Sri Lanka Institute of Information Technology

Distributed Systems (SE3020)

Assignment

2024

Submitted by:

IT21182396	Ranaweera A.P.
IT21118340	Kumarathunga S.A.D.S.
IT21173790	Sooriyaarachchi M.D.A.
IT21158322	Senanayake W.G.B

Introduction

Vidu Piyasa is a innovative educational platform that makes online learning much easier. It was developed drawing inspiration from platforms like Coursera and Udemy. This platform is designed to provide wide a array of features that will meet the needs of both course instructors and learners.

Instructors will have the ability to manage course content, including unloading, and removing materials such as lecture notes, videos, and quizzes. Instructors will also be able to monitor learner progress. The platform administrator will focus on validating and approving content related to the course before it is publicly released. Admin will also be responsible for managing financial transactions related to course enrollments.

This platform will provide learners with the ability to enroll in multiple courses simultaneously without scheduling conflicts, It also features an integrated payment gateway for course enrollment payments. Upon successful enrollment, learners will receive confirmation via SMS and email, utilizing third-party services for notification purposes.

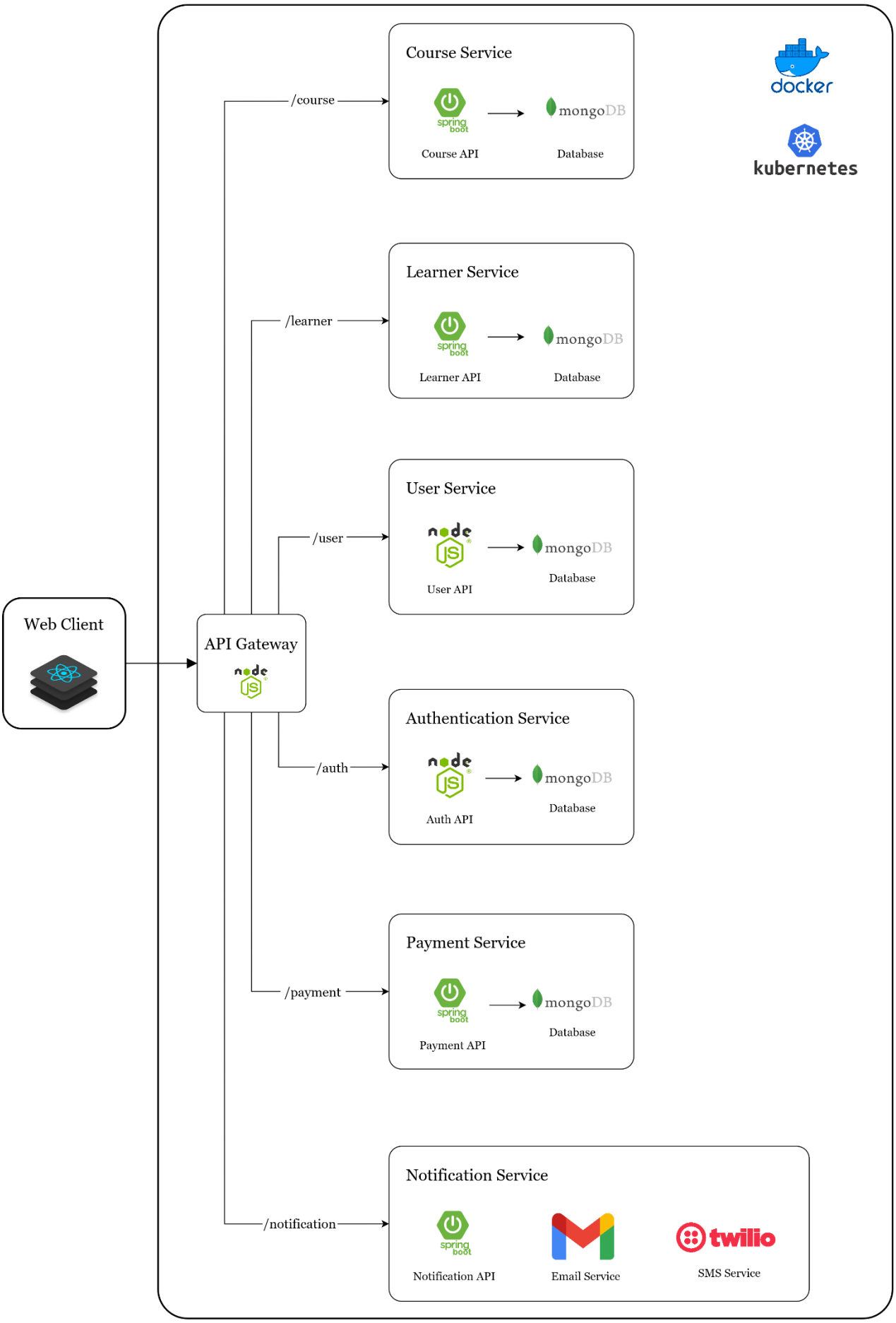
The primary goal behind this project is to build an functional educational platform that also serves as an hands-on example of micro-services architecture principles. It highlights core principles such as service independance, scalabillty, and the abillty to work with other services that are developed using different programming languages and web frameworks.

