Student <-> Teacher Relationship
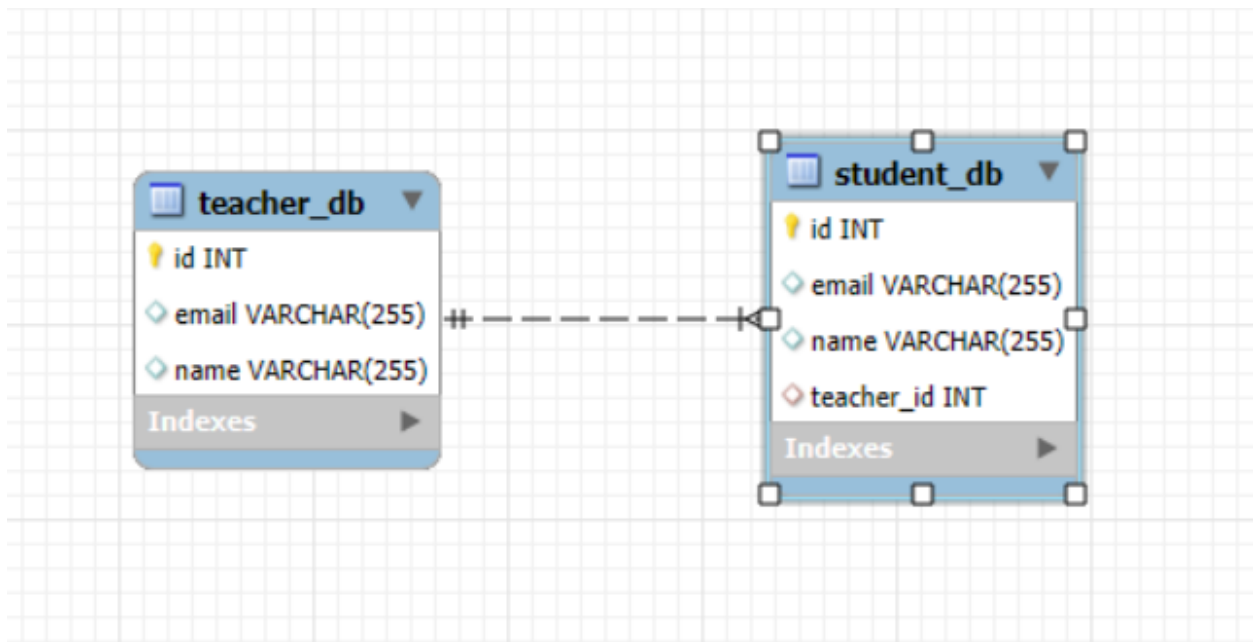
Controllers:

Student Controller


```java
public class StudentController {


    @Autowired
    private StudentService studentService;

    // Add a new student
    @PostMapping("/addStudent")
    public Student addStudent(@RequestBody Student student) {
        return studentService.saveDetails(student);
    }

    // Get all students
    @GetMapping("/all")
    public List<Student> getAllStudents() {
        return studentService.getAllStudents();
    }

    // Get students by teacher ID
    @GetMapping("/teacher/{teacherId}")
    public List<Student> getStudentsByTeacherId(@PathVariable int teacherId) {
        return studentService.getStudentsByTeacherId(teacherId);
    }

    // Update a student by ID
    @PutMapping("/update/{id}")
    public Student updateStudent(@PathVariable int id, @RequestBody Student student) {
        return studentService.updateStudent(id, student);
    }

    // Delete a student by ID
    @DeleteMapping("/delete/{id}")
    public String deleteStudent(@PathVariable int id) {
        return studentService.deleteStudent(id);
    }
}
```
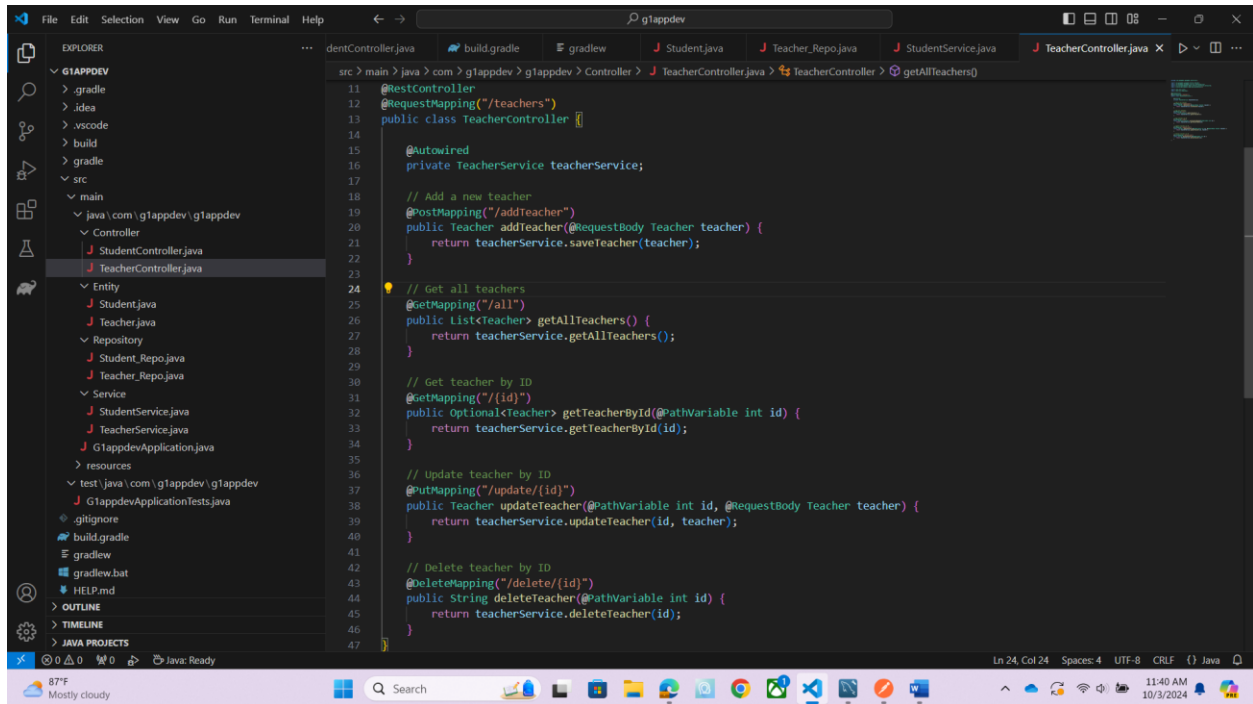
Teacher Controller


```java
@RestController
@RequestMapping("/teachers")
public class TeacherController {

    @Autowired
    private TeacherService teacherService;

    // Add a new teacher
    @PostMapping("/addTeacher")
    public Teacher addTeacher(@RequestBody Teacher teacher) {
        return teacherService.saveTeacher(teacher);
    }

    // Get all teachers
    @GetMapping("/all")
