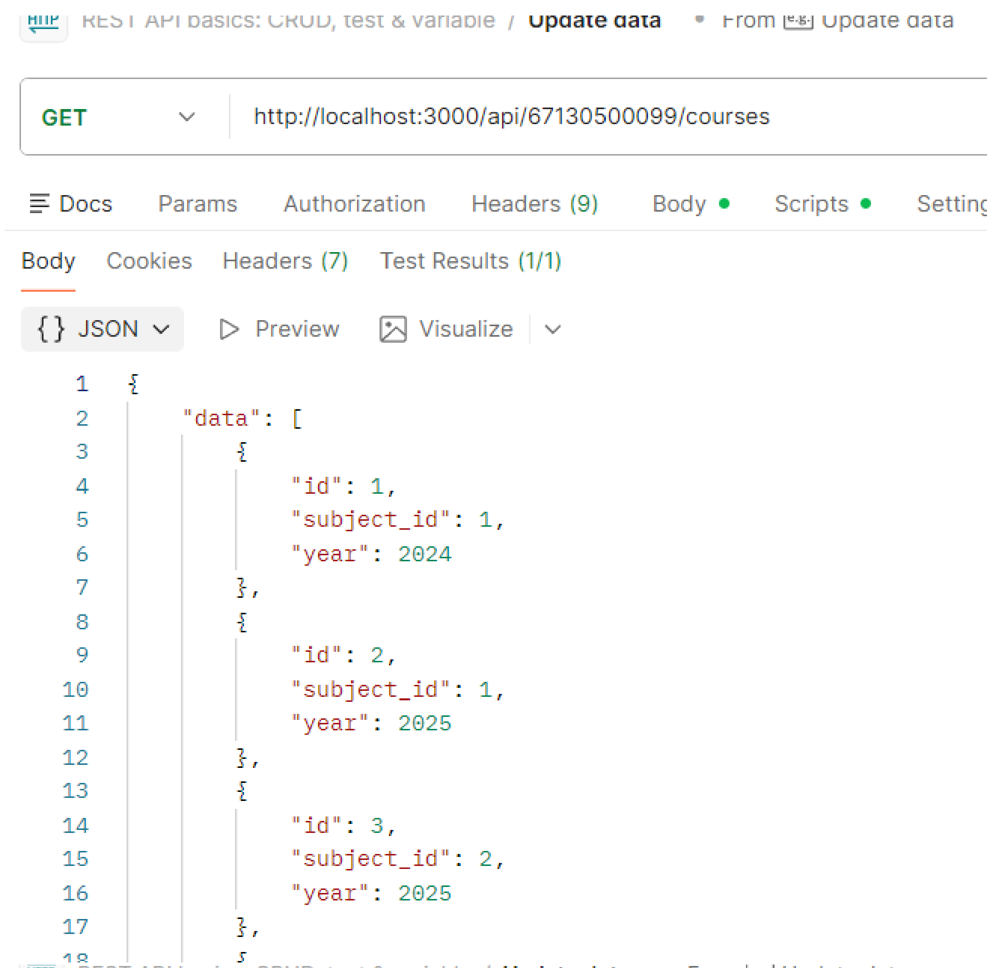
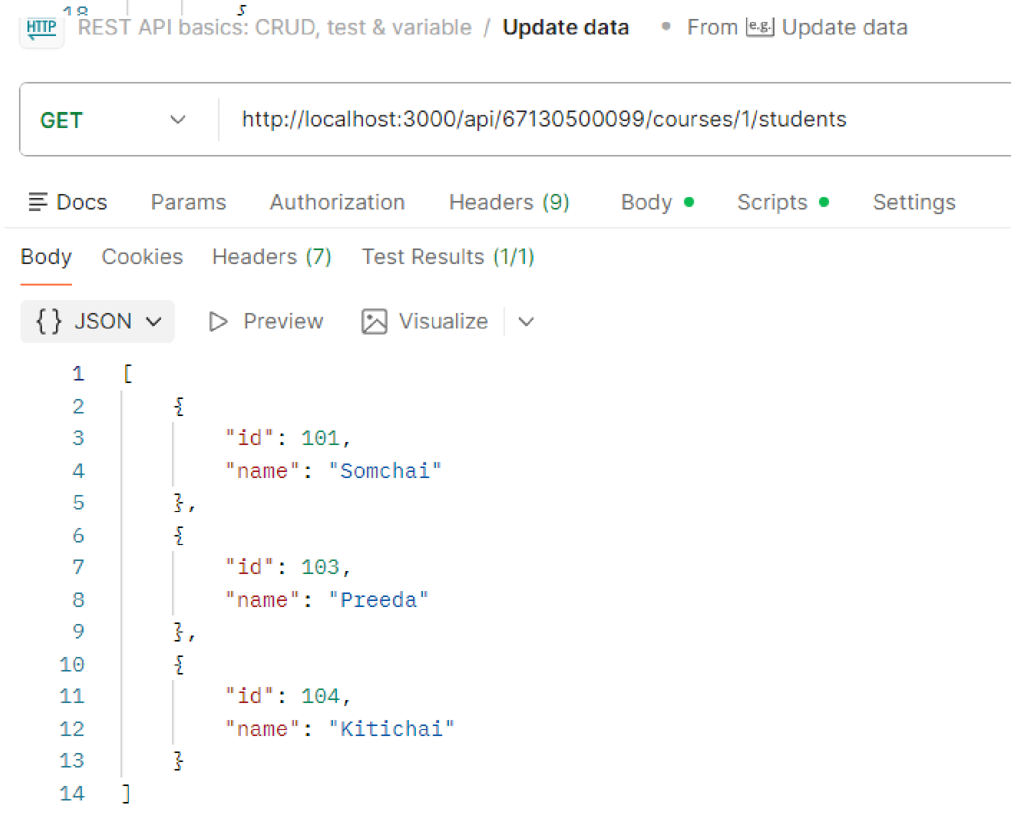

REST API basics: CRUD, test & variable / **Update data** • From Update data

GET http://localhost:3000/api/67130500099/courses

Docs Params Authorization Headers (9) Body ● Scripts ● Setting
Body Cookies Headers (7) Test Results (1/1)
{ } JSON Preview Visualize