GitHub Repo

https://github.com/IT21118340/ViduPiyasa

Presentation Video

High Level Architectural Diagram



Service Interfaces

Course Service

Interface	Description
getCourseById	Get course details by id
getCourses	Get list of all courses and their details
addCourse	Insert course
addCourseContent	Add content for the course. Ex:- Videos & Questions
updateCourseById	Update course
updateCourseContentById	Update content for the course by id
deleteCourseById	Delete course by id
approveCourseContentById	Approve course content
getAllStudentsByCourseId	Get list of students for a course

(See Appendix A)

Learner Service

Interface	Description
enrollCourse	Enroll student to a course
unenrollCourse	Unenroll student from course
getEnrolledCourses	Get list of enrolled courses for a student
getCourseProgress	Get progress of the student
updateCourseProgress	Update progress of the student

(See Appendix A)

Payment Service

Interface	Description
addPaymentDetails	Add verified payment details. (Verification is done by a third-party service)
viewPaymentDetails	Get Payment Details

(See Appendix A)

User Service

Interface	Description
login	Verify user credentials and login user (Returns a JWT token)
register	Register user
update	Update user details
delete	Remove user
getUserDetailsbytoken	Get user details by token
getUserDetailsbyID	Get user details by user id
getUserList	Get list of all user

(See Appendix A)

Authentication Service

Interface	Description
registerAuth	Add authentication configurations
authorize	Verify authentication token
authorizeStudent	Verify Student user role
authorizeInstructor	Verify Instructor user role
authorizeAdmin	Verify Admin user role

(See Appendix A)

Notification Service

Interface	Description
sendEmail	Send email to one recipient
batchSendEmails	Send email to multiple recipient
sendSMS	Send sms to one recipient
batchSendSMS	Send sms to multiple recipient

(See Appendix A)

Workflows

API Gateway

API Gateway is a core component in micro-services architecture. It's main purpose is to serve as an centralized entry point for API calls from web clients to target services.

It offers functionalities such as authentication, authorization, role-based access control, routing, and logging. API Gateway can simplify the user experience by handling requests invoking multiple micro-services and aggregating their responses.

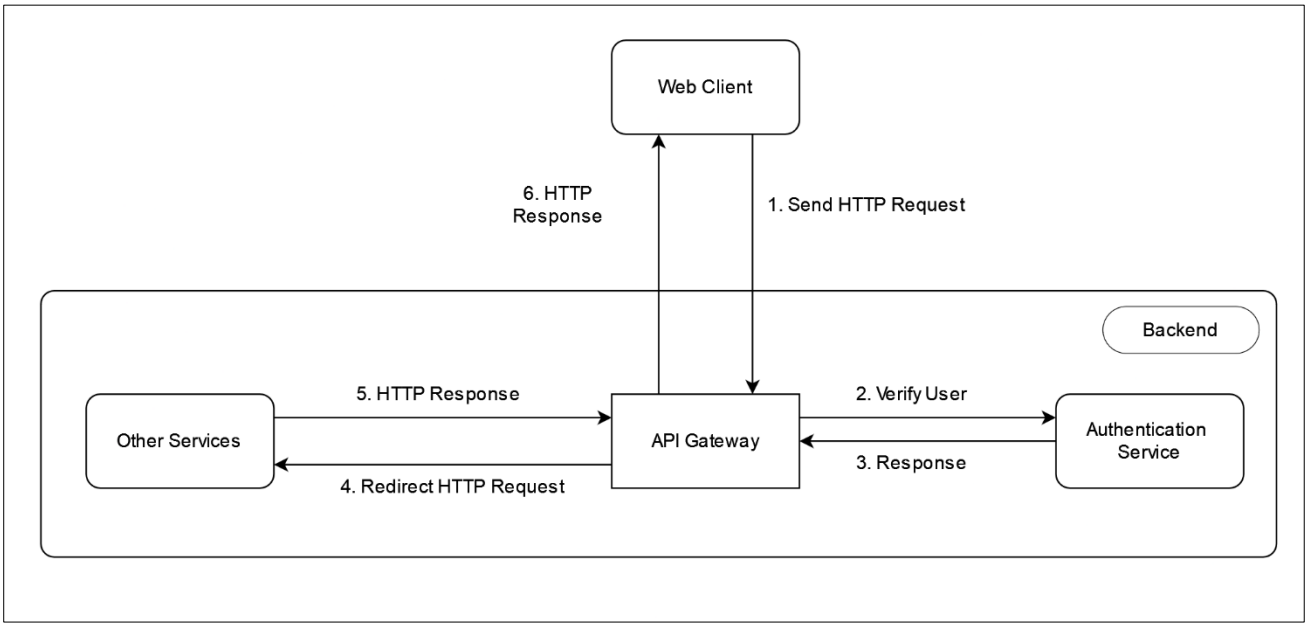


Diagram of the API Gateway

Considering minimalistic requirements of the project and the need to provide an centralized entry point for incoming traffic, we decided to implement custom API Gateway using NodeJS that is suitable for project needs.