    public List<Teacher> getAllTeachers() {
        return teacherService.getAllTeachers();
    }

    // Get teacher by ID
    @GetMapping("/{id}")
    public Optional<Teacher> getTeacherById(@PathVariable int id) {
        return teacherService.getTeacherById(id);
    }

    // Update teacher by ID
    @PutMapping("/update/{id}")
    public Teacher updateTeacher(@PathVariable int id, @RequestBody Teacher teacher) {
        return teacherService.updateTeacher(id, teacher);
    }

    // Delete teacher by ID
    @DeleteMapping("/delete/{id}")
    public String deleteTeacher(@PathVariable int id) {
        return teacherService.deleteTeacher(id);
    }
}
```
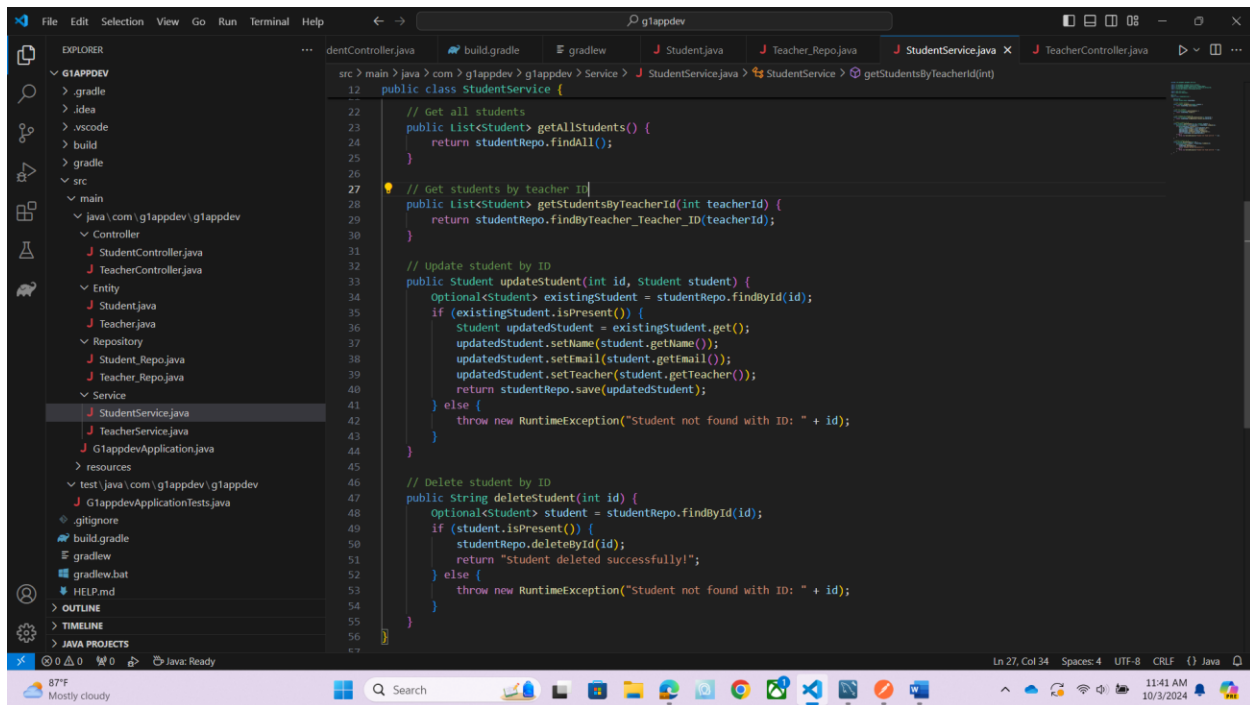
Services:

StudentService

```java
public class StudentService {

    // Get all students
    public List<Student> getAllStudents() {
        return studentRepo.findAll();
    }

    // Get students by teacher ID
    public List<Student> getStudentsByTeacherId(int teacherId) {
        return studentRepo.findByTeacher_Teacher_ID(teacherId);
    }

    // Update student by ID
    public Student updateStudent(int id, Student student) {
        Optional<Student> existingStudent = studentRepo.findById(id);
        if (existingStudent.isPresent()) {
            Student updatedStudent = existingStudent.get();
            updatedStudent.setName(student.getName());
            updatedStudent.setEmail(student.getEmail());
            updatedStudent.setTeacher(student.getTeacher());
            return studentRepo.save(updatedStudent);
        } else {
            throw new RuntimeException("Student not found with ID: " + id);