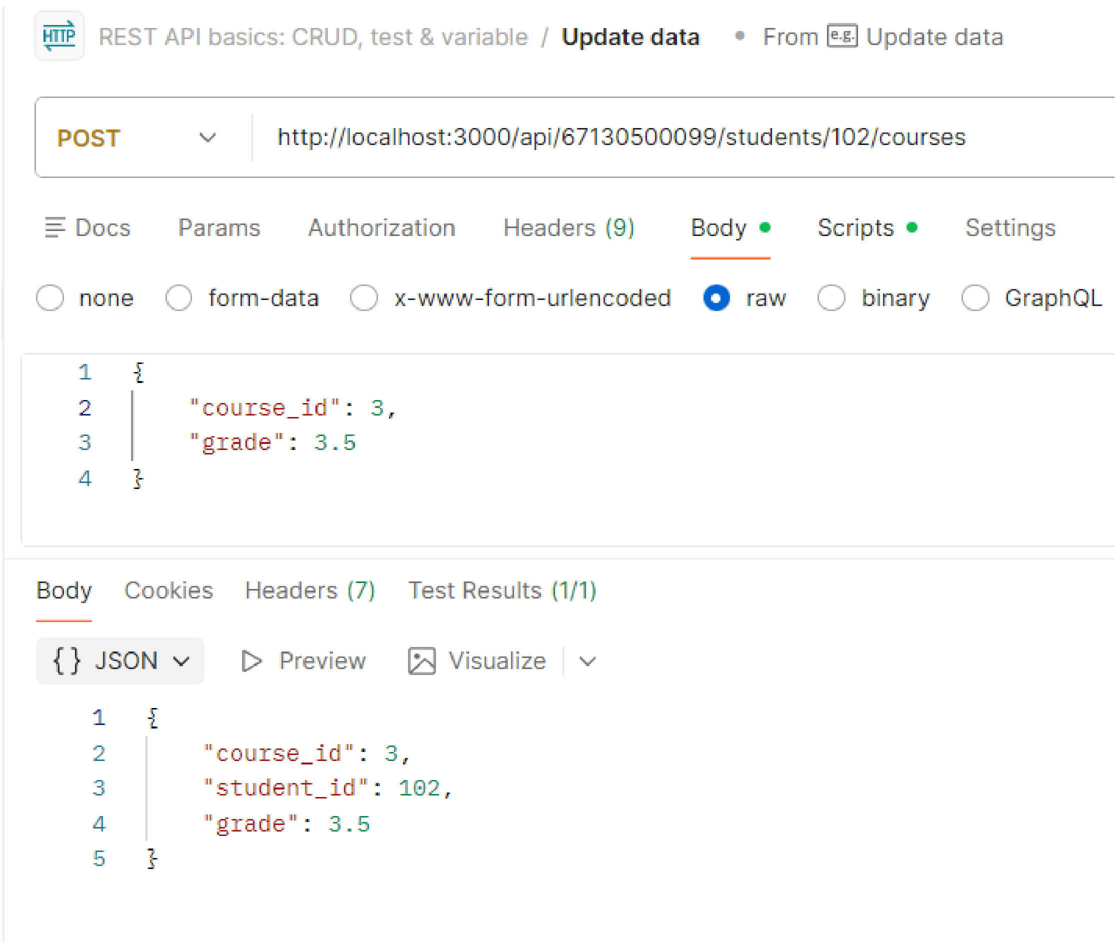
```
{
    "data": [
        {
            "id": 1,
            "subject_id": 1,
            "year": 2024
        },
        {
            "id": 2,
            "subject_id": 1,
            "year": 2025
        },
        {
            "id": 3,
            "subject_id": 2,
            "year": 2025
        },
```


REST API basics: CRUD, test & variable / **Update data** • From Update data

GET http://localhost:3000/api/67130500099/courses/1/students

Docs Params Authorization Headers (9) Body ● Scripts ● Settings
Body Cookies Headers (7) Test Results (1/1)
{ } JSON Preview Visualize

```
[
    {
        "id": 101,
        "name": "Somchai"
    },
    {
        "id": 103,
        "name": "Preeda"
    },
    {
        "id": 104,
        "name": "Kitichai"
    }
]
```



REST API basics: CRUD, test & variable / **Update data** • From Update data

POST http://localhost:3000/api/67130500099/students/102/courses

Docs Params Authorization Headers (9) Body Scripts Settings

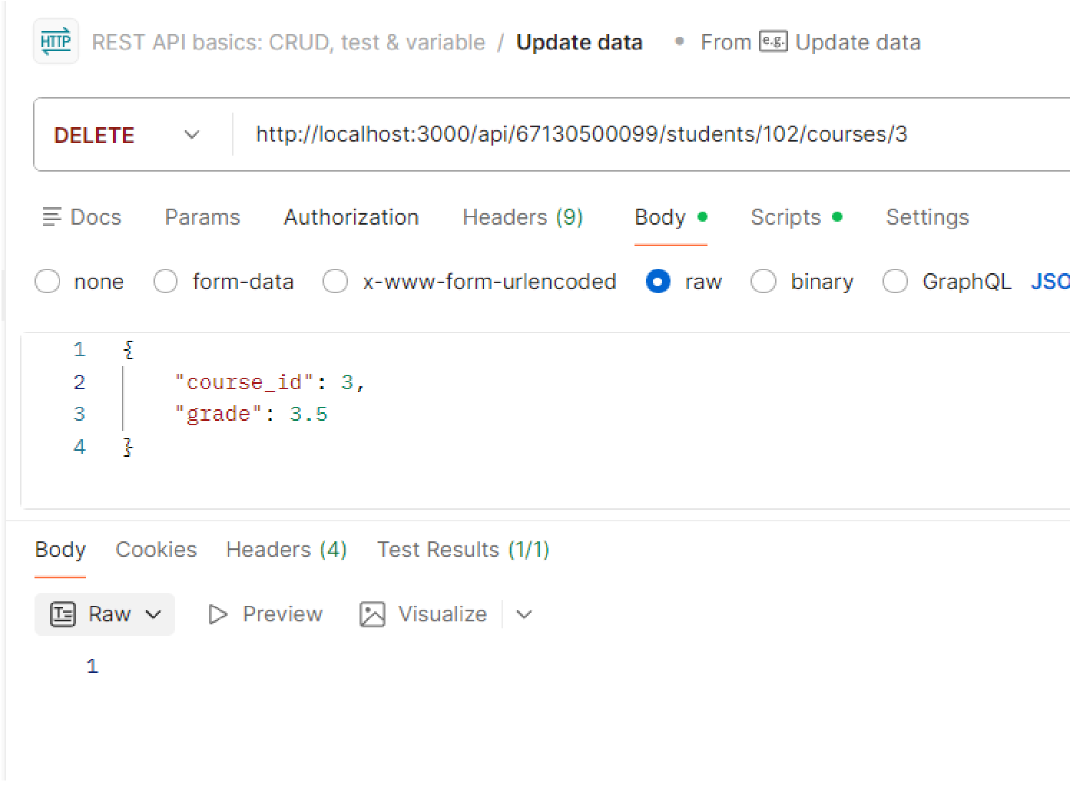
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL

```
{
    "course_id": 3,
    "grade": 3.5
}
```

Body Cookies Headers (7) Test Results (1/1)