Course Service

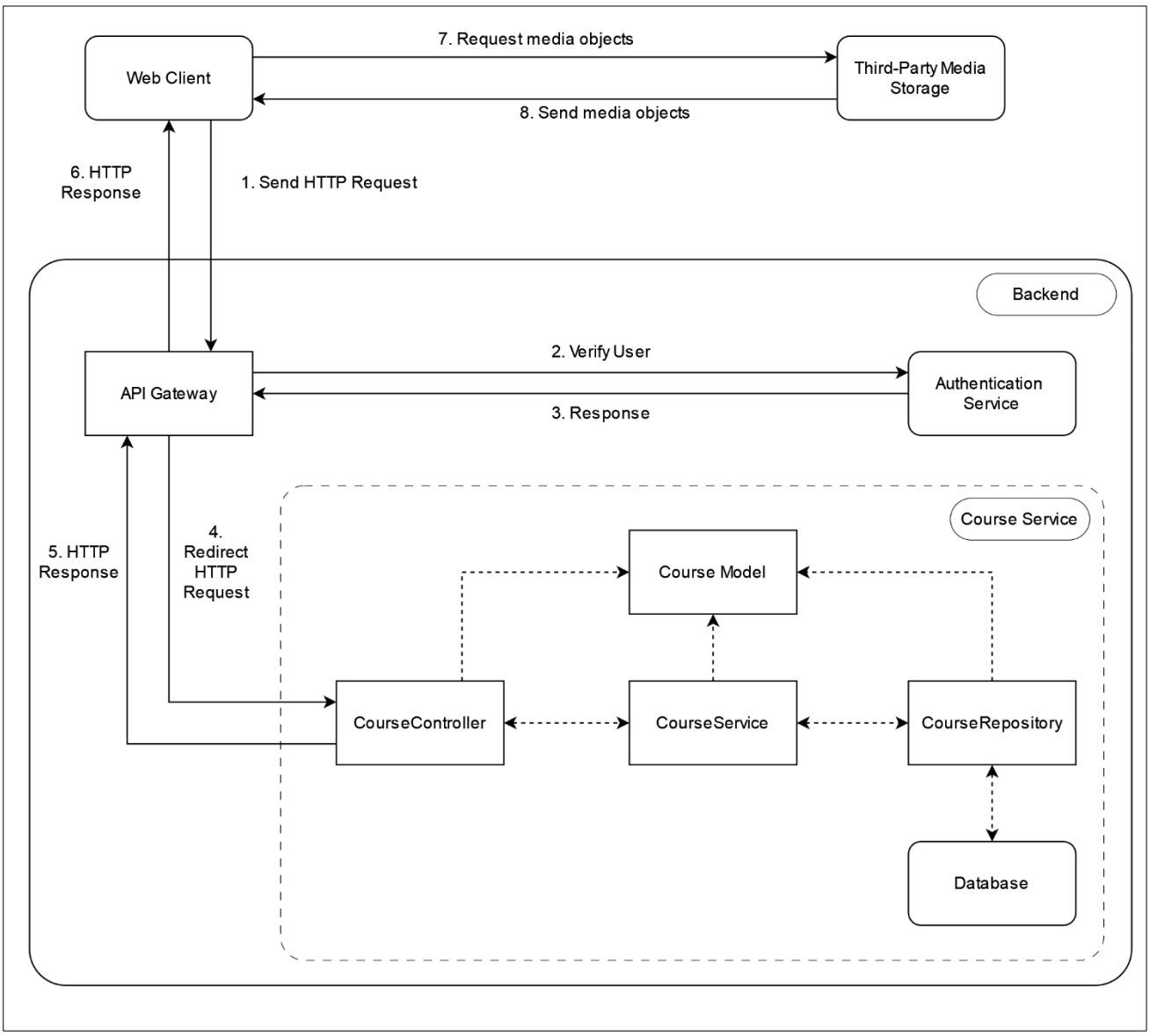


Diagram of Course Service

Purpose of this API is to handle task related to course management such creating course, uploading materials, approve course content, updating course details, remove course materials, and remove course. Layered architecture and repository pattern is used to structure this API.

Once API Gateway receives a request, It will call authentication service to verify user credentials inside authentication header. If included user credentials are valid, API gateway will redirect the request to Course Service. Once API Gateway receives a response from the service, It will be redirected to web client.

