Node.js Assignment

Introduction

This assignment focuses on creating RESTful Web services using Node.js and the popular Express library as well as integrating a React front end to interact with a set of Web APIs. The assignment will add quizzes to the WhiteBoard application we've been working througout the semester. A quizzes React component will pull a set of quizzes from a Node.js server and render list of quizzes. Selecting a quiz will navigate to a React quiz component which will pull a set of questions for the quiz from a Node.js server. The React quiz component will render the quiz and grade it.

Features

The features implement it in this assignment are

- Create RESTful Web services with Node.js and Express
- Create Node.js servers
- Integrate React UI with a Node.js server

Create Node.js Express Web Services

Create a Quiz Service

In file `/services/quiz-service.js`, create a JavaScript Web service that retrieves all the quizzes as well a quiz for a given quiz ID. Here's an example on how the service might look like

`/services/quiz-service.js`

```
let quizzes = require('./quizzes.json')        // quizzes declared in separate file
const findAllQuizzes = () => quizzes           // retrieve all quizzes
const findQuizById = (quizId) =>               // retrieve specific quiz by its ID
    quizzes.find(quiz => quiz._id === quizId)
module.exports = {                             // export functions for controller
    findAllQuizzes,
    findQuizById
}
```

In file `/services/quizzes.json`, create a JSON file that contains an array of quizzes. Use the following content for the JSON file. This is the JSON file the service requires above

`/services/quizzes.json`

```
[
  {"_id": "123", "title": "Quiz 1"},
  {"_id": "234", "title": "Quiz 2"},
```

```
  {"_id": "345", "title": "Quiz 3"}
]
```

Create Quiz RESTful Web Service Controller

In file `/controllers/quizzes-controller.js`, create a RESTful Web service controller that defines HTTP GET endpoints and uses the quiz service to retrieve quizzes for a given quiz ID as well as all the quizzes. The controller should implement the following endpoints

Method	URL	Description
GET	/api/quizzes	Retrieves all the quizzes
GET	/api/quizzes/:qid	Retrieves a quiz whose _id is qid

Here's an example of what the controller might look like

/controllers/quizzes-controller.js

```
const quizService                                    // require service
    = require('../services/quiz-service)
module.exports = (app) => {                          // accept app, express
                                                     // instance
    const findAllQuizzes = (req, res) =>             // use service to retrieve all
        res.send(quizService.findAllQuizzes())       // quizzes
    const findQuizById = (req, res) => {             // use service to retrieve
        const quizId = req.params['qid']             // single quiz by its _id
        const quiz = quizService.findQuizById(quizId)
        res.json(quiz)
    }

    app.get('/api/quizzes', findAllQuizzes)          // create GET HTTP endpoints
    app.get('/api/quizzes/:qid', findQuizById)       // to retrieve quizzes
}
```

Create a Question Service

In file `/services/questions-service.js`, create a JavaScript Web service that retrieves the questions for a given quiz. Here's an example on how the service might look like

/services/questions-service.js

```
let questions = require('./questions.json')
findQuestionsForQuiz = (quizId) =>
    questions.filter(question => question.quizId === quizId)
module.exports = {
    findQuestionsForQuiz
}
```

In file `/services/questions.json`, create a JSON file that contains an array of questions. Use the following content for the JSON file

`/services/questions.json`

```
[
    {"_id": "123", "title": "JPA True False", "quizId": "234", "question":
      "JPA stands for Java Persistence API", "correct": "true", "type": "TRUE_FALSE"},
    {"_id": "234", "title": "JPA Multiple Choice", "quizId": "234", "question":
      "What does JPA stand for?", "correct": "Java Persistence API",
      "type": "MULTIPLE_CHOICE", "choices":
      ["Java Pseudo API", "Java Persistence API", "JSON Persistence API",
      "JavaScript Persistence API"]},
    {"_id": "345", "title": "JSON True False", "quizId": "234", "question":
      "JSON stands for Java Object Notation", "correct": "false","type": "TRUE_FALSE"},
    {"_id": "456", "title": "JSON Multiple Choice", "quizId": "345", "question":
      "What does JSON stand for?", "correct": "JavaScript Object Notation",
      "type": "MULTIPLE_CHOICE", "choices": ["JavaScript Object Notation",
      "JavaScript Object Normalization", "Java Object Normalization", "Java Object
      Notation"]},
    {"_id": "567", "title": "CSS True False", "quizId": "345", "question":
      "CSS stands for Cascading Style Sheet", "correct": "true", "type": "TRUE_FALSE"},
    {"_id": "678", "title": "CSS Multiple Choice", "quizId": "123", "question":
      "What does CSS stand for?", "correct": "Cascading Style Sheet", "type":
      "MULTIPLE_CHOICE", "choices": ["Class Style Sheet", "Cascading Screen Style",
      "Cascading Style Sheet", "Cascading Style Screen"]}
]
```

