAt: {
type: Date,
default: Date.now,
},
},
],

assignments: [
{
title: {
type: String,
required: true,
},
description: {
type: String,
},
file: {
type: String,
},
marks:{
type:Number,
default:0,
},
createdAt: {
type: Date,
default:Date.now()
},
submit: [
{
student: {
type: Schema.Types.ObjectId,
ref: "User",
required: true,
},
file: {
type: String,
},
description: {
type: String,
},
```

```
submittedAt: {
type: Date,
default:Date.now(),
},
status: {
type: String,
enum: ["pending", "submitted"],
default: "pending",
},
marks: {
type: Number,
default: 0,
},},
],},],},
{
timestamps: true,
}
);

const CourseModel = model("course", CourseSchema);
module.exports = CourseModel;

//Orders
const mongoose = require('mongoose');
const {Schema,model}=mongoose;
const PurchaseSchema = new Schema({
  userId: {
    type:Schema.Types.ObjectId, ref: 'User'
    },
  courseId: {
     type: Schema.Types.ObjectId, ref: 'course'
    },
  paymentId:{
    type:String,
  }

});

const purchaseModel=model('Purchase',PurchaseSchema);

module.exports=purchaseModel


//Contact
```

```
const mongoose = require('mongoose');

const {Schema,model}=mongoose;

const PurchaseSchema = new Schema({
userId: {
type:Schema.Types.ObjectId, ref: 'User'
},
courseId: {
type: Schema.Types.ObjectId, ref: 'course'
},
paymentId:{
type:String,
}

});


const purchaseModel=model('Purchase',PurchaseSchema);

module.exports=purchaseModel
```

7. Testing

7.1 Introduction

In this software testing has been defined as the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. Software testing is the process used to assess the quality of computer software. It involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they should.

