Sri Lanka Institute of Information Technology

Distributed Systems - SE3020

Assignment

Learnify - Development of an Educational Platform for Online Learning

S2-SE-WE-44

	Student Registration Number	Student Name
1	IT21318320	Silva T.U.D
2	IT21189944	Madusanka G.K.I
3	IT21223594	Thalangama T.P
4	IT21319174	Dissanayake M.G.T.W

Table of Contents

Table of Figures

Introduction

Learnify is an online educational platform which enable learners to browse through a variety of course list and follow them in order to improve their knowledge. The application has been developed using several technologies to cater to the differed requirements. As the initiate step design phase was conducted to make a sketch of the Learnify platform and Figma was used to it. Then the development phase was started, and the user interfaces has been created in a manner which attracts users to the application with eye catching images and animations, and to enhance the user experience different frameworks have been used to ensure the responsiveness and the compatibility. To ensure security, session management and for the backend services a separate backend code have been developed. As the tools and technologies MERN stack has been used. React JS has been used for the front-end development along with tailwind CSS for styling. For the backend NodeJS along with Express have been used and to store data MongoDB has been used. To serve the application it has been deployed using Docker and Kubernetes. For the payment gateway and to send notifications via mail and SMS, Stripe, Nodemailer and Notify.lk has been used respectively. Overall, Learnify caters functionalities for main three roles including learner, instructor, and admin. And the main services are authentication service, user service, course service, learner service, payment service and notification service. In summary, Learnify provides all the capabilities for the learners to make their learning experience interactive and effective.

Services

API Gateway

The API Gateway in Learnify serves as the primary access point for clients who want to interact with the microservices based architecture in the online learning platform. The advantage of implementing the API gateway reveals a pathway for all types of clients including web browsers and mobile apps to access the developed services. And also the gateway hides the implementation complexity hence provides a high level of abstraction. Main services provided

by this gateway are request routing, protocol translation, authentication and authorization, request and response transformation, load balancing and rate limiting. The API gateway directs received requests from clients to the appropriate microservice within the system. This functions based on the provided route configurations ensuring accurate request forwarding to the intended service and provides a high level of abstraction. Furthermore this service has the capability to support many communication protocols and initially for Learnify platform HTTP protocol is being supported. In this service it has the ability to handle authentication and authorization by validating credentials and tokens preventing any unexpected and unprotected access to the services. And also this reduces the necessity of performing authentication and authorization mechanisms in each individual microservices. The rate limiting feature ensures the protection against excessive resource consumption from clients incoming requests. Overall, the API Gateway plays a main role in managing request communications and protecting microservices.

Authentication Service

Authentication service is mainly focused on security aspects ensuring to provide a more secure authentication feature for the end users. There are multiple functions which are performed by the authentication service including user registration, user authentication, session management, token-based authentication, access control. In user registration it provides the feature for users to provide the required data on registering to the system and then validates the details and stores them securely in the databases. In this process username validations, password validation will be performed in order to strengthen the security and also when storing the data encryptions will be taken in place. Once registered, users can use their own username and passwords to access the system functionalities. In this process the provided username and password will be checked against the data saved inside the database by using cryptographic techniques such as hashing to securely store and compare passwords. After authenticating the user an authentication token will be granted which then can be used to prove the identity and access rights. Furthermore the authentication service takes care of access control policies based on role-based access. This ensures the user with specific roles such as learner, instructor and admin only can access the allowed tasks inside the Learnify platform.

User Service

The user service manages user-related data and provides functionality of profile management, to ensure personalized user experiences across the platform. The User Service facilitates the registration process for new users joining the platform. Users can manage and update their profiles through the User Service, allowing them to modify personal information, preferences, and settings according to their evolving needs and preferences. The User Service provides endpoints for users to update profile details such as contact information, profile pictures, and notification preferences, ensuring that user profiles remain accurate and up to date over time.

Furthermore, the user service provides CRUD (Create, Read, Update, Delete) operations for managing user data, enabling functionalities such as user search, retrieval, creation, and deletion. In summary, the user service is a component, responsible for managing user-related data and interactions to ensure personalized, secure, and seamless user experiences across the platform. By providing functionalities for profile management it enables the platform to deliver value to users.

Course Service

The course service is designed to manage the educational courses offered on the Learnify platform. And it has functionalities to facilitate the creation, management of courses, ensuring an engaging learning experience for users. The course service provides functionality for course creation, allowing administrators and content creators to author and publish new courses on the platform. It supports the creation of course content, including text, multimedia, questionnaires, providing a flexible environment for course creators. Additionally, it facilitates course management tasks such as updating course content, setting course schedules, and configuring enrollment options to meet the evolving needs of learners and instructors. The course service maintains a comprehensive list of available courses within the platform, organizing courses into categories, topics, and levels to facilitate user discovery and navigation. It supports features such as course search, filtering, and sorting, enabling

users to find and explore relevant courses based on their interests, preferences, and learning objectives. Overall, the course service is a useful component within the microservices architecture of Learnify, responsible for managing the courses and supporting a range of functionalities to deliver personalized, and effective learning experiences for learners. By providing capabilities for course creation, cataloging enables platforms to offer high-quality educational content while enabling learner engagement and satisfaction.

Learner Service

The learner service manages learner-related data and interactions on Learnify. It facilitates tracking learner progress by ensuring a supportive and motivating learning environment for learners. The learner service tracks and records learner progress across courses and learning activities within the platform. It captures data on course enrollment, completion status, assessment scores. By maintaining detailed learner performances, the learner service offers insights into individual learning journeys, helping learners and instructors monitor progress, identify areas for improvement, and celebrate achievements. In summary, the Learner Service is a main component within a microservices architecture, dedicated to improve learner experiences, and motivating learner success on Learnify. By providing capabilities for progress tracking, competency tracking, and performance analytics, it encourages learners to achieve their learning goals, acquire new skills, and thrive in their educational journey.

Payment Service

The payment service, integrated with Stripe, is responsible for facilitating secure and efficient payment processing for course enrollments on Learnify, the online educational platform. The payment service utilizes Stripe's payment processing capabilities to handle financial transactions securely and efficiently. It integrates with Stripe's APIs to initiate payment requests, collect payment information from users, and process transactions using various payment methods, including credit/debit cards. By utilizing Stripe's capabilities, the payment service enables users to complete course enrollments effortlessly, providing an error free payment experience. Security is important in the payment service to protect sensitive

payment information and avoid the risk of vulnerabilities. Therefore it has ensured the security of the user by taking required actions. The payment service tracks payment statuses and updates in real-time, providing users with instant feedback on the result of their payment transactions. It integrates with the notification service in Learnify to send notifications on real-time updates on payment events, such as successful payments. Additionally, it sends automated email notifications to users, confirming successful course enrollments, by interacting with the mentioned notification service. By keeping users informed throughout the payment process, the payment service enhances user experience of the Learnify platform. In summary, the payment service, integrated with Stripe, is responsible for enabling secure, efficient, payment processing for course enrollments on Learnify.

Notification Service

The notification service is responsible for delivering timely notifications to users upon confirmation of course enrollment and successful payment verifications on Learnify. By utilizing Nodemailer for email notifications and Notify.lk for SMS service, the Notification Service ensures that users receive notifications via their preferred communication channels. Email notification service works as follows. Upon confirmation of course enrollment and successful payment verifications, the notification service utilizes Nodemailer, a powerful Node.js module for sending emails, to deliver notifications to users via email. It generates dynamic email templates containing relevant information such as course details, enrollment confirmation, payment receipts. By using Nodemailer's features, the notification service ensures that email notifications are informative, visually appealing, and customized to each user's enrollment status. In addition to email notifications, the notification service utilizes Notify.lk, a reliable SMS service provider, to deliver notifications to users via SMS messages. It sends personalized SMS messages containing essential information such as course enrollment confirmation, payment receipts. By integrating with Notify.lk's API, the notification service ensures that SMS notifications are delivered reliably and promptly to users' mobile devices, providing an additional communication channel for users who prefer receiving notifications via SMS. The notification service utilizes event-based triggers to deliver notifications in response to specific user actions or events within the platform. Upon

confirmation of course enrollment and successful payment verifications, it triggers notification workflows to generate and send email and SMS notifications to users, providing instant confirmation and acknowledgment of their enrollment and payment status. Consuming user data and enrollment details, the notification service delivers personalized notifications to each user's enrollment status and preferences. In summary, the notification service, integrated with Nodemailer for email notifications and Notify.lk for SMS service, is an important component within Learnify, responsible for delivering timely and personalized notifications to users upon confirmation of course enrollment and successful payment verifications. The Notification Service enhances user experience, engagement, and satisfaction while providing an effective communication channel for users throughout their learning journey.

