Thanks for sharing all the details! Based on your backend and frontend code for the quiz functionality, I've corrected the integration so that:

1. **Tutor can submit options properly including `isCorrect`.**
2. **Student selection is tracked and submitted.**
3. **Backend calculates scores correctly based on `isCorrect` from options.**

Here's the **corrected code**:

---

# ✅ Frontend

## `quiz.service.ts`

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class QuizService {
  private baseUrl = 'http://localhost:3004/quiz';

  constructor(private http: HttpClient) {}

  createQuiz(quiz: any, questions: any): Observable<any> {
    return this.http.post(`${this.baseUrl}/newCreateQuiz`, { quizData: quiz, questions });
  }

  getQuizByCourseId(courseId: number): Observable<any> {
    return this.http.get(`${this.baseUrl}/getQuizByCourseId/${courseId}`);
  }

  submitQuiz(attemptData: any, answers: any[]): Observable<any> {
    return this.http.post(`${this.baseUrl}/submitQuiz`, {
      attemptData,
      answers
    });
  }

  getResultsByUserId(userId: number): Observable<any> {
    return this.http.get(`${this.baseUrl}/getResultByUserId/${userId}`);
```

```
  }
}
```

---

## get-quiz-by-course-id.component.ts

```
import { Component } from '@angular/core';
import { QuizService } from '../../services/quiz.service';

@Component({
  selector: 'app-get-quiz-by-course-id',
  templateUrl: './get-quiz-by-course-id.component.html',
  styleUrls: ['./get-quiz-by-course-id.component.css']
})
export class GetQuizByCourseIdComponent {
  courseId: number = 19;
  userId: number = 14;
  quiz: any;
  answers: any[] = [];
  submitted = false;
  score: number | null = null;

  constructor(private quizService: QuizService) {}

  fetchQuiz() {
    this.quizService.getQuizByCourseId(this.courseId).subscribe((data) => {
      if (data.length > 0) {
        this.quiz = data[0];
        this.submitted = false;
        this.answers = this.quiz.questions.map((q: any) => ({
          questionId: q.questionId,
          selectedOptionId: null
        }));
      }
    });
  }

  selectOption(questionId: number, optionId: number) {
    const answer = this.answers.find(a => a.questionId === questionId);
    if (answer) {
      answer.selectedOptionId = optionId;
    }
  }

  submitQuiz() {
    const attemptData = {
      userId: this.userId,
```

```
      quizId: this.quiz.quizId,
      attemptDate: new Date()
    };
    this.quizService.submitQuiz(attemptData, this.answers).subscribe(res => {
      this.score = res.score;
      this.submitted = true;
      this.getResults();
    });
  }

  getResults() {
    this.quizService.getResultsByUserId(this.userId).subscribe(results => {
      console.log('Your past results:', results);
    });
  }
}
```

---

## get-quiz-by-course-id.component.html

```
<div class="container p-4">
  <h2 class="text-xl font-bold mb-4">Take Quiz</h2>
  <label class="block mb-2 font-medium">Enter Course ID:</label>
  <input [(ngModel)]="courseId" type="number" class="border p-2 mb-4 w-full"
placeholder="e.g. 1" />
  <button (click)="fetchQuiz()" class="bg-blue-600 text-white px-4 py-2 rounded">Get
Quiz</button>

  <div *ngIf="quiz">
    <h3 class="text-lg font-semibold mt-4">{{ quiz.quizName }}</h3>
    <p>{{ quiz.description }}</p>
    <form *ngFor="let q of quiz.questions">
      <div class="mt-4">
        <p class="font-medium">{{ q.questionText }}</p>
        <div *ngFor="let opt of q.options">
          <input type="radio" [name]="'question_' + q.questionId" [value]="opt.optionId"
            (change)="selectOption(q.questionId, opt.optionId)" [disabled]="submitted" />
          <label>{{ opt.optionText }}</label>
        </div>
      </div>
    </form>
    <button *ngIf="!submitted" (click)="submitQuiz()" class="mt-4 bg-green-600 text-white
px-4 py-2 rounded">Submit Quiz</button>
    <div *ngIf="submitted" class="mt-4 text-green-700">
      <p>✅ Quiz submitted!</p>
      <p>Your score: {{ score }}/{{ quiz.questions.length }}</p>
    </div>
```

```
  </div>
</div>
```

---

## ✅ Backend - Fixes to `QuizService`

### Changes made:

- Corrected saving of options with `isCorrect`.
- Updated scoring logic to check `isCorrect` flag.
- Ensured relation loading works correctly.

```
async createQuiz(quizData: Partial<Quiz>, questions: any[]): Promise<Quiz> {
  const course = await courseRepository.findOne({ where: { courseId:
quizData.course?.courseId } });
  if (!course) throw new Error('Course not found');

  const quiz = QuizRepository.create({
    quizName: quizData.quizName,
    description: quizData.description,
    totalmarks: quizData.totalmarks,
    course: course
  });

  const savedQuiz = await QuizRepository.save(quiz);

  for (const q of questions) {
    const question = questionRepository.create({
      questionText: q.questionText,
      correctOptionId: q.correctOptionId,
      quiz: savedQuiz
    });

    const savedQuestion = await questionRepository.save(question);

    for (const opt of q.options) {
      const option = optionRepository.create({
        optionText: opt.optionText,
        isCorrect: opt.isCorrect || false,
        question: savedQuestion
      });
      await optionRepository.save(option);
    }
```

```
  }

  return savedQuiz;
}

async submitQuiz(
  attemptData: { userId: number; quizId: number },
  answers: { questionId: number; selectedOptionId: number }[]
): Promise<{ score: number }> {
  const { userId, quizId } = attemptData;

  let score = 0;

  for (const answer of answers) {
    const selectedOption = await optionRepository.findOne({
      where: {
        question: { questionId: answer.questionId },
        optionId: answer.selectedOptionId
      },
      relations: ['question']
    });

    if (selectedOption?.isCorrect) {
      score++;
    }
  }

  const attempt = QuizAttemptRepository.create({
    userId,
    quizId,
    score,
    attemptDate: new Date()
  });

  await QuizAttemptRepository.save(attempt);

  return { score };
}
```

---

Would you like me to zip the corrected frontend and backend folders and share a downloadable link now?