7.2 Testing Objectives

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality
- To prevent defects
- To make sure that the end results meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specification.

7.3 Testing steps

7.3.1 Unit Testing

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

7.3.2 Integration Testing

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole.

7.3.3 Validation Testing

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

- The function or performance characteristics confirm to specification and are accepted.
- A deviation from specification is uncovered and a deficiency lists is created.
- Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

7.3.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. Also for hard copy the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

7.3.5 User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point:

- Input screen design.
- Output screen design.
- Online message should be guide to the user.

Login

Input	Test condition	Test output	Comments
Login	If the admin/ Instructor/Student email_id is not entered.	Please fill out this field.	Username is required
	If the password is not entered	Please fill out this field	Password is required
	If the username and password are not valid	User not found	Enter valid username and password
	If the student/Instructor email_id and password are valid.	Successfully logged in.	Appropriate password and email is entered and the users are now logged in.
	If the admin email_id and password are valid.	Successfully logged in.	Appropriate password and email is entered and the Admin is now logged in to the admin page.

Register

Input	Test condition	Test output	Comments
Register	If the username is not entered	Please fill out this field.	Username is required
	If the email_id is not entered	Please fill out this field.	Email is required
	If the entered email_id exists	User already exists	Register through another email_id
	If the password is not entered	Please fill out this field. Minimum 8 letters required	Password required
	If the above fields are valid.	Registered successfully	User registered

7.4 Integration Testing

Admin Login

Test No.	Test condition	Test output	Expected output
1	Admin login	The site is redirected to admin page	Success

Instructor/Student Register and login page

Test No.	Test condition	Test output	Expected output
1	Register	The entered data is stored in database and the site is redirected to home page	Success
2	Login	The site is redirected to homepage	Success

Admin Index page

Test No.	Test condition	Test output	Expected output
1	Dashboard	The site is redirected to Admin Dashboard page	Success
2	Manage Students	The site is redirected to Students list page	Success
3	Manage Instructors	The site is redirected to Instructors list page	Success
4	Manage Courses	The site is redirected to Courses list page	Success

5	View Feedbacks	The site is redirected to feedbacks list page	Success

Student Index page

Test No.	Test condition	Test output	Expected output