Use case Diagram

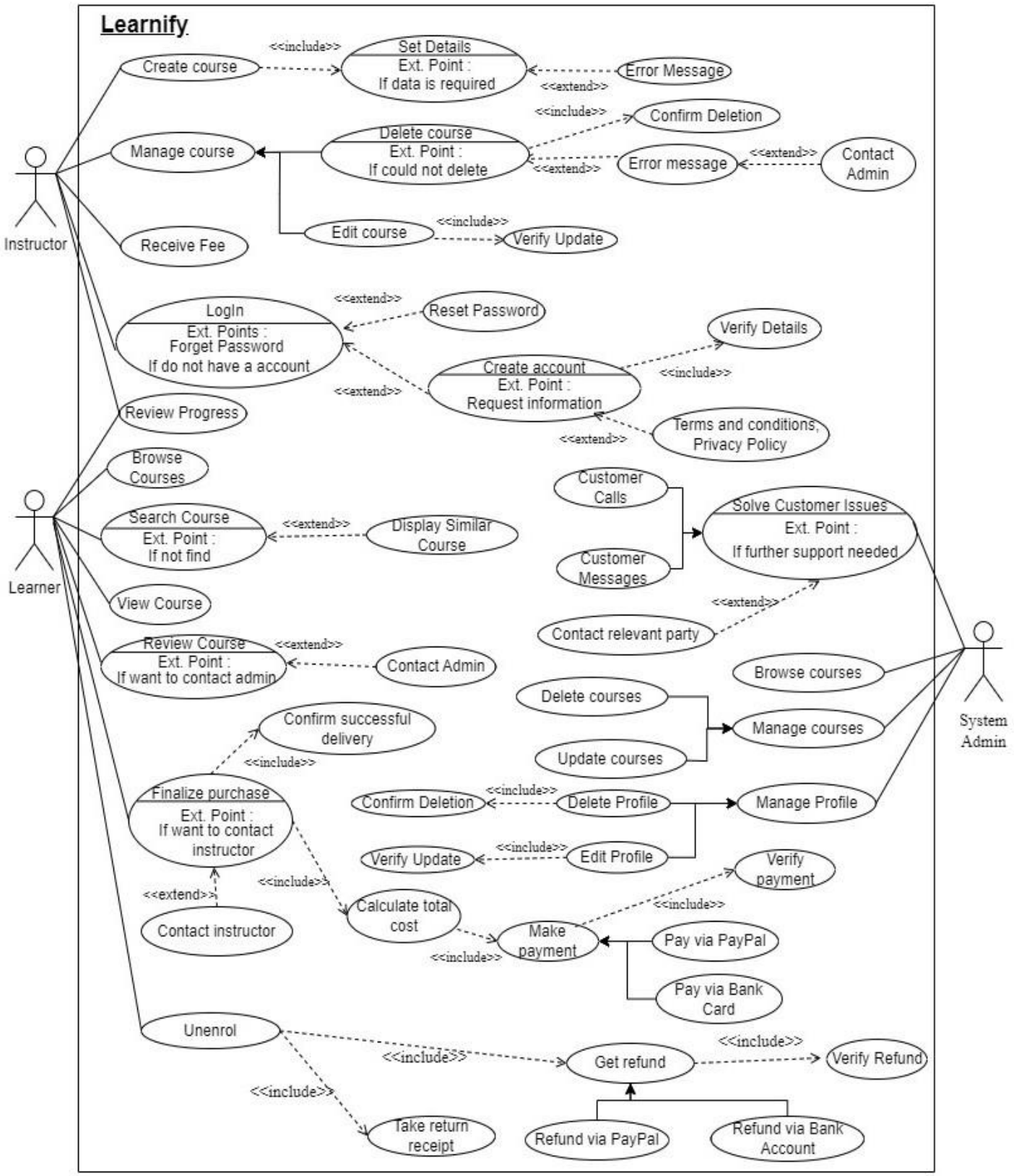

Figure 1: Use case Diagram

System Overview Diagram

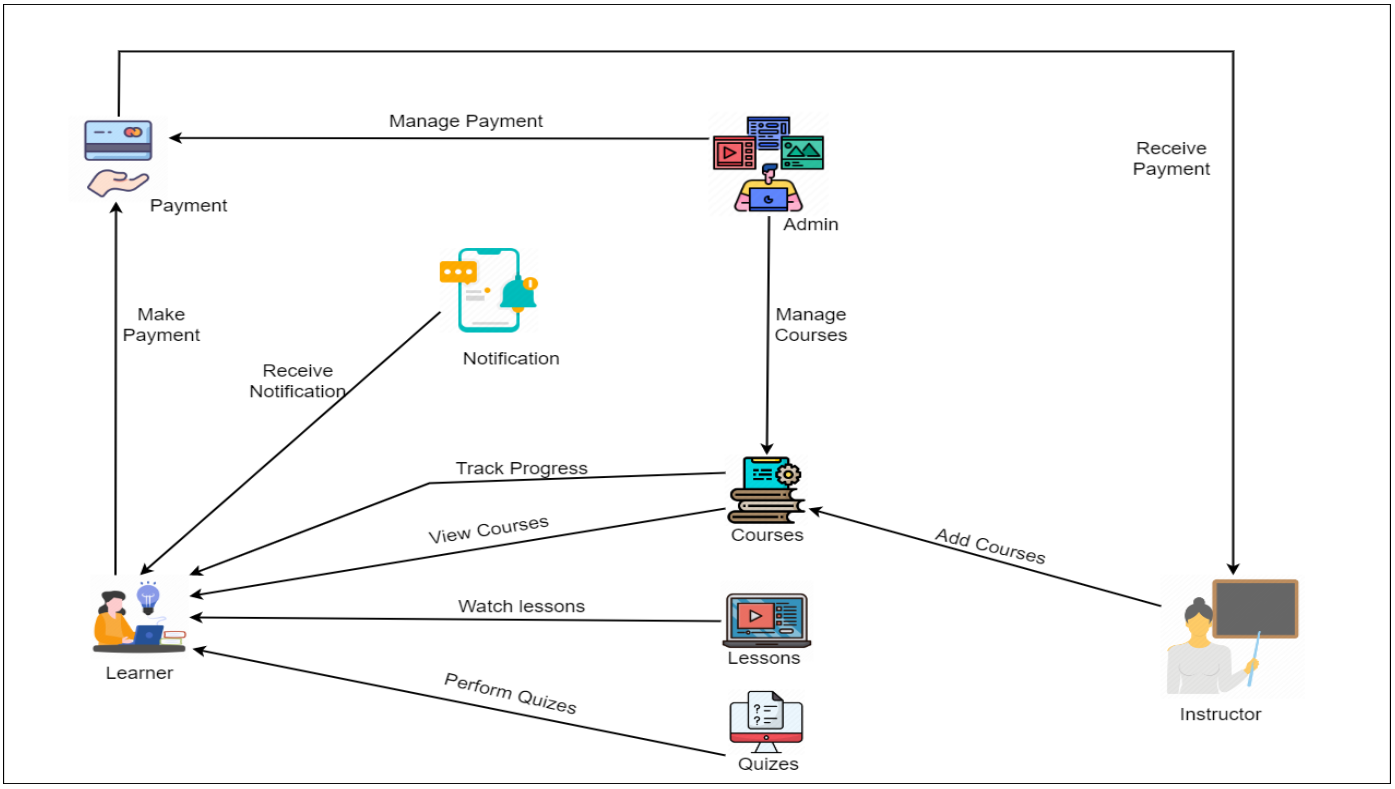

Figure 2: System Overview Diagram

Architectural Diagram

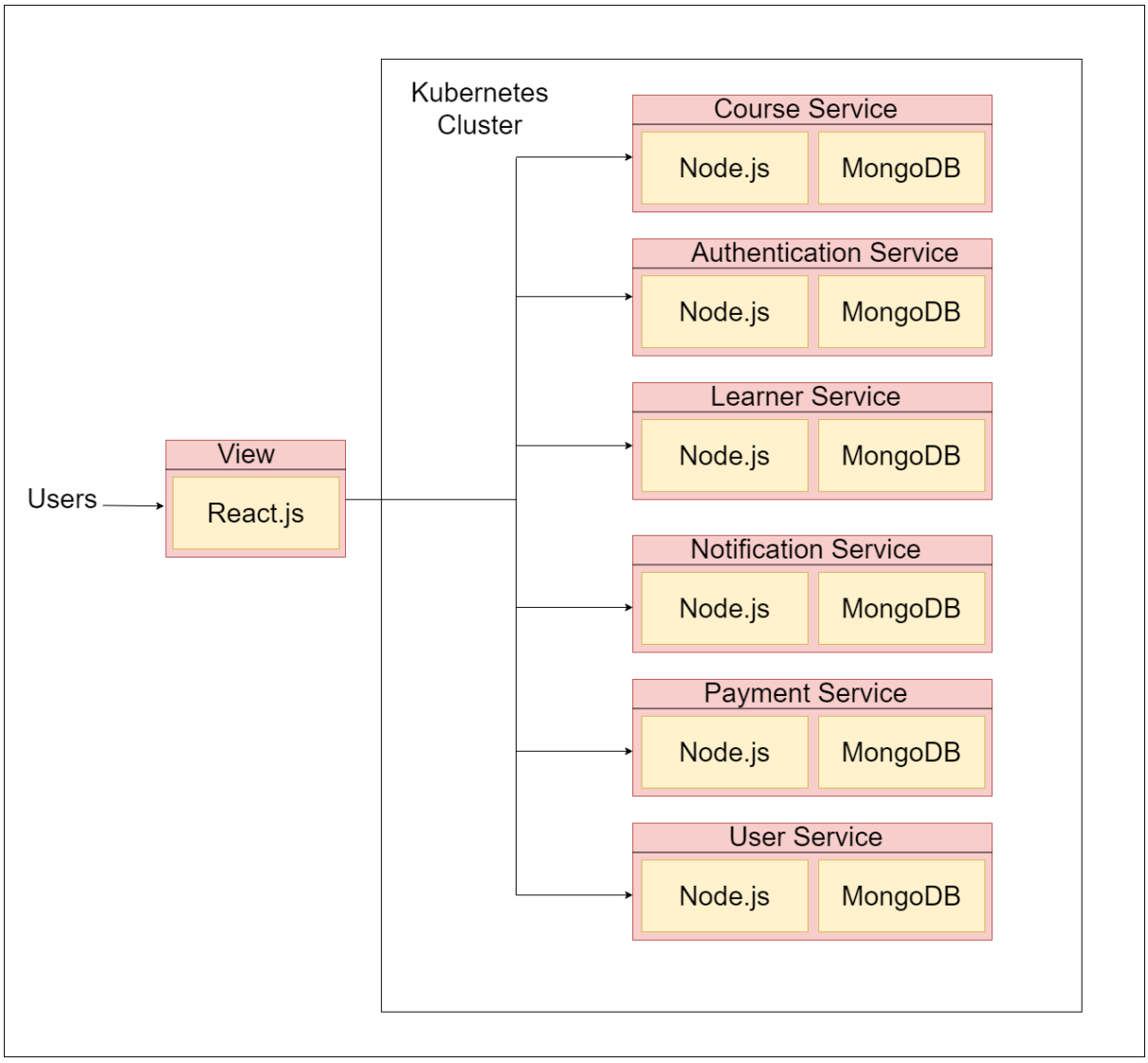

Figure 3: Architecture Diagram

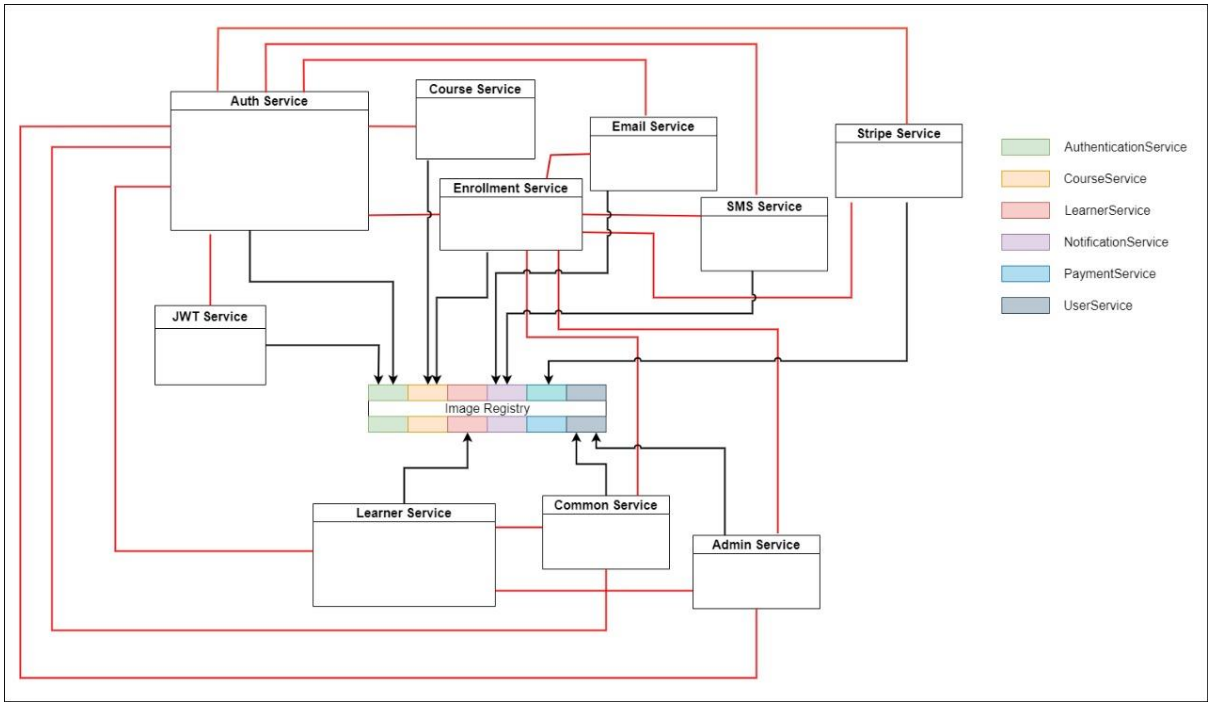

Figure 4: Service Connections

Service Structure

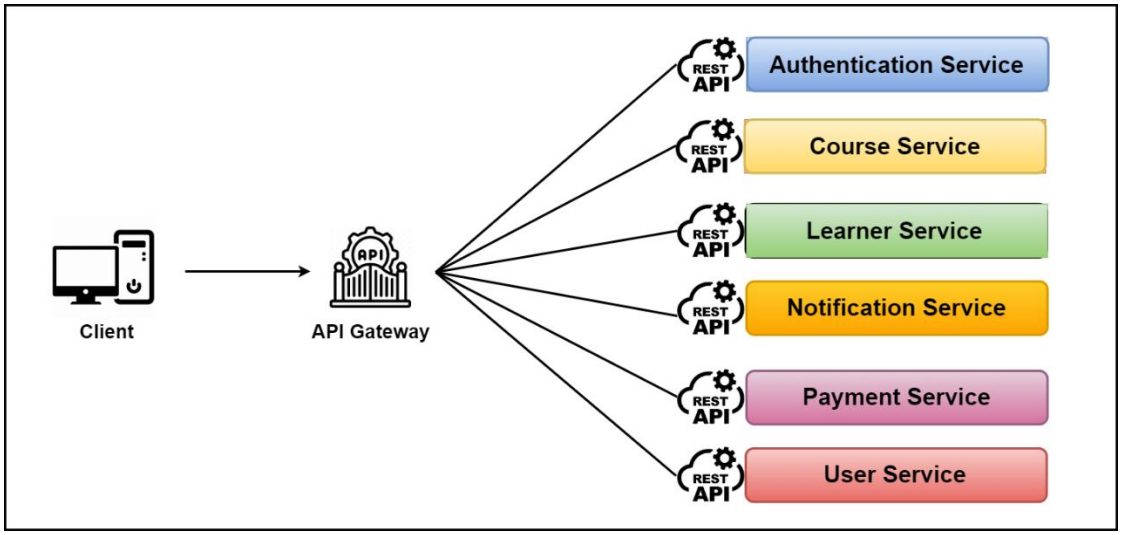

Figure 5: Service Structure

ER Diagram

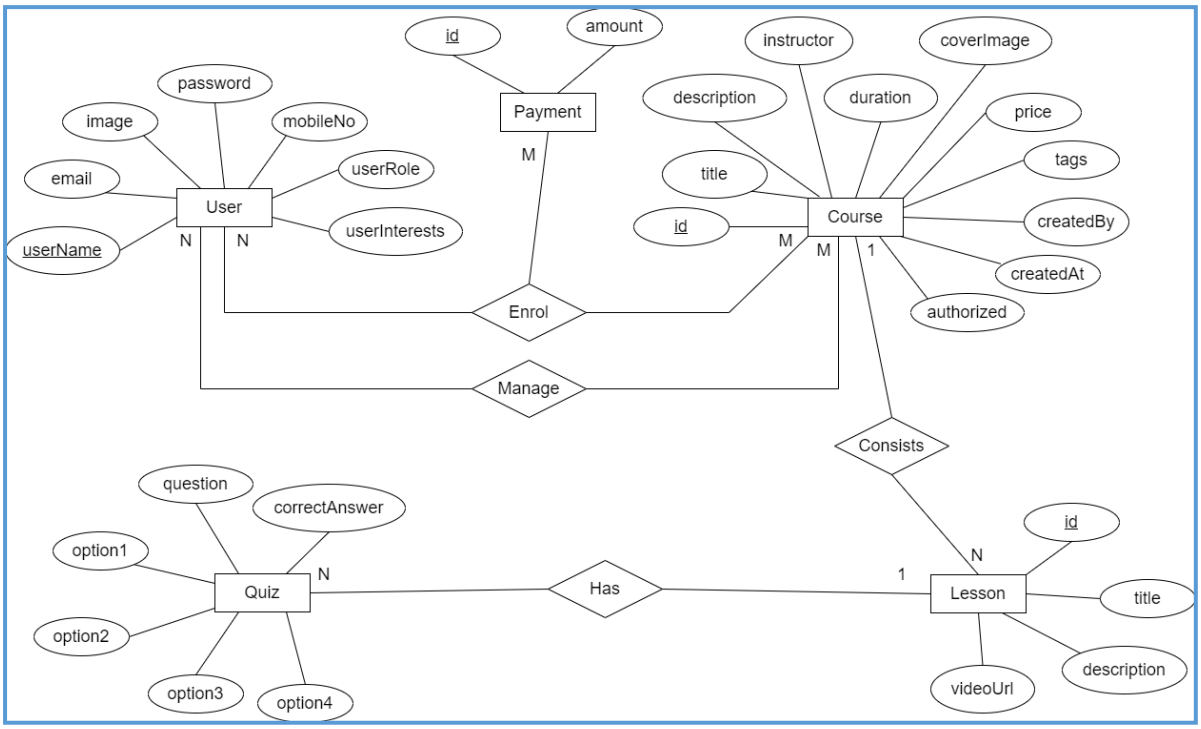

Figure 6: ER Diagram

MongoDB Scheme

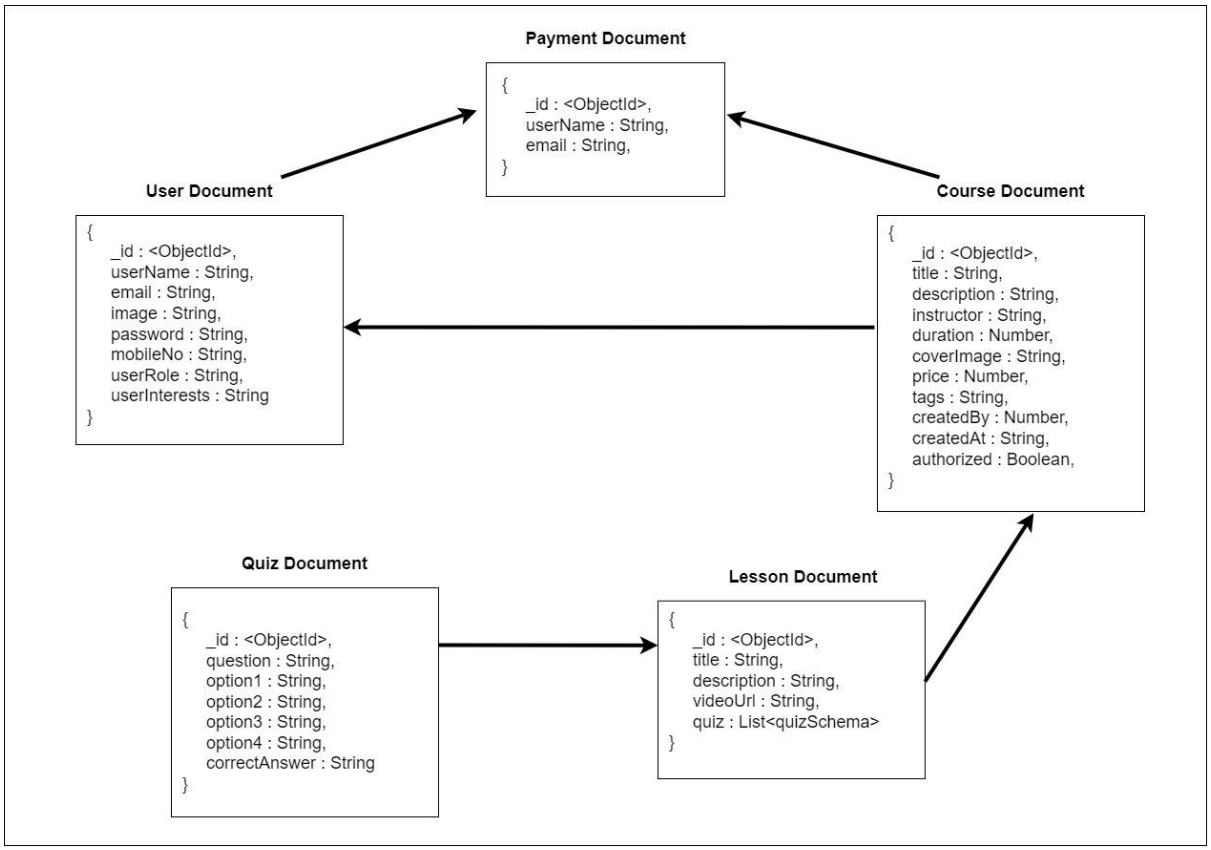

Figure 7: MongoDB Scheme

Sequence Diagram

User Profile Management

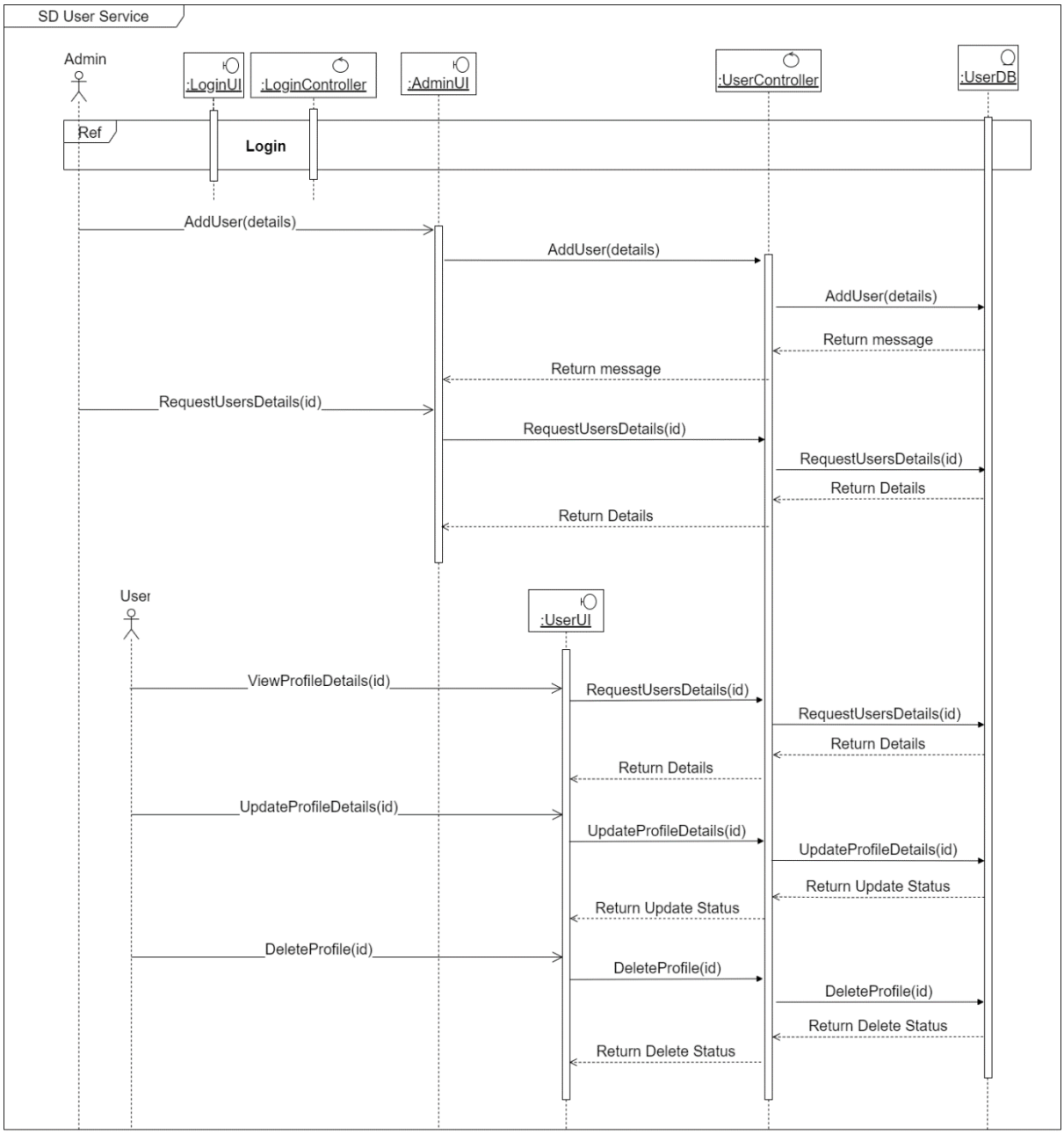

Figure 8: User Profile Management Sequence Diagram

Course Management

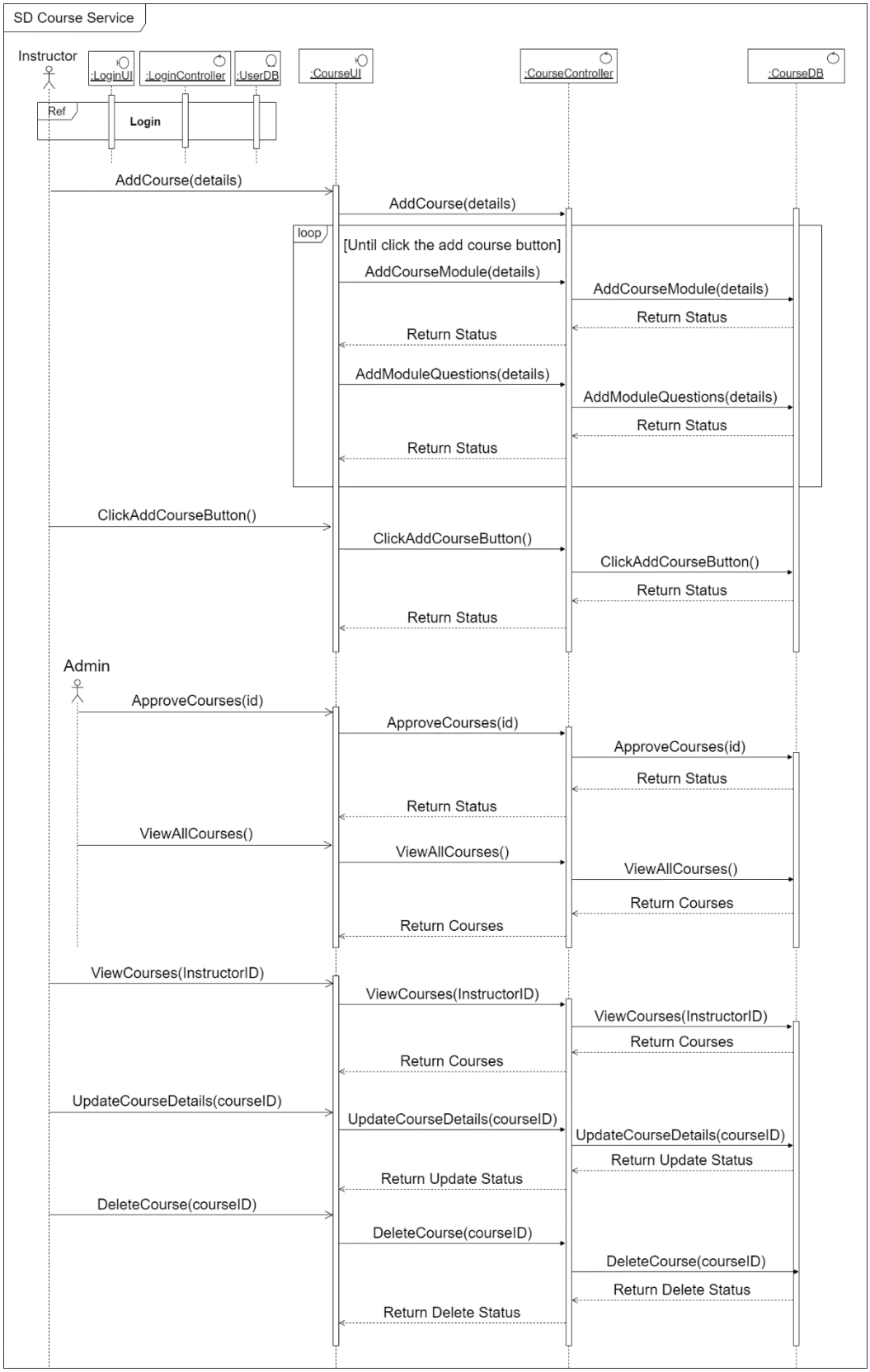

Figure 9: Course Management Sequence Diagram

Course Enrollment

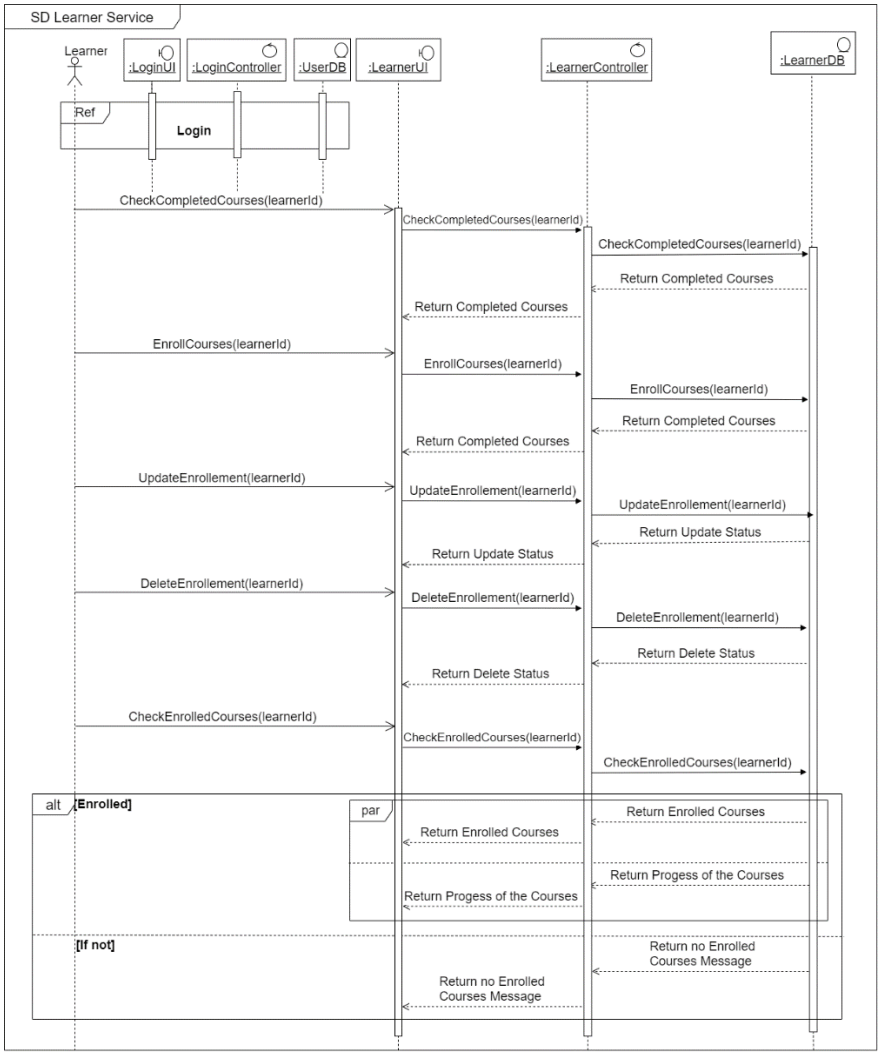

Figure 10: Course Enrollment Sequence Diagram

Payment Handling

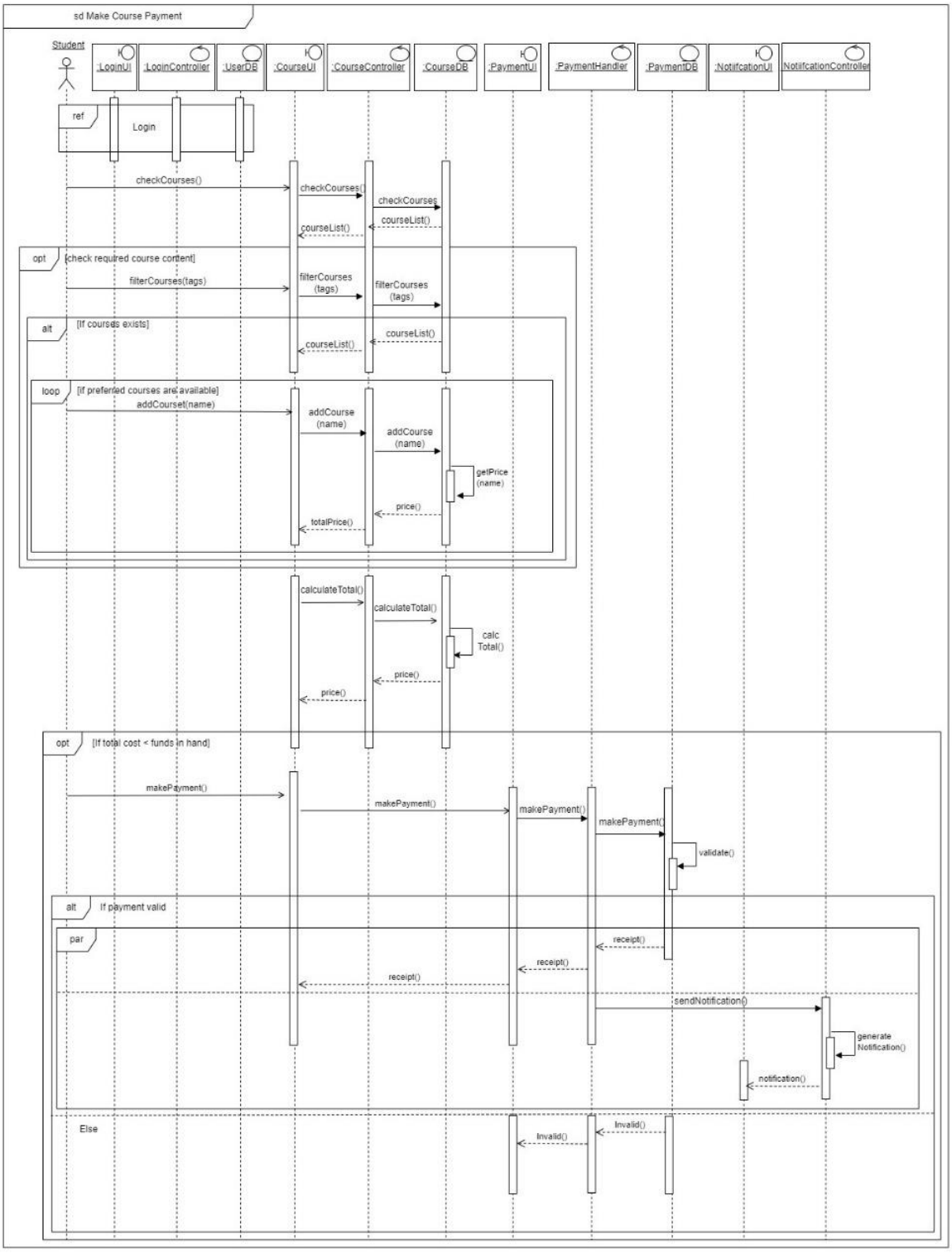

Figure 11: Payment Handling Sequence Diagram

Tools and Technologies

- Code Editor - VS Code
- React Version - 18.2.0
- Node Version - 18.12.1
- Docker Version – 3.8
- Package Manager - Node Package Manager

Figma Design

Figma Design was created before the implementing the Astro Journey. Some designs are as follows:

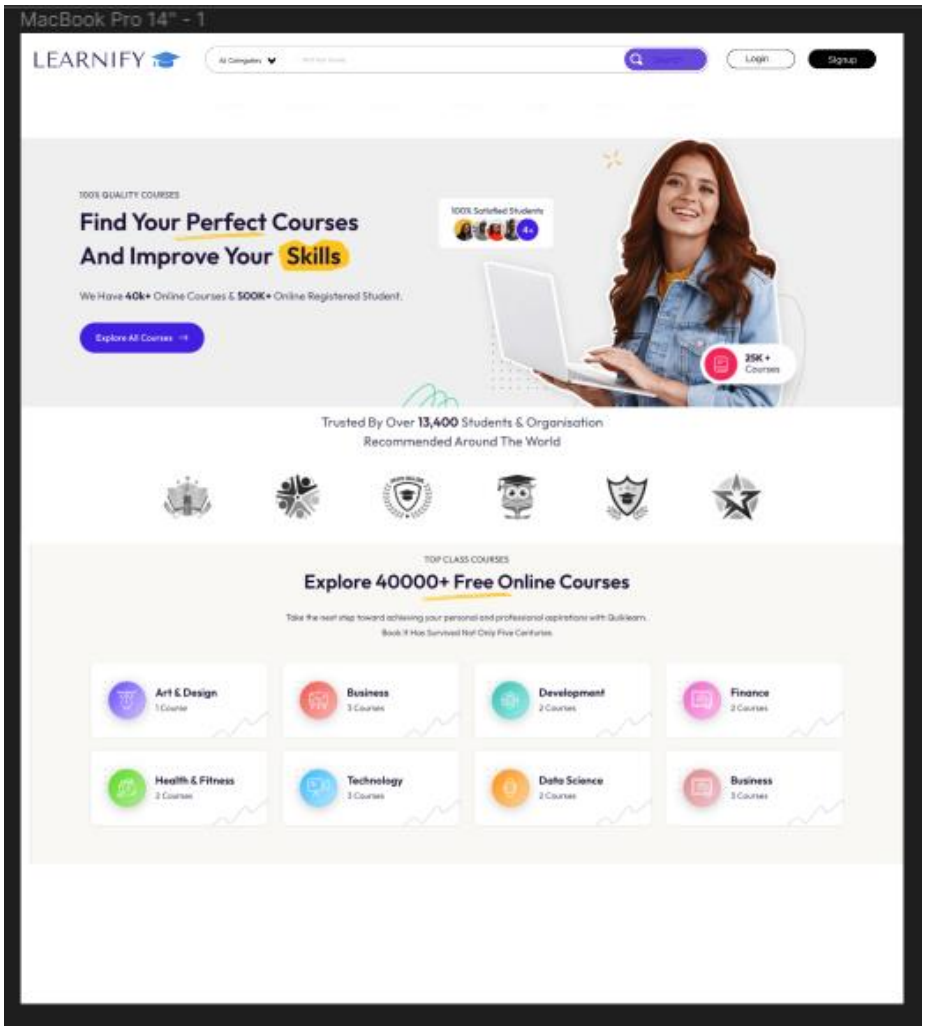

Figure 12: Figma Home Page Design

Please find the figma design by navigating

https://www.figma.com/file/Ckg8mccpPmfjz2uACpbjkn/Learnify?type=design&node-id=0-1&mode=design&t=K2SqelLli4WTdx4R-0

API Documentation

Please find the API Documentation from the below link:

https://documenter.getpostman.com/view/26795401/2sA3JNcM9m

Frontend Folder Structure

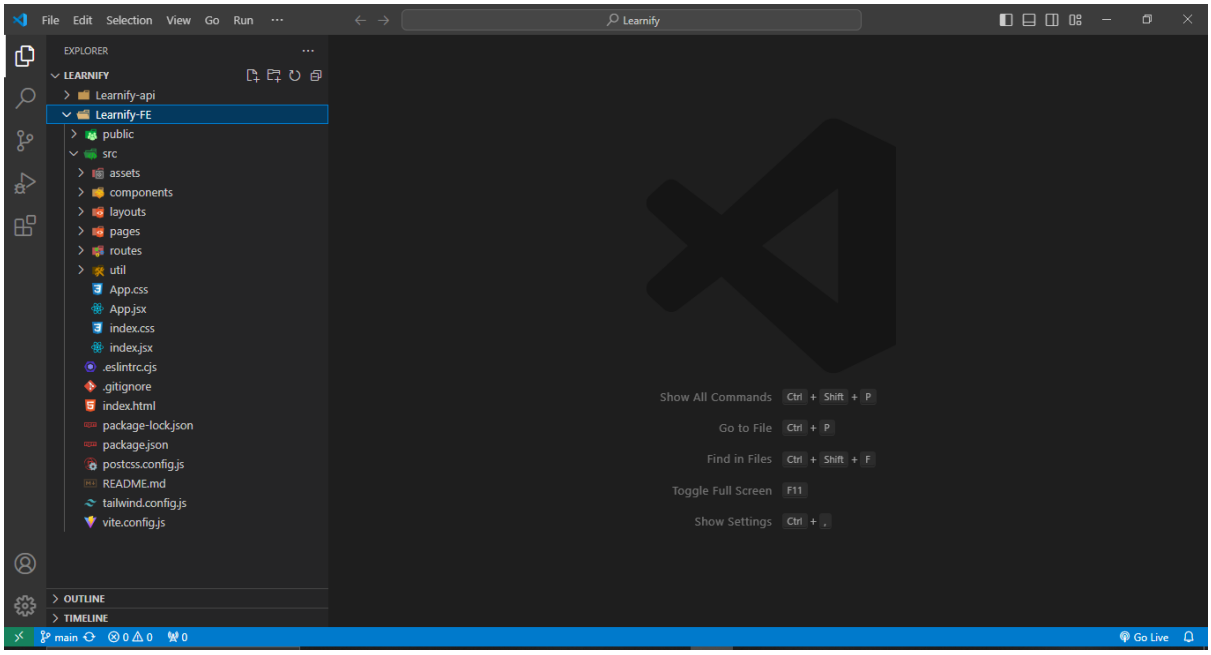

Figure 13: Frontend Folder Structure

Backend Folder Structure

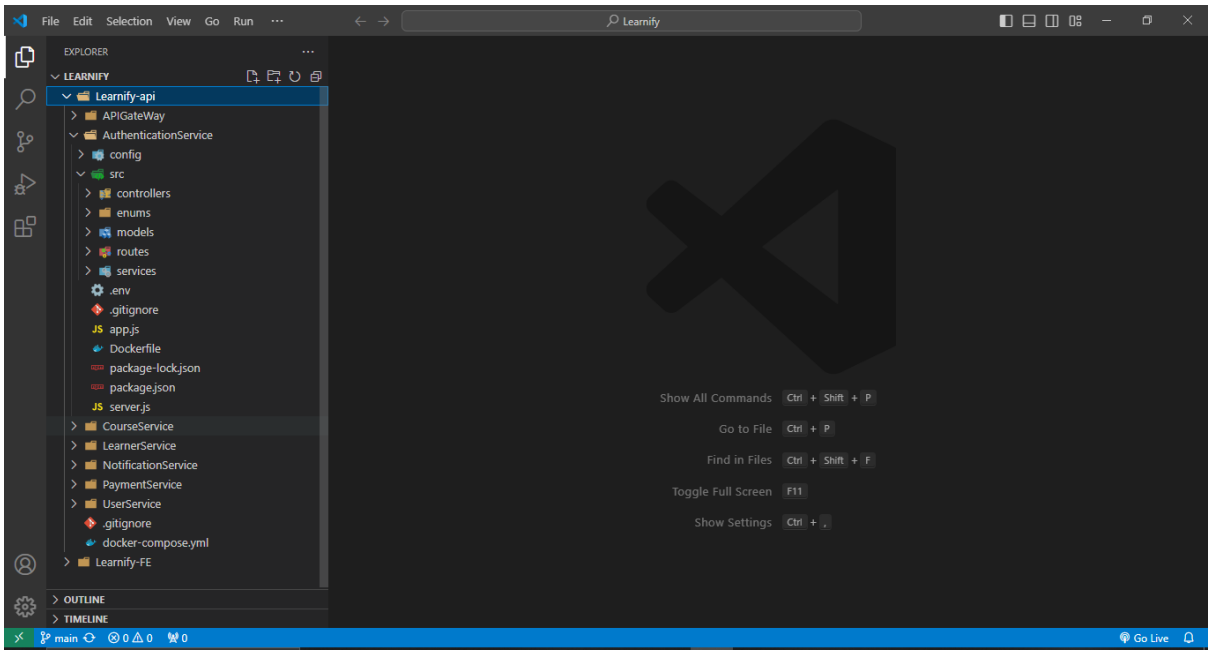

Figure 14: Backend Folder Structure

GitHub Repository

GitHub Repository Link :

Frontend - https://github.com/ChillBroh/Learnify-FE

Backend - https://github.com/ChillBroh/Learnify-api

Clone the project from the above link and run the following commands to run the application under development mode.