{ } JSON Preview Visualize

```
{
    "course_id": 3,
    "student_id": 102,
    "grade": 3.5
}
```



REST API basics: CRUD, test & variable / **Update data** • From Update data

DELETE http://localhost:3000/api/67130500099/students/102/courses/3

Docs Params Authorization Headers (9) Body Scripts Settings

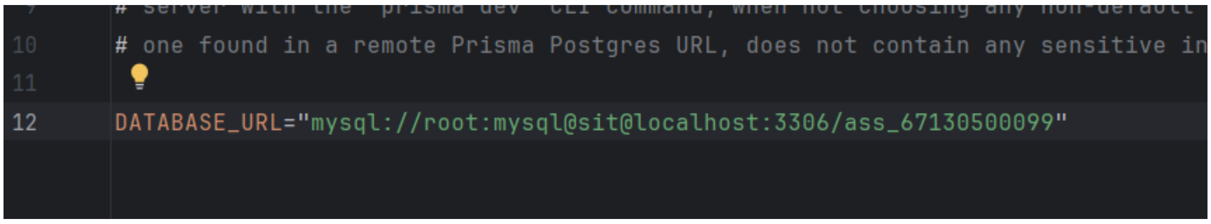
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSO

```
{
    "course_id": 3,
    "grade": 3.5
}
```

Body Cookies Headers (4) Test Results (1/1)

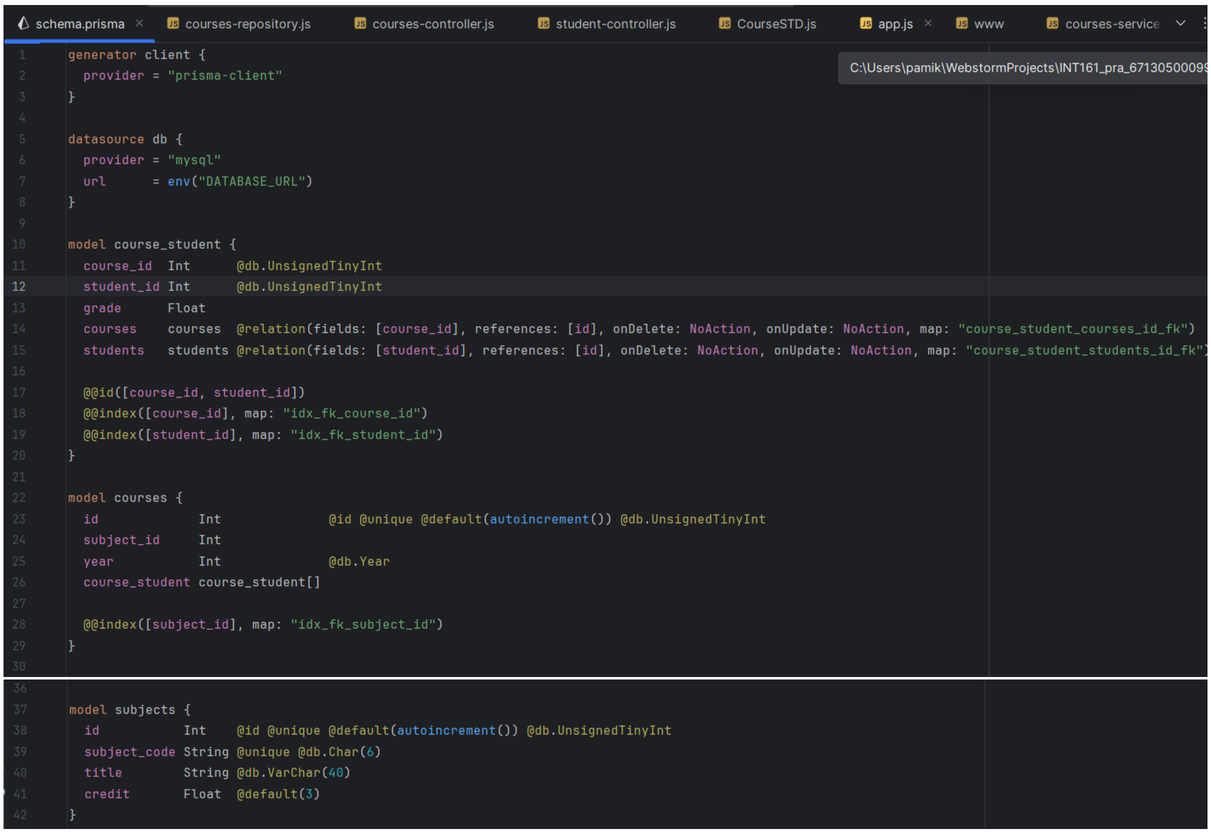
Raw Preview Visualize

.env



```
# server with the  prisma dev  CLI command, when not choosing any non-default
# one found in a remote Prisma Postgres URL, does not contain any sensitive in

DATABASE_URL="mysql://root:mysql@sit@localhost:3306/ass_67130500099"
```