1	Manage Profile	The site is redirected to Student Profile page	Success
2	View Course List	The site is redirected to all Courses page	Success
3	View Courses	The site is redirected to Purchased courses page to view course content	Success
4	View purchases	The site is redirected to purchases page to view the order details information	Success
5	Logout	The site is redirected to homepage	Success

Instructor Index page

Test No.	Test condition	Test output	Expected output
1	Manage Profile	The site is redirected to Instructor Profile page	Success
2	View Course List	The site is redirected to all Courses page	Success
3	Create Course	The site is redirected to Create Course page	Success
4	View Courses	The site is redirected to view created course page	Success
5	Logout	The site is redirected to homepage	Success

View Course page(Instructor)

Test No.	Test condition	Test output	Expected output
1	Add assignment	Opens the add Assignment modal	Success
2	Add recordings	Opens the add Recordings modal	Success

3	View student assignments	The site is redirected to Student assignments page	Success
4	View lectures	The site is redirected to view lecture page	Success
5	View student performance	The site is redirected to performance page	Success

View Course Page(Student)

Test No.	Test condition	Test output	Expected output
1	View lectures	The site is redirected to view lectures page	Success
2	View Assignments	The site is redirected to view assignments page	Success
3	Submit assignment	Opens the submission box	Success
4	View course performance	The site is redirected to performance page	Success