Note : Ensure to replace the MongoDB URL as to run the application without any inconvenience.

YouTube Video Link

Please find the YouTube video link below :

https://youtu.be/JRwDU77Ax44

Contribution

Registration Number	Student Name	Contribution
IT21318320	Silva T.U.D	• Introduction • Payment Gateway and Notification Service Description • Use Case Diagram • System Overview Diagram • ER Diagram • Sequence Diagram for payment handling • Figma Design • API Documentation • Code of Payment Gateway and Notification Service
IT21189944	Madusanka G.K.I	• User Authentication and Course Service Description • Use Case Diagram • System Overview Diagram • ER Diagram • Sequence Diagram for course management • Figma Design • API Documentation • Code of User Authentication and Course Service
IT21223594	Thalangama T.P	• API Gateway and User Service Description • Use Case Diagram • System Overview Diagram • ER Diagram • Sequence Diagram for user profile management

		• Figma Design • API Documentation • Code of API Gateway and User Service
IT21319174	Dissanayake M.G.T.W	• Learner Service and Course Enrollment Description • Use Case Diagram • System Overview Diagram • ER Diagram • Sequence Diagram for course enrollment • Figma Design • API Documentation • Code of Learner Service and Course Enrollment

Appendix

- **Learnify-API**

API Gateway

rateLimiter.js

```
const {rateLimit} = require('express-rate-limit')

//Rate limiting
exports.applyRateLimiter = rateLimit({
   windowMs: 15 * 60 * 1000,
   limit:100,
 })
```

requestAuthenticator.js

```
const jwt = require("jsonwebtoken");

exports.authenticateRequest = async (req, res, next) => {
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];

  if (!token) {
    return res.status(401).json({ error: "Missing token" });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, decoded) => {
    if (err) {
      console.log(err.message);
      return res.status(403).json({ error: "Invalid token" });
    }
    req.user = decoded;
    next();
  });
};
```

Authentication Service

database.js

```
/**
 * @description - Database config file
 */
```

```
const mongoose = require('mongoose')
require('dotenv').config()


//Database URI
const mongoDB_URI = process.env.DATABASE_URI
const DB_CONNECTION_TIMEOUT = process.env.DB_CONNECTION_TIMEOUT
const DB_NAME = process.env.DB_NAME
const DB_MIN_POOL_SIZE = process.env.DB_MIN_POOL_SIZE
const DB_MAX_POOL_SIZE = process.env.DB_MAX_POOL_SIZE

//Connection instance
let mongoInstance = null

// Setting up mongoose connection
const createConnection = async () => {
   try {
      const conn = await mongoose.connect(mongoDB_URI, {
         connectTimeoutMS: DB_CONNECTION_TIMEOUT,
         dbName: DB_NAME,
         minPoolSize: DB_MIN_POOL_SIZE, //Creating a connection pool
         maxPoolSize: DB_MAX_POOL_SIZE
      })


      console.log('Connected to MongoDB')

      return conn

   } catch (error) {

      console.log('Error at creating MongoDB connection : database.js : '+  error)
```

```
    }
}

//Using singleton pattern to ensure there's only one reference to the mongoose connection pool to avoid overhead.
//Multiple clients can access this object same time and accquire a connection from the mongoose connection pool
const getInstance = () => {

    if(!mongoInstance){
        mongoInstance = createConnection()
    }
    return mongoInstance
}

module.exports = {
    getInstance
}
```

userController.js

```
/**
 * @description - All the user related CRUD operations are happening here
 */

const authModel = require("../models/authModel");
const HttpStatus = require("../enums/httpStatus");

exports.createUser = async (userData) => {
  try {
    const response = await authModel.create(userData);
```

```
    return { status: HttpStatus.CREATED, body: response };
  } catch (error) {
    console.log(
      "Internal server error at createUser(). More details : " + error
    );
    return { status: HttpStatus.INTERNAL_SERVER_ERROR, body: error };
  }
};

exports.getUserByName = async (userData) => {
  try {
    const response = await authModel.findOne({ userName: userData });
    return { status: HttpStatus.OK, body: response };
  } catch (error) {
    console.log(
      "Internal server error at getUserByName(). More details : " + error
    );
    return { status: HttpStatus.NOT_FOUND, body: error };
  }
};
```

httpStatus.js

```
//Creating custom status codes for better readability
const HttpStatus = Object.freeze({
    OK: 200,
    CREATED: 201,
    BAD_REQUEST: 400,
    UNAUTHORIZED: 401,
    FORBIDDEN: 403,
    NOT_FOUND: 404,
    INTERNAL_SERVER_ERROR: 500
```

```
})

module.exports = HttpStatus
```

systemWarnings.js

```
//Creating custom warnings
const SysWarn = Object.freeze({
   MISSING_JWT_TOKEN: 'Unauthorized: Missing token',
   INVALID_JWT_TOKEN: 'Unauthorized: Invalid token',
   USER_TYPE_NOT_AUTHORIZED: 'Unauthorized: User does not have authorization to perform this action'
})

module.exports = SysWarn
```

authModel.js

```
/**
 * @description - This file defines the authentication model for mongoDB
 */

const mongoose = require("mongoose");
// Define enums for user roles
const UserRole = {
  LEARNER: "learner",
  INSTRUCTOR: "instructor",
  ADMIN: "admin",
};
//Schema for authentication
const authSchema = new mongoose.Schema(
  {
```

```
    userName: { type: String, required: true, unique: true },
    email: { type: String, required: true, unique: true },
    image: { type: String, required: false },
    password: { type: String, required: true },
    mobileNo: { type: String, required: false },
    userRole: {
      type: String,
      required: false,
      enum: Object.values(UserRole),
      default: "learner",
    }, //Roles : learner, instructor, admin
    userInterests: { type: [String], required: false }
  },
  { collection: "authUsers" }
);

//Creating mongoose model using Schema
const authModel = mongoose.model("authModel", authSchema);

//Exporting model to be used by authController.js
module.exports = authModel;
```

authRoutes.js

```
/**
 * @description - Route file responsible for the authentication process
 */

//Requires
const express = require("express");
const router = express.Router();
const { register, login, logout, } = require("../services/authService");
```

```
const {generateNewAccessToken} = require('../services/jwtService')

router.post("/register", async (req, res) => {
  await register(req, res);
});

router.post("/login", async (req, res) => {
  await login(req, res);
});

router.post("/logout", async (req, res) => {
  await logout(req, res);
});

router.get('/'), async(req,res) => {
  generateNewAccessToken(req,res)
}

//Exporting router to be used by the app.js
module.exports = router;
```

authService.js

```
/**
 * @description - Authentication related business logic handled here
 * @functionality - extract JSON payload, setting response header, handling business logic, and setting up JWT tokens
 */
const bcryptjs = require("bcryptjs");
const jwt = require("jsonwebtoken");
const { createUser, getUserByName } = require("../controllers/userController");
const saltCount = 10;
```

```
const HttpStatus = require("../enums/httpStatus");
const { getAccessToken, getRefreshToken } = require("../services/jwtService");

require("dotenv").config();

//Hasing the password added data privacy & security
const hashPasswordGen = async (plainPsw) => {
  const hashPsw = await bcryptjs.hash(plainPsw, saltCount);
  return hashPsw;
};

exports.register = async (req, res) => {
  const payload = req.body;

  const hashedPassword = await hashPasswordGen(payload.password);

  //Constructing the new user payload
  const newUser = {
    userName: payload.userName,
    email: payload.email,
    image: payload.image,
    password: hashedPassword,
    mobileNo: payload.mobileNo,
    userRole: payload.userRole,
  };

  console.log(newUser);

  //Registering the user with createUser function in userController.js
  const response = await createUser(newUser);

  //Auto-login the user upon successful registration to the system.
```

```
  if (response.status === HttpStatus.CREATED) {
    await this.login(req, res);
  } else {
    res.status(response.status).json(response.body);
  }
};

exports.login = async (req, res) => {
  const payload = req.body;

  const foundUser = await getUserByName(payload.userName);

  if (foundUser.status === HttpStatus.OK) {
    const retrivedUser = foundUser.body;
    console.log("user is ", retrivedUser);

    if (retrivedUser) {
      //Once the user is found in the system check for password matching to validate the user
      const checkPswMatch = await bcryptjs.compare(
        payload.password,
        retrivedUser.password
      );

      //if password matches, authenticate the user with JWT Token
      if (checkPswMatch === true) {
        //Setting up the JWT tokens. Access token contains userID and user type. Expires in 1 hour. Refresh token only store the user id. Expires in 3 months
        const accessToken = getAccessToken(
          retrivedUser._id,
          retrivedUser.userRole,
          retrivedUser.userName,
          retrivedUser.image
```

```
      );
      const refreshToken = getRefreshToken(
        retrivedUser._id,
        retrivedUser.userRole,
        retrivedUser.userName,
        retrivedUser.image
      );

      //Setting the response header with OK and dispatching the JWT Tokens in a JSON body
      res
        .status(HttpStatus.OK)
        .json({ accessToken: accessToken, refreshToken: refreshToken });
    } else {
      //If password didn't match, unauthorize the login request
      res
        .status(HttpStatus.UNAUTHORIZED)
        .json({ body: "User authentication failed!" });
    }
  } else {
    res
      .status(HttpStatus.UNAUTHORIZED)
      .json({ body: "User authentication failed!" });
  }
 } else {
  //if user not found sets request headers to unauthorized and sends authentication failed
message to the client
  res
    .status(HttpStatus.UNAUTHORIZED)
    .json({ body: "User authentication failed!" });
 }
};
```

```
//logout user
exports.logout = async (req, res, next) => {
  res.cookie("jwt", "loggedout", {
    expires: new Date(Date.now() + 10 * 1000),
    httpOnly: true,
  });
  res.status(200).json({
    status: "success",
    message: "logged Out!",
  });
};
```

jwtService.js

```
/**
 * @description - This service class is used to authenticate the request by checking the validity
of JWT token
 * @functionality - Check for the JWT token, validate if available, extract user ID.
 */

//Requires
const jwt = require("jsonwebtoken");
require("dotenv").config();
const sysWarn = require("../enums/systemWarnings");
const HttpStatus = require("../enums/httpStatus");

//Verifying JWT token
exports.verifyToken = async (req) => {
  //Extracting token from the header
  const token =
    req.headers.authorization && req.headers.authorization.split(" ")[1];
```

```
  //Checks for missing tokens
  if (!token) {
    return { status: HttpStatus.UNAUTHORIZED, body: sysWarn.MISSING_JWT_TOKEN };
  }

  try {
    //Decodes token and extract user data
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const userData = {
      userId: decoded.userId,
      userType: decoded.userType,
      userName: decoded.userName
    };

    return { status: HttpStatus.OK, body: userData };
  } catch (error) {
    //If error occured during decode phase, responds with Invalid token message.
    return { status: HttpStatus.UNAUTHORIZED, body: sysWarn.INVALID_JWT_TOKEN };
  }
};

//Regenerate a new access token for the admin
exports.generateNewAccessToken = (req, res) => {
  const token =
    req.headers.authorization && req.headers.authorization.split(" ")[1];

  //Checks for missing tokens
  if (!token) {
    res
      .status(HttpStatus.UNAUTHORIZED)
      .json({ body: sysWarn.MISSING_JWT_TOKEN });
  }
```

```
  try {
    //Decodes token and extract user data
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    const userId = decoded.userId;

    const accessToken = jwt.sign(
      { userId: userId, userType: "ADMIN" },
      process.env.JWT_SECRET,
      { expiresIn: "1h" }
    );

    res.status(HttpStatus.OK).json({ accessToken: accessToken });
  } catch (error) {
    //If error occured during decode phase, responds with Invalid token message.
    res
      .status(HttpStatus.UNAUTHORIZED)
      .json({ body: sysWarn.INVALID_JWT_TOKEN });
  }
};

//Generates access token
exports.getAccessToken = (userId, userRole, userName, image) => {
  const accessToken = jwt.sign(
    { userId: userId, userRole: userRole, userName: userName, image: image },
    process.env.JWT_SECRET,
    { expiresIn: "3h" }
  );

  return accessToken;
};
```

```
//Generates refresh token
exports.getRefreshToken = (userId, userRole, userName, image) => {
  const refreshToken = jwt.sign(
    { userId: userId, userRole: userRole, userName: userName, image: image },
    process.env.JWT_SECRET,
    { expiresIn: "90d" }
  );

  return refreshToken;
};
```

app.js

```
const express = require("express");
const app = express();
const appRouter = express.Router();
const cors = require("cors");
const connectDB = require("./config/database");

//Requires - Route classes
const authRoutes = require("./src/routes/authRoutes");

connectDB.getInstance();

app.use(cors());

app.use(express.json());

app.use("/", authRoutes);

//Exporting app to be used by the server.js
module.exports = app;
```

Dockerfile

```
FROM node:20-alpine

WORKDIR /app

COPY .env /app/.env

# Copy package.json and package-lock.json (if exists)
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the port your app runs on
EXPOSE 8000

# Command to run your application
CMD ["node", "server.js"]
```

server.js

```
/**
 * @description - Initiating HTTP server and listning to incoming http requests
 */

//Imports and Requires
const http = require("http");
```

```
const app = require("./app");

require("dotenv").config();

console.log("Server Js executing... Initiating HTTP server");
const port = process.env.PORT;

//Creates HTTP server
const server = http.createServer(app);

//Server then  listen to the port (4000)
server.listen(port, () => {
  console.log(`Authentication Server is running on port ${port}`);
});
```

Course Service

db.js

```
/**
 * @description - Database config file
 */

const mongoose = require('mongoose')
require('dotenv').config()



//Database URI
const mongoDB_URI = process.env.DATABASE_URI
const DB_CONNECTION_TIMEOUT = process.env.DB_CONNECTION_TIMEOUT
const DB_NAME = process.env.DB_NAME
```

```
const DB_MIN_POOL_SIZE = process.env.DB_MIN_POOL_SIZE
const DB_MAX_POOL_SIZE = process.env.DB_MAX_POOL_SIZE

//Connection instance
let mongoInstance = null

// Setting up mongoose connection
const createConnection = async () => {
   try {
      const conn = await mongoose.connect(mongoDB_URI, {
         connectTimeoutMS: DB_CONNECTION_TIMEOUT,
         dbName: DB_NAME,
         minPoolSize: DB_MIN_POOL_SIZE, //Creating a connection pool
         maxPoolSize: DB_MAX_POOL_SIZE
      })


      console.log('Connected to MongoDB')

      return conn

   } catch (error) {

      console.log('Error at creating MongoDB connection : database.js : '+  error)

   }
}
```

//Using singleton pattern to ensure there's only one reference to the mongoose connection pool to avoid overhead.
//Multiple clients can access this object same time and accquire a connection from the mongoose connection pool

```
const getInstance = () => {

   if(!mongoInstance){
      mongoInstance = createConnection()
   }
   return mongoInstance
}

module.exports = {
   getInstance
}
```

authController.js

```
const jwt = require("jsonwebtoken");
const AppError = require("../utils/AppError");
//Give rolebase authentication for protected routes
exports.restrictTo = (...roles) => {
  return (req, res, next) => {
    const user = authenticateRequest(req, res, next);
    if (!roles.includes(user.userRole)) {
      return next(
        new AppError("You do not have permission to perform this action", 403)
      );
    }
    console.log(user);
    req.user = user;
    next();
  };
};

const authenticateRequest = (req, res, next) => {
```

```
  const authHeader = req.headers["authorization"];
  const token = authHeader && authHeader.split(" ")[1];
  return (userToken = jwt.decode(token));
};
```

courseController.js

```
const Course = require("../models/courseModel");
const AppError = require("../utils/AppError");
const catchAsync = require("../utils/catchAsync");

const createCourse = catchAsync(async (req, res, next) => {
  const {
    title,
    description,
    instructor,
    duration,
    coverImage,
    price,
    tags,
    lessons,
  } = req.body;

  const existingCourse = await Course.findOne({ title: title });
  if (existingCourse) {
    return next(new AppError("Course already exists!", 400));
  }

  const newCourse = new Course({
    title,
    description,
    instructor,
```

```
    duration,
    coverImage,
    price,
    tags,
    createdBy: req.user.userId,
    createdAt: new Date(),
    lessons,
  });

  const savedCourse = await newCourse.save();
  res.status(201).json({
    data: savedCourse,
    message: "Course Created",
  });
});

const getAllCourses = catchAsync(async (req, res, next) => {
  const courses = await Course.find();
  res.status(200).json({
    status: "success",
    data: courses,
  });
});

const getCoursesByUser = catchAsync(async (req, res, next) => {
  const courses = await Course.find({ createdBy: req.user.userId });
  res.status(200).json({
    status: "success",
    data: courses,
  });
});
```

```
const getOneCourse = catchAsync(async (req, res, next) => {
  const courseId = req.params.id;
  const course = await Course.findOne({ _id: courseId });

  if (!course) {
    return next(new AppError("Course Not Found!", 404));
  }
  res.status(201).json({
    status: "success",
    course,
  });
});

const updateCourse = catchAsync(async (req, res, next) => {
  const courseId = req.params.id;
  console.log(courseId);
  const updatedDetails = req.body;
  console.log(updatedDetails);

  const course = await Course.findOneAndUpdate(
    { _id: courseId }, // Filter criteria
    { $set: updatedDetails }, // Update data using $set operator
    { new: true } // Return the updated document
  );

  if (!course) {
    return next(new AppError("Course Not Found!", 404));
  }
  res.status(201).json({
    status: "success",
    data: {
      course,
```

```
    },
  });
});

const deleteCourse = catchAsync(async (req, res, next) => {
  const courseId = req.params.id;
  const course = await Course.findOneAndDelete({ _id: courseId });

  if (!course) {
    return next(new AppError("Course Not Found!", 404));
  }
  res.status(204).end();
});

const approveCourse = catchAsync(async(req,res,next) => {
  const courseId = req.params.id
  const updateBody = {
    authorized: req.body.authorized
  }
  const response = await Course.findByIdAndUpdate(courseId, updateBody)
  if (!response) {
    return next(new AppError("Course Not Approved correctly!", 500));
  }
  res.status(201).end();
})

const getCourseByApproval = catchAsync(async(req,res,next) => {

  const approvedState = req.params.id
  const coursesList = await Course.find({authorized: approvedState})

  if (coursesList.length < 1) {
```

```
    return next(new AppError("No pending courses!", 404));
  }
  res.status(201).json(coursesList);
})

module.exports = {
  createCourse,
  getAllCourses,
  updateCourse,
  deleteCourse,
  getOneCourse,
  getCoursesByUser,
  approveCourse,
  getCourseByApproval,
};
```

enrollmentController.js

```
const Enrollment = require("../models/enrollmentModel");
const Course = require("../models/courseModel");
const catchAsync = require("../utils/catchAsync");
const AppError = require("../utils/AppError");

// Create a new enrollment
const createEnrollment = catchAsync(async (req, res, next) => {
  const { courseId, userId } = req.body;

  const existingEnrollment = await Enrollment.findOne({ courseId, userId });
  if (existingEnrollment) {
    return next(new AppError("Already enrolled!", 400));
  }

  const course = await Course.findOne({ _id: courseId });
```

```
  if (!course) {
    return next(new AppError("Course not found", 404));
  }

  const newEnrollment = new Enrollment({
    courseId,
    instructorId: course.createdBy,
    learnerId: userId,
    paymentStatus: true,
  });

  const savedEnrollment = await newEnrollment.save();
  res.status(201).json({
    data: savedEnrollment,
    message: "Enrolled",
  });
});

// Get all enrollments
const getAllEnrollments = catchAsync(async (req, res, next) => {
  const enrollments = await Enrollment.find();
  res.status(200).json({
    data: enrollments,
  });
});
//get courses enrolled related to instructor
const getAllEnrollmentsByInstructor = catchAsync(async (req, res, next) => {
  const instructorId = req.user.userId;
  console.log(req.user.userId);
  console.log("hello");
  const enrollments = await Enrollment.find({ instructorId: instructorId });
  console.log(enrollments);
```

```
  res.status(200).json({
    data: enrollments,
  });
});

// Get a single enrollment by ID
const getEnrollmentById = catchAsync(async (req, res, next) => {
  const enrollment = await Enrollment.find({ learnerId: req.params.id });
  if (!enrollment) {
    return next(new AppError("Enrollment not found", 404));
  }
  res.status(200).json({
    data: enrollment,
  });
});

// Update an enrollment by ID
const updateEnrollmentById = catchAsync(async (req, res, next) => {
  const userId = req.params.userId;
  const courseId = req.params.id;
  const enrollment = await Enrollment.findOne({
    courseId: courseId,
    learnerId: userId,
  });
  console.log(enrollment);
  const updatedEnrollment = await Enrollment.findByIdAndUpdate(
    enrollment._id,
    req.body,
    {
      new: true,
      runValidators: true,
    }
```

```
  );
  if (!updatedEnrollment) {
    return next(new AppError("Enrollment not found", 404));
  }
  res.status(200).json({
    data: updatedEnrollment,
    message: "Enrollment updated successfully",
  });
});
// Delete an enrollment by ID
const deleteEnrollmentById = catchAsync(async (req, res, next) => {
  const userId = req.user.userId;
  const courseId = req.params.id;
  console.log(userId);
  console.log(courseId);
  const enrollment = await Enrollment.findOne({
    courseId: courseId,
    learnerId: userId,
  });
  const deletedEnrollment = await Enrollment.findByIdAndDelete(enrollment._id);
  if (!deletedEnrollment) {
    return next(new AppError("Enrollment not found", 404));
  }
  res.status(204).json({
    data: null,
    message: "Enrollment deleted successfully",
  });
});
// Export the controller functions
module.exports = {
  createEnrollment,
  getAllEnrollments,
```

```
  getEnrollmentById,
  updateEnrollmentById,
  deleteEnrollmentById,
  getAllEnrollmentsByInstructor,
};
```

errorHandler.js

```
const errorHandler = (error, req, res, next) => {
  const statusCode = error.statusCode || 500;
  const status = error.status || "error";
  const message = error.message || "Something went wrong";
  res.status(statusCode).json({
    status,
    message,
  });
};

module.exports = errorHandler;
```

courseModel.js

```
/**
 * @description - This file defines the course model for MongoDB
 */

const mongoose = require("mongoose");

// Schema for quiz
const quizSchema = new mongoose.Schema({
  question: {
    type: String,
```

```
    required: true,
  },
  option1: {
    type: String,
    required: true,
  },
  option2: {
    type: String,
    required: true,
  },
  option3: {
    type: String,
    required: true,
  },
  option4: {
    type: String,
    required: true,
  },
  correctAnswer: {
    type: String,
    required: true,
  },
});

// Schema for lessons
const lessonSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
```

```
    required: true,
  },
  videoUrl: {
    type: String,
  },
  quiz: [quizSchema],
});

// Schema for courses
const courseSchema = new mongoose.Schema(
  {
    title: {
      type: String,
      required: true,
    },
    description: {
      type: String,
      required: true,
    },
    instructor: {
      type: String,
      required: true,
    },
    duration: {
      type: Number,
      required: true,
    },
    coverImage: {
      type: String,
      required: true,
    },
    price: {
```

```
    type: Number,
    required: true,
  },
  tags: {
    type: [String],
    required: true,
  },
  createdBy: {
    type: String,
    required: true,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
  authorized: {
    type:Boolean,
    default:false
  },

  lessons: [lessonSchema],
 },
 { collection: "courses" }
);

// Creating mongoose model using Schema
const CourseModel = mongoose.model("Course", courseSchema);

module.exports = CourseModel;
```

enrollmentModel.js

```
/**
 * @description - This file defines the enrollment model for MongoDB
 */

const mongoose = require("mongoose");

// Schema for courses
const enrollmentSchema = new mongoose.Schema(
  {
    courseId: {
      type: String,
      required: true,
    },
    instructorId: {
      type: String,
      required: true,
    },
    learnerId: {
      type: String,
      required: true,
    },
    completionLevel: {
      type: Number,
      required: true,
      default: 0,
    },
    paymentStatus: {
      type: Boolean,
      required: true,
      default: false,
    },
  },
```

```
  { collection: "enrollment" }
);

// Creating mongoose model using Schema
const EnrollmentModel = mongoose.model("enroll", enrollmentSchema);

module.exports = EnrollmentModel;
```

courseRoutes.js

```
const express = require("express");
const router = express.Router();
const {
  createCourse,
  updateCourse,
  deleteCourse,
  getCoursesByUser,
  approveCourse,
  getCourseByApproval,

} = require("../controllers/courseController");
const authController = require("../controllers/authController");

router.route("/").post(authController.restrictTo("instructor"), createCourse);

router
  .route("/:id")
  .patch(authController.restrictTo("instructor"), updateCourse)
  .delete(authController.restrictTo("instructor"), deleteCourse);

router
  .route("/course/user")
```

```
  .get(authController.restrictTo("instructor"), getCoursesByUser);

router
  .route("/course/:id")
  .patch(authController.restrictTo("admin"), approveCourse)
  .get(authController.restrictTo("admin"), getCourseByApproval)

module.exports = router;
```

enrollmentRoutes.js

```
const express = require("express");
const router = express.Router();
const {
  createEnrollment,
  getAllEnrollments,
  getEnrollmentById,
  updateEnrollmentById,
  deleteEnrollmentById,
  getAllEnrollmentsByInstructor,
} = require("../controllers/enrollmentController");
const authController = require("../controllers/authController");

router
  .route("/")
  .post(createEnrollment)
  .get(authController.restrictTo("admin", "instructor"), getAllEnrollments);
router
  .route("/:id/userId")
  .get(getEnrollmentById)
  .patch(authController.restrictTo("learner"), updateEnrollmentById)
  .delete(authController.restrictTo("learner"), deleteEnrollmentById);
```

```
module.exports = router;
```

guestRoutes.js

```
const express = require("express");
const router = express.Router();
const {
  getAllCourses,
  getOneCourse,
} = require("../controllers/courseController");

router.route("/").get(getAllCourses);
router.route("/:id").get(getOneCourse);
module.exports = router;
```

