Music School Course Management System

Bing Wen

Client Background

This project involves a small music school with a student population ranging between 300-500 and a part-time teaching staff of 10-30. The school offers instruction in various instruments, including piano, violin, cello, guitar, and drums, as well as composition. Courses are categorized into group lessons (2-6 students) and one-on-one private lessons.

Currently, the school manages courses manually using Excel, which is time-consuming and error prone. This course management task is crucial for the school's daily operations, and they are seeking to implement an IT system for more efficient management. The school previously attempted to use a SaaS system but abandoned it due to a lack of customization options and high costs for tailored features.

Project Objectives

The primary objective of this project is to design and implement a course management system tailored to the school's specific needs, with the capability to provide data to their financial system.

Requirements Overview

1. The current system is designed for single-user use, but it needs to be prepared for expansion to a multi-user system.

2. The system must be able to manage a variety of courses:

    1） Course Categories: Group lessons and one-on-one lessons; piano, violin, cello, guitar, drums, and composition; fixed-schedule recurring courses and on-demand booking courses.

    2） Weekly Summary Schedule: Ability to generate a weekly summary schedule.

    3） Course Management: Ability to add, modify, and cancel each class.

3. The system must be able to manage teachers and handle tasks such as teacher onboarding, leave, vacations, class cancellations, substitutions, and resignations.

4. The system must be able to manage students and handle tasks such as student registration, leave, class cancellations, and bookings.

5. The system must be able to manage classrooms.

6. The system must be able to export data for financial use.

Architecture

The system development will utilize the following technologies:

- **Frontend (React.js):**
  - Construct the user interface using React.js.
  - Communicate with the backend RESTful API via Axios.
- **Backend (Node.js):**
  - Build RESTful API using Node.js.
  - Handle frontend requests and perform business logic.
  - Interact with the MySQL database for data storage and retrieval.
- **Database (MySQL):**
  - Store application data.
  - Perform data insertion, updating, deletion, and querying through SQL.

Week 2 report

Tasks for This Week

1. Complete the setup of the development environment.
2. Learn React.js and Node.js.
3. Start functional design.
4. Start Database design.

Challenges This Week

1. **Limited Knowledge of React.js and Node.js**: Need to spend time reading documentation and watching tutorial videos to learn.
2. **Analysis of Original Requirements**: There are ambiguities in the definitions, which need to be addressed based on experience.

Plans for Next Week

1. Continue learning React.js and Node.js.
2. Complete the main functional design.
3. Complete the main database design.
4. Start coding.