7.5 System Testing Table

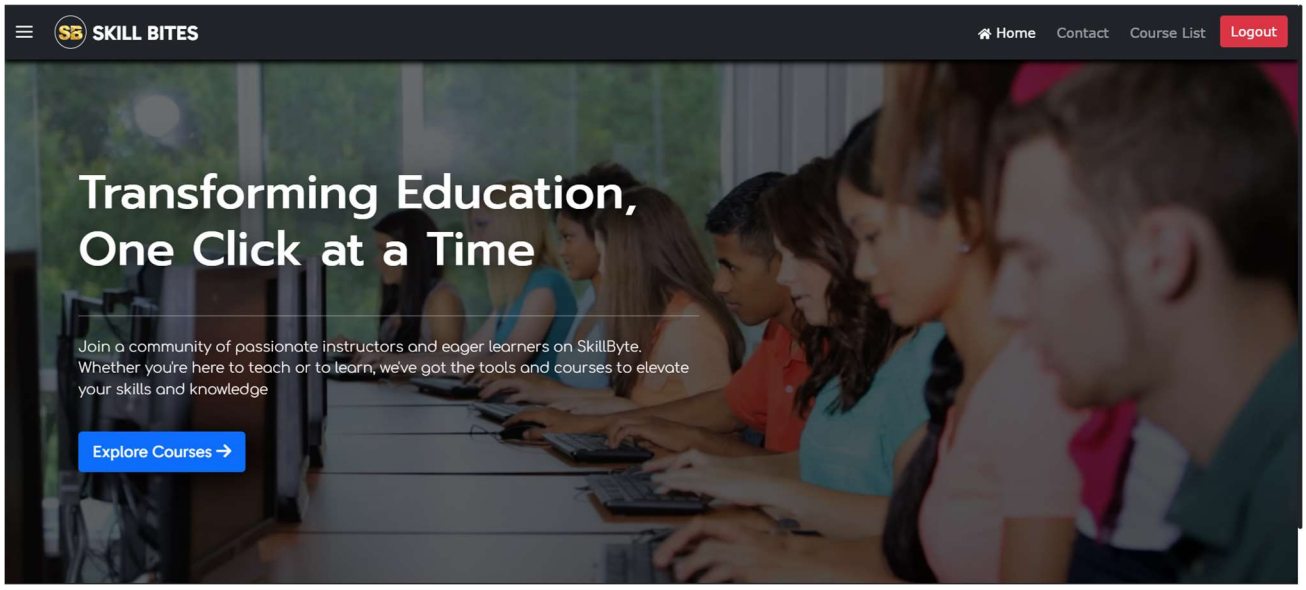
System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

Sl.No	Test condition	Test report
1	System lading	Successful
2	System run procedure	Successful
3	File I/O operation	Successful
4	Database communication	Successful
5	Server/client interaction	Successful
6	Memory usage	Normal
7	System processor usage	Normal
8	Authentication/Authorization	Successful

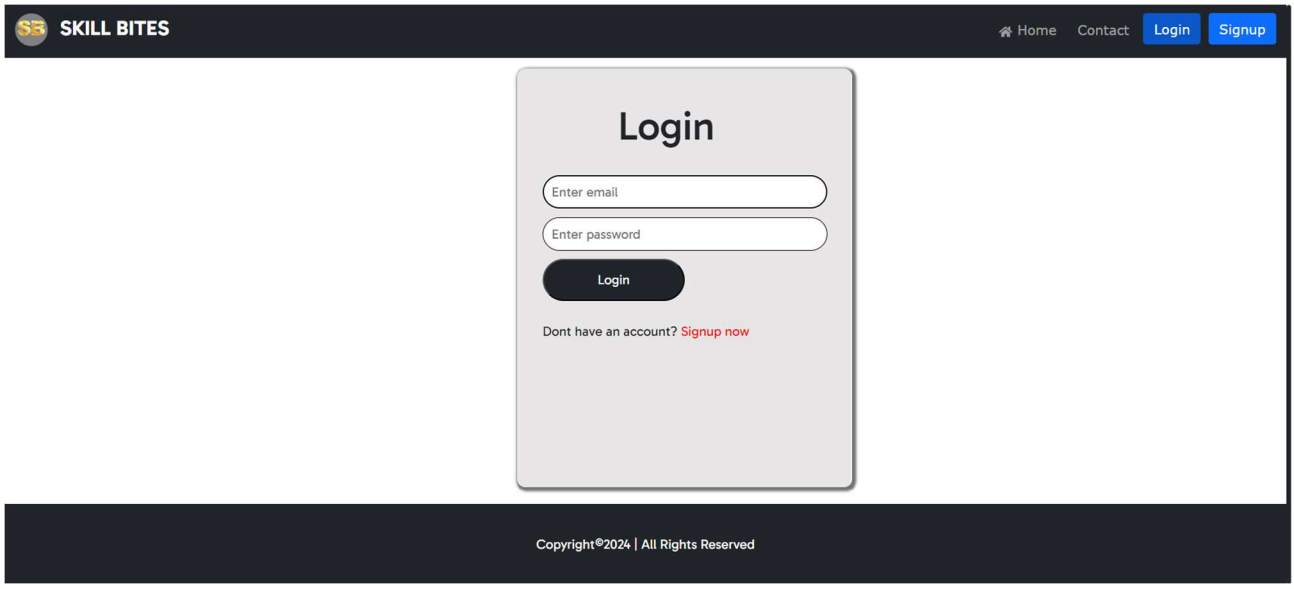
8. User Interface

8.1 Screenshots:

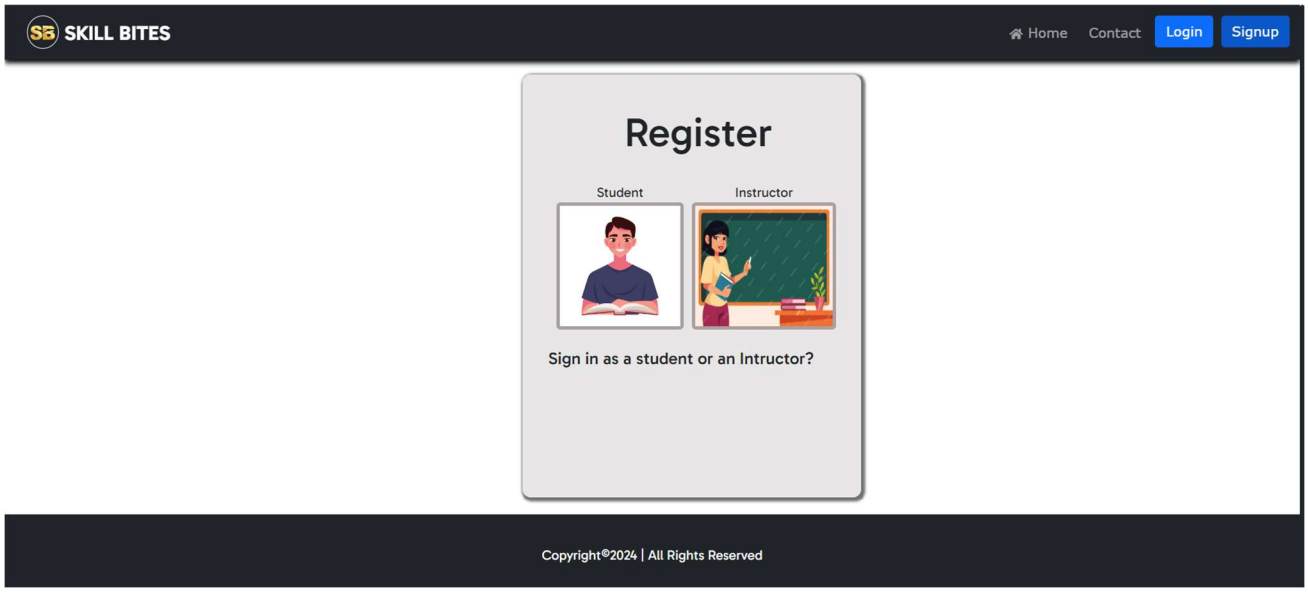
Homepage

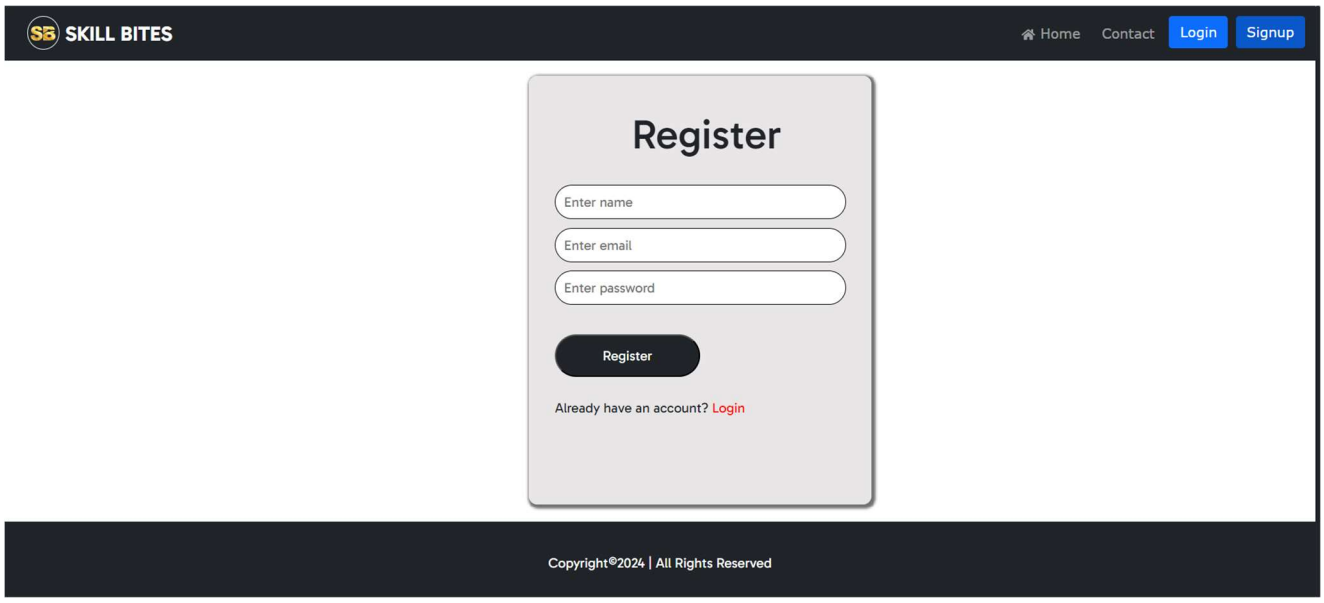


Login:

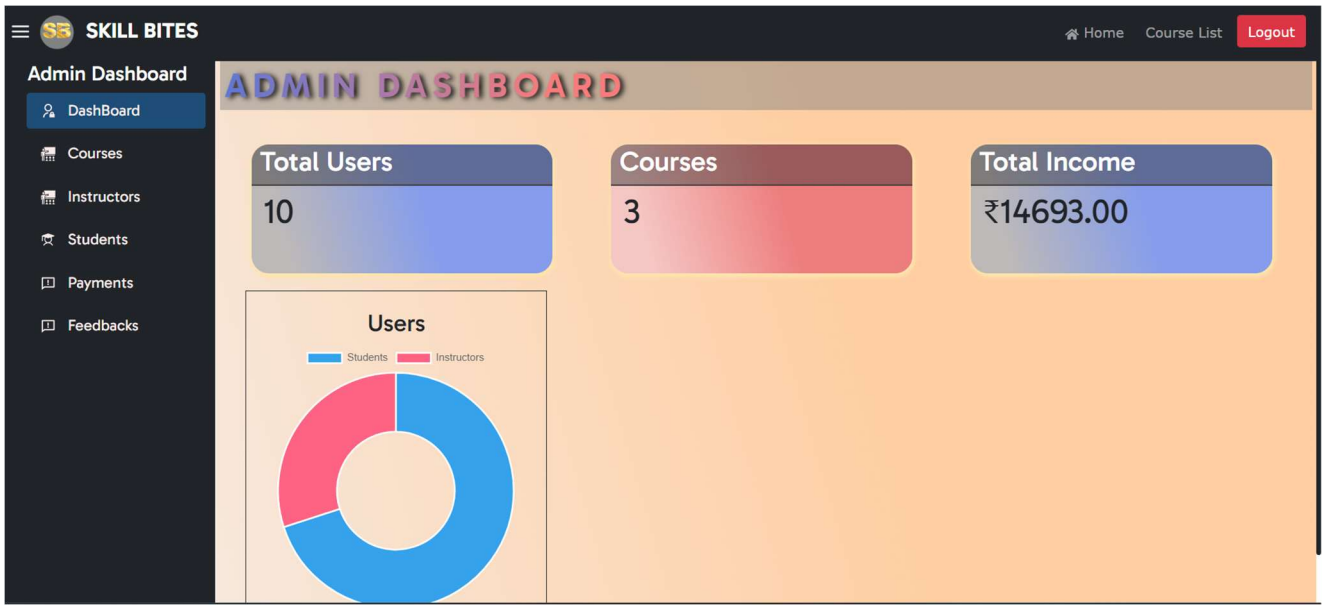


Register page

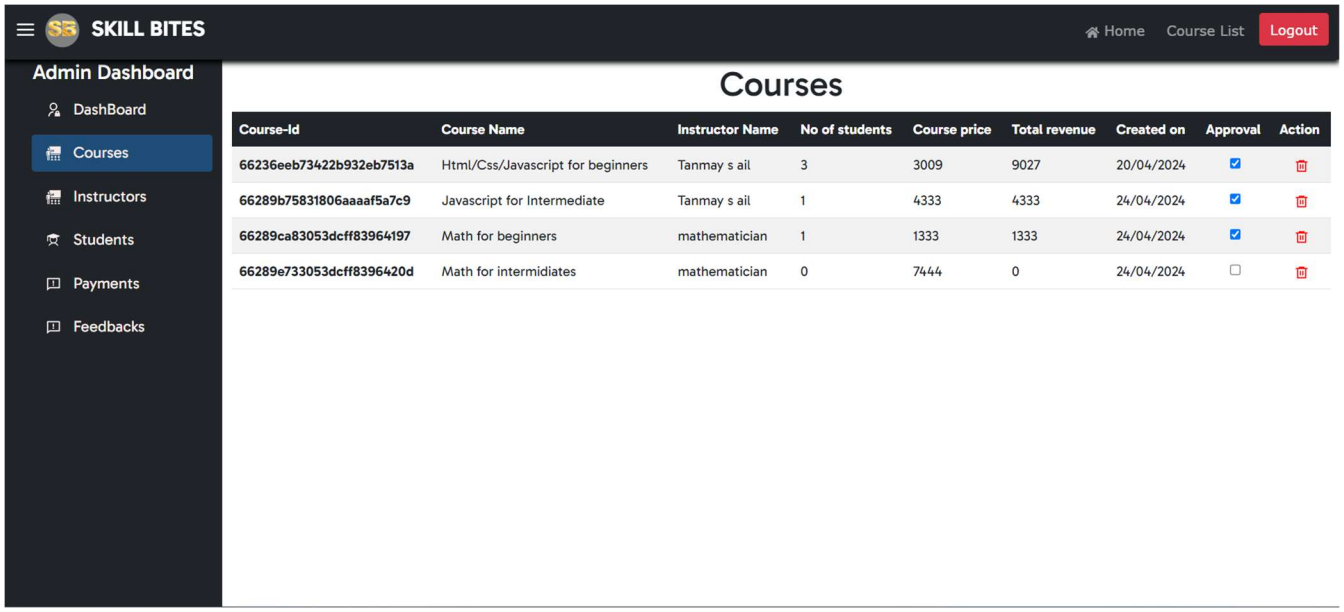




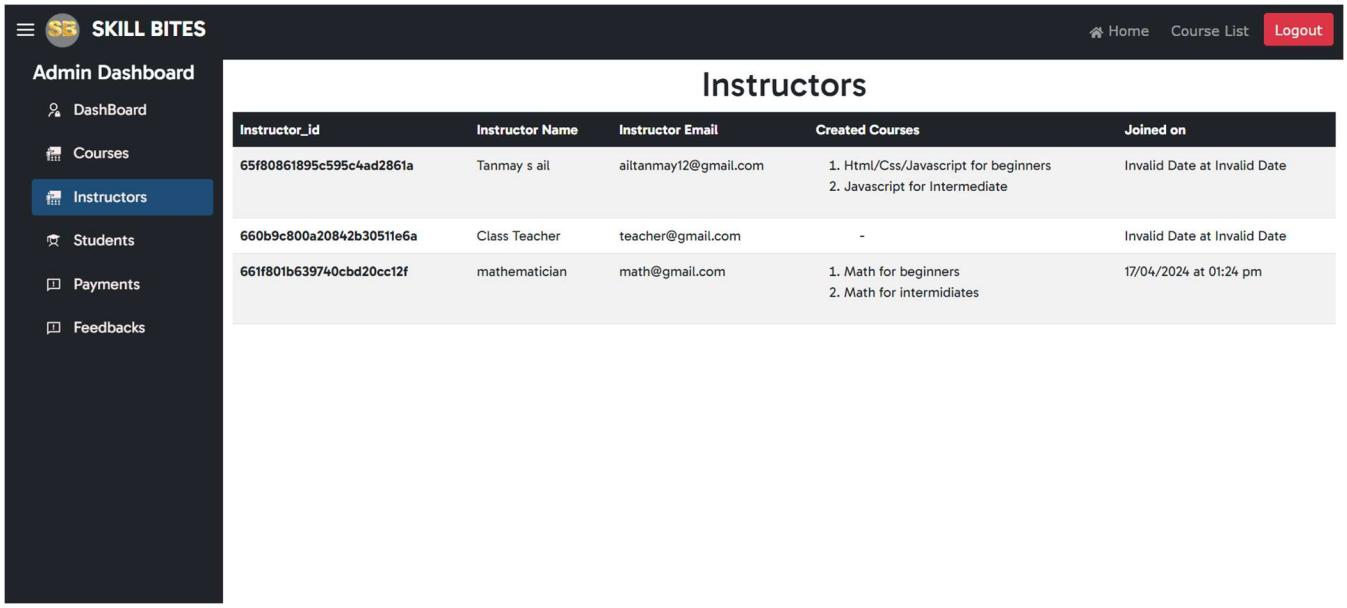
Admin Dashboard page



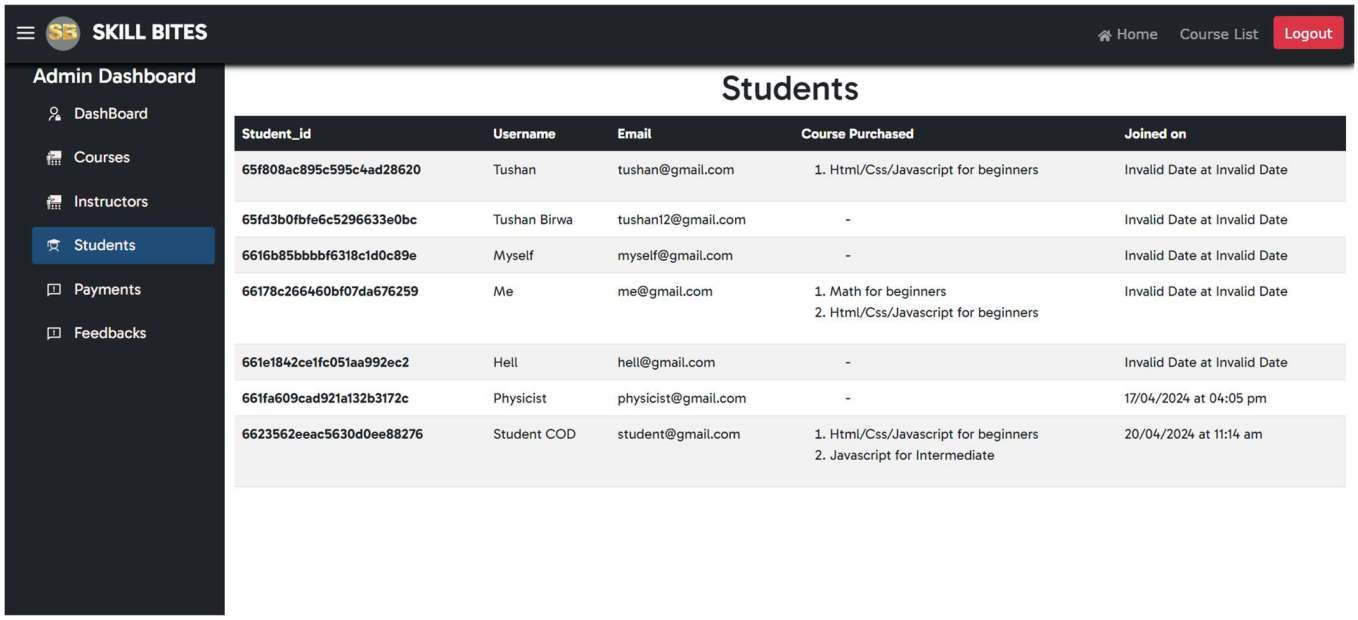
Manage Courses page



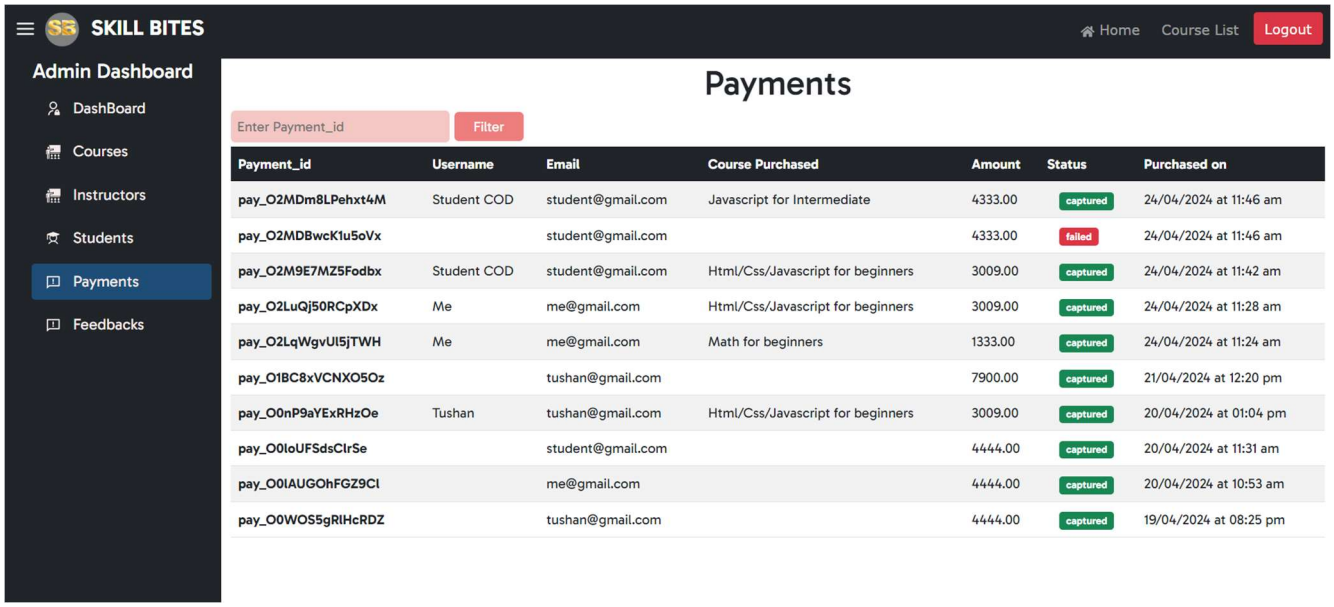
Manage Instructor page



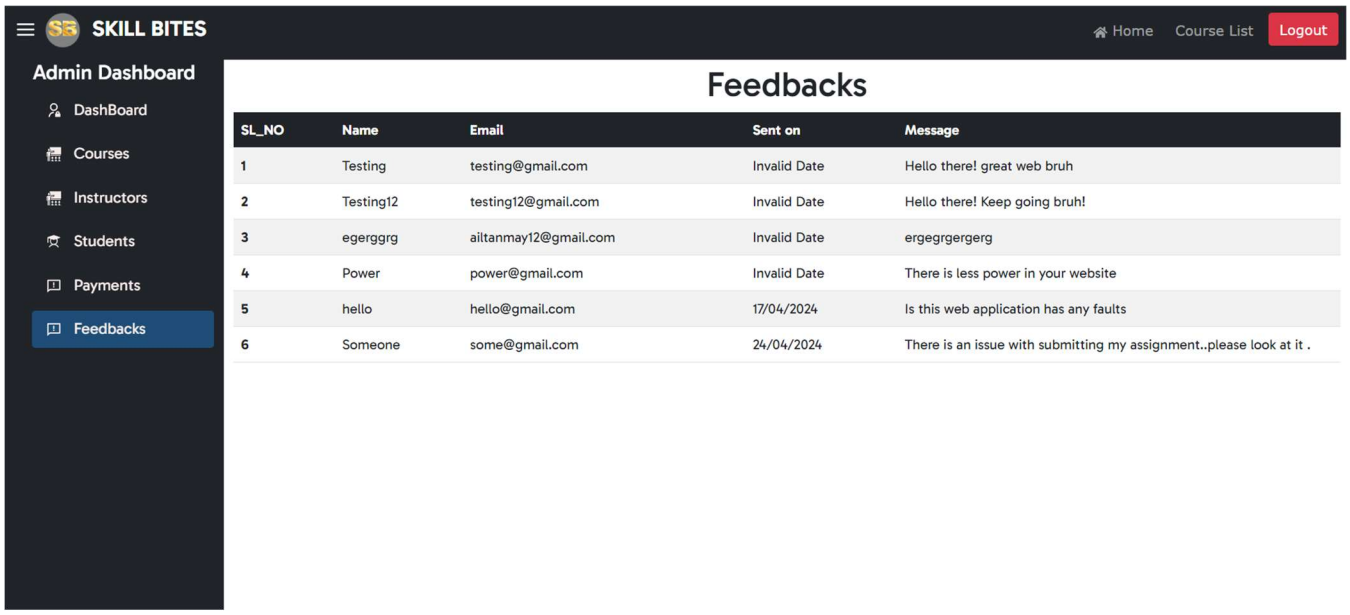
Manage Students page



Manage Payments page

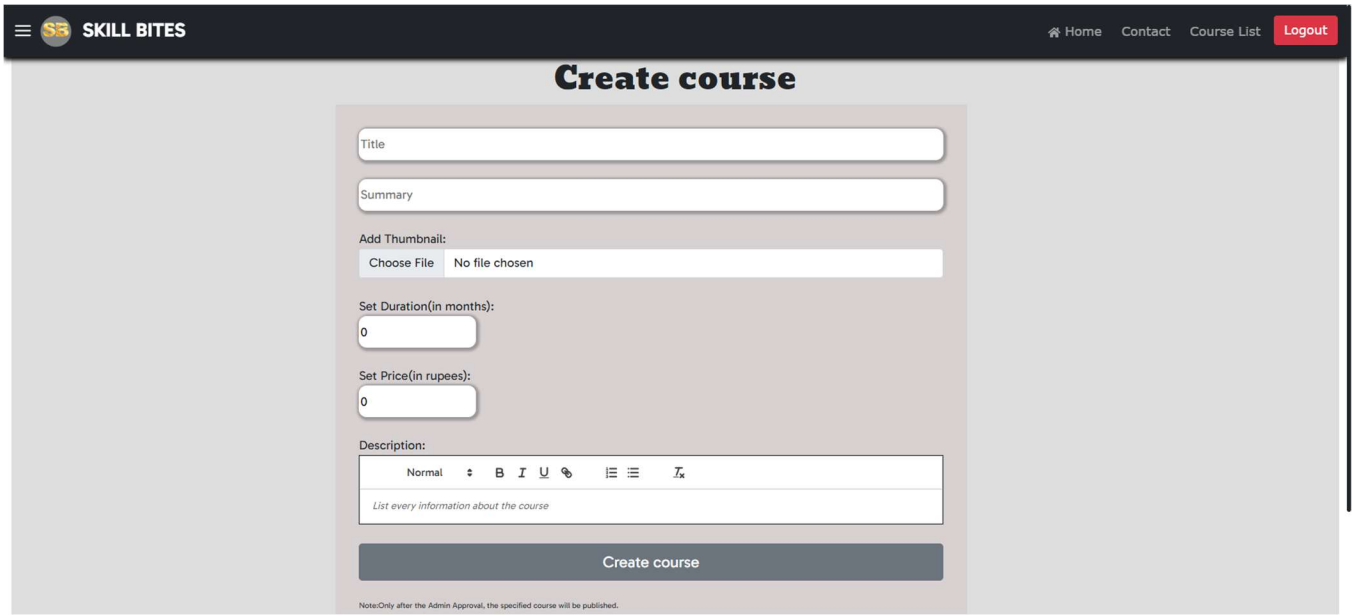


View Feedbacks page

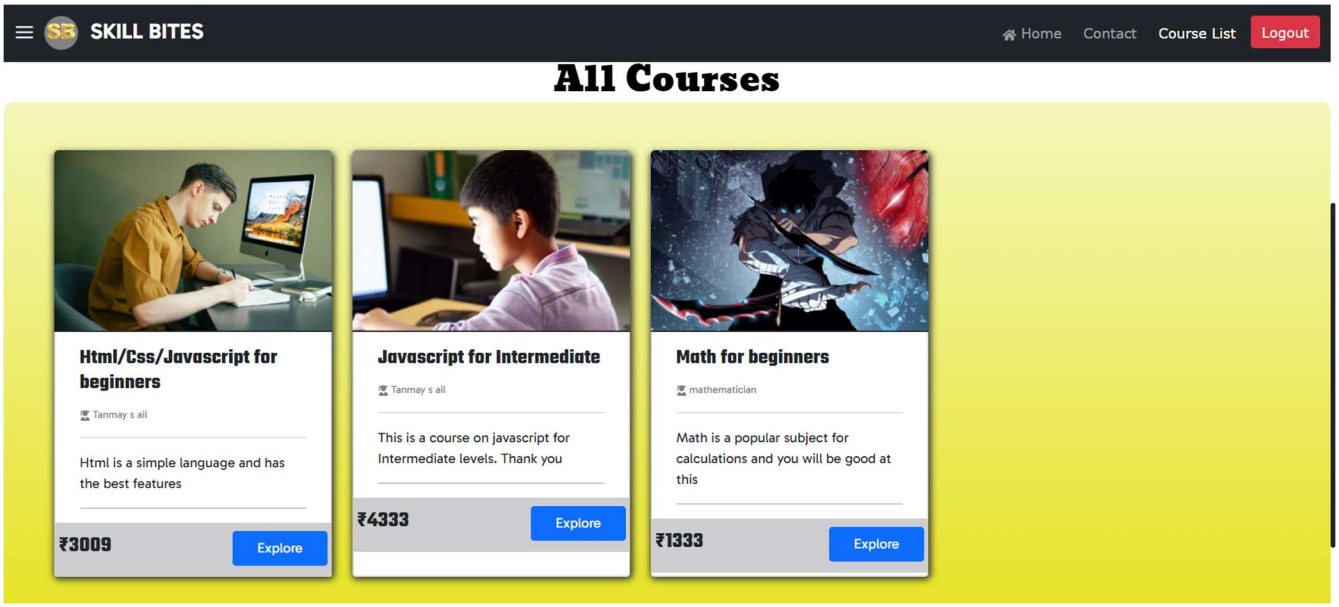


Profile page

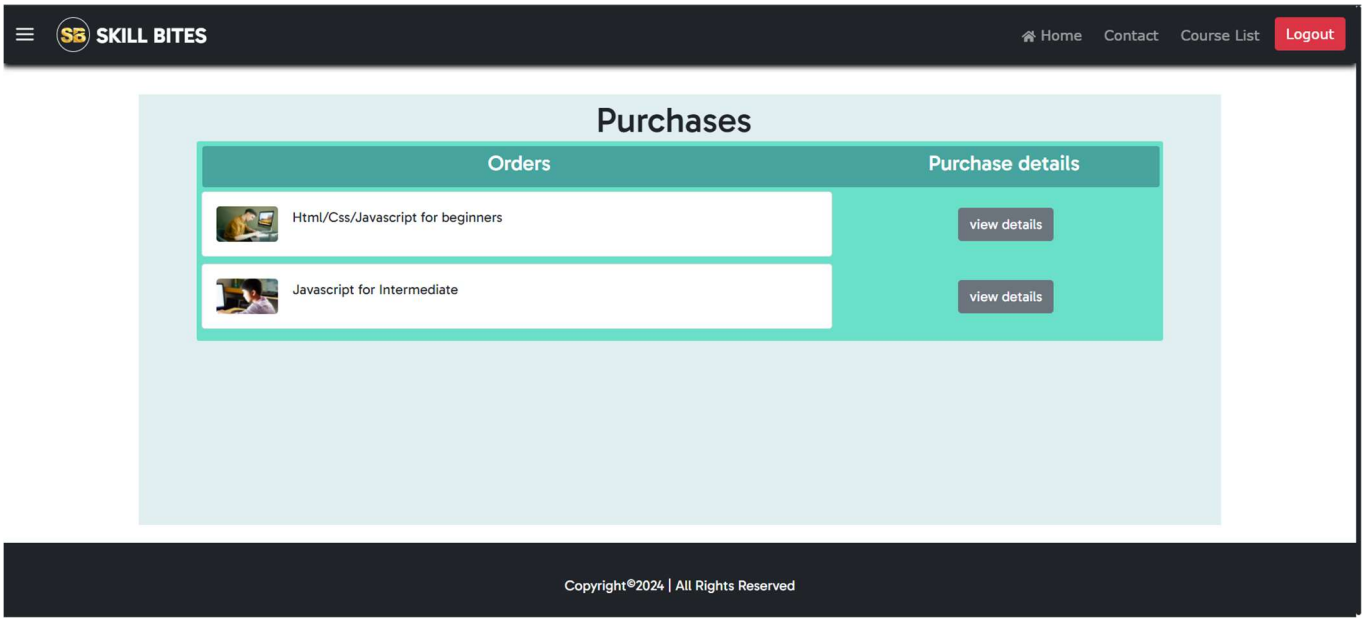
Create Course page



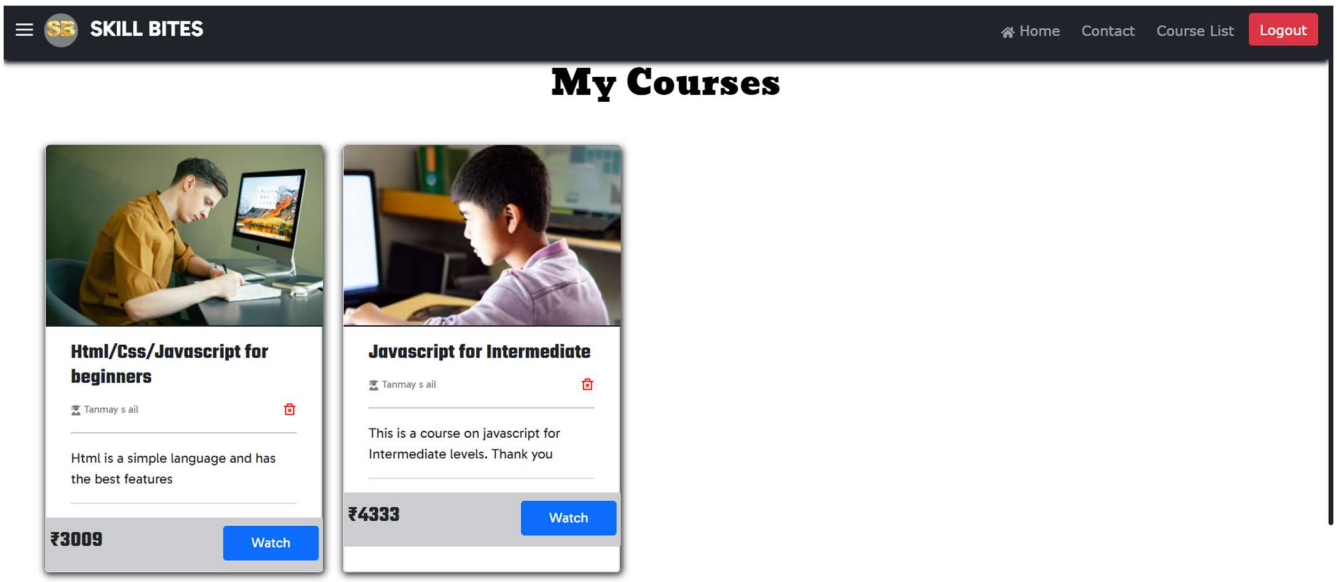
View All Courses page



View Purchases

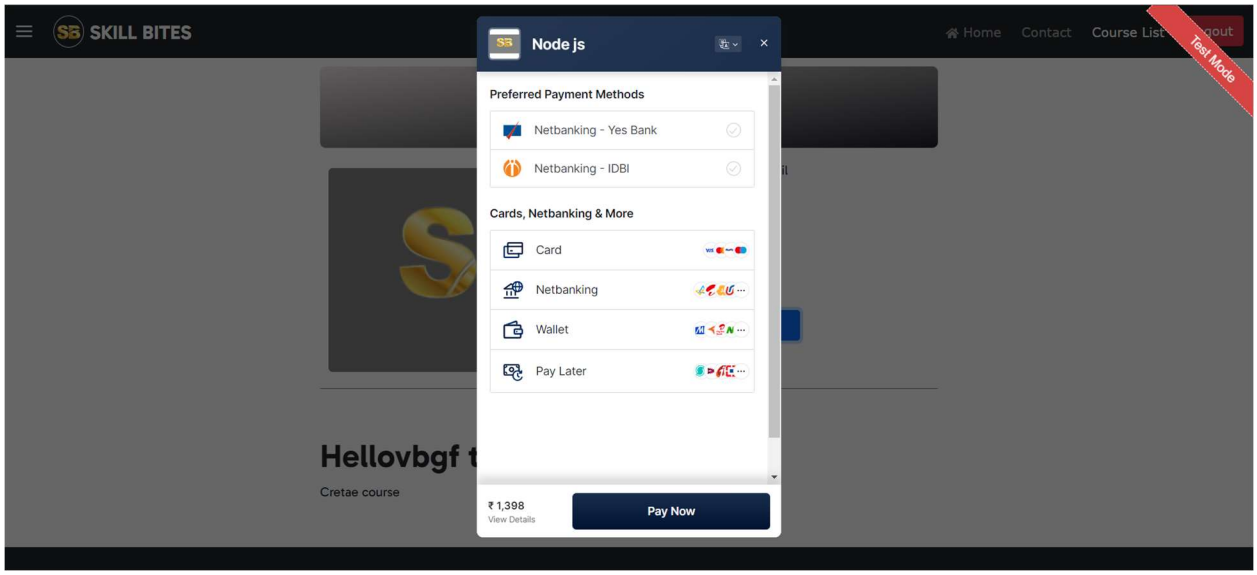


View Created / Purchased courses

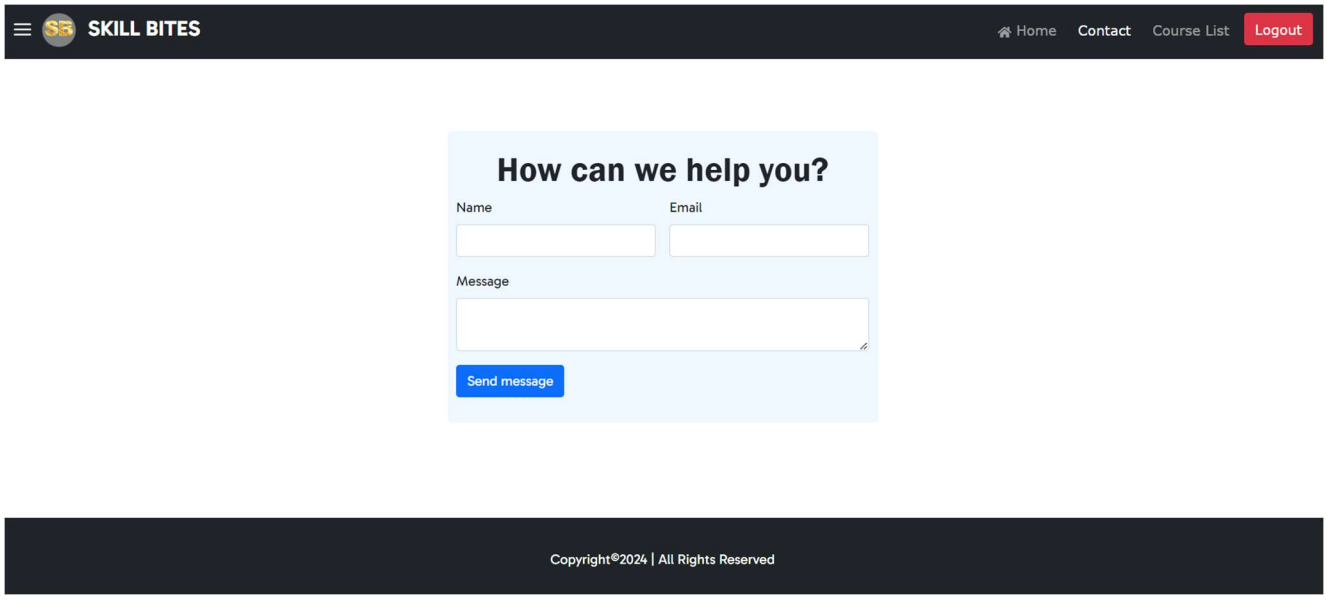


View Course description

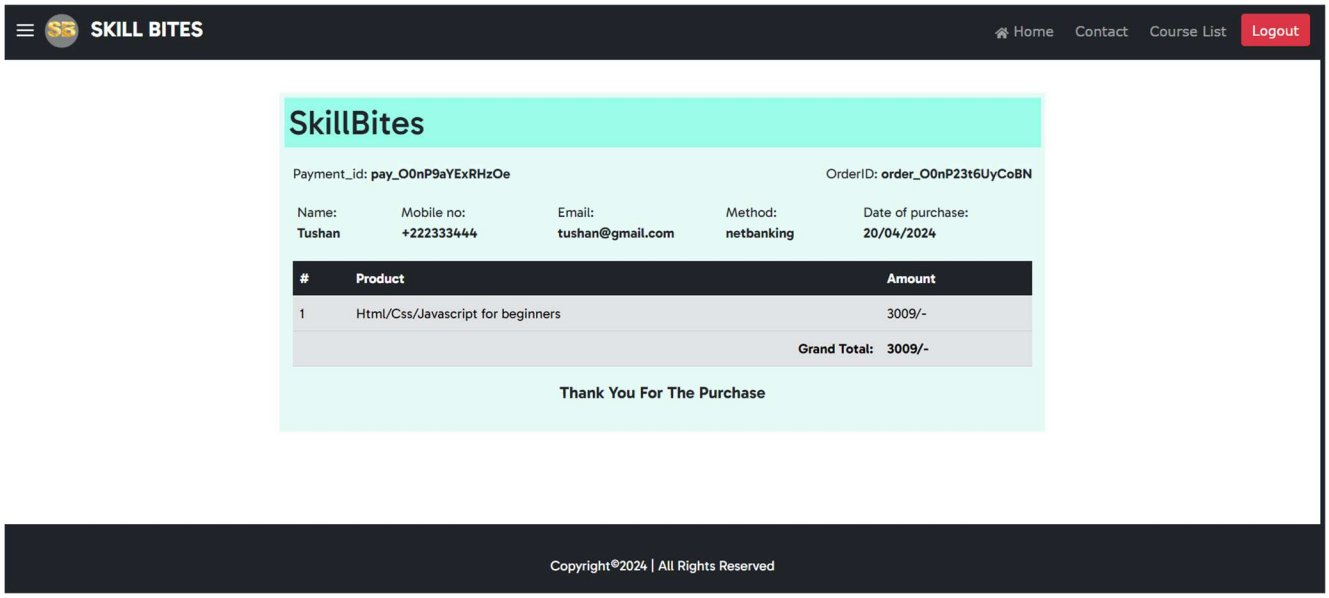
Payment page



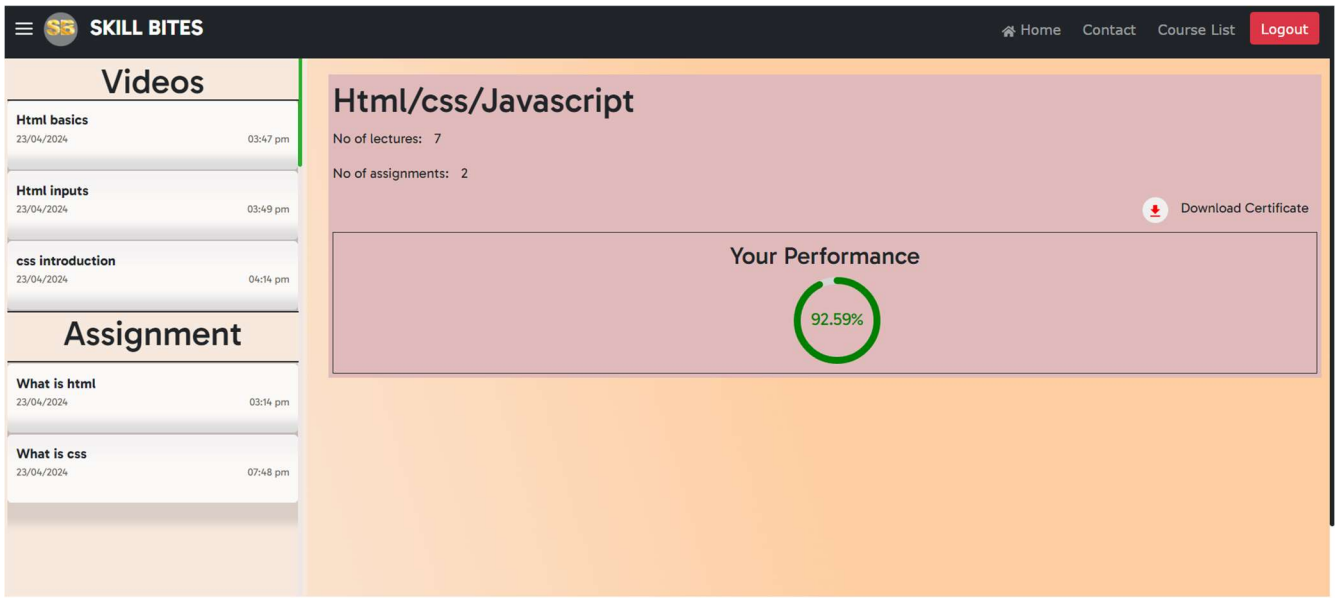
Contact page



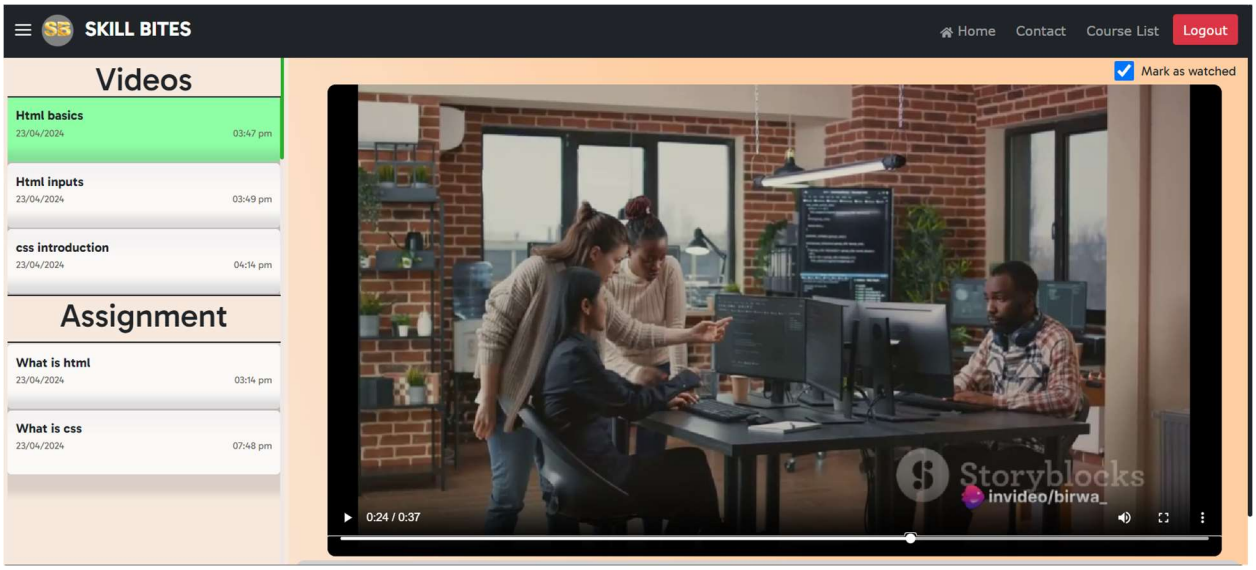