Learner Service

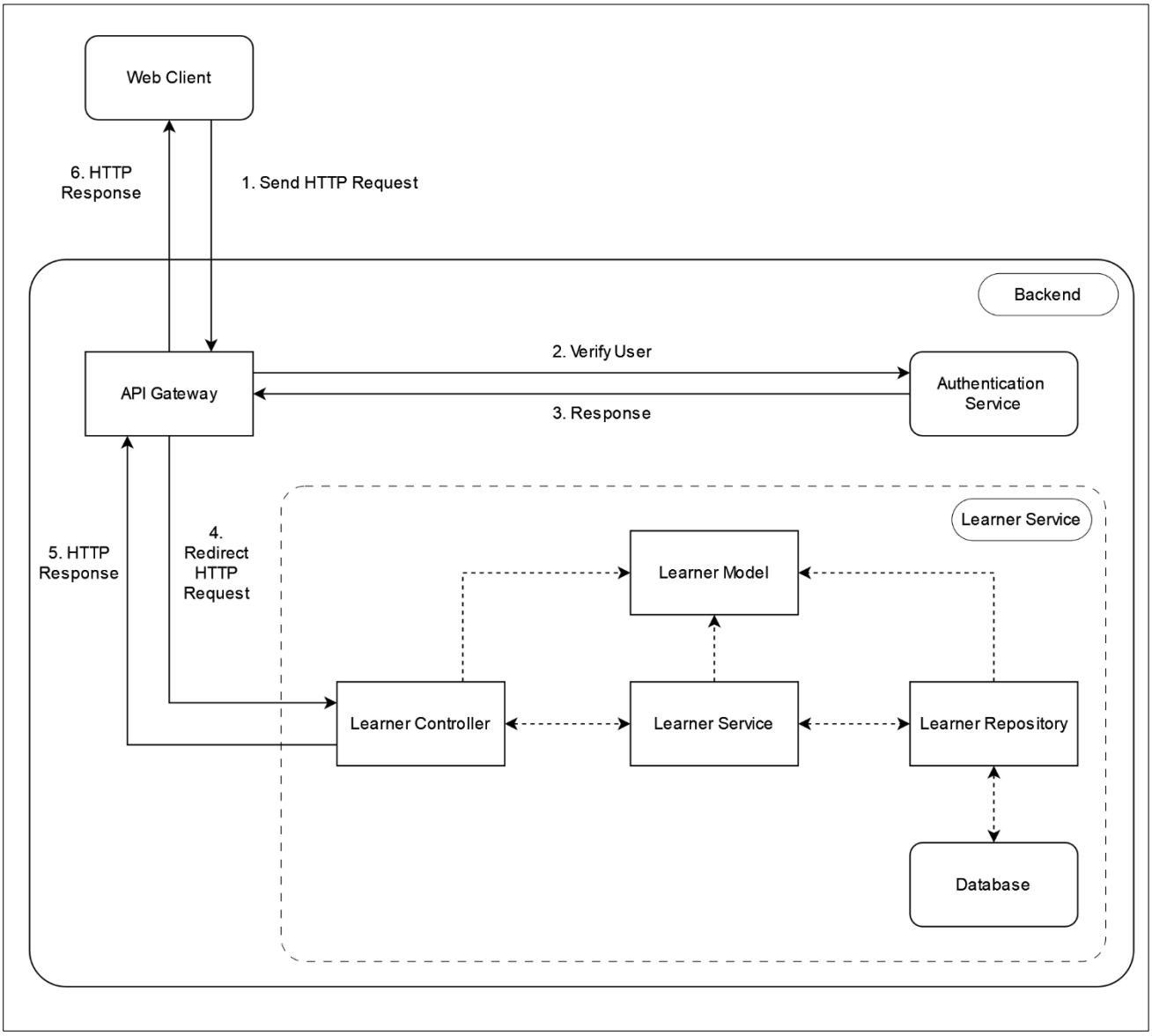


Diagram of Learner Service

Purpose of this API is to handle task related to learner management such enroll to course/s, unenroll from course, view progress and update progress.

Once API Gateway receives a request, It will call authentication service to verify user credentials inside authentication header. If included user credentials are valid, API gateway will redirect the request to Learner Service. Once API Gateway receives a response from the service, It will be redirected to web client.

User Service

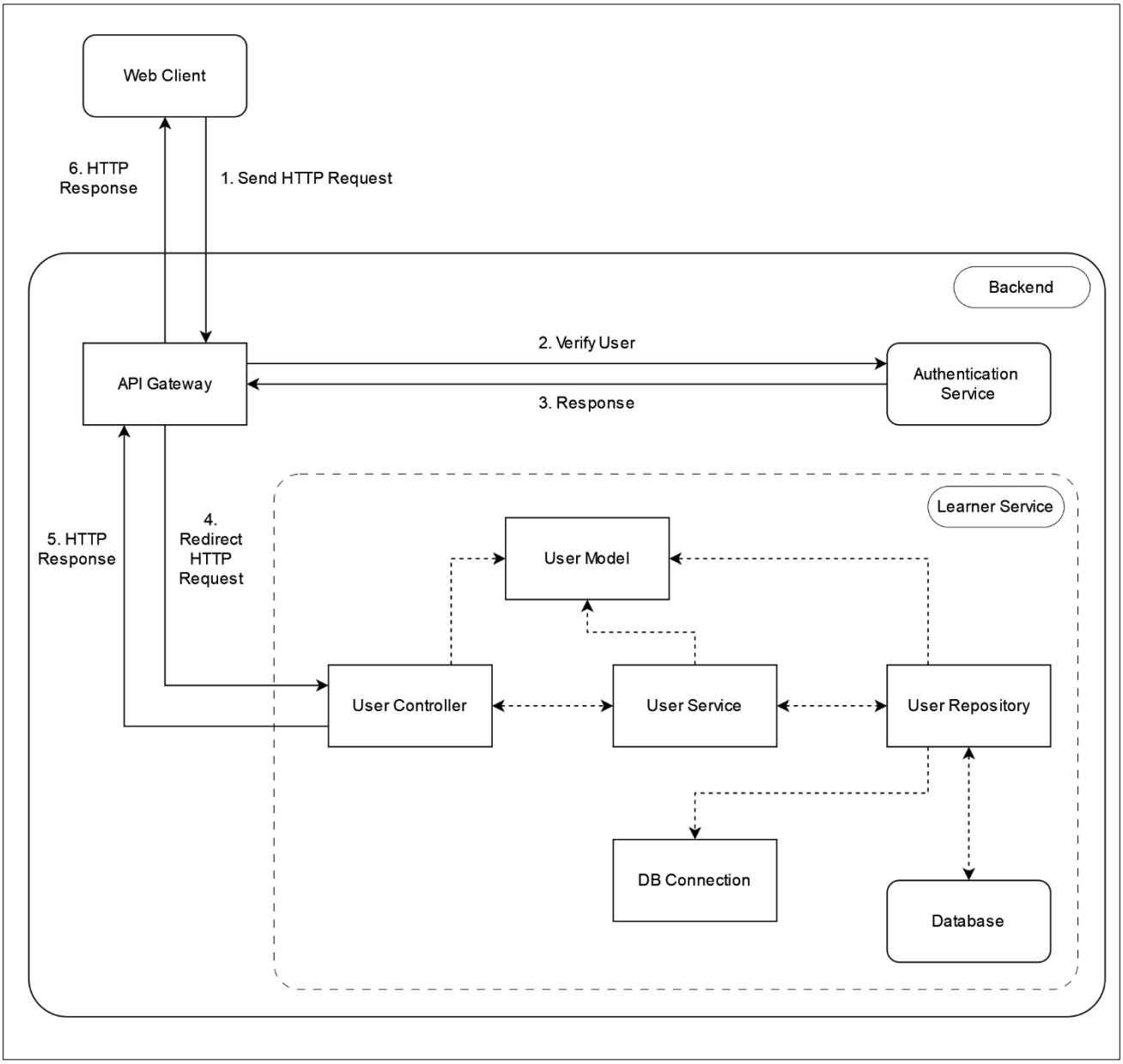


Diagram of User Service

Purpose of this API is to handle task related to user management such register user, get user details, update details, and remove user.

Once API Gateway receives a request, It will call authentication service to verify user credentials inside authentication header. If included user credentials are valid, API gateway will redirect the request to User Service. Once API Gateway receives a response from the service, It will be redirected to web client.