AppErrors.js

```
class AppError extends Error {
  constructor(message, statusCode) {
    super(message);

    this.statusCode = statusCode;
    this.status = `${statusCode}`.startsWith("4") ? "fail" : "error";
    this.isOperational = true;

    console.log(this);
    console.log(this.constructor);
    Error.captureStackTrace(this, this.constructor);
  }
}
```

```
module.exports = AppError;
```

catchAsync.js

```
//This middleware used to handle errors globally
module.exports = (fn) => {
  return (req, res, next) => {
    fn(req, res, next).catch(next);
  };
};
```

app.js

```
const express = require("express");
const cors = require("cors");
const AppError = require("./src/utils/AppError");
const errorHandler = require("./src/middlewares/errorHandler");
const courseRoutes = require("./src/routes/courseRoutes");
const connectDB = require("./config/database");
const guestRoutes = require("./src/routes/guestRoutes");
const enrollRoutes = require("./src/routes/enrollmentRoutes");

const app = express();
connectDB.getInstance();
app.use(cors());
app.use(express.json());

//Add routes here
app.use("/", courseRoutes);
app.use("/enroll", enrollRoutes);
app.use("/guest-routes", guestRoutes);
```

```
app.use(errorHandler);

app.all("*", (req, res, next) => {
  next(new AppError(`Can't find ${req.originalUrl} on this server!`, 404));
});
//Exporting app to be used by the server.js
module.exports = app;
```

DockerFile

```
FROM node:20-alpine

WORKDIR /app

COPY .env /app/.env

# Copy package.json and package-lock.json (if exists)
COPY package*.json ./



# Install dependencies
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the port your app runs on
EXPOSE 8000

# Command to run your application
CMD ["node", "server.js"]
```

server.js

```
/**
 * @description - Initiating HTTP server and listning to incoming http requests
 */

//Imports and Requires
const http = require("http");
const app = require("./app");

require("dotenv").config();

console.log("Server Js executing... Initiating HTTP server");
const port = process.env.PORT;

//Creates HTTP server
const server = http.createServer(app);

//Server then  listen to the port (8002)
server.listen(port, () => {
  console.log(`Course Server is running on port ${port}`);
});
```

Learner Service

database.js

```
/**
 * @description - Database config file
 */

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

```
require('dotenv').config()


//Database URI
const mongoDB_URI = process.env.DATABASE_URI
const DB_CONNECTION_TIMEOUT = process.env.DB_CONNECTION_TIMEOUT
const DB_NAME = process.env.DB_NAME
const DB_MIN_POOL_SIZE = process.env.DB_MIN_POOL_SIZE
const DB_MAX_POOL_SIZE = process.env.DB_MAX_POOL_SIZE

//Connection instance
let mongoInstance = null

// Setting up mongoose connection
const createConnection = async () => {
   try {
      const conn = await mongoose.connect(mongoDB_URI, {
         connectTimeoutMS: DB_CONNECTION_TIMEOUT,
         dbName: DB_NAME,
         minPoolSize: DB_MIN_POOL_SIZE, //Creating a connection pool
         maxPoolSize: DB_MAX_POOL_SIZE
      })


      console.log('Connected to MongoDB')

      return conn

   } catch (error) {

      console.log('Error at creating MongoDB connection : database.js : '+  error)
```

```
    }
}

//Using singleton pattern to ensure there's only one reference to the mongoose connection pool to avoid overhead.
//Multiple clients can access this object same time and accquire a connection from the mongoose connection pool
const getInstance = () => {

    if(!mongoInstance){
        mongoInstance = createConnection()
    }
    return mongoInstance
}

module.exports = {
    getInstance
}
```

learnerController.js

```
const EnrollmentModel = require("../model/enrollmentModel")
const HTTPStatus = require('../enums/httpStatus')

exports.getEnrolledCourses = async(payload) => {
    try {
        const response = await EnrollmentModel.find({learnerId:payload})
        return { status: HTTPStatus.OK, body: response }
    } catch (error) {

        return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }
```

```
    }
}

exports.getCompletedCourses = async(payload) => {
    try {
        const response = await EnrollmentModel.find({learnerId:payload, completionLevel:100})
        return { status: HTTPStatus.OK, body: response }
    } catch (error) {

        return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }

    }
}
```

httpStatus.js

```
//Creating custom status codes for better readability
const HttpStatus = Object.freeze({
    OK: 200,
    CREATED: 201,
    BAD_REQUEST: 400,
    UNAUTHORIZED: 401,
    FORBIDDEN: 403,
    NOT_FOUND: 404,
    INTERNAL_SERVER_ERROR: 500
})

module.exports = HttpStatus
```

enrollmentModel.js

```
/**
```

```
 * @description - This file defines the enrollment model for MongoDB
 */

const mongoose = require("mongoose");

// Schema for courses
const enrollmentSchema = new mongoose.Schema(
  {
    courseId: {
      type: String,
      required: true,
    },
    learnerId: {
      type: String,
      required: true,
    },
    completionLevel: {
      type: Number,
      required: true,
      default: 0,
    },
    paymentStatus: {
      type: Boolean,
      required: true,
      default: false,
    },
  },
  { collection: "enrollment" }
);

// Creating mongoose model using Schema
const EnrollmentModel = mongoose.model("enroll", enrollmentSchema);
```

```
module.exports = EnrollmentModel;
```

learnerRoutes.js

```
/**
 * @description - Route file responsible for the learner processes
 */

//Requires
const express = require("express");
const router = express.Router();
const {
  enrolledCourses,
  completedCoursesList,
} = require("../services/LearnerService");

router.get("/completed/:id", async (req, res) => {
  await completedCoursesList(req, res);
});

router.get("/enrolled/:id", async (req, res) => {
  await enrolledCourses(req, res);
});
//Exporting router to be used by the app.js
module.exports = router;
```

learnerService.js

```
const {
  getCompletedCourses,
  getEnrolledCourses,
```

```
} = require("../controller/LearnerController");
const HttpStatus = require("../enums/httpStatus");

exports.completedCoursesList = async (req, res) => {
  //Expects learner id as a param
  const payload = req.params.id;

  const response = await getCompletedCourses(payload);

  if (response.body.length < 1) {
    res.status(HttpStatus.NOT_FOUND).json("No completed courses found!");
  } else {
    res.status(response.status).json(response.body);
  }
};

exports.enrolledCourses = async (req, res) => {
  //Expects learner id as a param
  const payload = req.params.id;

  const response = await getEnrolledCourses(payload);

  if (response.body.length < 1) {
    res.status(HttpStatus.NOT_FOUND).json("Not enrolled in any courses yet!");
  } else {
    res.status(response.status).json(response.body);
  }
};
```

app.js

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

```
const cors = require("cors");
const connectDB = require("./config/database");
const LearnerRoutes = require("./src/routes/LearnerRoutes");
connectDB.getInstance();

const app = express();

app.use(cors());
app.use(express.json());

//Add routes here
app.use("/courses", LearnerRoutes);

//Exporting app to be used by the server.js
module.exports = app;
```

DockerFile

```
FROM node:20-alpine

WORKDIR /app

COPY .env /app/.env

# Copy package.json and package-lock.json (if exists)
COPY package*.json ./



# Install dependencies
RUN npm install

# Copy the rest of the application
```

```
COPY . .

# Expose the port your app runs on
EXPOSE 8000

# Command to run your application
CMD ["node", "server.js"]
```

server.js

```
/**
 * @description - Initiating HTTP server and listning to incoming http requests
 */

//Imports and Requires
const http = require('http')
const app = require('./app')

require('dotenv').config()

console.log( 'Server Js executing... Initiating HTTP server')
const port = process.env.PORT



//Creates HTTP server
const server = http.createServer(app)

//Server then  listen to the port (8003)
server.listen(port, () => {

   console.log(`Enrollment Server is running on port ${port}`)
})
```

Notification Service

database.js

```
/**
 * @description - Database config file
 */

const mongoose = require('mongoose')
require('dotenv').config()


//Database URI
const mongoDB_URI = process.env.DATABASE_URI
const DB_CONNECTION_TIMEOUT = process.env.DB_CONNECTION_TIMEOUT
const DB_NAME = process.env.DB_NAME
const DB_MIN_POOL_SIZE = process.env.DB_MIN_POOL_SIZE
const DB_MAX_POOL_SIZE = process.env.DB_MAX_POOL_SIZE

//Connection instance
let mongoInstance = null

// Setting up mongoose connection
const createConnection = async () => {
   try {
      const conn = await mongoose.connect(mongoDB_URI, {
         connectTimeoutMS: DB_CONNECTION_TIMEOUT,
         dbName: DB_NAME,
         minPoolSize: DB_MIN_POOL_SIZE, //Creating a connection pool
         maxPoolSize: DB_MAX_POOL_SIZE
```

```
        })


        console.log('Connected to MongoDB')

        return conn

    } catch (error) {

        console.log('Error at creating MongoDB connection : database.js : '+  error)

    }
}

//Using singleton pattern to ensure there's only one reference to the mongoose connection pool to avoid overhead.
//Multiple clients can access this object same time and accquire a connection from the mongoose connection pool
const getInstance = () => {

    if(!mongoInstance){
        mongoInstance = createConnection()
    }
    return mongoInstance
}

module.exports = {
    getInstance
}
```

emailController.js

```
const nodeMail = require("nodemailer");

const EmailController = async (req, res) => {
  const { email, subject, text } = req.body;

  console.log(email, subject, text);
  try {
    const transporter = nodeMail.createTransport({
      service: "gmail",
      auth: {
        user: "isharamadusanka0714@gmail.com",
        pass: "oddlakwxossufvui",
      },
    });

    await transporter.sendMail({
      from: "isharamadusanka0714@gmail.com",
      to: email,
      subject: subject,
      text: text,
    });
    console.log("Email Sent Successfully");
  } catch (error) {
    console.log("Email Not Sent");
    console.log(error);
  }
};

module.exports = EmailController;
```

smsController.js

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

const SMSController = async (req, res) => {

  try {
    const user_id = '27156';
    const api_key = '2wswJTwUKGNu1Rwbd6N9';

    let receiver = req.body.receiver;
    const msg = req.body.message;

    // Validate receiver phone number format
    if (!receiver.startsWith('+')) {
      if (receiver.length === 10 && receiver.startsWith('0')) {
        // Remove the leading '0' and add '+94' instead
        receiver = '+94' + receiver.substring(1);
      } else {
        throw new Error('Receiver phone number should start with "+".');
      }
    } else if (!receiver.match(/^\+94\d{9}$/)) {
      throw new Error('Invalid receiver phone number format.');
    }

    const url = `https://app.notify.lk/api/v1/send?user_id=${user_id}&api_key=${api_key}&sender_id=NotifyDEMO&to=${receiver}&message=${msg}`;

    const response = await axios.post(url);

    console.log(response.data);
```

```
    res.status(200).send(response.data);
  } catch (error) {
    console.error('Error:', error);
    res.status(500).send('Internal Server Error');
  }
};

module.exports = SMSController;
```

router.js

```
const express = require("express");
const router = express.Router();
const EmailController = require("../controllers/EmailController");
const SMSController = require("../controllers/SMSController");

router.post("/send-email", EmailController);
// router.post("/send-sms", SMSController);

module.exports = router;
```

app.js

```
const express = require("express");
const cors = require("cors");
const connectDB = require("./config/database");
const Router = require("./src/routers/Routers");

connectDB.getInstance();

const app = express();
```

```
app.use(cors());
app.use(express.json());

app.use("/", Router);

//Exporting app to be used by the server.js
module.exports = app;
```

DockerFile

```
FROM node:20-alpine

WORKDIR /app

COPY .env /app/.env

# Copy package.json and package-lock.json (if exists)
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the port your app runs on
EXPOSE 8000

# Command to run your application
CMD ["node", "server.js"]
```

server.js

```
/**
 * @description - Initiating HTTP server and listning to incoming http requests
 */

//Imports and Requires
const http = require('http')
const app = require('./app')

require('dotenv').config()

console.log( 'Server Js executing... Initiating HTTP server')
const port = process.env.PORT



//Creates HTTP server
const server = http.createServer(app)

//Server then  listen to the port (8004)
server.listen(port, () => {

   console.log(`Notification Server is running on port ${port}`)
})
```

Payment Service

database.js

```
/**
 * @description - Database config file
 */
```

```
const mongoose = require('mongoose')
require('dotenv').config()


//Database URI
const mongoDB_URI = process.env.DATABASE_URI
const DB_CONNECTION_TIMEOUT = process.env.DB_CONNECTION_TIMEOUT
const DB_NAME = process.env.DB_NAME
const DB_MIN_POOL_SIZE = process.env.DB_MIN_POOL_SIZE
const DB_MAX_POOL_SIZE = process.env.DB_MAX_POOL_SIZE

//Connection instance
let mongoInstance = null

// Setting up mongoose connection
const createConnection = async () => {
   try {
      const conn = await mongoose.connect(mongoDB_URI, {
         connectTimeoutMS: DB_CONNECTION_TIMEOUT,
         dbName: DB_NAME,
         minPoolSize: DB_MIN_POOL_SIZE, //Creating a connection pool
         maxPoolSize: DB_MAX_POOL_SIZE
      })


      console.log('Connected to MongoDB')


      return conn

   } catch (error) {
```

```
        console.log('Error at creating MongoDB connection : database.js : '+  error)

    }
}

//Using singleton pattern to ensure there's only one reference to the mongoose connection pool to avoid overhead.
//Multiple clients can access this object same time and accquire a connection from the mongoose connection pool
const getInstance = () => {

    if(!mongoInstance){
        mongoInstance = createConnection()
    }
    return mongoInstance
}

module.exports = {
    getInstance
}
```

stripeController.js

```
const Stripe = require("stripe");
const stripe = Stripe(process.env.STRIPE_KEY);
const axios = require("axios");
require("dotenv").config();

const CreateCheckout = async (req, res) => {
  try {
    const session = await stripe.checkout.sessions.create({
      payment_method_types: ["card"],
```

```
    mode: "payment",
    line_items: [
     {
       price_data: {
         currency: "usd",
         product_data: {
           name: req.body.name,
           images: [req.body.image],
         },
         unit_amount: req.body.price * 100,
       },
       quantity: req.body.quantity,
     },
    ],
    success_url: `${process.env.CLIENT_URL}course/detailed/${req.body.id}`,
    cancel_url: `${process.env.CLIENT_URL}`,
   });
   const response = await axios.post(`http://localhost:8002/enroll/`, {
    courseId: req.body.id,
    userId: req.body.userId,
   });

   console.log(response);
   res.json({ url: session.url });
  } catch (e) {
   res.status(500).json({ error: e.message });
  }
};

module.exports = { CreateCheckout };
```

stripeRouter.js

```
const express = require("express");
const StripeController = require("../controllers/stripeController");

const router = express.Router();

router.post("/create-checkout-session", StripeController.CreateCheckout);

// Create order function

module.exports = router;
```

app.js

```
const express = require("express");
const cors = require("cors");
const connectDB = require("./config/database");
const stripeRouter = require("./src/routes/stripeRouter");

connectDB.getInstance();

const app = express();

app.use(cors());
app.use(express.json());

app.use("/", stripeRouter);

//Add routes here

//Exporting app to be used by the server.js
module.exports = app;
```

DockerFile

```
FROM node:20-alpine

WORKDIR /app

COPY .env /app/.env

# Copy package.json and package-lock.json (if exists)
COPY package*.json ./



# Install dependencies
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the port your app runs on
EXPOSE 8000

# Command to run your application
CMD ["node", "server.js"]
```

server.js

```
/**
 * @description - Initiating HTTP server and listning to incoming http requests
 */

//Imports and Requires
const http = require("http");
```

```
const app = require("./app");

require("dotenv").config();

console.log("Server Js executing... Initiating HTTP server");
const port = process.env.PORT;

//Creates HTTP server
const server = http.createServer(app);

//Server then  listen to the port (8005)
server.listen(port, () => {
  console.log(`Payment Server is running on port ${port}`);
});
```

User Service

database.js

```
/**
 * @description - Database config file
 */

const mongoose = require('mongoose')
require('dotenv').config()


//Database URI
const mongoDB_URI = process.env.DATABASE_URI
const DB_CONNECTION_TIMEOUT = process.env.DB_CONNECTION_TIMEOUT
const DB_NAME = process.env.DB_NAME
```

```
const DB_MIN_POOL_SIZE = process.env.DB_MIN_POOL_SIZE
const DB_MAX_POOL_SIZE = process.env.DB_MAX_POOL_SIZE

//Connection instance
let mongoInstance = null

// Setting up mongoose connection
const createConnection = async () => {
   try {
      const conn = await mongoose.connect(mongoDB_URI, {
         connectTimeoutMS: DB_CONNECTION_TIMEOUT,
         dbName: DB_NAME,
         minPoolSize: DB_MIN_POOL_SIZE, //Creating a connection pool
         maxPoolSize: DB_MAX_POOL_SIZE
      })


      console.log('Connected to MongoDB')

      return conn

   } catch (error) {

      console.log('Error at creating MongoDB connection : database.js : '+  error)

   }
}
```

//Using singleton pattern to ensure there's only one reference to the mongoose connection pool to avoid overhead.
//Multiple clients can access this object same time and accquire a connection from the mongoose connection pool

```
const getInstance = () => {

   if(!mongoInstance){
      mongoInstance = createConnection()
   }
   return mongoInstance
}

module.exports = {
   getInstance
}
```

adminController.js

```
const learnerBaseModel = require("../models/userModel");
const HttpStatus = require("../enums/httpStatus");

exports.createUser = async (userData) => {
   try {
     const response = await learnerBaseModel.create(userData);
     return { status: HttpStatus.CREATED, body: response };
   } catch (error) {
     console.log(
       "Internal server error at createUser(). More details : " + error
     );
     return { status: HttpStatus.INTERNAL_SERVER_ERROR, body: error };
   }
 };
```

commonController.js

```
const userBaseModel = require("../models/userModel")
```

```
const HTTPStatus = require('../enums/httpStatus')




exports.getUserByName = async (Name) => {
  try {
    const response = await userBaseModel.findOne({ userName: Name })

    return { status: HTTPStatus.OK, body: response }
  } catch (error) {
    console.log(error)
    return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }
  }
}

//Not working yet - ToBe Fixed
exports.saveUserPreferences = async (username, payload) => {
  try {

    const response = await userBaseModel.findOneAndUpdate(
      { userName: username },
      payload,
      { new: true }
    )

    return { status: HTTPStatus.OK, body: response }
  } catch (error) {
    console.log(error)
    return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }
  }
}
```

```
exports.deleteUser = async (userName) => {
   try {
      const response = await userBaseModel.findOneAndDelete({ userName: userName })
      return { status: HTTPStatus.OK, body: response }
   } catch (error) {

      return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }
   }
}

exports.getUsersByType = async (userType) => {
   try {
      const response = await userBaseModel.find({ userRole: userType })
      return { status: HTTPStatus.OK, body: response }
   } catch (error) {
      return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }
   }
}

exports.getUser = async(userId) => {
   try {
      const response = await userBaseModel.findById(userId)
      return { status: HTTPStatus.OK, body: response }
   } catch (error) {
      return { status: HTTPStatus.INTERNAL_SERVER_ERROR, body: error }
   }
}
```

httpStatus.js

```
//Creating custom status codes for better readability
```

```
const HttpStatus = Object.freeze({
   OK: 200,
   CREATED: 201,
   BAD_REQUEST: 400,
   UNAUTHORIZED: 401,
   FORBIDDEN: 403,
   NOT_FOUND: 404,
   INTERNAL_SERVER_ERROR: 500
})

module.exports = HttpStatus
```

userModel.js

```
/**
 * @description - This file defines the base user model for mongoDB
 */

const mongoose = require('mongoose')

const UserRole = {
   user: "user",
   INSTRUCTOR: "instructor",
   ADMIN: "admin",
  }

const userBaseSchema = new mongoose.Schema({
   userName: { type: String, required: true, unique: true },
   email: { type: String, required: true, unique: true },
   image: { type: String, required: false },
   password: { type: String, required: true },
   mobileNo: { type: String, required: false },
```

```
    userRole: {
      type: String,
      required: false,
      enum: Object.values(UserRole),
      default: "learner",
    }, //Roles : learner, instructor, admin
    userInterests: { type: [String], required: false }
  }, { collection: 'authUsers'})

//Creating mongoose model using Schema
const userBaseModel = mongoose.model('userBaseModel', userBaseSchema)

//Exporting model to be used by userController.js
module.exports = userBaseModel
```

adminRoutes.js

```
/**
 * @description - Route file responsible for the admin processes
 */

//Requires
const express = require('express')
const router = express.Router()
const { addNewUser } = require('../services/adminService')

//Save preferences
router.post('/', async(req,res) => {
    await addNewUser(req,res)
})

//Exporting router to be used by the app.js
```

```
module.exports = router
```

commonRoutes.js

```
/**
 * @description - Route file responsible for the learner processes
 */

//Requires
const express = require('express')
const router = express.Router()
const { updateUser, deleteUser, getAllUserByType, getUserById } = require('../services/commonService')

router.get('/:id', async (req, res) => {
  await getAllUserByType(req, res)
})

router.get('/user/:id', async(req,res) => {
  await getUserById(req,res)
})

//Save preferences
router.patch('/', async (req, res) => {

  await updateUser(req, res)
})

//Delete learner
router.delete('/', async (req, res) => {
  await deleteUser(req, res)
})
```

```
//Exporting router to be used by the app.js
module.exports = router
```

adminService.js

```
const {createUser} = require('../controllers/adminController')
const bcryptjs = require("bcryptjs");
const saltCount = 10;


const hashPasswordGen = async (plainPsw) => {
   const hashPsw = await bcryptjs.hash(plainPsw, saltCount);
   return hashPsw;
 };

exports.addNewUser = async(req,res) => {

   const payload = req.body;
   const hashedPassword = await hashPasswordGen(payload.password);

   //Constructing the new user payload
   const newUser = {
     userName: payload.userName,
     email: payload.email,
     image: payload.image,
     password: hashedPassword,
     mobileNo: payload.mobileNo,
     userRole: payload.userRole,
   };
```

```
    console.log(newUser);

    //Registering the user with createUser function in userController.js
    const response = await createUser(newUser);

    res.status(response.status).json(response.body);
}
```

commonService.js

```
const { saveUserPreferences, deleteUser, getUsersByType, getUser } = require('../controllers/commonController')


exports.updateUser = async(req,res) => {


    const payload = req.body
    const userName = req.body.userName

    const response = await saveUserPreferences(userName,payload)

    res.status(response.status).json(response.body)
}

exports.deleteUser = async(req,res) => {
    const name = req.body.userName

    const response = await deleteUser(name)

    res.status(response.status).json(response.body)
}
```

```
exports.getAllUserByType = async(req,res) => {

    const userType = req.params.id

    const response = await getUsersByType(userType)

    res.status(response.status).json(response.body)
}

exports.getUserById = async(req,res) => {
    const userId = req.params.id

    const response = await getUser(userId)

    res.status(response.status).json(response.body)
}
```

app.js

```
const express = require("express")
const cors = require("cors")
const connectDB = require('./config/database')
const commonRoutes = require('./src/routes/commonRoutes')
const adminRoutes =  require('./src/routes/adminRoutes')

connectDB.getInstance()

const app = express()

app.use(cors())
app.use(express.json())
```

```
//routes here
app.use('/common', commonRoutes)
app.use('/admin',adminRoutes)

//Exporting app to be used by the server.js
module.exports = app
```

DockerFile

```
FROM node:20-alpine

WORKDIR /app

COPY .env /app/.env

# Copy package.json and package-lock.json (if exists)
COPY package*.json ./



# Install dependencies
RUN npm install

# Copy the rest of the application
COPY . .

# Expose the port your app runs on
EXPOSE 8000

# Command to run your application
CMD ["node", "server.js"]
```

server.js

```
/**
 * @description - Initiating HTTP server and listning to incoming http requests
 */

//Imports and Requires
const http = require('http')
const app = require('./app')

require('dotenv').config()

console.log( 'Server Js executing... Initiating HTTP server')
const port = process.env.PORT



//Creates HTTP server
const server = http.createServer(app)

//Server then  listen to the port (8002)
server.listen(port, () => {

   console.log(`User Server is running on port ${port}`)

})
```

docker-compose.yml

```
version: "3.8"

services:
  gateway:
```

```
  container_name: gateway
  build: ./ApiGateWay
  ports:
    - "8000:8000"

auth:
  container_name: auth
  build: ./AuthenticationService
  ports:
    - "8001:8001"

course:
  container_name: course
  build: ./CourseService
  ports:
    - "8002:8002"

learner:
  container_name: learner
  build: ./LearnerService
  ports:
    - "8003:8003"

notification:
  container_name: notification
  build: ./NotificationService
  ports:
    - "8004:8004"

payment:
  container_name: payment
  build: ./PaymentService
```

```
  ports:
   - "8005:8005"

 user:
  container_name: user
  build: ./UserService
  ports:
   - "8006:8006"
```

