Based on your backend and files, here's the **complete frontend implementation** for the following flow:

🎯 Flow Overview:

1. **User selects a course** → fetch quiz questions using `getQuizByCourseId`.
2. **User attempts quiz** by selecting options.
3. **User submits quiz** → selected answers are sent to backend.
4. **User sees score/result** via `getResultsByUserId`.

✅ 1. `quiz.service.ts` (Angular Service)

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

@Injectable({
  providedIn: 'root'
})
export class QuizService {
  private baseUrl = 'http://localhost:3003/quiz';

  constructor(private http: HttpClient) {}

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

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

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

✅ 2. `quiz-attempt.component.ts`

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

@Component({
  selector: 'app-quiz-attempt',
  templateUrl: './quiz-attempt.component.html',
  styleUrls: ['./quiz-attempt.component.css']
})
export class QuizAttemptComponent {
  courseId!: number;
  userId: number = 1; // replace with actual user ID logic
  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
```

```
    };

    this.quizService.submitQuiz(attemptData, this.answers).subscribe(res => {
      this.score = res.score;
      this.submitted = true;
    });
  }

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

✅ 3. `quiz-attempt.component.html`

```
<div class="container p-4">
  <h2 class="text-xl font-bold mb-4">Take Quiz</h2>

  <label for="courseId" 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>
```

✅ 4. Notes on Enhancements

- If you want to select course via dropdown, you can add another service method to fetch all courses and bind to `<select>`.
- Replace hardcoded `userId` with the logged-in user's ID (from session or token).

If you'd like, I can also include:

- A **quiz history table** for results.
- A **course dropdown integration** instead of manual ID.

Want that too?