Authentication Service

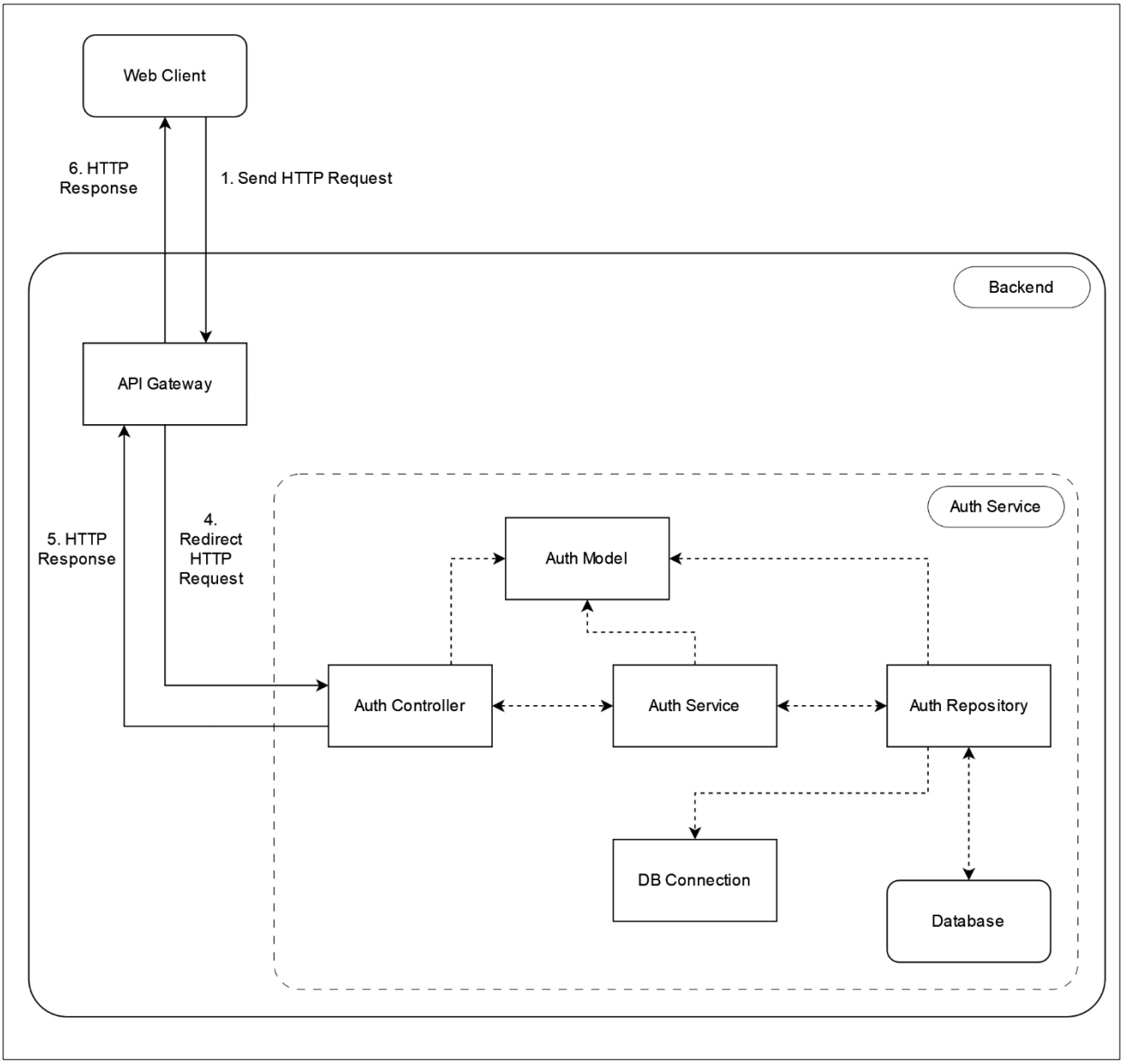


Diagram of Authentication Service

Purpose of this API is to handle task related to authentication management such authenticate user, verify user credentials and verify user role.

Once API Gateway receives a request, It will be redirect to Authentication Service. When API Gateway receives a response from the service, It will be redirected to web client.