- **Learnify-FE**

AddUserPanel.jsx

```
import React, { useEffect, useState } from 'react'
import axios from "../../util/AxiosInstance"
import Swal from "sweetalert2";

export default function AddUserPanel() {

  const [userName, setUserName] = useState('')
  const [email, setEmail] = useState('')
  const [mobileNo, setMobileNo] = useState('')
  const [userRole, setUserRole] = useState('instructor')
  const [psw, setPsw] = useState('')
  const [rePsw, setRePsw] = useState('')

  const handleUserName = (e) => {
    setUserName(e.target.value)
  }

  const handleEmail = (e) => {
```

```
    setEmail(e.target.value)
}

const handleMobileNo = (e) => {
    setMobileNo(e.target.value)
}

const handleUserRole = (e) => {
    setUserRole(e.target.value)
}

const handlePsw = (e) => {
    setPsw(e.target.value)
}

const handleRePsw = (e) => {
    setRePsw(e.target.value)
}

const handleAddUser = async(e) => {

    e.preventDefault()

    if (psw !== rePsw) {

        Swal.fire(
            'Password Mismatch',
            'Please make sure the passwords match.',
            'error'
        );
        return; // Exit function if passwords don't match
    }
```

```
      const newUser = {
         userName : userName,
         email : email,
         mobileNo : mobileNo,
         userRole : userRole,
         password : psw,
         image                                                                    :
'https://th.bing.com/th/id/OIP.MtqzdcGHvL8kNnw5IDh1iQHaHa?rs=1&pid=ImgDetMain'
      }

      const res = await axios.post('auth/register', newUser )

      clearStates()

      Swal.fire(
         ` Successfully Added ${userRole} user`,
         "",
         "success"
       )




   }

   const clearStates = () => {
      setUserName('')
      setEmail('')
      setMobileNo('')
      setPsw('')
      setRePsw('')
      setUserRole('instructor')
```

```
  }

  return (
    <form className=' w-full h-full rounded-xl pt-5 bg-white flex flex-col justify-between shadow' onSubmit={handleAddUser}>
      <div className='flex flex-col items-center'>
        <span className=' font-FutuBt font-bold  text-3xl text-[#575757]'>Add New User</span>
        <div className=' w-full flex justify-between px-5 my-5'>
          <input type='text' required placeholder='User Name' className=' py-2 w-[45%] border-b-2 pl-2 border-b-[#575757] rounded' onChange={handleUserName} />
          <input type='email' required placeholder='Email' className=' py-2 w-[45%] border-b-2 pl-2 border-b-[#575757] rounded' onChange={handleEmail} />
        </div>
        <div className=' w-full flex justify-between px-5 '>
          <input type='number' required placeholder='Mobile no' className='  py-2 w-[45%] border-b-2 pl-2 border-b-[#575757] rounded' onChange={handleMobileNo} />
          <div className=' border-b-2 pl-2 border-b-[#575757] bg-white rounded w-[45%] flex items-center'>
            <label> Select role : </label>
            <select onChange={handleUserRole} defaultValue='instructor' >
              <option value='instructor'> Instructor</option>
              <option value='admin'> Admin </option>
            </select>
          </div>

        </div>
        <div className=' w-full flex justify-between px-5 my-5'>
          <input type='password' required placeholder='Password' className=' py-2 w-[45%] border-b-2 pl-2 border-b-[#575757] rounded' onChange={handlePsw} />
          <input type='password' required placeholder='Confirm Password' className=' py-2 w-[45%] border-b-2 pl-2 border-b-[#575757] rounded' onChange={handleRePsw} />
```

```
            </div>
          </div>

          <div className='flex w-full justify-end pr-5  items mb-5'>
            <button type='submit' className={`px-5 py-2 border-blue-500 border-2 rounded-2xl text-[#575757] font-semibold hover:scale-105 cursor-pointer hover:text-white hover:bg-blue-500 hover:border-0 ${psw !== rePsw && 'opacity-50 pointer-events-none'}`} >Add User</button>
          </div>
        </form>
    )
}
```

CourseDetailModel.jsx

```
import React from 'react'

import 'react-toastify/dist/ReactToastify.css'




export default function CourseDetailModel({ open, onClose, course, approveHandler }) {


    const handleApprove = async () => {

        approveHandler()
        onClose()

    }
```

```
  return (
    //Background
    <div onClick={onClose} className={` absolute z-[999] inset-0 flex justify-center items-center ${open ? 'visible bg-black/50 ' : 'invisible'}`}>
      {/**Actual modal */}

      <div onClick={(e) => e.stopPropagation()} className={`bg-white p-5 items-start flex flex-row w-fit h-fit rounded-md shadow transition-all ${open ? 'opacity-100' : 'opacity-0'}`}>

        <div className=' flex w-full  flex-col h-fit items-center'>
          <span className=' text-4xl '> {course.title} </span>
          <img src={course.coverImage} className=' my-5 w-full h-[30vh]'/>
          <div className=' font-FuturaMdBt w-full flex flex-row justify-start items-center'>
            <span className=' text-2xl '>Description :  </span>
            <span> {course.description} </span>
          </div>
          <div className=' font-FuturaMdBt w-full flex flex-row justify-start items-center'>
            <span className=' text-2xl '>Instructor :   </span>
            <span> {course.instructor}</span>
          </div>
          <div className=' font-FuturaMdBt w-full flex flex-row justify-start items-center'>
            <span className=' text-2xl '>Duration :   </span>
            <span>{course.duration} Days</span>

          </div>
          <div className=' font-FuturaMdBt w-full flex flex-row justify-start items-center'>
            <span className=' text-2xl '>Price :   </span>
            <span>{course.price}$ </span>

          </div>
```

```
            <div className=' mx-10 mb-10 w-[30vw] mt-10 h-10 items-center  flex justify-between'>
              <span className=' px-10 py-2 border-green-500 border-2 text-[#575757] rounded-xl cursor-pointer font-semibold hover:scale-105 hover:text-white hover:bg-green-500 hover:border-0' onClick={() => handleApprove()}>Approve</span>
              <span className=' px-10 py-2 border-red-500 border-2 text-[#575757] rounded-xl cursor-pointer font-semibold hover:scale-105 hover:text-white hover:bg-red-500 hover:border-0' onClick={() => onClose()}>Cancel</span>
            </div>
          </div>
          <div className=' text-4xl cursor-pointer justify-center items-center flex text-[#575757]'>
            <span onClick={onClose}>&times;</span>
          </div>
        </div>

      </div>
  )
}
```

PendingCourses.jsx

```
import React, { useEffect, useState } from 'react'
import axios from "../../util/AxiosInstance";
import { ToastContainer, toast } from 'react-toastify'
import 'react-toastify/dist/ReactToastify.css'
import CourseDetailModel from './CourseDetailModel';

export default function PendingCoursesPanel() {

  const [pendingCourses, setPendingCourses] = useState([])
```

```
const [open, setOpen] = useState(false)

useEffect(() => {
  getAllPendingCourses()
}, []);

const getAllPendingCourses = async () => {

  try {
    const res = await axios.get('course//course/false')
    console.log(res.data)
    setPendingCourses(res.data)
  } catch (error) {
    console.log(error)
  }



}

const onApprove = async (id) => {

  const courseId = id
  const payload = {
    authorized: true
  }

  try {
    const res = await axios.patch(`course//course/${courseId}`, payload)

    if (res.status === 201) {

      setPendingCourses((prevPendingCourses) => {
```

```
        getAllPendingCourses()
        return prevPendingCourses
      })

      toast("Course approved successfully!")
    } else {
      console.log("GetAllPendingCourses NotRunning!")
    }

  } catch (error) {
    console.log(error)
  }
}

const onCancel = async (id) => {

  const courseId = id
  const payload = {
    authorized: false
  }

  try {
    const res = await axios.patch(`course//course/${courseId}`, payload)

    if (res.status === 201) {

      setPendingCourses((prevPendingCourses) => {
        getAllPendingCourses()
        return prevPendingCourses
      })

    } else {
```

```
      console.log("GetAllPendingCourses NotRunning!")
    }

  } catch (error) {
    console.log(error)
  }
}


 return (
  <div className=' w-full h-full  '>
    <ToastContainer autoClose={3000} />
    <div className='flex flex-col items-center  '>
      <span className=' text-3xl font-bold my-5 text-[#575757]'> Pending Courses</span>
      <div  className='p-2  w-[calc(30vw)]  h-[68vh]  bg-white  rounded-2xl  overflow-y-auto shadow'>
        {pendingCourses.map((item) => (
          <div className=' p-2 w-full h-fit flex flex-row justify-between items-center border-gray-500 border-2 rounded-lg mt-5 shadow-lg  hover:bg-gray-200' >
            <div className=' w-fit h-full  py-3 cursor-pointer' onClick={() => setOpen(true)}>
              <div  className=' font-FuturaMdBd     font-semibold  mx-2  text-[#575757]  ' >{item.title}</div>
            </div>


            <div className=' flex font-FuturaMdBt text-md text-white items-center'>
              <span className=' px-5 py-2 border-green-500 border-2  cursor-pointer rounded-xl shadow hover:scale-105 text-[#575757] hover:text-white hover:bg-green-500 hover:border-0 ' onClick={() => onApprove(item._id)}>approve</span>
              <span  className='p-1  m-2  cursor-pointer  text-[#575757]' onClick={() => onCancel(item._id)}>&times; </span>
            </div>
```

```
          <CourseDetailModel open={open} onClose={() => setOpen(false)} course={item}
approveHandler={() => onApprove(item._id)} />
          </div>

        ))}
       </div>

     </div>
   </div>
  )
}
```

UserDetailsPanel.jsx

```
import React, { useEffect, useState } from 'react'
import axios from '../../util/AxiosInstance'
import UserUpdateModel from './UserUpdateModel'

export default function UserDetailsPanel() {

  const [userData, setUserData] = useState('')
  const [updateModel , setUpdateModel] = useState(false)

  useEffect(() => {
    getUserDetails()
  }, [updateModel])

  const getUserDetails = async () => {
    const user = localStorage.getItem('jsonwebtoken')
    const userData = JSON.parse(user)
    const userID = userData.decodedJWT.userId
```

```
  const res = await axios.get(`user/common/user/${userID}`)

  if (res.status == 200) {
    setUserData(res.data)
  }
 }

 const toggleUpdateModel = () => {
  if(updateModel){
    setUpdateModel(false)
  }else{
    updateModel(true)
  }
 }

 return (
  <div className=' flex flex-col mt-16 p-5 h-[68vh] justify-between bg-white rounded-lg shadow'>
    <div className='flex flex-col text-[#575757]'>
      <div className='flex justify-center items-center flex-col p-5 font-FuturaMdBt'>
        <img src={userData.image} className='w-28 h-28 rounded-full my-5' />
        <span className=' text-3xl font-bold'>{userData.userName}</span>
      </div>
      <div className=' flex flex-col justify-between'>
        <div className=' flex flex-col items-start'>
          <span className='text-lg'>My email </span>
          <span className=' text-sm'>{userData.email}</span>
        </div>
        <div className=' flex flex-col items-start mt-10'>
          <span className='text-lg'>My contact no </span>
          <span className='text-sm'>{userData.mobileNo}</span>
```

```
        </div>
       </div>
     </div>
     <div className='flex place-self-end items-end'>
      <span className=' px-5 py-2 border-blue-500 border-2 rounded-2xl text-[#575757] font-semibold hover:scale-105 cursor-pointer hover:text-white hover:bg-blue-500 hover:border-0' onClick={() => setUpdateModel(true)}>update</span>
     </div>
     <UserUpdateModel open ={updateModel} onClose={() =>toggleUpdateModel()} user={userData} />

   </div>
  )
}
```

UserUpdateModel.jsx

```
import React, { useState, useEffect } from 'react'
import { ToastContainer, toast } from 'react-toastify'
import 'react-toastify/dist/ReactToastify.css'
import axios from '../../util/AxiosInstance'



export default function UserUpdateModel({ open, onClose, user }) {

   const [UserName, setUserName] = useState(user.userName)
   const [Email, setEmail] = useState(user.email)
   const [Number, setNumber] = useState(user.mobileNo)



   useEffect(() => {
```

```
    setUserName(user.userName);
    setEmail(user.email);
    setNumber(user.mobileNo);
}, [user]);

const handleUserName = (e) => {
    setUserName(e.target.value)

}

const handleEmail = (e) => {
    setEmail(e.target.value)
}

const handleNumber = (e) => {
    setNumber(e.target.value)
}

const handleUpdate = async () => {

    console.log("Stored : " + UserName)

    const payload = {
        userName: UserName,
        email: Email,
        mobileNo: Number
    }

    const res = await axios.patch('user/common/', payload)
    console.log(res.status)
    if (res.status === 200) {
```

```
        toast("User details updated successfully!")



        setTimeout(function () {
           onClose()
        }, 3000);
        setTimeout();


    }
  }


  return (
    //Background
    <div onClick={onClose} className={` absolute z-[999] inset-0 flex justify-center items-center ${open ? 'visible bg-black/50 ' : 'invisible'}`}>
        {/**Actual modal */}
        <ToastContainer autoClose={3000} />
        <div onClick={(e) => e.stopPropagation()} className={`bg-white p-2 items-start flex flex-row w-fit h-fit rounded-md shadow transition-all ${open ? 'opacity-100' : 'opacity-0'}`}>

          <div className=' flex w-full  flex-col h-fit items-center'>
            <span className=' font-semibold font-FuturaMdBt mt-10 text-3xl '>Update My Details</span>
            <div className=' mx-10 w-[30vw] mt-10 h-10 border-b-2 border-black flex justify-center'>
              <input type='text' className='pl-5 font-FuturaMdBt w-full ' placeholder={user.userName} onChange={handleUserName} />
```

```
            </div>
            <div className=' mx-10 w-[30vw] mt-10 h-10 border-b-2 border-black flex justify-center'>
              <input type='email' className=' pl-5 font-FuturaMdBt w-full ' placeholder={user.email} onChange={handleEmail} />
            </div>
            <div className=' mx-10 w-[30vw] mt-10 h-10 border-b-2 border-black flex justify-center'>
              <input type='text' className=' pl-5 font-FuturaMdBt w-full ' placeholder={user.mobileNo} onChange={handleNumber} />
            </div>
            <div className=' mx-10 mb-10 w-[30vw] mt-10 h-10 items-center  flex justify-between'>
              <span className=' px-10 py-2 border-blue-500 border-2 text-[#575757] rounded-xl cursor-pointer font-semibold hover:scale-105 hover:text-white hover:bg-blue-500 hover:border-0' onClick={() => handleUpdate()}>Save</span>
              <span className=' px-10 py-2 border-red-500 border-2 text-[#575757] rounded-xl cursor-pointer font-semibold hover:scale-105 hover:text-white hover:bg-red-500 hover:border-0' onClick={() => onClose()}>Cancel</span>
            </div>

          </div>
          <div className=' text-4xl  cursor-pointer justify-center items-center flex text-[#575757]'>
            <span onClick={onClose}>&times;</span>
          </div>
        </div>

      </div>
    )
}
```

ViewUserListModel.jsx

```
import React, { useEffect, useState } from 'react'
import axios from '../../util/AxiosInstance'

export default function ViewUserListModel({ open, onClose, userType }) {

    const [userList, setUserList] = useState([])



    useEffect(() => {
        console.log('User type is : ' + userType)
        getUserList()
    }, [userType])

    const getUserList = async () => {

        if (userType === 'instructor') {
            const res = await axios.get(`user/common/instructor`)

            if (res.status === 200) {
                setUserList(res.data)
                console.log(res.data)

            }

        }

        if (userType === 'admin') {
            const res = await axios.get(`user/common/admin`)

            if (res.status === 200) {
```

```
            setUserList(res.data)
            console.log(res.data)
        }

    }

}

return (
    <div onClick={onClose} className={` absolute z-[999] inset-0 flex justify-center items-center ${open ? 'visible bg-black/50 ' : 'invisible'}`}>
        {/**Actual modal */}

        <div onClick={(e) => e.stopPropagation()} className={`bg-white p-2 items-start flex flex-row w-fit h-fit rounded-md shadow transition-all ${open ? 'opacity-100' : 'opacity-0'}`}>

            <div className=' flex flex-col w-[50vw] h-[50vh] text-[#575757] items-center'>
                <div className=' flex py-2 w-full justify-center mb-3'>
                    <span className={`${userType === 'instructor' ? 'block' : 'hidden'} text-2xl text-[#575757] font-semibold`}>Current Instructors List</span>
                    <span className={`${userType === 'admin' ? 'block' : 'hidden'} text-2xl text-[#575757] font-semibold`}>Current Administrators List</span>
                </div>
                {userList.map((item) => (
                    <div className='w-[90%] flex my-2 justify-between py-2 px-4 border-2 text-[#575757] border-[#575757] font-FuturaMdBt rounded-xl' key={item.id}>
                        <span>{item.userName}</span>
                        <span> Mobile no : {item.mobileNo ? item.mobileNo:'No mobile Number yet'}</span>
                    </div>
                ))}
```

```
            </div>
            <div  className=' text-4xl   cursor-pointer  justify-center  items-center  flex  text-[#575757]'>
               <span onClick={onClose}>&times;</span>
            </div>
          </div>

       </div>
    )
}
```

AwardsSection.jsx

```
import React from "react";
import awards from "../../../assets/awards.png";

const AwardsSection = () => {
  return (
    <div className="bg-white h-auto">
      <div className="text-center mt-5 mb-5 text-md lg:text-xl">
        <p>Trusted By Over 13400 Students & Organizations</p>
        <p>Recommended Around The world</p>
      </div>
      <div className="flex pt-10 justify-center">
        <img src={awards} alt="" className="w-[50%] h-auto" />
      </div>
    </div>
  );
};

export default AwardsSection;
```

CoursesCategory.jsx

```
import React from "react";

import { SiInteractiondesignfoundation } from "react-icons/si";
import { FaBusinessTime } from "react-icons/fa6";
import { FaComputer } from "react-icons/fa6";
import { GiReceiveMoney } from "react-icons/gi";
import { GiHealthNormal } from "react-icons/gi";
import { FaRobot } from "react-icons/fa6";
import { MdDataThresholding } from "react-icons/md";

const categories = [
 {
  name: "Art & Design",
  icon: <SiInteractiondesignfoundation className="h-16 w-auto" />,
  courseCount: 100,
 },
 {
  name: "Business",
  icon: <FaBusinessTime className="h-16 w-auto" />,
  courseCount: 10,
 },
 {
  name: "Development",
  icon: <FaComputer className="h-16 w-auto" />,
  courseCount: 1000,
 },
 {
  name: "Finance",
  icon: <GiReceiveMoney className="h-16 w-auto" />,
  courseCount: 500,
```

```
  },
  {
    name: "Health & Fitness",
    icon: <GiHealthNormal className="h-16 w-auto" />,
    courseCount: 30,
  },
  {
    name: "Technology",
    icon: <FaRobot className="h-16 w-auto" />,
    courseCount: 10000,
  },
  {
    name: "Data Science",
    icon: <MdDataThresholding className="h-16 w-auto" />,
    courseCount: 50,
  },
  {
    name: "Business",
    icon: <FaBusinessTime className="h-16 w-auto" />,
    courseCount: 100,
  },
];

const CoursesCategory = () => {
  return (
    <div className=" pb-10 bg-[#F9F8F4]">
      <div className="text-center mt-10 ">
        <div className="text-md pt-10 lg:text-xl">TOP CLASS COURSES</div>
        <div className="text-lg pt-10 pb-10 lg:text-5xl">
          Explore 4000+ Online Courses
        </div>
      </div>
```

```
    <div className="mx-12 lg:mx-48 grid grid-cols-2 xl:grid-cols-4 gap:4 lg:gap-6">
      {categories.map((category) => (
        <div  className="mb-8  rounded-[20px]  bg-white  p-2  shadow-md   hover:shadow-lg hover:cursor-pointer hover:bg-[#411EE2] hover:text-white  flex flex-rows items-center gap-6 justify-start lg:justify-center">
          <div className="  text-black text-3xl  h-[70px] w-[70px] flex items-center justify-center rounded-2xl">
            {category.icon}
          </div>
          <div className=" text-dark mb-3 mt-5 text-lg justify-start flex flex-col font-semibold">
            <div> {category.name}</div>

            <div className="font-light">{category.courseCount} Courses</div>
          </div>
        </div>
      ))}
    </div>
  </div>
 );
};

export default CoursesCategory;
```

HeroSection.css

```
.hero {
 background-image: url("../../../assets/bg.png");
 background-size: cover;
 background-attachment: fixed;
}
```

HeroSection.jsx

```
import React from "react";
import "./HeroSection.css";
import { FaArrowRight } from "react-icons/fa6";
import { Link } from "react-router-dom";

const HeroSection = () => {
  return (
    <>
      <section className="hero">
        <div className="flex px-12   lg:px-48">
          <div className=" mb-48 mt-36 ">
            <p className="whitespace-nowrap text-md lg:text-xl mb-5 text-left">
              100% Quality Courses
            </p>
            <p className="whitespace-nowrap text-lg  lg:text-5xl mb-5 font-extrabold">
              Find Your Perfect Courses
            </p>
            <p className=" text-md lg:text-xl">
              We Have 40K+ Online Courses and 10K+ Instructors
            </p>

            <Link to={"/courses"}>
              <button className="bg-[#411EE2] p-6 mt-12 text-white">
                <div className="flex flex-row">
                  <div>Explore All Courses</div>
                  <div className="ml-5 mt-1">
                    <FaArrowRight />
                  </div>
                </div>
              </button>
            </Link>
```

```
        </div>
      </div>
    </section>
  </>
 );
};

export default HeroSection;
```

AddCourseModal.jsx

```
import React, { useState } from "react";
import { MinusCircleOutlined, PlusOutlined } from "@ant-design/icons";
import { Button, Modal, Checkbox, Divider } from "antd";
import { Form, Input, Radio } from "antd";
import axios from "../../util/AxiosInstance";
import uploadFileToFirebase from "../../util/UploadFilesToFIreBase";
import Swal from "sweetalert2";
const { TextArea } = Input;

const plainOptions = [
  "Art & Design",
  "Business",
  "Development",
  "Finance",
  "Health & Fitness",
  "Technology",
  "Data Science",
];
const defaultCheckedList = [""];

const AddCourseModal = ({ open, close }) => {
```

```
const [confirmLoading, setConfirmLoading] = useState(false);
const [form] = Form.useForm();
const [file, setFile] = useState("");
const [checkedList, setCheckedList] = useState(defaultCheckedList);

const handleCancel = () => {
  close();
};

const onFinish = async (values) => {
  try {
    let uploadImg = "No Image";
    setConfirmLoading(true);
    if (file) {
      const response = await uploadFileToFirebase(file);
      uploadImg = response;
    }
    const response = await axios.post("course/", {
      title: values.title,
      description: values.description,
      instructor: values.instructor,
      duration: values.duration,
      coverImage: uploadImg,
      price: values.price,
      tags: checkedList,
      lessons: values.lessons.map((lesson, lessonIndex) => ({
        title: lesson.title,
        description: lesson.description,
        videoUrl: lesson.video,
        quiz: lesson.quiz.map((question, questionIndex) => ({
          question: question.question,
          option1: question.option1,
```

```
        option2: question.option2,
        option3: question.option3,
        option4: question.option4,
        correctAnswer: question.correctAnswer,
      })),
    })),
  });

  if (response.status === 201) {
    console.log("Course added successfully!");
    await Swal.fire({
      icon: "success",
      title: "Course added successfully!",
      showConfirmButton: false,
      timer: 1500,
    });
    close();
  } else {
    console.error("Failed to add course:", response.statusText);
    await Swal.fire({
      icon: "error",
      title: "Failed to add course",
      text: response.statusText,
    });
  }
} catch (error) {
  console.error("Error adding course:", error);
  await Swal.fire({
    icon: "error",
    title: "Error",
    text: error.response.data.message,
  });
```

```
  } finally {
    setConfirmLoading(false);
  }
};
const onChange = (list) => {
  setCheckedList(list);
};

const onCheckAllChange = (e) => {
  setCheckedList(e.target.checked ? plainOptions : []);
};

return (
  <>
    <Modal
      title="Add Course Details"
      open={open}
      confirmLoading={confirmLoading}
      width={1000}
      footer={null}
      onCancel={handleCancel}
    >
      <div>
        <Form form={form} name="form" onFinish={onFinish} autoComplete="off">
          <div className="pb-3 font-bold text-lg"> Basic Details</div>
          <div className="pb-3">Title</div>
          <Form.Item
            name="title"
            rules={[
              {
                required: true,
                message: "Please input title for the course!",
```

```
    },
  ]}
>
  <Input placeholder="Course Title" />
</Form.Item>
<div className="pb-3"> Description</div>
<Form.Item
  name="description"
  rules={[
    {
      required: true,
      message: "Please input description for the course!",
    },
  ]}
>
  <TextArea
    placeholder="Description of the course"
    autoSize={{
      minRows: 2,
      maxRows: 6,
    }}
  />
</Form.Item>
<div className="pb-3"> Instructor</div>
<Form.Item
  name="instructor"
  rules={[
    {
      required: true,
      message: "Please input description for the course!",
    },
  ]}
```

```
>
  <Input placeholder="Instructor" />
</Form.Item>
<div className="pb-3"> Duration</div>
<Form.Item
  name="duration"
  rules={[
    {
      required: true,
      message: "Please input total course duration!",
    },
  ]}
>
  <Input placeholder="Total course duration(hours)" type="number" />
</Form.Item>
<div className="pb-3"> Cover Image </div>
<label>
  <img
    className=""
    src={
      file
        ? URL.createObjectURL(file)
        : "https://icon-library.com/images/no-image-icon/no-image-icon-0.jpg"
    }
    alt="avatar"
    style={{
      width: "120px",
      height: "120px",
      cursor: "pointer",
    }}
  />
</label>
```

```
<Form.Item
  name="coverImage"
  rules={[
    {
      required: true,
      message: "Please input Cover image for the course!",
    },
  ]}
>
  <input
    type="file"
    accept="image/*"
    onChange={(e) => {
      const file = e.target.files[0];
      setFile(file);
    }}
  />
</Form.Item>
<div className="pb-3"> Price ($)</div>
<Form.Item
  name="price"
  rules={[
    {
      required: true,
      message: "Please input total price!",
    },
  ]}
>
  <Input placeholder="Total course price($)" type="text" />
</Form.Item>
<div className="pb-3">Tags</div>
<Checkbox
```

```
  indeterminate={
    checkedList.length > 0 &&
    checkedList.length < plainOptions.length
  }
  onChange={onCheckAllChange}
  checked={checkedList.length === plainOptions.length}
  className="pb-3"
>
  Check all
</Checkbox>
<br />
<Checkbox.Group
  options={plainOptions}
  value={checkedList}
  onChange={onChange}
  className="pb-3"
/>
{/* Dynamic fields lesson */}
<div className="pb-3 font-bold text-lg"> Lessons</div>
<Form.List name="lessons">
  {(fields, { add: addLesson, remove: removeLesson }) => (
    <>
      {fields.map((lessonField, lessonIndex) => (
        <div key={lessonField.key}>
          <div className="flex justify-between items-center pb-3">
            <h3>Lesson {lessonIndex + 1}</h3>
            {fields.length > 1 && (
              <MinusCircleOutlined
                onClick={() => removeLesson(lessonField.name)}
              />
            )}
          </div>
```

```
<Form.Item
 {...lessonField}
 name={[lessonField.name, "title"]}
 fieldKey={[lessonField.fieldKey, "title"]}
 rules={[
  {
   required: true,
   message: "Please input title for the lesson!",
  },
 ]}
>
 <Input placeholder="Lesson Title" />
</Form.Item>
<Form.Item
 {...lessonField}
 name={[lessonField.name, "description"]}
 fieldKey={[lessonField.fieldKey, "description"]}
 rules={[
  {
   required: true,
   message: "Please input description for the lesson!",
  },
 ]}
>
 <TextArea
  placeholder="Description of the lesson"
  autoSize={{ minRows: 2, maxRows: 6 }}
 />
</Form.Item>
<Form.Item
 {...lessonField}
 name={[lessonField.name, "video"]}
```

```
 fieldKey={[lessonField.fieldKey, "video"]}
 rules={[
  {
   required: true,
   message: "Please input video URL for the lesson!",
  },
 ]}
>
 <Input placeholder="Lesson Video URL" />
</Form.Item>
{/* Dynamic fields for quizzes */}
<div className="pb-3 font-bold text-lg">
 Questions and Answers
</div>
<Form.List name={[lessonField.name, "quiz"]}>
 {(quizFields, { add: addQuiz, remove: removeQuiz }) => (
  <>
   {quizFields.map((quizField, quizIndex) => (
    <div key={quizField.key}>
     <div className="flex justify-between items-center pb-3">
      <h4>Question {quizIndex + 1}</h4>
      {quizFields.length > 1 && (
       <MinusCircleOutlined
        onClick={() => removeQuiz(quizField.name)}
       />
      )}
     </div>
     <Form.Item
      {...quizField}
      name={[quizField.name, "question"]}
      fieldKey={[quizField.fieldKey, "question"]}
      rules={[
```

```
    {
      required: true,
      message: "Please input question!",
    },
  ]}
>
  <Input placeholder="Question" />
</Form.Item>
<Form.Item
  {...quizField}
  name={[quizField.name, "option1"]}
  fieldKey={[quizField.fieldKey, "question"]}
  rules={[
    {
      required: true,
      message: "Please input Option 1!",
    },
  ]}
>
  <Input placeholder="Option 1" />
</Form.Item>

<Form.Item
  {...quizField}
  name={[quizField.name, "option2"]}
  fieldKey={[quizField.fieldKey, "question"]}
  rules={[
    {
      required: true,
      message: "Please input Option 2!",
    },
  ]}
```

```
>
  <Input placeholder="Option 2" />
</Form.Item>
<Form.Item
  {...quizField}
  name={[quizField.name, "option3"]}
  fieldKey={[quizField.fieldKey, "question"]}
  rules={[
    {
      required: true,
      message: "Please input Option 3!",
    },
  ]}
>
  <Input placeholder="Option 3" />
</Form.Item>
<Form.Item
  {...quizField}
  name={[quizField.name, "option4"]}
  fieldKey={[quizField.fieldKey, "question"]}
  rules={[
    {
      required: true,
      message: "Please input Option 4!",
    },
  ]}
>
  <Input placeholder="Option 4" />
</Form.Item>
<Form.Item
  {...quizField}
  name={[quizField.name, "correctAnswer"]}
```

```
        fieldKey={[
          quizField.fieldKey,
          "correctAnswer",
        ]}
        rules={[
          {
            required: true,
            message:
              "Please select the correct answer!",
          },
        ]}
      >
        <Radio.Group>
          <Radio value="option1">Option 1</Radio>
          <Radio value="option2">Option 2</Radio>
          <Radio value="option3">Option 3</Radio>
          <Radio value="option4">Option 4</Radio>
        </Radio.Group>
      </Form.Item>
    </div>
  ))}
  <Form.Item>
    <Button
      type="dashed"
      onClick={() => addQuiz()}
      icon={<PlusOutlined />}
    >
      Add Question
    </Button>
  </Form.Item>
 </>
)}
```

```
            </Form.List>
          </div>
        ))}
        <Form.Item>
          <Button
            type="dashed"
            onClick={() => addLesson()}
            icon={<PlusOutlined />}
          >
            Add Lesson
          </Button>
        </Form.Item>
      </>
    )}
  </Form.List>

  <div className="flex felx-row justify-end gap-3">
    <Form.Item>
      <Button onClick={handleCancel}>Cancel</Button>
    </Form.Item>
    <Form.Item>
      <Button
        type="primary"
        htmlType="submit"
        loading={confirmLoading}
      >
        Add Course
      </Button>
    </Form.Item>
  </div>
 </Form>
</div>
```

```
    </Modal>
   </>
  );
};

export default AddCourseModal;
```