Create Question RESTful Web Service Controller

In file `/controllers/question-controller.js`, create a RESTful Web service controller that defines an HTTP GET endpoint that uses the quesion service to retrieve questions for a given quiz ID. The controller should implement the following endpoints

Method	URL	Description
GET	/api/quizzes/:qid/questions	Retrieves the questions for a quiz whose _id is qid

Here's an example of what the controller might look like

`/controllers/question-controller.js`

```
const questionService
    = require("../services/questions-service")
module.exports = (app) => {
    const findQuestionsForQuiz = (req, res) => {
        const quizId = req.params['qid']
```

```
        const questions
            = questionService.findQuestionsForQuiz(quizId)
        res.json(questions)
    }
    app.get("/api/quizzes/:qid/questions",
        findQuestionsForQuiz)
}
```

Create a Node.js Server

In file `server.js`, create a Node.js server that exposes the quiz and question RESTful controllers and configures CORS to allow JavaScript clients to connect to the server from other domains

server.js

```
var express = require('express')
var app = express()

app.use(function (req, res, next) {
    res.header('Access-Control-Allow-Origin', '*');
    res.header('Access-Control-Allow-Headers',
        'Content-Type, X-Requested-With, Origin');
    res.header('Access-Control-Allow-Methods',
        'GET, POST, PUT, PATCH, DELETE, OPTIONS');
    next();
});

require('./controllers/quizzes-controller)(app)
require('./controllers/question-controller)(app)

app.listen(3000)
```

Create React RESTful Web Service Clients

Create React Web service clients to integrate a React application with the Node.js server created in later sections. Put all services under a common directory called services.

Create Quiz Web Service Client

In file `/services/quiz-service.js`, create a React Web service client. The client should be able to retrieve all the quizzes as well as a quiz for a given ID. Here's what the service might look like.

/services/quiz-service.js

```
const QUIZZES_URL = 'http://localhost:3000/api/quizzes';
const findAllQuizzes = () => {
   return fetch(QUIZZES_URL)
      .then(response => response.json())
}
const findQuizById = (qid) => {
```

```
   return fetch(`${QUIZZES_URL}/${qid}`)
      .then(response => response.json())
}
export default {
   findAllQuizzes, findQuizById
}
```

Create Question Web Service Client

In file `/services/question-service.js`, create a React Web service client. The client should be able to retrieve all the quesions for a given quiz ID. Here's what the service might look like

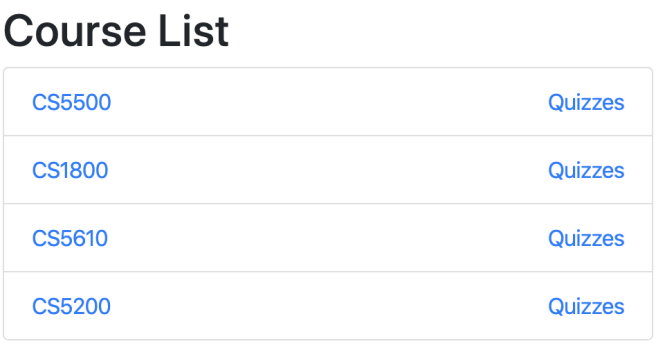
```
/services/question-service.js
```

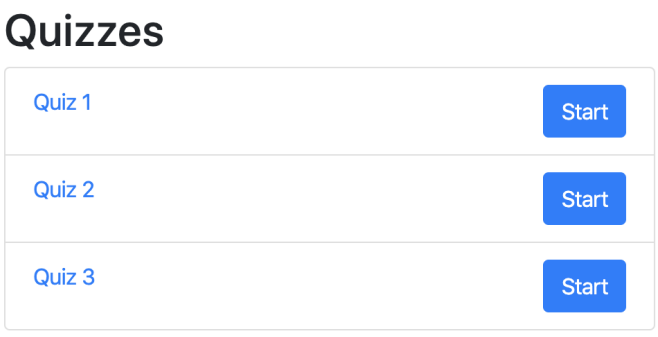
```
const QUIZZES_URL = 'http://localhost:3000/api/quizzes';
const findQuestionsForQuiz = (qid) => {
   return fetch(`${QUIZZES_URL}/${qid}/questions`)
      .then(response => response.json())
}
export default {
   findQuestionsForQuiz
}
```

Create a React User Interface

Create Quizzes Component

Create a component called quizzes that renders a list of all the quizzes. Use the wireframes shown below as a guideline.





Map the quizzes component to the following URL pattern

courses/:courseId/quizzes

In the course list component created in earlier assignments, add a link to the new quizzes component. Here's an example of how the link might be created. Note that the actual name of the folders and component files might differ for you from the ones shown here.