Payment Service

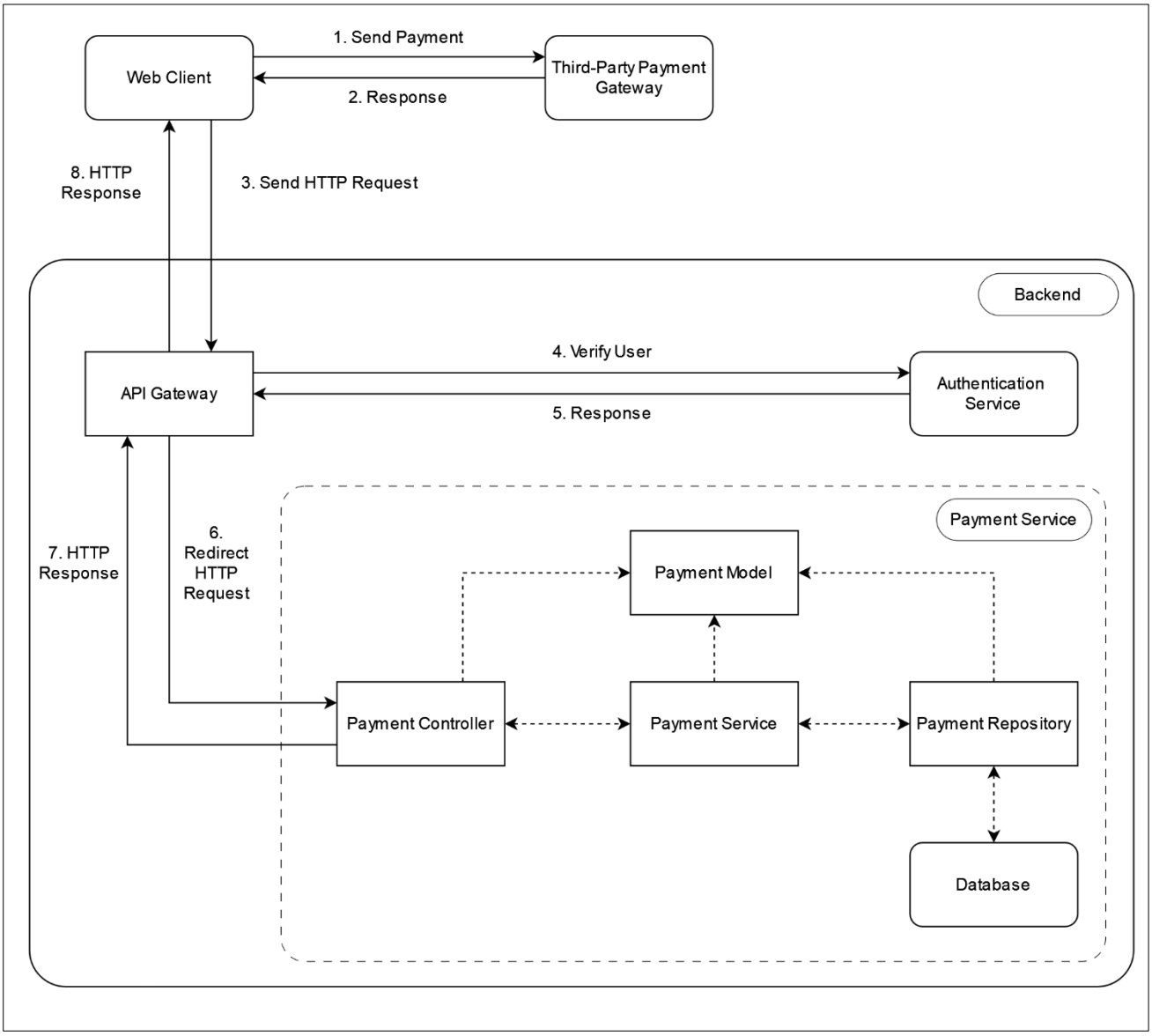


Diagram of Payment Service

Purpose of this API is to handle task related to payment management such add payment details and view payment details.

First payment details will be validated using a third-party payment gateway in the web client. Once API Gateway receives a request with validated payment details, It will call authentication service to verify user credentials inside authentication header. If included user credentials are valid, API gateway will redirect the request to Payment Service. Once API Gateway receives a response from the service, It will be redirected to web client.

Notification Service

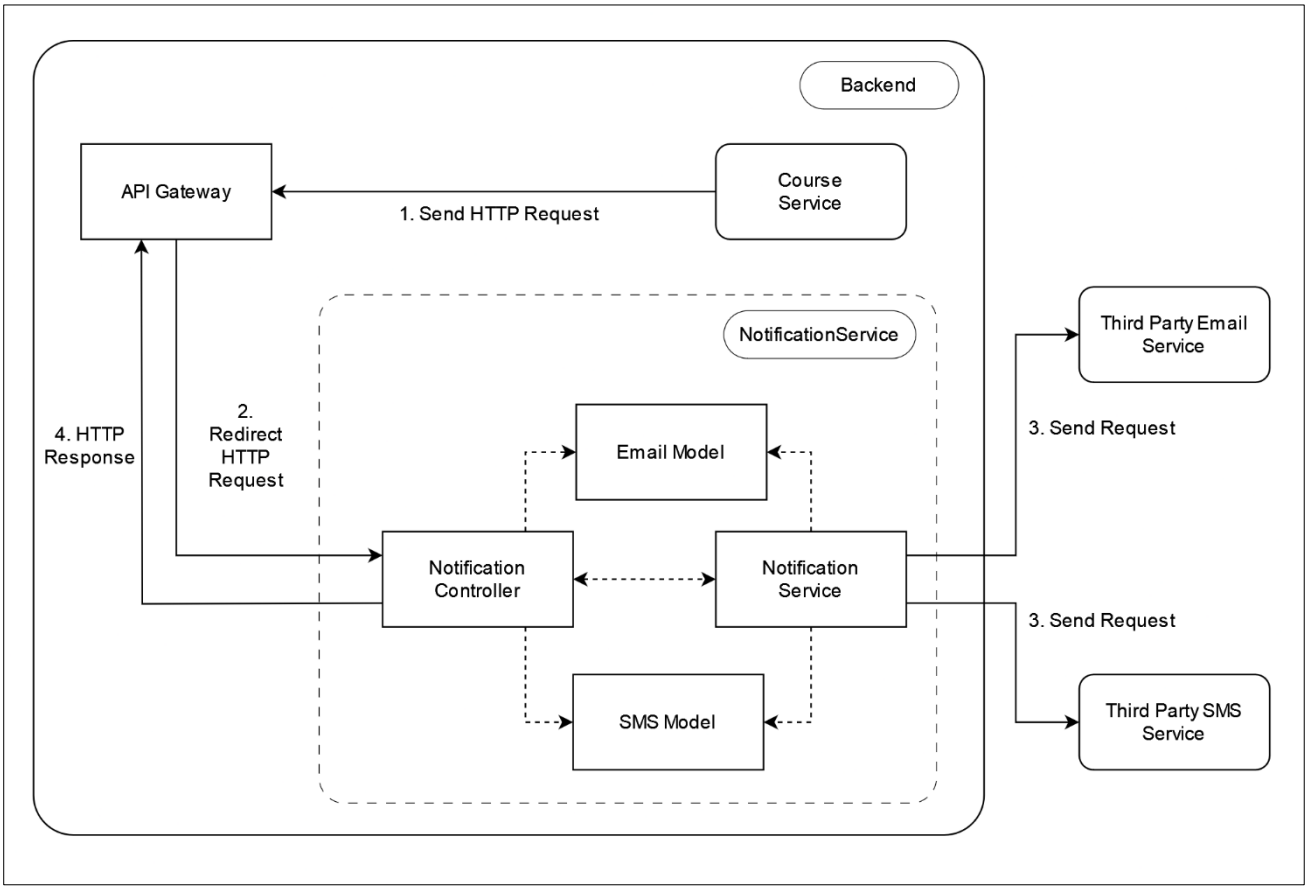


Diagram of Notification Service

Purpose of this API is to handle task related to notification management such send emails and SMS using third party services.