AdminNavBar.jsx

```
import React, { useState } from "react";
import { AiOutlineClose, AiOutlineMenu } from "react-icons/ai";
import { Link, useNavigate } from "react-router-dom";
import { Fragment } from "react";
import { Menu, Transition } from "@headlessui/react";
import Swal from "sweetalert2";
import logo from "../assets/logo.png";
import axios from "../util/AxiosInstance";
import { Input, Select } from "antd";
const { Search } = Input;

const options = [
  {
    value: "zhejiang",
    label: "Zhejiang",
  },
  {
    value: "jiangsu",
    label: "Jiangsu",
  },
];

const AdminNavBar = () => {
```

```
const navigate = useNavigate();
const userToken = localStorage.getItem("jsonwebtoken");
const userDetails = JSON.parse(userToken);
const user = userDetails ? userDetails.decodedJWT : "";
const accessToken = userDetails ? userDetails.accessToken : "";
const [nav, setNav] = useState(false);
const [buttonLoading, setButtonLoading] = useState(false);

const handleLogout = async (e) => {
 e.preventDefault();
 try {
  const res = await axios.post("auth/logout", {
   headers: {
    Authorization: `Bearer ${accessToken}`,
   },
  });
  console.log(res);

  localStorage.removeItem("jsonwebtoken");
  localStorage.removeItem("role");
  Swal.fire({
   icon: "success",
   title: "Logged Out",
   text: "Successfully LoggedOut!!",
  });
  navigate("/login");
 } catch (error) {
  console.error("Logout failed:", error);
  // Show error message
  Swal.fire({
   icon: "error",
   title: "Logout Failed",
```

```
      text: error.response.data.message || "Something went wrong!",
    });
  }
};

const searchHandler = () => {
  setButtonLoading(true);
  setTimeout(() => {
    setButtonLoading(false);
  }, 2000);
};

const handleNav = () => {
  setNav(!nav);
};

function classNames(...classes) {
  return classes.filter(Boolean).join(" ");
}

return (
  <nav className="flex flex-row w-full px-16 justify-between py-4 sticky bg-white top-0 z-[999]">
    <div className="flex items-center">
      <Link to="/admin/home">
        <img
          src={logo}
          alt="Logo"
          className="h-10 lg:h-10 w-auto rounded-full"
        />
      </Link>
    </div>
```

```
    <div className="flex flex-row ">
      {/* <!-- right header section --> */}
      <div className="items-center space-x-3 hidden md:flex">
        {user ? (
          <>
            <div className="text-xl">Hello {user.userName}</div>
            <button className=""></button>
            <Menu as="div" className="relative inline-block text-left">
              <div>
                <Menu.Button className="flex rounded-full bg-gray-800 text-sm focus:outline-
none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">
                  {/* {user.name} */}
                  <img
                    className="h-12 w-12 rounded-full"
                    src={user.image}
                    alt=""
                  ></img>
                </Menu.Button>
              </div>

              <Transition
                as={Fragment}
                enter="transition ease-out duration-100"
                enterFrom="transform opacity-0 scale-95"
                enterTo="transform opacity-100 scale-100"
                leave="transition ease-in duration-75"
                leaveFrom="transform opacity-100 scale-100"
                leaveTo="transform opacity-0 scale-95"
              >
                <Menu.Items className="absolute right-0 z-10 mt-2 w-40 h-auto origin-top-right
rounded-md bg-black shadow-lg ring-1 ring-[#333333] ring-opacity-5 focus:outline-none">
                  <div className="py-1">
```

```
<h2 className="block px-4 py-2 text-sm text-[#41A4FF]">
 {user.userName}
</h2>
<Menu.Item>
 {({ active }) => (
  <Link
   className={classNames(
    active
     ? "bg-gray-100 text-[#41A4FF]"
     : "text-[#41A4FF]",
    "block px-4 py-2 text-sm"
   )}
   to="/user"
  >
   Profile
  </Link>
 )}
</Menu.Item>

<Menu.Item>
 {({ active }) => (
  <button
   type="button"
   onClick={handleLogout}
   className={classNames(
    active
     ? "bg-gray-100 text-[#41A4FF]"
     : "text-[#41A4FF]",
    "block w-full px-4 py-2 text-left text-sm"
   )}
  >
   Sign out
```

```
                </button>
              )}
            </Menu.Item>
          </div>
        </Menu.Items>
      </Transition>
    </Menu>
  </>
 ) : (
  <div className="items-center space-x-3 hidden md:flex">
   <>
    <Link
     to="/login"
     className="px-4 py-2 text-black font-bold bg-white border border-black text-center cursor-pointer rounded-3xl"
    >
     Sign in
    </Link>
    <Link
     to="/register"
     className="px-4 py-2 text-white font-bold bg-gray-800 text-center cursor-pointer rounded-3xl"
    >
     Sign up
    </Link>
   </>
  </div>
 )}
 </div>
</div>

{/* <div onClick={handleNav} className="block md:hidden">
```

```
      {nav ? (
        <AiOutlineClose size={20} style={{ color: "black" }} />
      ) : (
        <AiOutlineMenu size={20} style={{ color: "black" }} />
      )}
    </div> */}

    {/* <div
      className={
        !nav
          ? "fixed left-[-100%] top-0 w-[60%] h-full border-r border-r-gray bg-white ease-in-out duration-500 md:hidden"
          : "fixed left-0 top-0 w-[60%] h-full border-r border-r-gray bg-white ease-in-out duration-500 md:hidden"
      }
    >
      <div className="flex items-center">
        <Link to="/">
          <img src={logo} alt="Logo" className="w-48 h-auto" />
        </Link>
      </div>
      <ul className="p-4 mt-20">
        <li className="p-4 border-b border-gray-600">
          <Link
            to="/"
            onClick={() => {
              setNav(false);
            }}
            spy={true}
            smooth={true}
            duration={500}
          >
```

```
          Home
         </Link>
        </li>

        <li className="p-4 border-b border-gray-600">
         <Menu as="div" className="relative inline-block text-left">
          <div>
           <Menu.Button className="inline-flex w-full justify-center gap-x-1.5 rounded-md
px-3 py-2">
            My Travel Details
           </Menu.Button>
          </div>
         </Menu>
        </li>
        <li className="p-4 border-b border-gray-600">
         <Menu as="div" className="relative inline-block text-left">
          <div>
           <Menu.Button className="inline-flex w-full justify-center gap-x-1.5 rounded-md
px-3 py-2">
            Balance
           </Menu.Button>
          </div>
         </Menu>
        </li>
        <li className="p-4 border-b border-gray-600">
         {user ? (
          <>
           <button className=""></button>
           <Menu as="div" className="relative inline-block text-left">
            <div>
             <Menu.Button className="flex rounded-full bg-gray-800 text-sm focus:outline-
none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">
```

```
            <img
              class="h-8 w-8 rounded-full"
              src={profile}
              alt=""
            ></img>
          </Menu.Button>
        </div>

        <Transition
          as={Fragment}
          enter="transition ease-out duration-100"
          enterFrom="transform opacity-0 scale-95"
          enterTo="transform opacity-100 scale-100"
          leave="transition ease-in duration-75"
          leaveFrom="transform opacity-100 scale-100"
          leaveTo="transform opacity-0 scale-95"
        >
          <Menu.Items className="absolute z-10 mt-2 w-40 origin-top-right rounded-md
bg-white shadow-lg ring-1 ring-[#333333] ring-opacity-5 focus:outline-none">
            <div className="py-1">
              <h2 className="block px-4 py-2 text-sm text-[#9744BE]">
                {user.name}
              </h2>
              <Menu.Item>
                {({ active }) => (
                  <Link
                    onClick={() => {
                      setNav(false);
                    }}
                    spy={true}
                    smooth={true}
```

```
        duration={500}
        className={classNames(
          active
            ? "bg-gray-100 text-gray-900"
            : "text-gray-700",
          "block px-4 py-2 text-sm"
        )}
        to="/user"
      >
        Profile
      </Link>
    )}
  </Menu.Item>

  <Menu.Item>
    {({ active }) => (
      <button
        type="button"
        onClick={handleLogout}
        className={classNames(
          active
            ? "bg-gray-100 text-gray-900"
            : "text-gray-700",
          "block w-full px-4 py-2 text-left text-sm"
        )}
      >
        Sign out
      </button>
    )}
  </Menu.Item>
 </div>
</Menu.Items>
```

```
              </Transition>
            </Menu>
          </>
        ) : (
          <div className="items-center space-x-3 hidden md:flex">
            <>
              <Link
                to="/login"
                className="px-4 py-2 text-white font-bold bg-[#9744BE] text-center hover:bg-[#7d5391] cursor-pointer rounded-md"
              >
                Sign in
              </Link>
              <Link
                to="/register"
                className="px-4 py-2 text-white font-bold bg-gray-800 text-center hover:bg-gray-600 cursor-pointer rounded-md"
              >
                Sign up
              </Link>
            </>
          </div>
        )}
      </li>
    </ul>
  </div> */}
 </nav>
 );
};

export default AdminNavBar;
```

DetailsDrawer.jsx

```
import React from "react";
import { Drawer, Tag } from "antd";

const DetailsDrawer = ({ visible, close, selectedCourse }) => {
  return (
    <Drawer
      title={selectedCourse ? selectedCourse.title : "Course Details"}
      onClose={close}
      open={visible}
      width={700}
    >
      {selectedCourse && (
        <div className="flex flex-col gap-4 text-lg">
          <p className="text-xl font-bold">Course Details</p>
          <div>
            {" "}
            <img
              className="w-28 h-auto"
              src={
                selectedCourse.coverImage === "no image"
                  ? "https://icon-library.com/images/no-image-icon/no-image-icon-0.jpg"
                  : selectedCourse.coverImage
              }
              alt=""
            />
          </div>
          <p>Title: {selectedCourse.title}</p>
          <p>Instructor: {selectedCourse.instructor}</p>
          <p>Description {selectedCourse.description}</p>
          <p>Duration: {selectedCourse.duration}</p>
```

```
<p>Price: {selectedCourse.price}$</p>
<p>
 Tags:{" "}
 {selectedCourse.tags.map((tag) => {
  let color = tag.length > 5 ? "geekblue" : "green";
  if (tag === "loser") {
   color = "volcano";
  }
  return (
   <Tag color={color} key={tag}>
    {tag.toUpperCase()}
   </Tag>
  );
 })}
</p>
<p>createdAt: {selectedCourse.createdAt}</p>
<p className="text-xl font-bold">Lessons</p>
{selectedCourse.lessons.map((lesson, index) => (
 <div className="text-lg" key={index}>
  <p>Title: {lesson.title}</p>
  <p>Description: {lesson.description}</p>
  <p>
   Video URL : <a href={lesson.videoUrl}>{lesson.videoUrl}</a>{" "}
  </p>
  {lesson.quiz.map((quiz, index2) => (
   <div className="text-lg" key={index2}>
    <p>Question: {quiz.question}</p>
    <p>Answer: {quiz.correctAnswer}</p>
   </div>
  ))}
 </div>
))}
```

```
      </div>
    )}
   </Drawer>
  );
};

export default DetailsDrawer;
```

Footer.jsx

```
import React from "react";
import logo from "../assets/logo.png";

const Footer = () => {
  return (
   <div className="fixed bottom-0 left-0 right-0 mt-5  bg-gray-400  p-4 px-10 flex flex-rows justify-between gap-8 text-white">
    <div className="flex flex-row">
     <div className="text-black mr-5"> Copyright @2023 </div>
     <div>
      <img src={logo} alt="" className="h-6 w-auto" />
     </div>
    </div>
    <div></div>
   </div>
  );
};

export default Footer;
```

InstructorNavBar.jsx

```
import React, { useState } from "react";
import { AiOutlineClose, AiOutlineMenu } from "react-icons/ai";
import { Link, useNavigate } from "react-router-dom";
import { Fragment } from "react";
import { Menu, Transition } from "@headlessui/react";
import Swal from "sweetalert2";
import logo from "../assets/logo.png";
import axios from "../util/AxiosInstance";
import { Input, Select } from "antd";
const { Search } = Input;

const options = [
  {
    value: "zhejiang",
    label: "Zhejiang",
  },
  {
    value: "jiangsu",
    label: "Jiangsu",
  },
];

const InstructorNavBar = () => {
  const navigate = useNavigate();
  const userToken = localStorage.getItem("jsonwebtoken");
  const userDetails = JSON.parse(userToken);
  const user = userDetails ? userDetails.decodedJWT : "";
  const accessToken = userDetails ? userDetails.accessToken : "";
  const [nav, setNav] = useState(false);
  const [buttonLoading, setButtonLoading] = useState(false);
  const handleLogout = async (e) => {
    e.preventDefault();
```

```
  try {
    const res = await axios.post("auth/logout", {
      headers: {
        Authorization: `Bearer ${accessToken}`,
      },
    });
    console.log(res);

    localStorage.removeItem("jsonwebtoken");
    localStorage.removeItem("role");
    Swal.fire({
      icon: "success",
      title: "Logged Out",
      text: "Successfully LoggedOut!!",
    });
    navigate("/login");
  } catch (error) {
    console.error("Logout failed:", error);
    // Show error message
    Swal.fire({
      icon: "error",
      title: "Logout Failed",
      text: error.response.data.message || "Something went wrong!",
    });
  }
};

const searchHandler = () => {
  setButtonLoading(true);
  setTimeout(() => {
    setButtonLoading(false);
  }, 2000);
```

```
};

const handleNav = () => {
  setNav(!nav);
};

function classNames(...classes) {
  return classes.filter(Boolean).join(" ");
}

return (
  <nav className="flex flex-row w-full px-16 justify-around py-4 sticky bg-white top-0 z-[999]">
    <div className="flex items-center">
      <Link to="/instructor/home">
        <img
          src={logo}
          alt="Logo"
          className="h-10 lg:h-10 w-auto rounded-full"
        />
      </Link>
    </div>

    {/* <!-- left header section --> */}
    <div className="items-center text-xl hidden space-x-5 md:flex gap-8">
      <Link to={"/instructor/courses"}>Courses</Link>
      <Link to={"/instructor/enrollment"}>Enrollment Details</Link>
    </div>
    {/* <!-- right header section --> */}
    <div className="items-center space-x-3 hidden md:flex">
      {user ? (
        <>
```

```
<div className="text-xl">Hello {user.userName}</div>
<button className=""></button>
<Menu as="div" className="relative inline-block text-left">
 <div>
  <Menu.Button className="flex rounded-full bg-gray-800 text-sm focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">
   {/* {user.name} */}
   <img
    className="h-12 w-12 rounded-full"
    src={user.image}
    alt=""
   ></img>
  </Menu.Button>
 </div>

 <Transition
  as={Fragment}
  enter="transition ease-out duration-100"
  enterFrom="transform opacity-0 scale-95"
  enterTo="transform opacity-100 scale-100"
  leave="transition ease-in duration-75"
  leaveFrom="transform opacity-100 scale-100"
  leaveTo="transform opacity-0 scale-95"
 >
  <Menu.Items className="absolute right-0 z-10 mt-2 w-40 h-auto origin-top-right rounded-md bg-black shadow-lg ring-1 ring-[#333333] ring-opacity-5 focus:outline-none">
   <div className="py-1">
    <h2 className="block px-4 py-2 text-sm text-[#41A4FF]">
     {user.userName}
    </h2>
    <Menu.Item>
     {({ active }) => (
```

```
        <Link
          className={classNames(
            active
              ? "bg-gray-100 text-[#41A4FF]"
              : "text-[#41A4FF]",
            "block px-4 py-2 text-sm"
          )}
          to="/user"
        >
          Profile
        </Link>
      )}
    </Menu.Item>

    <Menu.Item>
      {({ active }) => (
        <button
          type="button"
          onClick={handleLogout}
          className={classNames(
            active
              ? "bg-gray-100 text-[#41A4FF]"
              : "text-[#41A4FF]",
            "block w-full px-4 py-2 text-left text-sm"
          )}
        >
          Sign out
        </button>
      )}
    </Menu.Item>
  </div>
</Menu.Items>
```

```
          </Transition>
        </Menu>
      </>
    ) : (
      <div className="items-center space-x-3 hidden md:flex">
        <>
          <Link
            to="/login"
            className="px-4 py-2 text-black font-bold bg-white border border-black text-center cursor-pointer rounded-3xl"
            >
            Sign in
          </Link>
          <Link
            to="/register"
            className="px-4 py-2 text-white font-bold bg-gray-800 text-center   cursor-pointer rounded-3xl"
            >
            Sign up
          </Link>
        </>
      </div>
    )}
  </div>

  {/* <div onClick={handleNav} className="block md:hidden">
    {nav ? (
      <AiOutlineClose size={20} style={{ color: "black" }} />
    ) : (
      <AiOutlineMenu size={20} style={{ color: "black" }} />
    )}
  </div> */}
```

```
    {/* <div
      className={
        !nav
          ? "fixed left-[-100%] top-0 w-[60%] h-full border-r border-r-gray bg-white ease-in-out duration-500 md:hidden"
          : "fixed left-0 top-0 w-[60%] h-full border-r border-r-gray bg-white ease-in-out duration-500 md:hidden"
      }
    >
      <div className="flex items-center">
        <Link to="/">
          <img src={logo} alt="Logo" className="w-48 h-auto" />
        </Link>
      </div>
      <ul className="p-4 mt-20">
        <li className="p-4 border-b border-gray-600">
          <Link
            to="/"
            onClick={() => {
              setNav(false);
            }}
            spy={true}
            smooth={true}
            duration={500}
          >
            Home
          </Link>
        </li>

        <li className="p-4 border-b border-gray-600">
          <Menu as="div" className="relative inline-block text-left">
```

```
          <div>
            <Menu.Button className="inline-flex w-full justify-center gap-x-1.5 rounded-md
px-3 py-2">
              My Travel Details
            </Menu.Button>
          </div>
        </Menu>
      </li>
      <li className="p-4 border-b border-gray-600">
        <Menu as="div" className="relative inline-block text-left">
          <div>
            <Menu.Button className="inline-flex w-full justify-center gap-x-1.5 rounded-md
px-3 py-2">
              Balance
            </Menu.Button>
          </div>
        </Menu>
      </li>
      <li className="p-4 border-b border-gray-600">
        {user ? (
          <>
            <button className=""></button>
            <Menu as="div" className="relative inline-block text-left">
              <div>
                <Menu.Button className="flex rounded-full bg-gray-800 text-sm focus:outline-
none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">

                  <img
                    class="h-8 w-8 rounded-full"
                    src={profile}
                    alt=""
                  ></img>
```

```
          </Menu.Button>
        </div>

        <Transition
          as={Fragment}
          enter="transition ease-out duration-100"
          enterFrom="transform opacity-0 scale-95"
          enterTo="transform opacity-100 scale-100"
          leave="transition ease-in duration-75"
          leaveFrom="transform opacity-100 scale-100"
          leaveTo="transform opacity-0 scale-95"
        >
          <Menu.Items className="absolute z-10 mt-2 w-40 origin-top-right rounded-md bg-white shadow-lg ring-1 ring-[#333333] ring-opacity-5 focus:outline-none">
            <div className="py-1">
              <h2 className="block px-4 py-2 text-sm text-[#9744BE]">
                {user.name}
              </h2>
              <Menu.Item>
                {({ active }) => (
                  <Link
                    onClick={() => {
                      setNav(false);
                    }}
                    spy={true}
                    smooth={true}
                    duration={500}
                    className={classNames(
                      active
                        ? "bg-gray-100 text-gray-900"
                        : "text-gray-700",
                      "block px-4 py-2 text-sm"
```

```
            )}
            to="/user"
          >
            Profile
          </Link>
        )}
      </Menu.Item>

      <Menu.Item>
        {({ active }) => (
          <button
            type="button"
            onClick={handleLogout}
            className={classNames(
              active
                ? "bg-gray-100 text-gray-900"
                : "text-gray-700",
              "block w-full px-4 py-2 text-left text-sm"
            )}
          >
            Sign out
          </button>
        )}
      </Menu.Item>
    </div>
  </Menu.Items>
 </Transition>
</Menu>
</>
) : (
<div className="items-center space-x-3 hidden md:flex">
 <>
```

```
            <Link
              to="/login"
              className="px-4 py-2 text-white font-bold bg-[#9744BE] text-center hover:bg-[#7d5391] cursor-pointer rounded-md"
            >
              Sign in
            </Link>
            <Link
              to="/register"
              className="px-4 py-2 text-white font-bold bg-gray-800 text-center hover:bg-gray-600 cursor-pointer rounded-md"
            >
              Sign up
            </Link>
          </>
        </div>
      )}
    </li>
  </ul>
 </div> */}
 </nav>
 );
};

export default InstructorNavBar;
```

Loader.jsx

```
import React from "react";
import { LoadingOutlined } from "@ant-design/icons";
import { Spin } from "antd";
```

```
const Loader = () => {
  const overlayStyles = {
    position: "fixed",
    top: 0,
    left: 0,
    width: "100%",
    height: "100%",
    backgroundColor: "rgba(0, 0, 0, 0.7)",
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    zIndex: 1000,
  };

  return (
    <div className="text-center mb-[500px]" style={overlayStyles}>
      <Spin
        indicator={
          <LoadingOutlined
            style={{
              fontSize: 36,
            }}
          />
        }
      />
    </div>
  );
};

export default Loader;
```

MenuNavBar.jsx

```
import React from "react";
import { Link } from "react-router-dom";
import { Menu } from "@headlessui/react";

const nav_links = [
  {
    path: "/",
    display: "Home",
  },
  {
    path: "/courses",
    display: "Courses",
  },
  {
    path: "/modules",
    display: "Modules",
  },
  {
    path: "/blog",
    display: "Blog",
  },
  {
    path: "/about-us",
    display: "About Us",
  },
  {
    path: "/contact-us",
    display: "Contact",
  },
];
```

```
const MenuNav = () => {
  return (
    <div>
      <nav className="w-screen flex justify-around  h-12 py-4 sticky top-0">
        <div className="items-center hidden space-x-5 md:flex">
          {/* <!-- Hamburger button for mobile view --> */}
          <button
            className="block border-0 bg-transparent px-2 text-neutral-500 hover:no-underline hover:shadow-none focus:no-underline focus:shadow-none focus:outline-none focus:ring-0 dark:text-neutral-200 lg:hidden"
            type="button"
            data-te-collapse-init
            data-te-target="#navbarSupportedContent1"
            aria-controls="navbarSupportedContent1"
            aria-expanded="false"
            aria-label="Toggle navigation"
          >
            {/* <!-- Hamburger icon --> */}
            <span className="[&>svg]:w-7">
              <svg
                xmlns="http://www.w3.org/2000/svg"
                viewBox="0 0 24 24"
                fill="currentColor"
                className="h-7 w-7"
              >
                <path
                  fillRule="evenodd"
                  d="M3 6.75A.75.75 0 013.75 6h16.5a.75.75 0 010 1.5H3.75A.75.75 0 013 6.75zM3 12a.75.75 0 01.75-.75h16.5a.75.75 0 010 1.5H3.75A.75.75 0 013 12zm0 5.25a.75.75 0 01.75-.75h16.5a.75.75 0 010 1.5H3.75a.75.75 0 01-.75-.75z"
                  clipRule="evenodd"
                />
```

```
          </svg>
         </span>
        </button>
        <div>
         <Menu as="div" className="relative inline-block text-left">
          <div>
           <Menu.Button className="inline-flex space-x-8 text-xl w-full justify-center rounded-md px-3 py-2">
            {nav_links.map((item, index) => (
             <Menu>
              {
               <Link to={item.path}>
                <div className="hover:cursor-pointer  px-3  rounded-lg">
                 {item.display}
                </div>
               </Link>
              }
             </Menu>
            ))}
           </Menu.Button>
          </div>
         </Menu>
        </div>
       </div>
     </nav>
   </div>
  );
};

export default MenuNav;
```