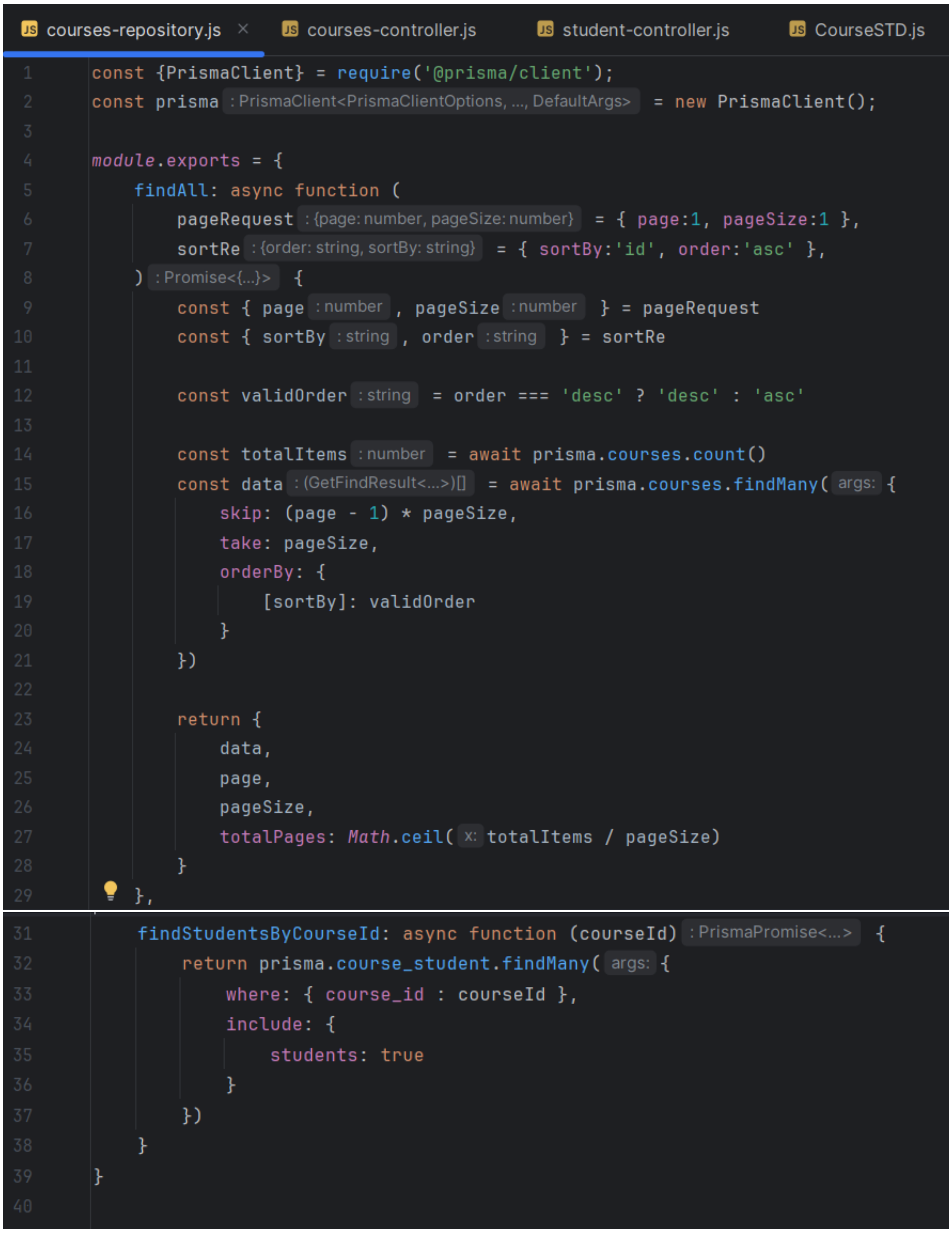
schema.prisma



```
generator client {
  provider = "prisma-client"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

model course_student {
  course_id  Int      @db.UnsignedTinyInt
  student_id Int      @db.UnsignedTinyInt
  grade      Float
  courses    courses  @relation(fields: [course_id], references: [id], onDelete: NoAction, onUpdate: NoAction, map: "course_student_courses_id_fk")
  students   students @relation(fields: [student_id], references: [id], onDelete: NoAction, onUpdate: NoAction, map: "course_student_students_id_fk"

  @@id([course_id, student_id])
  @@index([course_id], map: "idx_fk_course_id")
  @@index([student_id], map: "idx_fk_student_id")
}

model courses {
  id             Int              @id @unique @default(autoincrement()) @db.UnsignedTinyInt
  subject_id     Int
  year           Int              @db.Year
  course_student course_student[]

  @@index([subject_id], map: "idx_fk_subject_id")
}

```

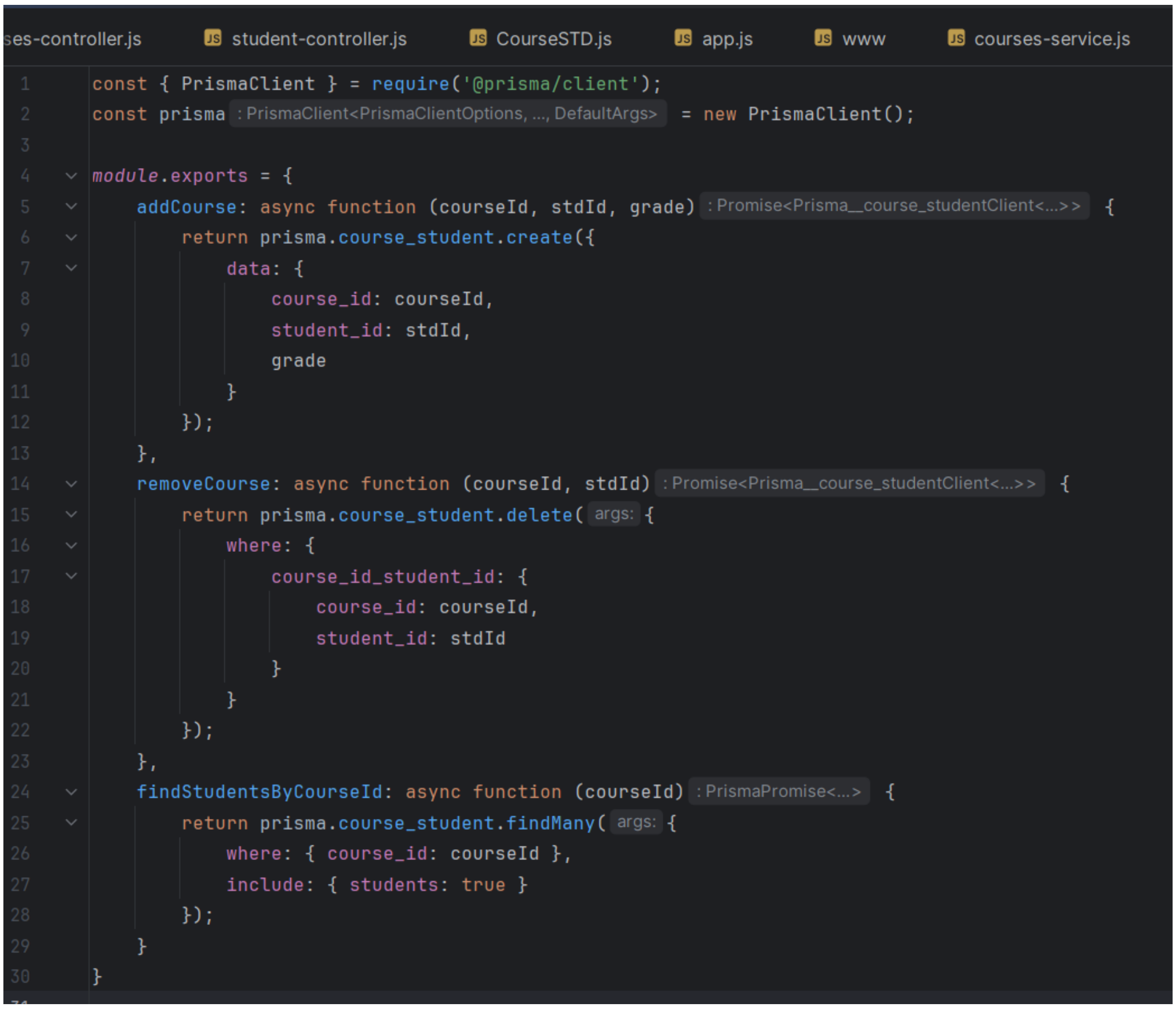
```

model subjects {
  id           Int    @id @unique @default(autoincrement()) @db.UnsignedTinyInt
  subject_code String @unique @db.Char(6)
  title        String @db.VarChar(40)
  credit       Float  @default(3)
}
```