Week 3 report

Tasks for This Week

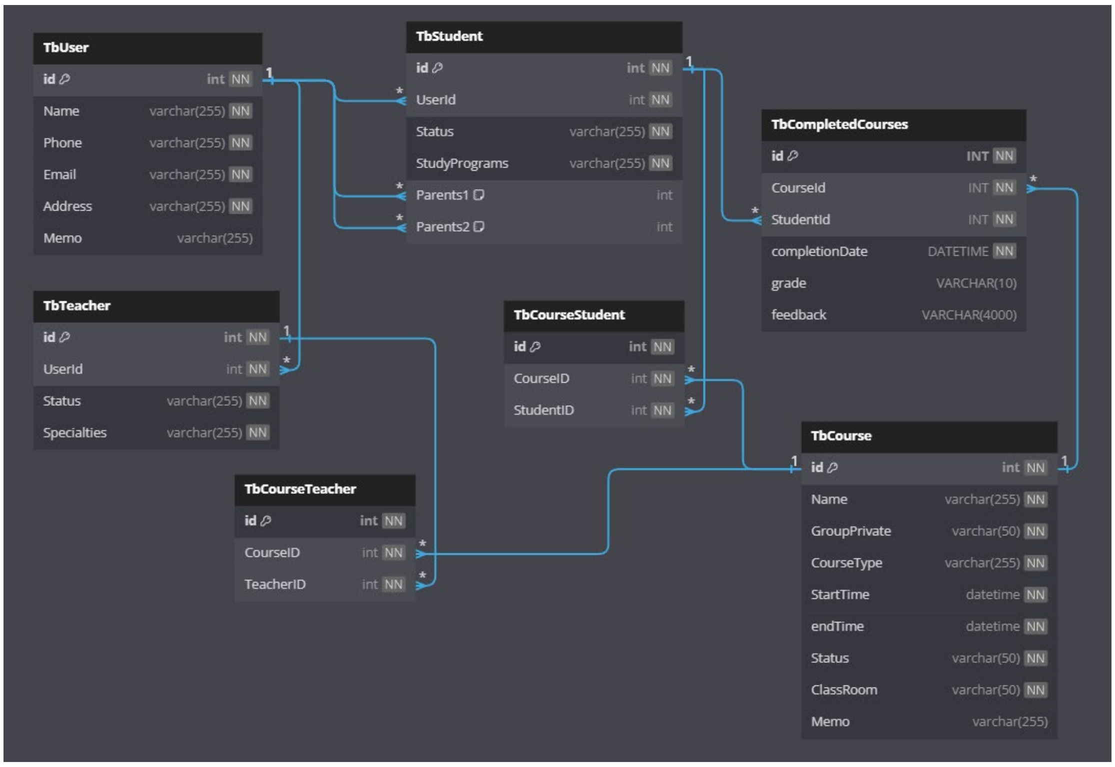
- According to Sapna's suggestion, I will first focus on database design, then proceed with backend development, and finally work on frontend development. Therefore, I have adjusted my plan. After discussing with Sapna this week, I have basically completed the design of the main database tables.Learn React.js and Node.js.

Challenges This Week

- There were some confusions in the design of the database abstraction for the requirements, but they have been corrected after discussion.

Plans for Next Week

- Developing backend code while learning Node.js and related technologies.
- The remaining details of the database design will be refined during the development process.



Week 4 report

Tasks for This Week

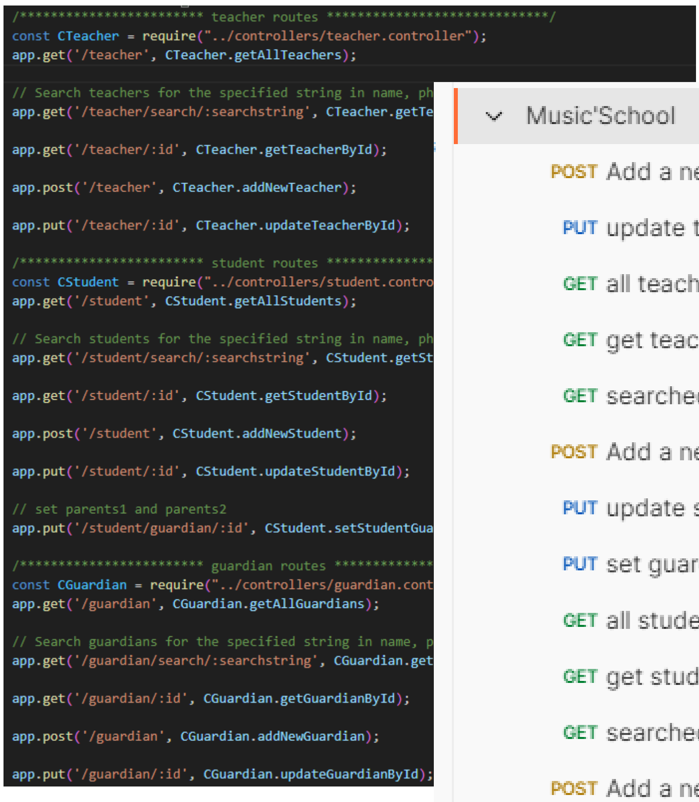
- Completed part of the backend code, primarily the modules for teachers, students, and guardians. And complete the corresponding test scripts in Postman.