This service is mostly used by Course Service to notify users about their enrollment to a course. Course Service will send a request containing relevant details to API Gateway, which will be redirected to the Notification Service. Then notifications are sent, API Gateway receives a response from the service that task is completed.

Security

Encrypted Password

This security measure used to protect sensitive information, such as user passwords, by converting them into a format that is unreadable without the decryption key.

This system uses an encryption algorithm to convert original user password into a scrambled version of it, before storing them in the database. In login process, encrypted version of the user given password will be matched with the stored password. This mechanism helps to protect sensitive user data and prevent unauthorized access to accounts, even if the database is compromised.

JWT Authentication

JSON Web Token (JWT) is an open standard for securely transmitting information between parties as a JSON object.

In the login process, system generates a JWT that includes the user's information and roles and sent it to the client. The client includes this token is all future requests, which allows user to verify user's identity and permissions without communicating with the database.

Role-Based Access Control (RBAC)

This security mechanism is used to restricts access to services based on the roles of individual users. In RBAC, permissions are grouped into roles, and users will be assigned a role to manage access rights for services and resources in the system.

This system has three user roles **Student, Instructor, Admin**. Upon registration, Users will be assigned to one of these roles. After login to the system, they can only access services permitted for their assigned role. For example, If Student try to upload or remove course materials their request will not be processed as only Instructors have that permission.

Individual Contribution

ID	Name	Contribution
IT21182396	Ranaweera A.P.	• Authentication Service • Web UIs for Course Management
IT21118340	Kumarathunga S.A.D.S.	• Course Service • Notification Service • API Gateway • Docker and Kubernetes Implementation • Web UIs for Platform Administrator
IT21173790	Sooriyaarachchi M.D.A.	• Learner Service • Payment Service • Web UIs for Learner Management
IT21158322	Senanayake W.G.B.	• User Service • Web UI for Instructor Management

Appendix

Appendix A

CourseController.java

```
package com.ds.courseservice.controller;


import com.ds.courseservice.model.Course;
import com.ds.courseservice.model.CourseContent;
import com.ds.courseservice.model.Msg;
import com.ds.courseservice.service.CourseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;


@CrossOrigin("*")
@RestController
@RequestMapping(value = "/api/course")
public class CourseController {


    @Autowired
    private CourseService courseService;


    @GetMapping("/{courseId}")
    public ResponseEntity<?> getCourseById(@PathVariable String courseId){
        return courseService.getCourseById(courseId);
    }


    @GetMapping("/")
    public ResponseEntity<?> getCourses(){
        return courseService.getCourses();
    }


    @PostMapping("/")
    public ResponseEntity<?> addCourse(@RequestBody Course course,){
        return courseService.insertCourse(course);
    }
```

```
    @PostMapping("/content")
    public ResponseEntity<?> addCourseContent(@RequestBody CourseContent courseContent,){
        return courseService.insertCourse(courseContent);
    }

    @PutMapping("/{courseId}")
    public ResponseEntity<?> updateCourseById(@PathVariable String courseId, @RequestBody Course course,){
        return courseService.updateCourseById(courseId, course);
    }

    @PutMapping("/content")
    public ResponseEntity<?> updateCourseContentById(@RequestBody CourseContent courseContent,){
        return courseService.updateCourseContentById(courseContent);
    }

   @DeleteMapping("/{courseId}")
    public ResponseEntity<?> deleteById(@PathVariable String courseId,){
        return courseService.deleteById(courseId);
    }

    @PostMapping("/approve/{courseId}")
    public ResponseEntity<?> deleteById(@PathVariable Msg msg,){
        return courseService.deleteById(courseId);
    }

    @GetMapping("/students/{courseId}")
    public ResponseEntity<?> getAllStudentsByCourseId(@PathVariable String courseId){
        return courseService.getAllStudentsByCourseId(courseId);
    }
}
```

LearnerController.java

```
package com.learner.learnerservice.controller;
import com.learner.learnerservice.model.LearnerDTO;
import com.learner.learnerservice.service.LearnerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
@CrossOrigin("*")
@RestController
@RequestMapping(value = "/api/learner")
public class LearnerController {
    @Autowired
    private LearnerService learnerService;

    @GetMapping("/courses/")
    public ResponseEntity<?> getEnrolledCourses(@RequestBody LearnerDTO learner){
        return learnerService.getEnrolledCourses(learner);
    }

    @PostMapping("/enroll/")
    public ResponseEntity<?> enrollCourse(@RequestBody LearnerDTO learner){
        return learnerService.enrollCourse(learner);
    }

    @PostMapping("/unenroll/")
    public ResponseEntity<?> unenrollCourse(@RequestBody LearnerDTO learner){
        return learnerService.unenrollCourse(learner);
    }

    @GetMapping("/progress/")
    public ResponseEntity<?> getCourseProgress(@RequestBody LearnerDTO learner){
        return learnerService.getCourseProgress(learner);
    }

   @PostMapping("/progress")
    public ResponseEntity<?> updateCourseProgress(@RequestBody LearnerDTO learner){
        return learnerService.updateCourseProgress(learner);
    `
```

```
    @GetMapping("/progress/")
    public ResponseEntity<?> getCourseProgress(@RequestBody LearnerDTO learner){
        return learnerService.getCourseProgress(learner);
    }

    @PostMapping("/progress")
    public ResponseEntity<?> updateCourseProgress(@RequestBody LearnerDTO learner){
        return learnerService.updateCourseProgress(learner);
    }
}
```