Courses-repository.js



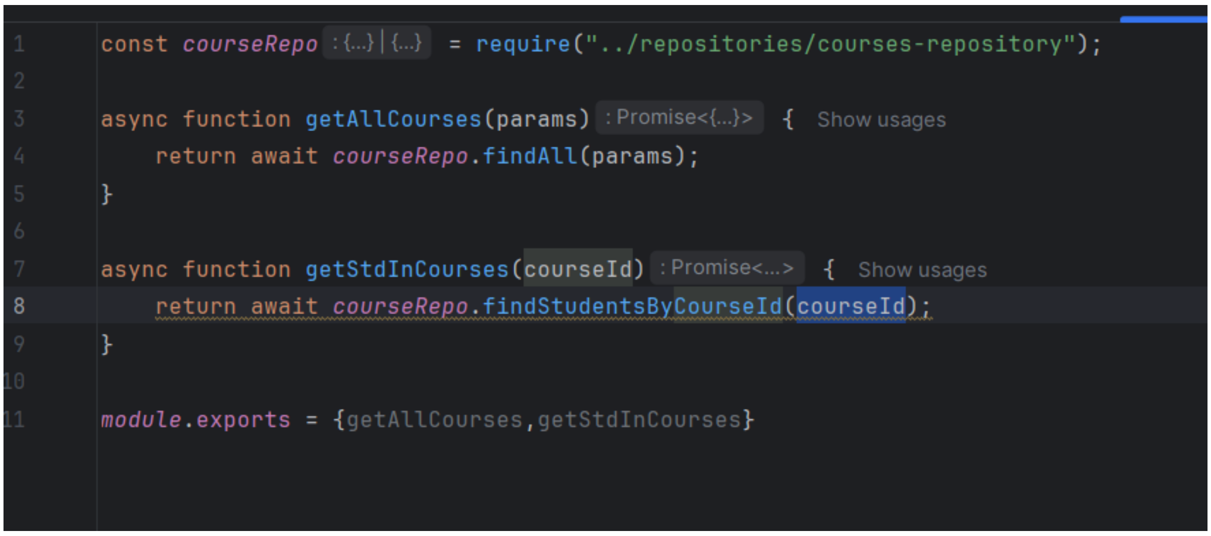
```
const {PrismaClient} = require('@prisma/client');
const prisma : PrismaClient<PrismaClientOptions, ..., DefaultArgs>  = new PrismaClient();

module.exports = {
    findAll: async function (
        pageRequest : {page: number, pageSize: number}  = { page:1, pageSize:1 },
        sortRe : {order: string, sortBy: string}  = { sortBy:'id', order:'asc' },
    ) : Promise<{...}>  {
        const { page : number , pageSize : number  } = pageRequest
        const { sortBy : string , order : string  } = sortRe

        const validOrder : string  = order === 'desc' ? 'desc' : 'asc'

        const totalItems : number  = await prisma.courses.count()
        const data : (GetFindResult<...>)[]  = await prisma.courses.findMany( args: {
            skip: (page - 1) * pageSize,
            take: pageSize,
            orderBy: {
                [sortBy]: validOrder
            }
        })

        return {
            data,
            page,
            pageSize,
            totalPages: Math.ceil( x: totalItems / pageSize)
        }
    },
    findStudentsByCourseId: async function (courseId) : PrismaPromise<...>  {
        return prisma.course_student.findMany( args: {
            where: { course_id : courseId },
            include: {
                students: true
            }
        })
    }
}
```

student-repository.js



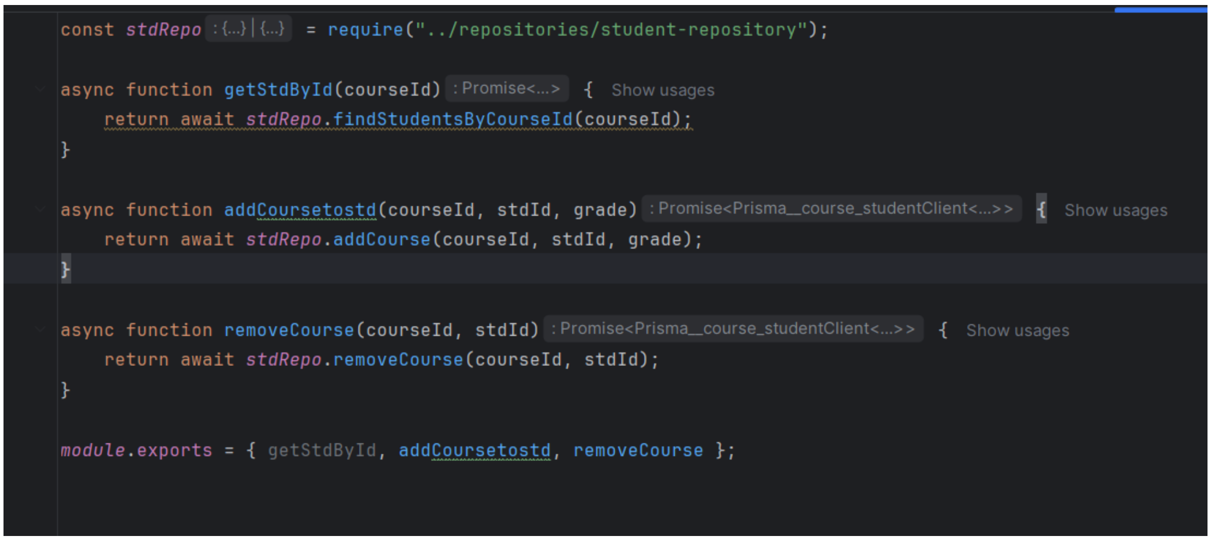
```javascript
const { PrismaClient } = require('@prisma/client');
const prisma : PrismaClient<PrismaClientOptions, ..., DefaultArgs> = new PrismaClient();

module.exports = {
    addCourse: async function (courseId, stdId, grade) : Promise<Prisma__course_studentClient<...>> {
        return prisma.course_student.create({
            data: {
                course_id: courseId,
                student_id: stdId,
                grade
            }
        });
    },
    removeCourse: async function (courseId, stdId) : Promise<Prisma__course_studentClient<...>> {
        return prisma.course_student.delete( args: {
            where: {
                course_id_student_id: {
                    course_id: courseId,
                    student_id: stdId
                }
            }
        });
    },
    findStudentsByCourseId: async function (courseId) : PrismaPromise<...> {
        return prisma.course_student.findMany( args: {
            where: { course_id: courseId },
            include: { students: true }
        });
    }
}
```