Purchase Page



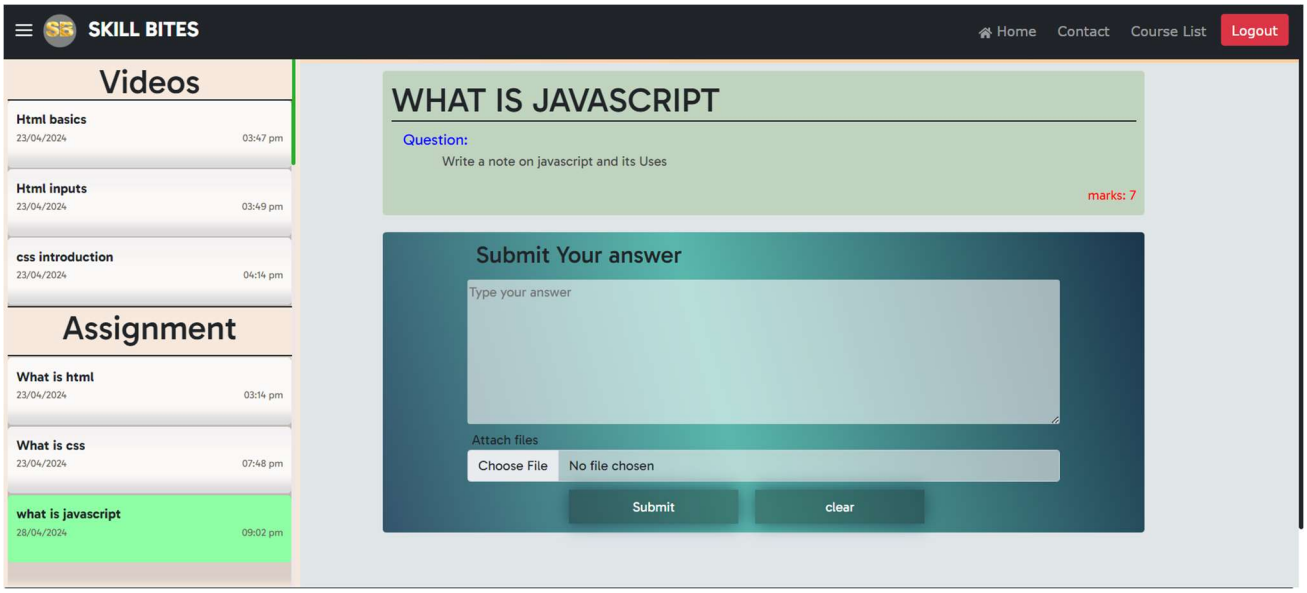
View Course Content with Performance page



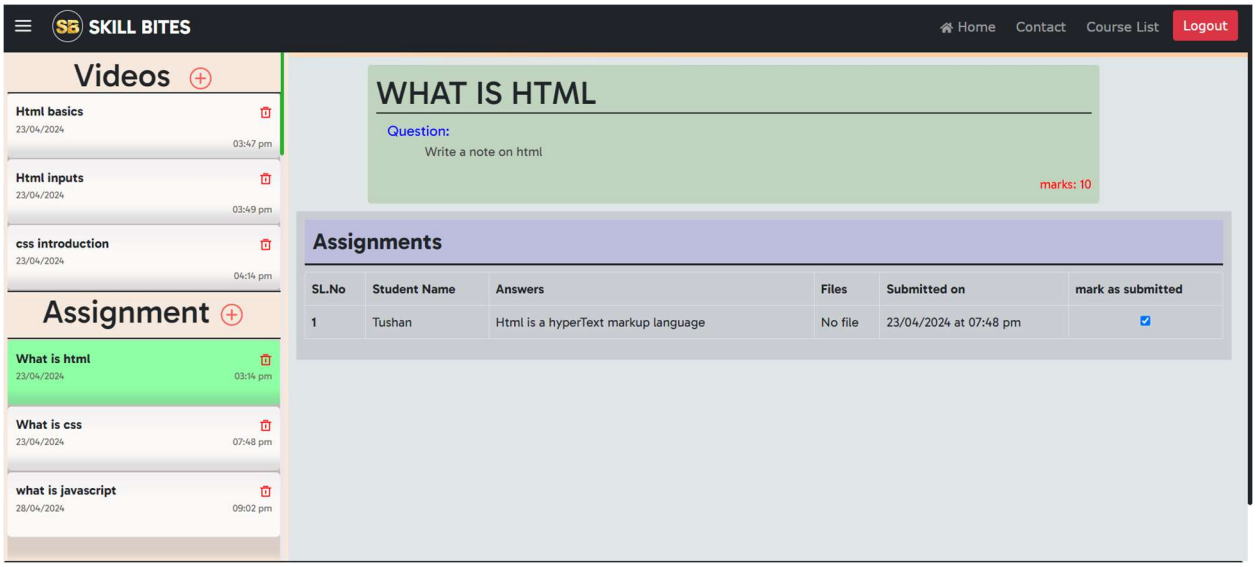
View Recordings



View Assignments



View Student assignments



Generate Certificate

9. User Manual

9.1 Introduction

The Skill Bites provide a platform for the instructors who is willing to start a paid online course to the students who want to learn specific courses. In these modern days there aren't people who is willing to teach students

9.2 Hardware Requirements

- Intel dual core processor or higher
- 4gb RAM and above
- 40gb hard disk
- Mouse/Keyboard

9.3 Software Requirements

- Language:Nodejs, ReactJS
- User interface design:HTML, CSS, BOOTSTRAP
- Scripting language: javascript, ReactJS
- Operating system:Windows
- Editor: Visual studio code
- Database:MongoDB
- Browsers: google chrome, microsoft edge,etc

10. Conclusion

The project "SkillBites" is a website-based initiative aimed at facilitating instructors in offering paid online courses to students interested in learning specific subjects. With its user-friendly website interface, SkillBites empowers instructors to effortlessly create and deliver high-quality courses while providing them with a seamless means to earn from their expertise. For students, SkillBites offers an immersive learning journey with interactive course materials, video recordings, assignments, and with additional resources. Navigating through the platform is not only straightforward but also enjoyable, fostering an ideal environment for knowledge absorption and skill development. By bridging the gap between instructors and learners, it fosters a vibrant community of knowledge exchange and personal development.