        }
    }

    // Delete student by ID
    public String deleteStudent(int id) {
        Optional<Student> student = studentRepo.findById(id);
        if (student.isPresent()) {
            studentRepo.deleteById(id);
            return "Student deleted successfully!";
        } else {
            throw new RuntimeException("Student not found with ID: " + id);
        }
    }
}
```

TeacherService


```java
public class TeacherService {

    // Get all teachers
    public List<Teacher> getAllTeachers() {
        return teacherRepo.findAll();
    }

    // Get teacher by ID
    public Optional<Teacher> getTeacherById(int id) {
        return teacherRepo.findById(id);
    }

    // Update teacher by ID
    public Teacher updateTeacher(int id, Teacher teacher) {
        Optional<Teacher> existingTeacher = teacherRepo.findById(id);
        if (existingTeacher.isPresent()) {
            Teacher updatedTeacher = existingTeacher.get();
            updatedTeacher.setName(teacher.getName());
            updatedTeacher.setEmail(teacher.getEmail());
            return teacherRepo.save(updatedTeacher);
        } else {
            throw new RuntimeException("Teacher not found with ID: " + id);
        }
    }

    // Delete teacher by ID
    public String deleteTeacher(int id) {
        Optional<Teacher> teacher = teacherRepo.findById(id);
        if (teacher.isPresent()) {
            teacherRepo.deleteById(id);
            return "Teacher deleted successfully!";
        } else {
            throw new RuntimeException("Teacher not found with ID: " + id);
        }
    }
}
```