courses-services.js



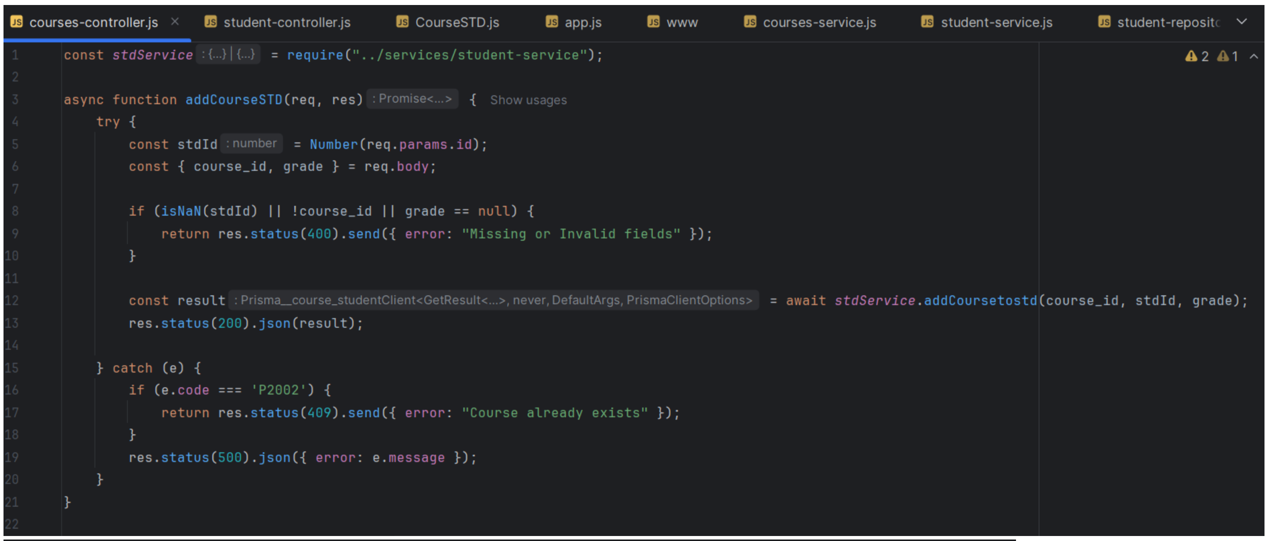
```
const courseRepo : {...}|{...} = require("../repositories/courses-repository");

async function getAllCourses(params) : Promise<{...}> {  Show usages
    return await courseRepo.findAll(params);
}

async function getStdInCourses(courseId) : Promise<...> {  Show usages
    return await courseRepo.findStudentsByCourseId(courseId);
}

module.exports = {getAllCourses,getStdInCourses}
```

students-services.js

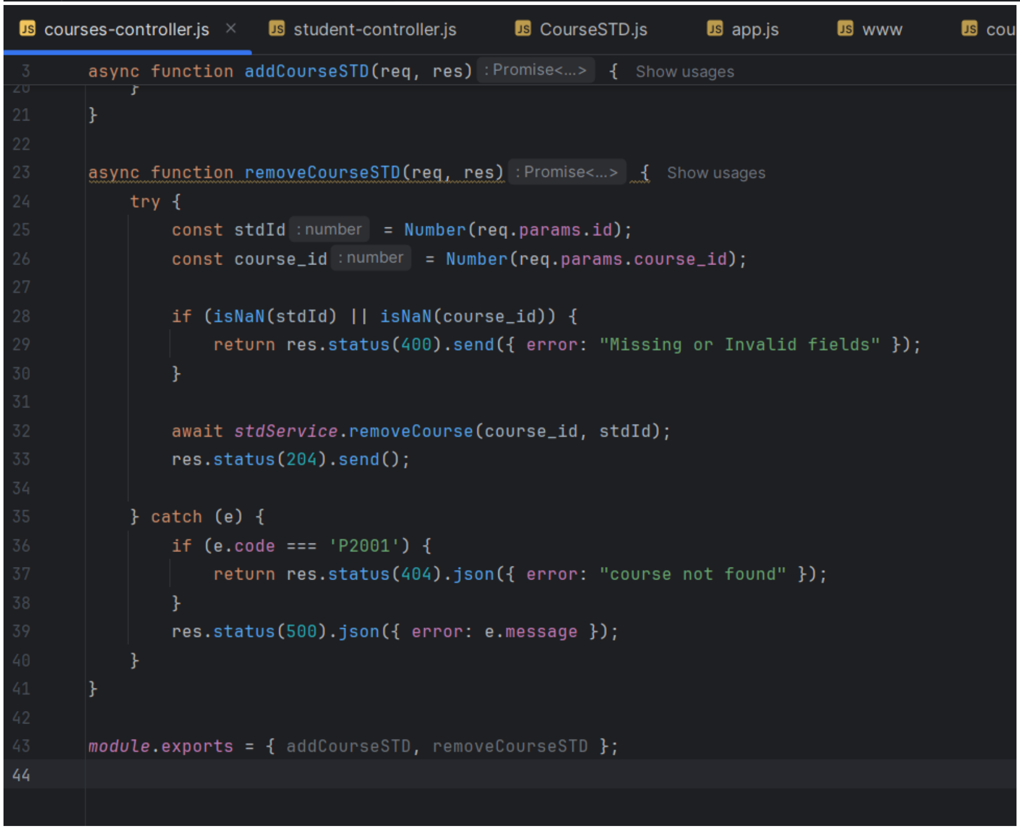


```
const stdRepo : {...}|{...} = require("../repositories/student-repository");

async function getStdById(courseId) : Promise<...> {  Show usages
    return await stdRepo.findStudentsByCourseId(courseId);
}

async function addCoursetostd(courseId, stdId, grade) : Promise<Prisma__course_studentClient<...>> {  Show usages
    return await stdRepo.addCourse(courseId, stdId, grade);
}

async function removeCourse(courseId, stdId) : Promise<Prisma__course_studentClient<...>> {  Show usages
    return await stdRepo.removeCourse(courseId, stdId);
}

module.exports = { getStdById, addCoursetostd, removeCourse };
```