Navbar.jsx

```
import React, { useState } from "react";
import { AiOutlineClose, AiOutlineMenu } from "react-icons/ai";
import { Link, useNavigate } from "react-router-dom";
import { Fragment } from "react";
import { Menu, Transition } from "@headlessui/react";
import Swal from "sweetalert2";
import logo from "../assets/logo.png";
import { Input, Select } from "antd";
import axios from "../util/AxiosInstance";
const { Search } = Input;

const options = [
  {
    value: "zhejiang",
    label: "Zhejiang",
  },
  {
    value: "jiangsu",
    label: "Jiangsu",
  },
];

const Navbar = () => {
  const navigate = useNavigate();
  const userToken = localStorage.getItem("jsonwebtoken");
  const userDetails = JSON.parse(userToken);
  const user = userDetails ? userDetails.decodedJWT : "";
  const accessToken = userDetails ? userDetails.accessToken : "";
  const [nav, setNav] = useState(false);
  const [buttonLoading, setButtonLoading] = useState(false);
  const handleLogout = async (e) => {
```

```
  e.preventDefault();
  try {
    const res = await axios.post("auth/logout", {
      headers: {
        Authorization: `Bearer ${accessToken}`,
      },
    });
    console.log(res);

    localStorage.removeItem("jsonwebtoken");
    localStorage.removeItem("role");
    Swal.fire({
      icon: "success",
      title: "Logged Out",
      text: "Successfully LoggedOut!!",
    });
    navigate("/login");
  } catch (error) {
    console.error("Logout failed:", error);
    // Show error message
    Swal.fire({
      icon: "error",
      title: "Logout Failed",
      text: error.response.data.message || "Something went wrong!",
    });
  }
};

const searchHandler = () => {
  setButtonLoading(true);
  setTimeout(() => {
    setButtonLoading(false);
```

```
  }, 2000);
 };

 const handleNav = () => {
  setNav(!nav);
 };

 function classNames(...classes) {
  return classes.filter(Boolean).join(" ");
 }

 return (
  <nav className="flex flex-row w-full px-16 justify-between py-4 sticky bg-white top-0 z-
[999]">
    <div className="flex items-center">
     <Link to="/">
      <img
       src={logo}
       alt="Logo"
       className="h-10 lg:h-10 w-auto rounded-full"
      />
     </Link>
    </div>
    <div className="flex flex-row ">
     {" "}
     {/* <!-- left header section --> */}
     <div className="items-center text-xl hidden space-x-5 md:flex w-96 gap-8">
      {/* <Select defaultValue="Zhejiang" options={options} /> */}{" "}
      {/* Adjust width here */}
      <Search
       placeholder="Find Your Course"
       enterButton="Search"
```

```
        size="large"
        allowClear
        style={{ width: 1000 }}
        className="mx-8"
        loading={buttonLoading}
        onSearch={searchHandler}
       />
      </div>
      {/* <!-- right header section --> */}
      <div className="items-center space-x-3 hidden md:flex">
       {user ? (
        <>
         <div className="text-xl">Hello {user.userName} !</div>
         <button className=""></button>
         <Menu as="div" className="relative inline-block text-left">
          <div>
           <Menu.Button className="flex rounded-full bg-gray-800 text-sm focus:outline-
none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">
            {/* {user.name} */}
            <img
             className="h-12 w-12 rounded-full"
             src={user.image}
             alt=""
            ></img>
           </Menu.Button>
          </div>

          <Transition
           as={Fragment}
           enter="transition ease-out duration-100"
           enterFrom="transform opacity-0 scale-95"
           enterTo="transform opacity-100 scale-100"
```

```
          leave="transition ease-in duration-75"
          leaveFrom="transform opacity-100 scale-100"
          leaveTo="transform opacity-0 scale-95"
        >
          <Menu.Items className="absolute right-0 z-10 mt-2 w-40 h-auto origin-top-right
rounded-md bg-black shadow-lg ring-1 ring-[#333333] ring-opacity-5 focus:outline-none">
            <div className="py-1">
              <h2 className="block px-4 py-2 text-sm text-[#41A4FF]">
                {user.userName}
              </h2>
              <Menu.Item>
                {({ active }) => (
                  <Link
                    className={classNames(
                      active
                        ? "bg-gray-100 text-[#41A4FF]"
                        : "text-[#41A4FF]",
                      "block px-4 py-2 text-sm"
                    )}
                    to="/user"
                  >
                    Profile
                  </Link>
                )}
              </Menu.Item>

              <Menu.Item>
                {({ active }) => (
                  <button
                    type="button"
                    onClick={handleLogout}
                    className={classNames(
```

```
                    active
                      ? "bg-gray-100 text-[#41A4FF]"
                      : "text-[#41A4FF]",
                    "block w-full px-4 py-2 text-left text-sm"
                  )}
                >
                  Sign out
                </button>
              )}
            </Menu.Item>
          </div>
        </Menu.Items>
      </Transition>
    </Menu>
  </>
) : (
  <div className="items-center space-x-3 hidden md:flex">
    <>
      <Link
        to="/login"
        className="px-4 py-2 text-black font-bold bg-white border border-black text-center cursor-pointer rounded-3xl"
      >
        Sign in
      </Link>
      <Link
        to="/register"
        className="px-4 py-2 text-white font-bold bg-gray-800 text-center cursor-pointer rounded-3xl"
      >
        Sign up
      </Link>
```

```
          </>
        </div>
      )}
    </div>
  </div>

  {/* <div onClick={handleNav} className="block md:hidden">
    {nav ? (
      <AiOutlineClose size={20} style={{ color: "black" }} />
    ) : (
      <AiOutlineMenu size={20} style={{ color: "black" }} />
    )}
  </div> */}

  {/* <div
    className={
      !nav
        ? "fixed left-[-100%] top-0 w-[60%] h-full border-r border-r-gray bg-white ease-in-out duration-500 md:hidden"
        : "fixed left-0 top-0 w-[60%] h-full border-r border-r-gray bg-white ease-in-out duration-500 md:hidden"
    }
  >
    <div className="flex items-center">
      <Link to="/">
        <img src={logo} alt="Logo" className="w-48 h-auto" />
      </Link>
    </div>
    <ul className="p-4 mt-20">
      <li className="p-4 border-b border-gray-600">
        <Link
          to="/"
```

```
            onClick={() => {
              setNav(false);
            }}
            spy={true}
            smooth={true}
            duration={500}
          >
            Home
          </Link>
        </li>

        <li className="p-4 border-b border-gray-600">
          <Menu as="div" className="relative inline-block text-left">
            <div>
              <Menu.Button className="inline-flex w-full justify-center gap-x-1.5 rounded-md
px-3 py-2">
                My Travel Details
              </Menu.Button>
            </div>
          </Menu>
        </li>
        <li className="p-4 border-b border-gray-600">
          <Menu as="div" className="relative inline-block text-left">
            <div>
              <Menu.Button className="inline-flex w-full justify-center gap-x-1.5 rounded-md
px-3 py-2">
                Balance
              </Menu.Button>
            </div>
          </Menu>
        </li>
        <li className="p-4 border-b border-gray-600">
```

```
{user ? (
 <>
  <button className=""></button>
  <Menu as="div" className="relative inline-block text-left">
   <div>
    <Menu.Button className="flex rounded-full bg-gray-800 text-sm focus:outline-none focus:ring-2 focus:ring-white focus:ring-offset-2 focus:ring-offset-gray-800">

     <img
      class="h-8 w-8 rounded-full"
      src={profile}
      alt=""
     ></img>
    </Menu.Button>
   </div>

   <Transition
    as={Fragment}
    enter="transition ease-out duration-100"
    enterFrom="transform opacity-0 scale-95"
    enterTo="transform opacity-100 scale-100"
    leave="transition ease-in duration-75"
    leaveFrom="transform opacity-100 scale-100"
    leaveTo="transform opacity-0 scale-95"
   >
    <Menu.Items className="absolute z-10 mt-2 w-40 origin-top-right rounded-md bg-white shadow-lg ring-1 ring-[#333333] ring-opacity-5 focus:outline-none">
     <div className="py-1">
      <h2 className="block px-4 py-2 text-sm text-[#9744BE]">
       {user.name}
      </h2>
      <Menu.Item>
```

```
 {({ active }) => (
  <Link
   onClick={() => {
    setNav(false);
   }}
   spy={true}
   smooth={true}
   duration={500}
   className={classNames(
    active
     ? "bg-gray-100 text-gray-900"
     : "text-gray-700",
    "block px-4 py-2 text-sm"
   )}
   to="/user"
  >
   Profile
  </Link>
 )}
</Menu.Item>

<Menu.Item>
 {({ active }) => (
  <button
   type="button"
   onClick={handleLogout}
   className={classNames(
    active
     ? "bg-gray-100 text-gray-900"
     : "text-gray-700",
    "block w-full px-4 py-2 text-left text-sm"
   )}
```

```
                >
                  Sign out
                </button>
              )}
            </Menu.Item>
          </div>
        </Menu.Items>
      </Transition>
    </Menu>
  </>
) : (
  <div className="items-center space-x-3 hidden md:flex">
    <>
      <Link
        to="/login"
        className="px-4 py-2 text-white font-bold bg-[#9744BE] text-center hover:bg-[#7d5391] cursor-pointer rounded-md"
      >
        Sign in
      </Link>
      <Link
        to="/register"
        className="px-4 py-2 text-white font-bold bg-gray-800 text-center hover:bg-gray-600 cursor-pointer rounded-md"
      >
        Sign up
      </Link>
    </>
  </div>
)}
</li>
</ul>
```

```
    </div> */}
  </nav>
 );
};

export default Navbar;
```

ViewCompletedCourses.jsx

```
import React, { useEffect, useState } from 'react';
import axios from '../util/AxiosInstance';

export default function ViewCompletedCourses() {
  const [completedList, setCompletedList] = useState([]);
  const [errorMsg, setErrorMsg] = useState(null);
  const [coursesList, setCourseList] = useState([]);

  useEffect(() => {
    getEnrolledCourses();
  }, []);

  const getEnrolledCourses = async () => {
    try {
      const user = localStorage.getItem('jsonwebtoken');
      const userData = JSON.parse(user);
      const userID = userData.decodedJWT.userId;

      const res = await axios.get(`learner/courses/completed/${userID}`);

      if (res.status === 200) {
        setCompletedList(res.data);
        await getEachCourse(res.data)
```

```
        } else {
            setErrorMsg(res.data);
        }
    } catch (error) {
        console.error('Error fetching completed courses:', error);
    }
};

const getEachCourse = async (completedList) => {
    try {
        const coursesArray = [];

        for (const element of completedList) {
            const courseID = element.courseId;
            const res = await axios.get(`course/guest-routes/${courseID}`);
            if (res.status === 201) {
                coursesArray.push(res.data.course);
            }
        }
        setCourseList(coursesArray);
    } catch (error) {
        console.error('Error fetching course details:', error);
    }
};

return (
    <div className='flex flex-col  w-full h-[45%] bg-white p-10 rounded-xl shadow'>
        <div className=' border-b-2 border-[#575757]'>
            <span className=' text-3xl font-semibold '>Completed Courses</span>
        </div>
        <div className=' flex flex-wrap mt-5 overflow-auto'>
```

```
            {coursesList.map((course, index) => (

                <div className='flex items-center group cursor-pointer'>
                  <div  className=' w-4 h-4 rounded-full m-2 border-2 border-[#575757] transition-colors duration-300 ease-in-out group-hover:bg-blue-500'></div>
                  <div key={index} className='flex flex-col px-1 w-72 h-fit py-2 bg-white rounded-lg shadow border-2 border-[#575757] '>

                    <div  className=' flex items-center'>
                      <img  src={course.coverImage}  className='w-5  h-5  rounded  m-2' alt='Course Cover' />
                      <span>{course.title}</span>
                    </div>



                  </div>
                </div>
              ))}
          </div>
        </div>
    );
}
```

ViewEnrolledCourses.jsx

```
import React, { useEffect, useState } from 'react';
import axios from '../util/AxiosInstance';

export default function ViewEnrolledCourses() {
    const [enrolledList, setEnrolledList] = useState([]);
    const [errorMsg, setErrorMsg] = useState(null);
    const [coursesList, setCourseList] = useState([]);
```

```
useEffect(() => {
   getEnrolledCourses();
}, []);

const getEnrolledCourses = async () => {
   try {
      const user = localStorage.getItem('jsonwebtoken');
      const userData = JSON.parse(user);
      const userID = userData.decodedJWT.userId;

      const res = await axios.get(`learner/courses/enrolled/${userID}`);

      if (res.status === 200) {
         setEnrolledList(res.data);
         await getEachCourse(res.data);
      } else {
         setErrorMsg(res.data);
      }
   } catch (error) {
      console.error('Error fetching enrolled courses:', error);
   }
};

const getEachCourse = async (enrolledList) => {
   try {
      const coursesArray = [];

      for (const element of enrolledList) {
         const courseID = element.courseId;
         const res = await axios.get(`course/guest-routes/${courseID}`);
         if (res.status === 201) {
```

```
            coursesArray.push(res.data.course);
          }
        }
        setCourseList(coursesArray);
      } catch (error) {
        console.error('Error fetching course details:', error);
      }
    };

    return (
      <div className='flex flex-col  w-full h-[45%] bg-white p-10 rounded-xl shadow'>
        <div className=' border-b-2 border-[#575757]'>
          <span className=' text-3xl font-semibold '>Enrolled Courses</span>
        </div>
        <div className=' flex flex-wrap mt-5 overflow-auto'>
          {coursesList.map((course, index) => (

            <div className='flex items-center group cursor-pointer'>
              <div  className=' w-4 h-4 rounded-full m-2 border-2 border-[#575757] transition-colors duration-300 ease-in-out group-hover:bg-blue-500'></div>
              <div key={index} className='flex flex-col px-1 w-72 h-fit py-2 bg-white rounded-lg shadow border-2 border-[#575757] '>

                <div className=' flex items-center'>
                  <img src={course.coverImage} className='w-5 h-5 rounded m-2' alt='Course Cover' />
                  <span>{course.title}</span>
                </div>
                <hr className='mx-2'/>
                <div className=' flex justify-between px-2'>
                  <span className=' text-sm '>Completion : </span>
                  <span>{enrolledList[index].completionLevel}%</span>
```

```
                    </div>
                  </div>
                </div>
              ))}
          </div>
      </div>
    );
}
```

Layout.jsx

```
import React, { useEffect, useState } from "react";
import Navbar from "../components/Navbar";
import Router from "../routes/Router";
import Footer from "../components/Footer";
import AdminNavBar from "../components/AdminNavBar";
import InstructorNavBar from "../components/InstructorNavBar";

const Layouts = () => {
  const token = localStorage.getItem("jsonwebtoken");
  const payload = JSON.parse(token);
  const role = payload?.decodedJWT.userRole;

  return (
    <div>
      {role === "admin" ? (
        <>
          <AdminNavBar />
          <Router />
          <Footer />
        </>
      ) : role === "instructor" ? (
```

```
      <>
        <InstructorNavBar />
        <Router />
        <Footer />
      </>
    ) : (
      <>
        <Navbar />
        <Router />
        <Footer />
      </>
    )}
  </div>
 );
};

export default Layouts;
```

Home.jsx

```
import React, { useState } from "react";
import PendingCoursesPanel from "../../components/admin/PendingCourses";
import UserDetailsPanel from "../../components/admin/UserDetailsPanel";
import AddUserPanel from "../../components/admin/AddUserPanel";
import ViewUserListModel from "../../components/admin/ViewUserListModel";

const Home = () => {

 const[openUserList, setUserListOpen] = useState(false)
 const[userRole, setUserType] = useState('')

 return (
```

```
   <div className=' bg-slate-100 w-screen h-[calc(100vh-154px)] flex flex-row justify-evenly'>
    <div className=" w-1/4 h-full bg-zinc-100">
     <UserDetailsPanel/>
    </div>
    <div className=" w-[36vw] h-full   py-5 flex flex-col justify-evenly">
     <div className=" w-full h-[60%]  ">
      <AddUserPanel/>
     </div>
     <div className=" w-full h-[40%] flex flex-col items-center justify-evenly font-FuturaMdBt font-semibold text-xl text-[#575757]">
      <div onClick={() =>{ setUserType('instructor'); setUserListOpen(true);}} className=" flex justify-center bg-white py-5 rounded-xl w-[80%] my-5 border-2 border-[#575757] hover:border-green-500 hover:bg-gray-200 ">
       <span >View Instructors</span>
      </div>
      <div onClick={() => { setUserType('admin'); setUserListOpen(true); }} className=" flex justify-center bg-white py-5 rounded-xl w-[80%] my-5 border-2 border-[#575757] hover:border-green-500 hover:bg-gray-200 ">
       <span >View Admins</span>
      </div>
     </div>
    </div>
    <div className=" flex justify-end"><PendingCoursesPanel/></div>
    <ViewUserListModel open={openUserList} onClose={() => setUserListOpen(false)} userType={userRole} />
   </div>
  )

};

export default Home;
```

Courses.jsx

```
import React, { useEffect, useState } from "react";
import { Space, Table, Tag, Button } from "antd";
import axios from "../../util/AxiosInstance";
import DetailsDrawer from "../../components/DetailsDrawer";
import Loader from "../../components/Loader";
import Swal from "sweetalert2";
import AddCourseModal from "../../components/instructor/AddCourseModal";

const Courses = () => {
  const [data, setData] = useState([]);
  const [drawerVisible, setDrawerVisible] = useState(false);
  const [selectedCourse, setSelectedCourse] = useState(null);
  const [loading, isLoading] = useState(false);
  const [modalOpen, setModalOpen] = useState(false);

  useEffect(() => {
    const getCourses = async () => {
      isLoading(true);
      try {
        const res = await axios.get("course/course/user");
        setData(res.data.data);
        isLoading(false);
      } catch (error) {
        isLoading(false);
        console.error("Error fetching courses:", error);
      }
    };
    getCourses();
  }, [modalOpen]);
```

```
const handleDrawer = (record) => {
  setSelectedCourse(record);
  setDrawerVisible(true);
};

const handleCloseDrawer = () => {
  setDrawerVisible(false);
};

const handleModalOpen = () => {
  setModalOpen(true);
};

const handleCloseModal = () => {
  setModalOpen(false);
};

const columns = [
  {
    title: "",
    dataIndex: "",
    key: "no",
    render: (_, __, index) => <p>{index + 1}</p>,
  },
  {
    title: "",
    dataIndex: "coverImage",
    key: "coverImage",
    render: (img, record) => (
      <p>
        <label>
          <img
```

```
          className=""
          src={
            img === "No Image"
              ? "https://icon-library.com/images/no-image-icon/no-image-icon-0.jpg"
              : img
          }
          alt="avatar"
          style={{
            width: "120px",
            height: "120px",
            cursor: "pointer",
          }}
          onClick={() => handleDrawer(record)}
        />
      </label>
    </p>
  ),
},
{
  title: "Title",
  dataIndex: "title",
  key: "title",
  render: (text, record) => (
    <p
      className="hover:cursor-pointer"
      onClick={() => handleDrawer(record)}
    >
      {text}
    </p>
  ),
},
{
```

```
  title: "Duration",
  dataIndex: "duration",
  key: "duration",
},
{
  title: "Price",
  dataIndex: "price",
  key: "price",
  render: (text) => <p>{text}$</p>,
},
{
  title: "Tags",
  key: "tags",
  dataIndex: "tags",
  render: (_, { tags }) => (
    <>
      {tags.map((tag) => {
        let color = tag.length > 5 ? "geekblue" : "green";
        if (tag === "loser") {
          color = "volcano";
        }
        return (
          <Tag color={color} key={tag}>
            {tag.toUpperCase()}
          </Tag>
        );
      })}
    </>
  ),
},
{
  title: "Action",
```

```
    key: "action",
    render: (_, record) => (
      <Space size="middle">
        <Button block>Update</Button>
        <Button danger onClick={() => handleDelete(record)}>
          Delete
        </Button>
      </Space>
    ),
  },
];

const handleDelete = async (record) => {
  const confirmation = await Swal.fire({
    title: "Are you sure?",
    text: `You are about to delete ${record.title}. This action cannot be undone.`,
    icon: "warning",
    showCancelButton: true,
    confirmButtonColor: "#d33",
    cancelButtonColor: "#3085d6",
    confirmButtonText: "Yes, delete it!",
    cancelButtonText: "Cancel",
  });

  if (confirmation.isConfirmed) {
    isLoading(true);
    try {
      await axios.delete(`course/${record._id}`);
      setData(data.filter((course) => course._id !== record._id));
      isLoading(false);
      Swal.fire(
        "Course Deleted",
```

```
      `${record.title} Course has been deleted`,
      "success"
     );
    } catch (error) {
     isLoading(false);
     console.error("Error deleting course:", error);
     Swal.fire("Error", "Error deleting course", "error");
    }
  }
};
return (
  <>
   {loading ? (
    <Loader />
   ) : (
    <div className="lg:px-48 pb-5 px-12">
     <div className="text-center lg:text-3xl text-md md:text-xl  font-bold pt-5">
      Your Course Details
     </div>
     <div>
      <div className="flex justify-center lg:justify-end">
       <Button type="primary" onClick={handleModalOpen}>
        Add Courses
       </Button>
      </div>
     </div>
     <div className="pt-10 pb-10">
      <Table columns={columns} dataSource={data} />
     </div>
     <DetailsDrawer
      visible={drawerVisible}
      close={handleCloseDrawer}
```

```
          selectedCourse={selectedCourse}
        />
        {modalOpen && (
          <AddCourseModal open={modalOpen} close={handleCloseModal} />
        )}
      </div>
    )}
  </>
 );
};

export default Courses;
```

Enrollments.jsx

```
import React, { useEffect, useState } from "react";
import { Table, Progress, Tag } from "antd";
import axios from "../../util/AxiosInstance";
import Loader from "../../components/Loader";

const Enrollments = () => {
 const [data, setData] = useState([]);
 const [drawerVisible, setDrawerVisible] = useState(false);
 const [selectedCourse, setSelectedCourse] = useState(null);
 const [loading, isLoading] = useState(false);
 const [modalOpen, setModalOpen] = useState(false);

 useEffect(() => {
  const getCourses = async () => {
   isLoading(true);
   try {
    const res = await axios.get("course/enroll");
```

```
      setData(res.data.data);
      isLoading(false);
    } catch (error) {
      isLoading(false);
      console.error("Error fetching courses:", error);
    }
  };
  getCourses();
}, [modalOpen]);

const columns = [
  {
    title: "",
    dataIndex: "",
    key: "no",
    render: (_, __, index) => <p>{index + 1}</p>,
  },
  {
    title: "Course",
    dataIndex: "courseId",
    key: "courseId",
  },
  {
    title: "User Name",
    dataIndex: "learnerId",
    key: "learnerId",
  },
  {
    title: "Completion Level",
    dataIndex: "completionLevel",
    key: "completionLevel",
    render: (text) => <Progress percent={text} />,
```

```
  },
  {
    title: "Payment Status",
    dataIndex: "paymentStatus",
    key: "paymentStatus",
    render: (text) =>
      text ? (
        <Tag color="green" className="text-xl">
          Paid
        </Tag>
      ) : (
        <Tag color="red" className="text-xl">
          Not Paid
        </Tag>
      ),
  },

  // {
  //   title: "Action",
  //   key: "action",
  //   render: (_, record) => (
  //     <Space size="middle">
  //       <Button block>Update</Button>
  //       <Button danger onClick={() => handleDelete(record)}>
  //         Delete
  //       </Button>
  //     </Space>
  //   ),
  // },
];

return (
```

```
    <>
      {loading ? (
        <Loader />
      ) : (
        <div className="lg:px-48 pb-5 px-12">
          <div className="text-center lg:text-3xl text-md md:text-xl  font-bold pt-5">
            Enrollment Details
          </div>
          <div className="pt-10 pb-10">
            <Table columns={columns} dataSource={data} />
          </div>
        </div>
      )}
    </>
  );
};

export default Enrollments;
```

Home.jsx

```
import React from "react";
import InstructorNavBar from "../../components/InstructorNavBar";

const Home = () => {
  return <div>instructor home</div>;
};

export default Home;
```

AboutUs.jsx

```
import React from "react";
import MenuNav from "../components/MenuNavBar";

const AboutUs = () => {
  return (
    <div>
      <MenuNav />
      <div className="bg-blue-500 h-fit py-24 flex flex-col justify-center  items-center">
        <h1 className="text-white text-4xl">About Us</h1>
      </div>
      <div className="bg-white  py-8 px-4">
        <div className="max-w-4xl mx-auto">
          <p className="text-black  text-lg mb-4">
            Welcome to our website! We are a team of passionate individuals
            dedicated to providing high-quality products and services.
          </p>
          <p className="text-black text-lg mb-4">
            Our mission is to make a positive impact on the world through
            innovative solutions and exceptional customer experiences.
          </p>
          <p className="text-black text-lg">
            Feel free to explore our website and learn more about what we do.
          </p>
        </div>
      </div>
    </div>
  );
};

export default AboutUs;
```

Blog.jsx

```
import React from "react";
import MenuNav from "../components/MenuNavBar";

const Blog = () => {
  return (
    <div>
      <MenuNav />
      Blog
    </div>
  );
};

export default Blog;
```

ContactUs.jsx

```
import React from "react";
import { Input, Button } from "antd";
import MenuNav from "../components/MenuNavBar";

const ContactUs = () => {
  return (
    <div>
      <MenuNav />
      <div className="bg-blue-500 h-fit py-24 flex flex-col justify-center  items-center">
        <h1 className="text-white text-4xl">Contact Us</h1>
      </div>
      <div className="flex flex-col items-center mt-8">
        <Input className="mb-4 w-64" placeholder="Name" />
        <Input className="mb-4 w-64" placeholder="Email" />
        <textarea
```

```
        className="mb-4 w-64 h-32 p-2 resize-none"
        placeholder="Message"
       />
       <Button type="primary">Submit</Button>
     </div>
   </div>
  );
};

export default ContactUs;
```