Future Scope:

- In present times where online education is gaining worldwide popularity, SkillBites has major advantage over this trend.
- Providing modules for community discussion tabs enabling instructors to engage in knowledge-sharing with their peers and facilitating student-to-student interactions, SkillBites can foster a vibrant learning community.
- Moreover, Integrating AI tools and real time video communication systems can provide better significance for the growth of SkillBites.

11. Bibliography

11.1 Web references:

- Bootstrap Documentation:

  URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/

- NodeJs Documentation:

  URL: https://nodejs.org/docs/latest/api/

- Razorpay Documentation:

  URL: https://razorpay.com/docs/api/

- Google Fonts:

  URL: https://fonts.google.com/

- React Icons documentation:

  Url: https://react-icons.github.io/react-icons/search/#q=

DrillBit

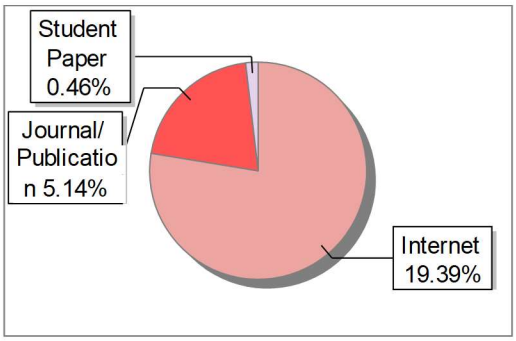
The Report is Generated by DrillBit Plagiarism Detection Software

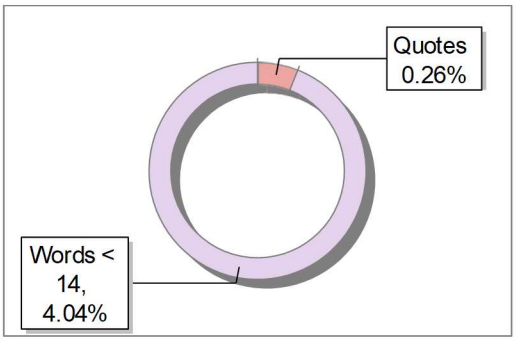
Submission Information

Author Name	Tanmay s ail and Tushan
Title	Skill Bites
Paper/Submission ID	1713137
Submitted by	simslibrarianmng@gmail.com
Submission Date	2024-04-29 11:40:08
Total Pages	37
Document type	Project Work

Result Information

Similarity **25 %**





Exclude Information

Quotes	Not Excluded
References/Bibliography	Excluded