courses-controller.js

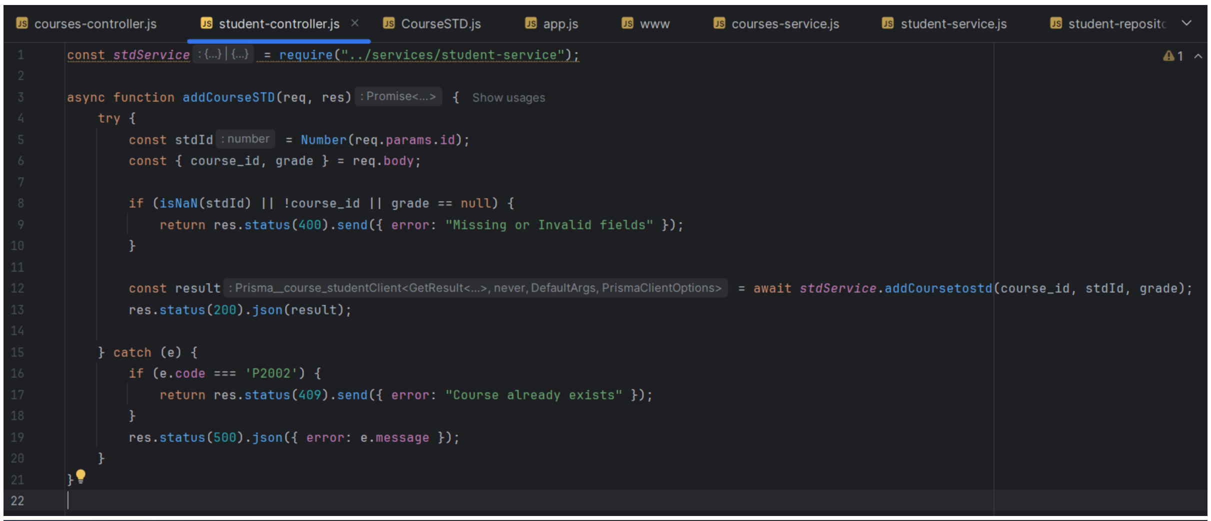


```
const stdService :{...}|{...} = require("../services/student-service");

async function addCourseSTD(req, res) :Promise<...> {  Show usages
    try {
        const stdId :number = Number(req.params.id);
        const { course_id, grade } = req.body;

        if (isNaN(stdId) || !course_id || grade == null) {
            return res.status(400).send({ error: "Missing or Invalid fields" });
        }

        const result :Prisma__course_studentClient<GetResult<...>, never, DefaultArgs, PrismaClientOptions> = await stdService.addCoursetostd(course_id, stdId, grade);
        res.status(200).json(result);

    } catch (e) {
        if (e.code === 'P2002') {
            return res.status(409).send({ error: "Course already exists" });
        }
        res.status(500).json({ error: e.message });
    }
}
```

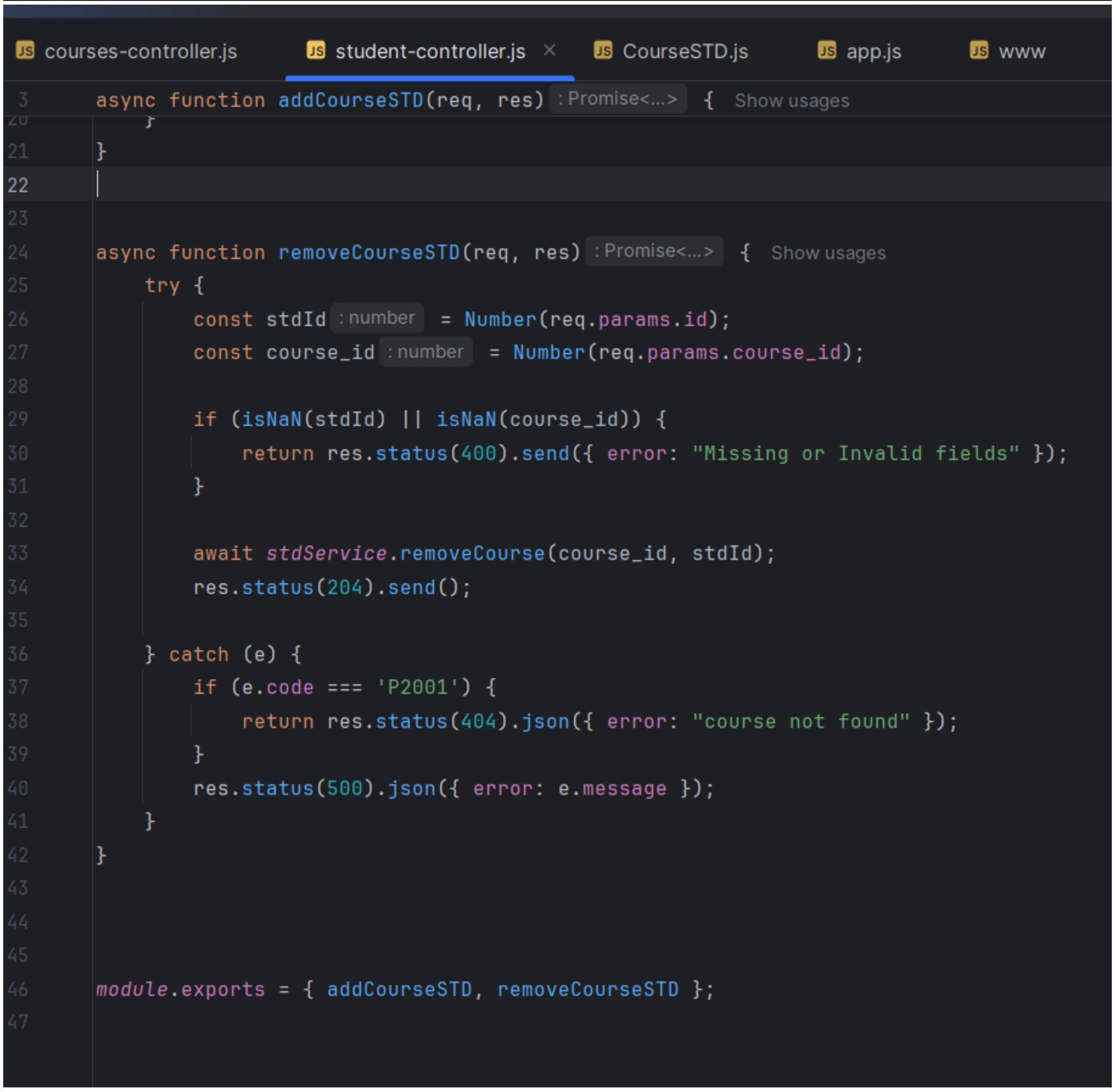


```
async function addCourseSTD(req, res) :Promise<...> {  Show usages
    }
}

async function removeCourseSTD(req, res) :Promise<...> {  Show usages
    try {
        const stdId :number = Number(req.params.id);
        const course_id :number = Number(req.params.course_id);

        if (isNaN(stdId) || isNaN(course_id)) {
            return res.status(400).send({ error: "Missing or Invalid fields" });
        }

        await stdService.removeCourse(course_id, stdId);
        res.status(204).send();

    } catch (e) {
        if (e.code === 'P2001') {
            return res.status(404).json({ error: "course not found" });
        }
        res.status(500).json({ error: e.message });
    }
}

module.exports = { addCourseSTD, removeCourseSTD };
```