CourseDetailedPage.jsx

```
import React, { useEffect, useState } from 'react'
import { useParams } from "react-router-dom";
import axios from "../util/AxiosInstance";

export default function CourseDetailedPage() {

   const { id } = useParams();
   const [data, setData] = useState();
   const [tags, setTags] = useState();

   const [openQuestions, setOpenQuestions] = useState(false)

   useEffect(() => {
      getCourse()
   }, []);

   const getCourse = async () => {

      const res = await axios.get(`guest/course/guest-routes/${id}`);
```

```
    console.log(res.data.course);

    setData(res.data.course);
    setTags(res.data.course.tags)
  };

  const toggleQuestion = () => {

    if(openQuestions){
      setOpenQuestions(false)
    }else{
      setOpenQuestions(true)
    }
  }



  return (
    <>
    <div className={`flex flex-col h-fit items-center justify-between ${openQuestions ? 'hidden':'block'}`} >

      {data && (
        <div className="flex flex-col absolute h-fit overflow-y-auto">
          <div className=' w-screen h-[80vh] overflow-hidden'>
            <img src={data.coverImage} className='w-full h-full object-cover'/>

          </div>
          <div className='bg-white absolute shadow bottom-0 w-screen rounded-t-3xl flex flex-col px-10 py-5 justify-between'>
            <div className='flex flex-col justify-start'>
```

```
                <span className=' text-4xl font-semibold '>{data.title}</span>
                <hr className=' bg-gray-200 w-[95vw] h-[2px] my-2 self-center' />
                <div className=' flex justify-between'>
                  <div className=' flex min-h-36 h-fit'>
                    <span className=' text-xl mt-5'>{data.description}</span>
                  </div>
                  <div className=' flex flex-col items-end text-xl'>
                    <span>Duration : <span className=' font-bold'>{data.duration}</span>
h </span>
                    <span>Assigned Instructor : <span className=' font-
bold'>{data.instructor}</span> </span>
                  </div>

                </div>



              </div>
              <div className='flex justify-between items-end'>
                <div className=' flex flex-col '>
                  <div className=' mb-5'>

                    <span className=' text-xl font-medium'> Tags</span>
                    <hr className=' mt-2' />
                  </div>
                  <div className=' flex'>
                    {tags.map((item) => (
                      <div className=' px-5 py-2 bg-blue-300 mx-2 rounded-lg'>
                        <span> {item} </span>
                      </div>
                    ))}
                  </div>
                </div>
```

```
                    <button className=' hover:bg-blue-500 border-2 border-blue-500 text-blue-500 px-10 py-2 font-bold hover:text-white rounded-3xl text-lg' onClick={() => toggleQuestion()}> Start </button>
                </div>

            </div>
        </div>
    )}
    </div>
    <div className={`flex flex-col h-fit items-center justify-between ${openQuestions ? 'block':'hidden'}`}>
    <button className=' hover:bg-blue-500 border-2 border-blue-500 text-blue-500 px-10 py-2 font-bold hover:text-white rounded-3xl text-lg' onClick={() => toggleQuestion()}> Go Back </button>
    </div>
    </>
  )
}
```

CourseDetails.jsx

```
import React, { useEffect, useState } from "react";
import { Button, Flex, Rate, Tag } from "antd";

import { useParams } from "react-router-dom";
import axios from "../util/AxiosInstance";
import { FaRegUser } from "react-icons/fa6";
const desc = ["terrible", "bad", "normal", "good", "wonderful"];

const CourseDetails = () => {
  const { id } = useParams();
  const [data, setData] = useState();
```

```
useEffect(() => {
  const getCourse = async () => {
    const res = await axios.get(`guest/course/guest-routes/${id}`);
    console.log(res.data.course);
    setData(res.data.course);
  };
  getCourse();
}, [id]);

const handleCheckout = async () => {
  try {
    const res = await axios.post("payment/create-checkout-session", {
      body: JSON.stringify({
        items: [{ name: data?.title, price: data?.price, quantity: 1 }],
      }),
    });
    console.log("res is", res);
    // window.location = url;
  } catch (error) {
    console.log(error);
  }
};
return (
  <div className="lg:px-32 lg:py-12 px-12 py-12">
    <div className="grid grid-cols-1 w-full h-[500px] md:grid-cols-2 gap-8">
      <div className="bg-white  shadow-md rounded-lg overflow-hidden">
        <img
          src={data?.coverImage}
          alt="course-logo"
          className="w-full h-full object-cover"
        />
```

```
</div>
<div className="bg-white shadow-md  rounded-lg px-6 py-8">
 <div className="text-3xl font-semibold mb-4">{data?.title}</div>
 <div className="text-xl mb-4">{data?.description}</div>
 <div className="text-lg flex flex-row mb-4 gap-3">
  <Flex gap="middle" vertical>
   <Rate tooltips={desc} value={5} />
  </Flex>
  <div className="text-xl">
   <FaRegUser />{" "}
  </div>
  <div>4500 Already Enrolled</div>
 </div>
 <div className="text-xl font-bold">Course Details</div>
 <div className="text-lg pb-4 pl-5">
  <ul>
   <li className="pt-4 pb-2">
    <span className="font-semibold pt-4 pb-4">Duration:</span>{" "}
    {data?.duration} hours
   </li>
   <li className="pt-2 pb-2">
    <span className="font-semibold pt-4 pb-4">Price: </span>
    {data?.price}
   </li>
   <li className="pt-2 pb-2">
    <span className="font-semibold pt-4 pb-4">Level: </span>
    {"Intermediate"}
   </li>
   <li className="pt-2 pb-2">
    <span className="font-semibold">Language: </span>
    {"English"}
   </li>
```

```
            <li className="pt-2 pb-2">
              <span className="font-semibold ">Categoris:</span>{" "}
              {data?.tags.map((tag) => {
                let color = tag.length > 5 ? "geekblue" : "green";
                if (tag === "loser") {
                  color = "volcano";
                }
                return (
                  <Tag color={color} key={tag} className="text-lg">
                    {tag.toUpperCase()}
                  </Tag>
                );
              })}
            </li>
          </ul>
        </div>
        <div className="flex items-end">
          <Button type="primary" onClick={handleCheckout} className="w-full">
            Enroll Now
          </Button>
        </div>
      </div>
    </div>
  </div>
 );
};

export default CourseDetails;
```

Courses.jsx

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

```
import MenuNav from "../components/MenuNavBar";
import axios from "../util/AxiosInstance";
import Loader from "../components/Loader";
import { Link } from "react-router-dom";
import { Button, Tag } from "antd";

const Courses = () => {
  const [loading, setLoading] = useState(false);
  const [data, setData] = useState([]);
  useEffect(() => {
    const getAllCourses = async () => {
      setLoading(true);
      try {
        const res = await axios.get("guest/course/guest-routes");
        setData(res.data.data);
        setLoading(false);
      } catch (error) {
        console.log(error);
        setLoading(false);
      }
    };
    getAllCourses();
  }, []);

  return (
    <div>
      <MenuNav />
      {loading ? (
        <Loader />
      ) : (
        <div className="lg:px-28 pb-24 px-12">
          <div className="text-center text-3xl font-bold text-[#411EE2] pt-10">
```

```
          All Courses In Our WebSite
        </div>
        <div className="grid pt-10 lg:grid-cols-4 md:grid-cols-3 gap-3 grid-cols-1">
          {data.map((item) => (
            <div
              key={item._id}
              className="group relative flex flex-col justify-between  shadow-2xl rounded-b-xl border-2 "
            >
              <div>
                <div className="min-h-80 aspect-h-1  aspect-w-1 w-full py-3 px-2 overflow-hidden lg:aspect-none group-hover:opacity-40 lg:h-80">
                  <img
                    src={item.coverImage}
                    alt="cover"
                    className="h-full w-full object-cover object-center lg:h-full lg:w-full"
                  />
                </div>
                <div className="flex justify-between p-3">
                  <h3 className="text-2xl  text-[#295cb1] ">
                    <Link to={`/course/${item._id}`}>
                      <span
                        aria-hidden="true"
                        className="absolute  inset-0 rounded-t-3xl "
                      />
                      {item.title}
                    </Link>
                    <p className="text-lg font-medium text-gray-900">
                      This is Dummy Descripion . You Can See the Full course
                      details by clicking on title
                    </p>
                  </h3>
```

```
          </div>

        </div>
        <div>
          <div className="px-2 pb-3">
            {item.tags.map((tag) => {
              let color = tag.length > 5 ? "geekblue" : "green";
              if (tag === "loser") {
                color = "volcano";
              }
              return (
                <Tag color={color} key={tag}>
                  {tag.toUpperCase()}
                </Tag>
              );
            })}
          </div>
          <div className="flex flex-row mr-2 space-x-3 justify-between">
            <p className="text-lg text-left p-2 ">
              Price : {item.price} $
            </p>

            <Link to={`/course/${item._id}`}>
              <Button type="primary">Enroll</Button>
            </Link>
          </div>
        </div>
      </div>
    ))}
  </div>
 </div>
)}
```

```
    </div>
  );
};

export default Courses;
```

Home.jsx

```
import React from "react";
import MenuNav from "../components/MenuNavBar";
import HeroSection from "../components/home/hero/HeroSection";
import AwardsSection from "../components/home/awards/AwardsSection";
import CoursesCategory from "../components/home/courses-category/CoursesCategory";

const Home = () => {
  return (
    <div>
      <MenuNav />
      <HeroSection />
      <AwardsSection />
      <CoursesCategory />

    </div>
  );
};

export default Home;
```

Login.jsx

```
import { Link, useNavigate } from "react-router-dom";
import Swal from "sweetalert2";
```

```
import login from "../assets/login.jpg";
import axios from "../util/AxiosInstance";
import Loader from "../components/Loader";
import { Form, Input } from "antd";
import { UserOutlined, LockOutlined } from "@ant-design/icons";

import { jwtDecode } from "jwt-decode";
import { useState } from "react";

const Login = () => {
  const navigate = useNavigate();
  const [loading, setLoading] = useState(false);

  const onFinish = async (values) => {
    setLoading(true);
    try {
      const res = await axios.post("auth/login", {
        userName: values.userName,
        password: values.password,
      });
      const decoded = jwtDecode(res.data.accessToken);
      const payload = {
        decodedJWT: decoded,
        accessToken: res.data.accessToken,
        refreshToken: res.data.refreshToken,
      };
      setLoading(false);
      localStorage.setItem("jsonwebtoken", JSON.stringify(payload));
      const role = decoded.userRole;
      if (role === "admin") {
        navigate("/admin/home");
      } else if (role === "instructor") {
```

```
      navigate("/instructor/home");
    } else {
      navigate("/");
    }
    window.location.reload();
  } catch (err) {
    setLoading(false);
    Swal.fire({
      icon: "error",
      title: "Oops...",
      text: err.response.data.body || "Something went wrong!",
    });
    console.log(err);
  }
};

const inputStyle =
  "p-3 w-96 rounded-3xl border border-purple focus:outline-none focus:border-blue-500";

return (
  <div
    className="min-h-screen bg-cover bg-no-repeat bg-center flex justify-center items-center"
    style={{ backgroundImage: `url(${login})` }}
  >
    <div className="absolute inset-0 flex justify-center items-center">
      <div className="border px-12 mx-12 mt-36 py-12  bg-opacity-20 bg-black border-black rounded-lg">
        <div className="px-12 pt-10 lg:px-24">
          <div className="pb-12 text-center ">
            <span className="text-[46px] text-[#411EE2] font-extrabold ">
              SIGN IN
```

```
</span>

<Form name="basic" onFinish={onFinish} autoComplete="off">
 <div className="mt-4">
  <Form.Item
   name="userName"
   rules={[
    {
     required: true,
     message: "Please input your username!",
    },
   ]}
   hasFeedback
  >
   <Input
    prefix={<UserOutlined className="site-form-item-icon" />}
    placeholder="userName"
    className={inputStyle}
   />
  </Form.Item>
 </div>

 <div className="mt-2">
  <Form.Item
   name="password"
   rules={[
    {
     required: true,
     message: "Please input your password!",
    },
   ]}
   hasFeedback
```

```
            >
              <Input.Password
                prefix={<LockOutlined className="site-form-item-icon" />}
                placeholder="Password"
                className={inputStyle}
              />
            </Form.Item>
          </div>
          <div className="mt-2">
            <Form.Item>
              <button
                type="submit"
                className="bg-[#411EE2] text-white font-bold px-6 py-3 rounded-md hover:bg-
[#333333]"
              >
                Login
              </button>
            </Form.Item>
          </div>
          <div>
            <Link to="/register" className="text-white hover:underline">
              Not a member ? Register
            </Link>
          </div>
        </Form>

        {loading && <Loader />}
      </div>
    </div>
  </div>
</div>
```

```
  );
};

export default Login;
```

Modules.jsx

```
import React from "react";
import MenuNav from "../components/MenuNavBar";

const Modules = () => {
  return (
    <div>
      <MenuNav />
      Modules
    </div>
  );
};

export default Modules;
```

Register.jsx

```
import { Link, useNavigate } from "react-router-dom";
import Swal from "sweetalert2";
import login from "../assets/login.jpg";
import axios from "../util/AxiosInstance";
import Loader from "../components/Loader";
import { Form, Input } from "antd";
import uploadFileToFirebase from "../util/UploadFilesToFIreBase";

import { useState } from "react";
```

```
const Register = () => {
  const navigate = useNavigate();
  const [loading, setLoading] = useState(false);
  const [file, setFile] = useState("");

  const onFinish = async (values) => {
    setLoading(true);
    console.log(values);
    let uploadImg = "No Image";
    try {
      if (
        values.name === "" ||
        values.email === "" ||
        values.password === "" ||
        values.contactNumber === ""
      ) {
        Swal.fire({
          icon: "error",
          title: "Oops...",
          text: "All Fields Required!",
        });
        return;
      }

      const result = await Swal.fire({
        title: "Do you want to Register With Learnify?",
        showDenyButton: true,
        confirmButtonText: "Yes",
        denyButtonText: "No",
      });
      if (result.isConfirmed) {
```

```
    if (file) {
      const response = await uploadFileToFirebase(file);
      uploadImg = response;
    }
    const res = await axios.post("auth/register", {
      userName: values.name,
      email: values.email,
      image: uploadImg,
      password: values.password,
      mobileNo: values.contactNumber,
      userRole: "learner",
    });
    console.log("response", res);
    Swal.fire(
      "Congratulations! You Have Successfully Registered with Laarnify",
      "",
      "success"
    );
    setLoading(false);
    navigate("/login");
  } else {
    setLoading(false);
    Swal.fire("Registraion Cancelled", "", "error");
  }
} catch (err) {
  setLoading(false);
  console.log(err);

  Swal.fire({
    icon: "error",
    title: "Oops...",
    text:
```

```
        err.response.data.code === 11000
          ? "User Already Exists!"
          : "Something went wrong!",
      });
    }
  };

  const inputStyle =
    "p-3 w-96 rounded-md border border-purple focus:outline-none focus:border-blue-500";

  return (
    <div>
      {loading ? (
        <Loader />
      ) : (
        <div
          className="min-h-screen  bg-cover bg-no-repeat bg-center flex justify-center items-center"
          style={{ backgroundImage: `url(${login})` }}
        >
          <div className="absolute inset-0  pt-24 flex justify-center items-center">
            <div  className="border  px-12  mx-12  mt-36   bg-black bg-opacity-10  border-black rounded-lg">
              <div className="px-12 pt-10 lg:px-24">
                <div className="pb-12 text-center ">
                  <span className="text-[46px] text-[#411EE2] font-extrabold ">
                    SIGN UP
                  </span>

                  <div className="mt-6 flex sm:flex-row justify-center">
                    <label htmlFor="fileInput">
                      <img
```

```
      className="rounded-full"
      src={
        file
          ? URL.createObjectURL(file)
          : "https://icon-library.com/images/no-image-icon/no-image-icon-0.jpg"
      }
      alt="avatar"
      style={{
        width: "120px",
        height: "120px",
        cursor: "pointer",
      }}
    />
  </label>
  <input
    type="file"
    id="fileInput"
    name="file"
    style={{ display: "none" }}
    accept="image/*"
    onChange={(e) => {
      const file = e.target.files[0];
      setFile(file);
    }}
    required
  />
</div>
<Form
  name="register"
  onFinish={onFinish}
  initialValues={{
    prefix: "86",
```

```
  }}
  scrollToFirstError
>
  <div className="pt-2">
    <Form.Item
      name="name"
      rules={[
        {
          required: true,
          message: "Please input your username!",
          whitespace: true,
        },
      ]}
      hasFeedback
    >
      <Input className={inputStyle} placeholder="username" />
    </Form.Item>
  </div>

  <div className="pt-2">
    <Form.Item
      name="email"
      rules={[
        {
          type: "email",
          message: "The input is not valid E-mail!",
        },
        {
          required: true,
          message: "Please input your E-mail!",
        },
      ]}
```

```
    hasFeedback
  >
    <Input className={inputStyle} placeholder="Email" />
  </Form.Item>
</div>

<div className="pt-2">
  <Form.Item
    name="contactNumber"
    rules={[
      {
        required: true,
        message: "Please input your phone number!",
      },
      ({ getFieldValue }) => ({
        validator(_, value) {
          if (/^\d{10}$/.test(value)) {
            return Promise.resolve();
          }
          return Promise.reject(
            "Phone number must be 10 digits and contain only numbers."
          );
        },
      }),
    ]}
    hasFeedback
  >
    <Input
      className={inputStyle}
      placeholder="Phone Number"
    />
  </Form.Item>
```

```
</div>

<div className="pt-2">
 <Form.Item
  noStyle
  shouldUpdate={(prevValues, currentValues) =>
   prevValues.role !== currentValues.role
  }
 >
  {({ getFieldValue }) =>
   getFieldValue("role") === "Foreigner" ? (
    <Form.Item
     name="uniqueField"
     rules={[
      {
       required: true,
      },
     ]}
    >
     <Input
      className={inputStyle}
      placeholder="Enter Your Passport Number"
     />
    </Form.Item>
   ) : getFieldValue("role") === "Passenger" ? (
    <Form.Item
     name="uniqueField"
     rules={[
      {
       required: true,
      },
     ]}
```

```
          >
            <Input
              className={inputStyle}
              placeholder="Enter Your Passport Number"
            />
          </Form.Item>
        ) : null
      }
    </Form.Item>
  </div>

  <div className="pt-2">
    <Form.Item
      name="password"
      rules={[
        {
          required: true,
          message: "Please input your password!",
        },
        {
          min: 8,
          message: "Password must be at least 8 characters.",
        },
        {
          pattern: /(?=.*\d)(?=.*[a-z])(?=.*[A-Z])/,
          message:
            "Password must contain at least one uppercase letter, one lowercase letter,
and one number.",
        },
      ]}
      hasFeedback
    >
```

```
    <Input.Password
      className={inputStyle}
      placeholder="Password"
    />
  </Form.Item>
</div>

<div className="pt-2">
  <Form.Item
    name="confirm"
    dependencies={["password"]}
    hasFeedback
    rules={[
      {
        required: true,
        message: "Please confirm your password!",
      },
      ({ getFieldValue }) => ({
        validator(_, value) {
          if (
            !value ||
            getFieldValue("password") === value
          ) {
            return Promise.resolve();
          }
          return Promise.reject(
            new Error(
              "The new password that you entered do not match!"
            )
          );
        },
      }),
```

```
            ]}
           >
            <Input.Password
              className={inputStyle}
              placeholder="Confirm Password"
            />
           </Form.Item>
          </div>

          <div className="pt-2">
           <div>
            <Form.Item>
             <button
              type="primary"
              htmlType="submit"
              className="bg-[#411EE2] text-white font-bold px-6 py-3 rounded-md
hover:bg-blue-800"
             >
              Register
             </button>
            </Form.Item>
           </div>
          </div>

          <div className="pt-2">
           <Link to="/login" className="text-white hover:underline">
            Already a member? Login
           </Link>
          </div>
         </Form>

         {loading && <Loader />}
```

```
            </div>
          </div>
        </div>
      </div>
    </div>
    )}
  </div>
 );
};

export default Register;
```

UserProfile.jsx

```
import React, { useEffect, useState } from 'react'
import UserDetailsPanel from '../components/admin/UserDetailsPanel'
import ViewEnrolledCourses from '../components/ViewEnrolledCourses'
import ViewCompletedCourses from '../components/ViewCompletedCourses'

export default function UserProfile() {



  return (
    <div className=' flex flex-row bg-slate-100 w-full h-full'>
      <div className=' w-[33vw] ml-10 mb-10'>
        <UserDetailsPanel />
      </div>
      <div className='h-[68vh] w-[60vw] mt-16 ml-10 flex flex-col justify-between '>

        <ViewEnrolledCourses />
        <ViewCompletedCourses/>
```

```
        </div>
      </div>
    )
}
```

Router.jsx

```
import React from "react";
import { Route, Routes, Navigate } from "react-router-dom";
import Home from "../pages/Home";
import Login from "../pages/Login";
import Register from "../pages/Register";
import AboutUs from "../pages/AboutUs";
import ContactUs from "../pages/ContactUs";
import Blog from "../pages/Blog";
import Courses from "../pages/Courses";
import Modules from "../pages/Modules";
import InstructorHome from "../pages/instructor/Home";
import AdminHome from "../pages/admin/Home";
import Swal from "sweetalert2";
import InstructorCourses from "../pages/instructor/Courses";
import InstructorEnrollment from "../pages/instructor/Enrollments";
import CourseDetails from "../pages/CourseDetails";
import UserProfile from "../pages/UserProfile"
import CourseDetailed from "../pages/CourseDetailedPage"

const Router = () => {
  const ProtectedRoute = ({ children }) => {
    const isAuthenticated = localStorage.getItem("jsonwebtoken") ? true : false;

    if (!isAuthenticated) {
      return <Navigate to="/login" />;
```

```
  }
  return children;
};
const AdminRoute = ({ children }) => {
  const token = localStorage.getItem("jsonwebtoken");
  const isAuthenticated = token ? true : false;
  const payload = JSON.parse(token);
  console.log(payload);
  const isAdmin = payload.decodedJWT.userRole === "admin";

  if (!isAuthenticated || !isAdmin) {
    Swal.fire({
      icon: "error",
      title: "Oops...",
      text: "Please Logout Login to your Admin Account",
    });
    return <Navigate to="/login" />;
  }
  return children;
};
const InstructorRoute = ({ children }) => {
  const token = localStorage.getItem("jsonwebtoken");
  const isAuthenticated = token ? true : false;
  const payload = JSON.parse(token);
  console.log(payload);
  const isAdmin = payload.decodedJWT.userRole === "instructor";

  if (!isAuthenticated || !isAdmin) {
    Swal.fire({
      icon: "error",
      title: "Oops...",
      text: "Please Logout Login to your Instructor Account",
```

```
    });
    return <Navigate to="/login" />;
  }
  return children;
};
const LoggedOutRoute = ({ children }) => {
  const token = localStorage.getItem("jsonwebtoken");
  const isAuthenticated = token ? true : false;
  let role;
  if (token) {
    const payload = JSON.parse(token);
    console.log(payload);
    role = payload.decodedJWT.userRole;
  }

  if (isAuthenticated) {
    if (role === "admin") {
      return <Navigate to="/admin/home" />;
    } else if (role === "instructor") {
      return <Navigate to="/instructor/home" />;
    } else {
      return <Navigate to="/" />;
    }
  }
  return children;
};
return (
  <Routes>
    <Route path="/" element={<Home />} />
    <Route
      path="/login"
      element={
```

```
    <LoggedOutRoute>
      <Login />
    </LoggedOutRoute>
  }
/>
<Route
  path="/register"
  element={
    <LoggedOutRoute>
      <Register />
    </LoggedOutRoute>
  }
/>
<Route path="/about-us" element={<AboutUs />} />
<Route path="/contact-us" element={<ContactUs />} />
<Route path="/blog" element={<Blog />} />
<Route path="/courses" element={<Courses />} />
<Route path="/modules" element={<Modules />} />
<Route path="/course/:id" element={<CourseDetails />} />
<Route path="/user" element={<UserProfile/>}/>
<Route path="/course/detailed/:id" element = {<CourseDetailed/>} />

<Route
  path="/instructor/home"
  element={
    <ProtectedRoute>
      <InstructorRoute>
        <InstructorHome />
      </InstructorRoute>
    </ProtectedRoute>
  }
/>
```

```
    <Route
      path="/instructor/courses"
      element={
        <ProtectedRoute>
          <InstructorRoute>
            <InstructorCourses />
          </InstructorRoute>
        </ProtectedRoute>
      }
    />
    <Route
      path="/instructor/enrollment"
      element={
        <ProtectedRoute>
          <InstructorRoute>
            <InstructorEnrollment />
          </InstructorRoute>
        </ProtectedRoute>
      }
    />
    <Route
      path="/admin/home"
      element={
        <ProtectedRoute>
          <AdminRoute>
            <AdminHome />
          </AdminRoute>
        </ProtectedRoute>
      }
    />
  </Routes>
);
```

```
};

export default Router;
```

AxiosInstance.js

```
// axiosInstance.js

import axios from "axios";

const axiosInstance = axios.create({
  baseURL: "http://localhost:8000/api/",
  // other default config
});

axiosInstance.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem("jsonwebtoken");
    const payload = JSON.parse(token);
    const accessToken = payload?.accessToken;
    if (token) {
      config.headers["Authorization"] = `Bearer ${accessToken}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

export default axiosInstance;
```

UploadFilesToFIreBase.jsx

```
import { ref, uploadBytes, getDownloadURL } from "firebase/storage";
import { storage } from "./firebase";

const uploadFileToFirebase = async (file) => {
  try {
    const timestamp = Date.now();
    const filename = `${file.name}_${timestamp}`;
    const imageRef = ref(storage, `images/${file.name + filename}`);

    // Upload the file
    const snapshot = await uploadBytes(imageRef, file);

    // Get the download URL
    const downloadURL = await getDownloadURL(snapshot.ref);

    console.log("File uploaded successfully. Download URL:", downloadURL);

    return downloadURL;
  } catch (error) {
    console.error("Error uploading file:", error);
    throw error;
  }
};

export default uploadFileToFirebase;
```

App.jsx

```
import { BrowserRouter } from "react-router-dom";
import Layouts from "./layouts/Layout";
```

```
function App() {
  let hours = 1;
  const now = new Date().getTime();
  let setupTime = localStorage.getItem("setupTime");
  if (setupTime == null) {
    localStorage.setItem("setupTime", now);
  } else {
    if (now - setupTime > hours * 60 * 60 * 1000) {
      localStorage.clear();
      localStorage.setItem("setupTime", now);
    }
  }
  return (
    <div>
      <BrowserRouter>
        <Layouts />
      </BrowserRouter>
    </div>
  );
}

export default App;
```

index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@font-face {
  font-family: FuturaMdBt;
```

```
  src: url("./assets/FuturaMediumBT.ttf");
}

@font-face {
  font-family: FutuBd;
  src: url("./assets/FutuBd.ttf");
}
```

tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./src/**/*.{html,js,jsx}"],
  theme: {
    extend: {
      fontFamily: {
        FuturaMdBt :['FuturaMdBt', 'sans'],
        FutuBd : ['FutuBd', 'sans']

      },
    },
  },
  plugins: [],
};
```

vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react-swc'

// https://vitejs.dev/config/
export default defineConfig({
```

```
  plugins: [react()],
})
```