Sources: Less than 14 Words %	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

25	35	B
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	pdfcoffee.com	8	Internet Data
2	vdocuments.mx	2	Internet Data
3	technodocbox.com	1	Internet Data
4	www.aalimec.ac.in	1	Publication
5	pt.slideshare.net	1	Internet Data
6	qdoc.tips	1	Internet Data
7	helplifeglobal.org	1	Internet Data
8	qdoc.tips	1	Internet Data
9	www.jntua.ac.in	1	Publication
10	www.slideshare.net	1	Internet Data
11	anilkbotta.blogspot.com	1	Internet Data
12	doku.pub	1	Internet Data
13	ignou.ac.in	<1	Internet Data
14	INFORMATICSwww.ijemr.net	<1	Publication

15	www.scribd.com	<1	Internet Data
16	qdoc.tips	<1	Internet Data
17	sist.sathyabama.ac.in	<1	Publication
18	qdoc.tips	<1	Internet Data
19	www.ccsenet.org	<1	Publication
20	INFORMATICSijera.com	<1	Publication
21	qdoc.tips	<1	Internet Data
22	Developing a comprehensive framework for eutrophication management in by Khorasani-2018	<1	Publication
23	Emerging Technologies, IT Infrastructure, and Economic Development in Mexico by Mejias-1999	<1	Publication
24	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
25	moam.info	<1	Internet Data
26	An analysis of medical device-related errors prevalence and possible solutions by Ward-2004	<1	Publication
27	ncert.nic.in	<1	Publication
28	Submitted to Visvesvaraya Technological University, Belagavi	<1	Student Paper
29	www.mdpi.com	<1	Internet Data
30	Contributions to Economic Analysis Dynamic General Equilibrium Mode, by Dixon, Peter B. Ri- 2001	<1	Publication
31	ejournal.undip.ac.id	<1	Internet Data
32	hnlu.ac.in	<1	Publication