student-controller.js

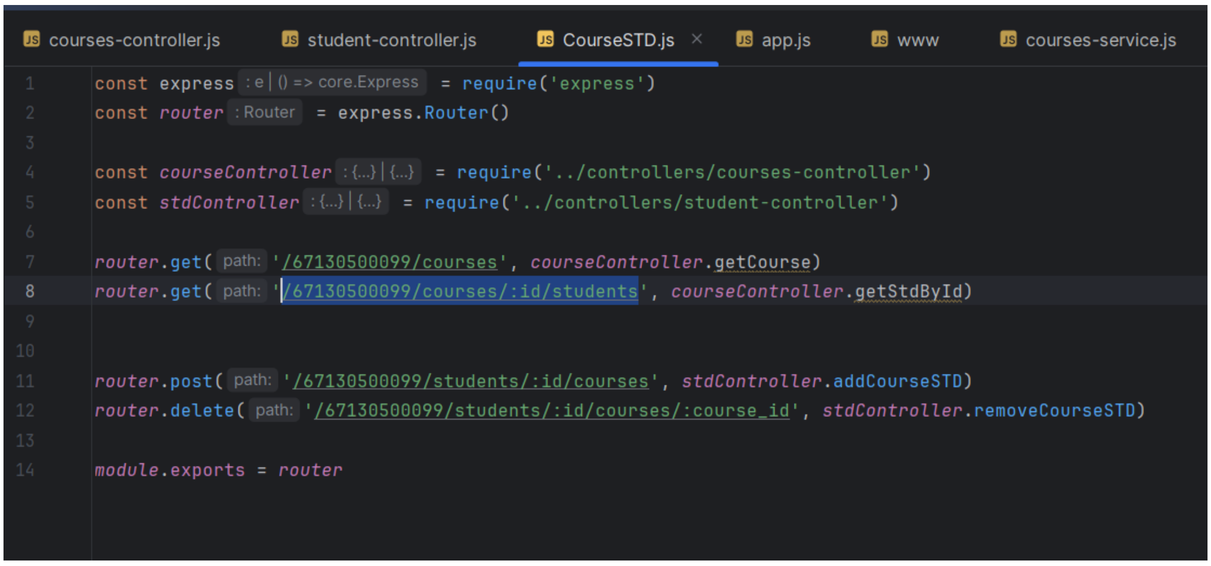


```
const stdService :{...}|{...} = require("../services/student-service");

async function addCourseSTD(req, res) :Promise<...> {  Show usages
    try {
        const stdId :number = Number(req.params.id);
        const { course_id, grade } = req.body;

        if (isNaN(stdId) || !course_id || grade == null) {
            return res.status(400).send({ error: "Missing or Invalid fields" });
        }

        const result :Prisma__course_studentClient<GetResult<...>, never, DefaultArgs, PrismaClientOptions> = await stdService.addCoursetostd(course_id, stdId, grade);
        res.status(200).json(result);

    } catch (e) {
        if (e.code === 'P2002') {
            return res.status(409).send({ error: "Course already exists" });
        }
        res.status(500).json({ error: e.message });
    }
}

```



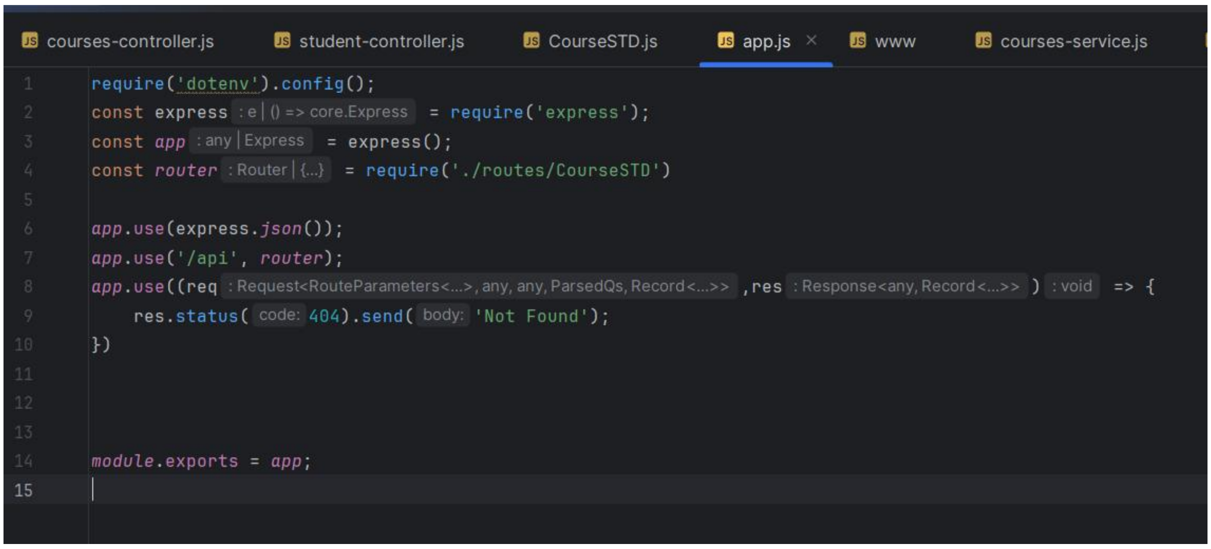
```
async function addCourseSTD(req, res) :Promise<...> {  Show usages
        }
}


async function removeCourseSTD(req, res) :Promise<...> {  Show usages
    try {
        const stdId :number = Number(req.params.id);
        const course_id :number = Number(req.params.course_id);

        if (isNaN(stdId) || isNaN(course_id)) {
            return res.status(400).send({ error: "Missing or Invalid fields" });
        }

        await stdService.removeCourse(course_id, stdId);
        res.status(204).send();

    } catch (e) {
        if (e.code === 'P2001') {
            return res.status(404).json({ error: "course not found" });
        }
        res.status(500).json({ error: e.message });
    }
}



module.exports = { addCourseSTD, removeCourseSTD };

```

Route/CourseSTD.js



```javascript
const express = require('express')
const router = express.Router()

const courseController = require('../controllers/courses-controller')
const stdController = require('../controllers/student-controller')

router.get('/67130500099/courses', courseController.getCourse)
router.get('/67130500099/courses/:id/students', courseController.getStdById)


router.post('/67130500099/students/:id/courses', stdController.addCourseSTD)
router.delete('/67130500099/students/:id/courses/:course_id', stdController.removeCourseSTD)

module.exports = router
```

app.js



```javascript
require('dotenv').config();
const express = require('express');
const app = express();
const router = require('./routes/CourseSTD')

app.use(express.json());
app.use('/api', router);
app.use((req ,res ) => {
    res.status(404).send('Not Found');
})



module.exports = app;
```