Challenges This Week

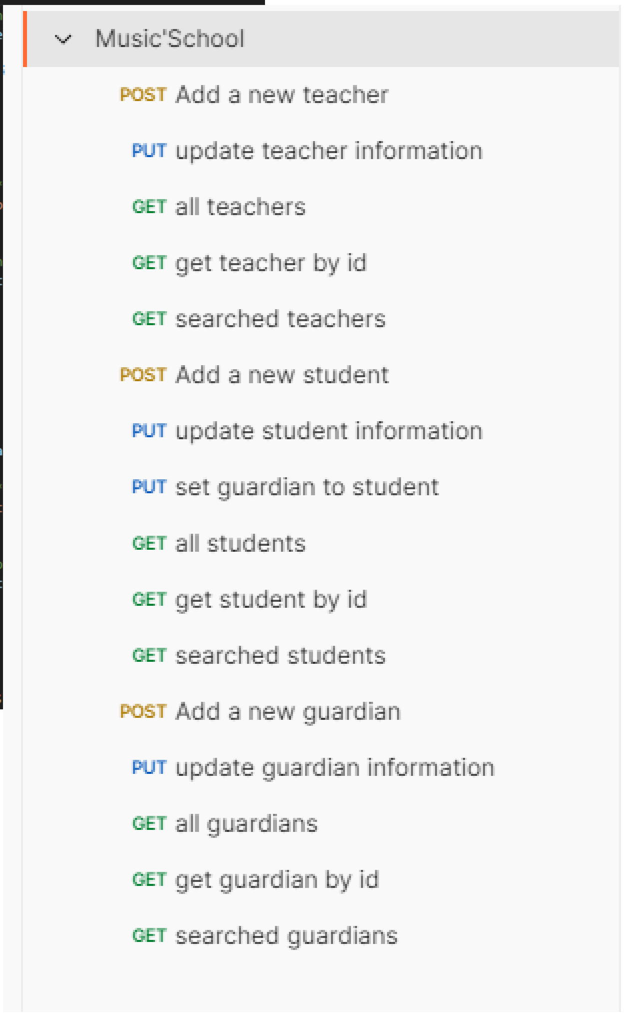
- Develop while getting familiar with Node.js technology.

Plans for Next Week

- Next week, I plan to finish the coding sections for courses and completed courses.
- And complete the automatic generation function for long-term courses.
- Develop database stored procedures to automate daily course settlements.



```
/*********************** teacher routes *****************************/
const CTeacher = require("../controllers/teacher.controller");
app.get('/teacher', CTeacher.getAllTeachers);

// Search teachers for the specified string in name, ph
app.get('/teacher/search/:searchstring', CTeacher.getTe

app.get('/teacher/:id', CTeacher.getTeacherById);

app.post('/teacher', CTeacher.addNewTeacher);

app.put('/teacher/:id', CTeacher.updateTeacherById);

/*********************** student routes *************
const CStudent = require("../controllers/student.contro
app.get('/student', CStudent.getAllStudents);

// Search students for the specified string in name, ph
app.get('/student/search/:searchstring', CStudent.getSt

app.get('/student/:id', CStudent.getStudentById);

app.post('/student', CStudent.addNewStudent);

app.put('/student/:id', CStudent.updateStudentById);

// set parents1 and parents2
app.put('/student/guardian/:id', CStudent.setStudentGua

/*********************** guardian routes ************
const CGuardian = require("../controllers/guardian.cont
app.get('/guardian', CGuardian.getAllGuardians);

// Search guardians for the specified string in name, p
app.get('/guardian/search/:searchstring', CGuardian.get

app.get('/guardian/:id', CGuardian.getGuardianById);

app.post('/guardian', CGuardian.addNewGuardian);

app.put('/guardian/:id', CGuardian.updateGuardianById);
```


Music'School
POST Add a new teacher
PUT update teacher information
GET all teachers
GET get teacher by id
GET searched teachers
POST Add a new student
PUT update student information
PUT set guardian to student
GET all students
GET get student by id
GET searched students
POST Add a new guardian
PUT update guardian information
GET all guardians
GET get guardian by id
GET searched guardians