UserRoutes.js

```
import express from "express"
import {login,register,updateUser,deleteUser,getUserDetailsbyID,getUserList,getUserDetailsbytoken} from "../controller/user.controller.js"
const userRoutes = express.Router();

userRoutes.post("/login",  login);
userRoutes.post("/register",  register);
userRoutes.put("/update/:id",  updateUser);
userRoutes.delete("/delete/:id",  deleteUser);
userRoutes.get("/user_token/getdetails",  getUserDetailsbytoken);
userRoutes.get("/:id",  getUserDetailsbyID);
userRoutes.get("/",  getUserList);

export default userRoutes;
```

AuthRoutes.js

```
import express from "express"
import {addAuthConfig, authorize, authorizeStudent, authorizeInstructor, authorizeAdmin} from "../controller/auth.controller.js"
const authRoutes = express.Router();

authRoutes.get("/authorize",  authorize);
authRoutes.get("/authorizeSeller",  authorizeStudent);
authRoutes.get("/authorizeBuyer",  authorizeInstructor);
authRoutes.get("/authorizeAdmin",  authorizeAdmin);
authRoutes.post("/registerAuth",  addAuthConfig);

export default authRoutes;
```

PaymentController.java

```
package com.pay.paymentservice.controller;


import com.pay.paymentservice.model.Payment;
import com.pay.paymentservice.service.PaymentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;


@CrossOrigin("*")
@RestController
@RequestMapping(value = "/api/payment")
public class PaymentController {


    @Autowired
    private PaymentService paymentService;


    @GetMapping("/{paymentId}")
    public ResponseEntity<?> getPaymentById(@PathVariable String paymentId){
        return paymentService.getPaymentById(paymentId);
    }


    @PostMapping("/")
    public ResponseEntity<?> addPayment(@RequestBody Payment payment){
        return paymentService.insertPayment(payment);
    }
}
```

NotificationController.java

```
package com.notify.notificationservice.controller;


import com.notify.notificationservice.model.Email;
import com.notify.notificationservice.model.Sms;
import com.notify.notificationservice.service.NotificationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;


@CrossOrigin("*")
@RestController
@RequestMapping(value = "/api/notification")
public class NotificationController {


    @Autowired
    private NotificationService notificationService;


    @PostMapping("/email")
    public ResponseEntity<?> sendEmail(@RequestBody Email mail){
        return notificationService.sendEmail(mail);
    }


    @PostMapping("/sms")
    public ResponseEntity<?> sendSMS(@RequestBody Sms sms){
        return notificationService.sendSMS(sms);
    }


            @PostMapping("/batchmail")
    public ResponseEntity<?> batchSendEmails(@RequestBody Email mail){
        return notificationService.batchSendEmails(mail);
    }


    @PostMapping("/batchsms")
    public ResponseEntity<?> batchSendSMS(@RequestBody Sms sms){
        return notificationService.batchSendSMS(sms);
    }
}
```

Appendix B

COURSE SERVICE

```
package com.ds.courseservice.service;

import com.ds.courseservice.model.Course;
import com.ds.courseservice.model.CourseContent;
import com.ds.courseservice.model.Msg;

public interface CourseService {

	public ResponseEntity<?> getCourseById(String courseId);

	public ResponseEntity<?> getCourses();

	public ResponseEntity<?> addCourse(Course course,);

	public ResponseEntity<?> addCourseContent(Course course);

	public ResponseEntity<?> updateCourseById(String courseId, Course course);

	public ResponseEntity<?> updateCourseContentById(Course course);

	public ResponseEntity<?> deleteById(String courseId);

	public ResponseEntity<?> approveCourse(String courseId);

	public ResponseEntity<?> getAllStudentsByCourseId(String courseId);
}
```

```
package com.ds.courseservice.repository;

import com.ds.courseservice.model.Course;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface CourseRepository extends MongoRepository<Course, String> {

}
```

LEARNER SERVICE

```
package com.learner.learnerservice.service;

import com.learner.learnerservice.model.LearnerDTO;

public interface LearnerService {

	public ResponseEntity<?> getEnrolledCourses(LearnerDTO learner);

	public ResponseEntity<?> enrollCourse(LearnerDTO learner);

	public ResponseEntity<?> unenrollCourse(LearnerDTO learner);

	public ResponseEntity<?> getCourseProgress(LearnerDTO learner);

	public ResponseEntity<?> updateCourseProgress(LearnerDTO learner);

}
```

LEARNER REPOSITORY

```
package com.learner.learnerservice.repository;

import com.learner.learnerservice.model.LearnerDTO;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface LearnerRepository extends MongoRepository<LearnerDTO, String> {

}
```

PAYMENT SERVICE

```
package com.pay.paymentservice.service;

import com.pay.paymentservice.model.Payment;

public interface ICourseService {

	public ResponseEntity<?> getPaymentById(String paymentId);

	public ResponseEntity<?> addPayment(Payment payment);

}
```

PAYMENT REPOSITORY

```
package com.pay.courseservice.repository;

import com.pay.paymentservice.model.Payment;

import org.springframework.data.mongodb.repository.MongoRepository;

public interface PaymentRepository extends MongoRepository<Payment, String> {

}
```

```
package com.notify.notificationservice.service;

import com.notify.notificationservice.model.Email;
import com.notify.notificationservice.model.Sms;

public interface NotificationService {

	public ResponseEntity<?> sendEmail(Email mail);

	public ResponseEntity<?> sendSMS(Sms sms);

	public ResponseEntity<?> batchSendEmails(Email mail);

	public ResponseEntity<?> batchSendSMS(Sms sms);

}
```