/src/components/course-list.js

```
    <Link className="float-right"
       to={`/courses/${course._id}/quizzes`}>
      Quizzes
    </Link>
```

In the quizzes component, use a service to retrieve all the quizzes and bind them to a local quizzes array. Iterate over the quizzes and render them as a list. Link the quiz title and a button to quiz component created in a later section

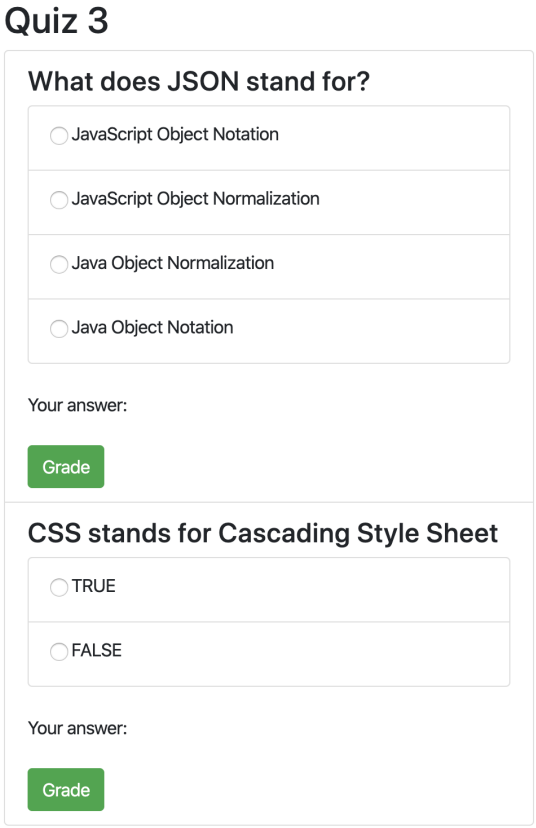
Create an Quiz Component

Create a React component called quiz that renders a quiz as a list of questions. Map the new quiz component to the following route:

courses/:courseId/quizzes/:quizId

In the quiz component, use the question service to retrieve all the quesions for the current quiz and bind them to a local questions array.

Iterate over the questions and render them as a list. Depending on the type of question, render either a true false question or multiple choice question, both components implemented in a later section of this assignment.



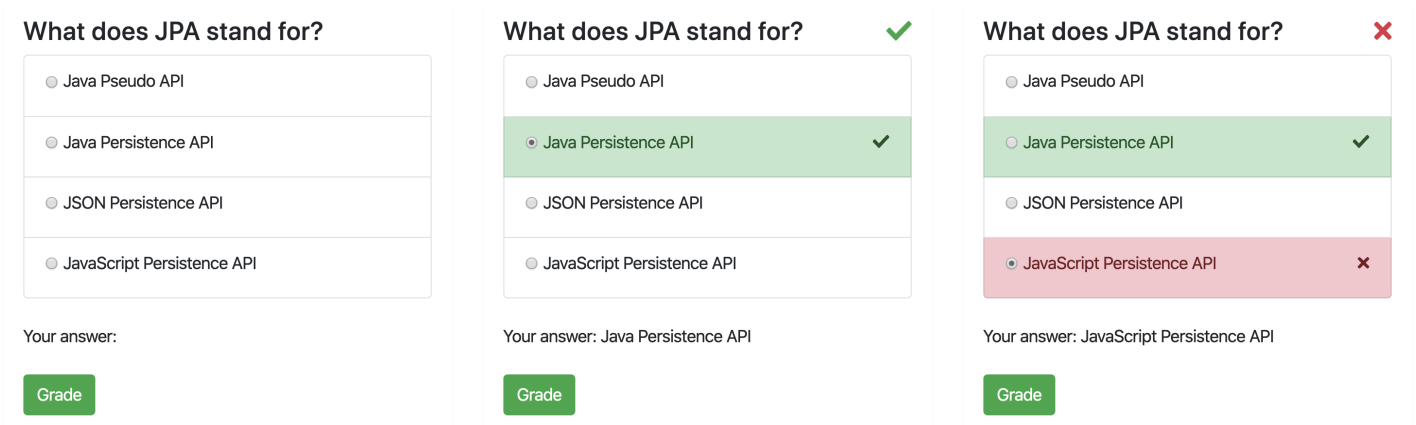
Create a True False Question Component

Create a component called true-false-question that renders a True/False question. Use the wireframes below as a guideline

The component should receive a question object as an input and use the object to render the question as shown in the wireframes above. The question should be rendered at the top of the component, then radio buttons representing the true and false choices, the answer selected, and then a grade button. The radio buttons should be mutually exclusive and independent from other questions. This can be achieved by using the unique ID of the question as the name for the radio buttons. When the user makes a selection, the answer is shown. When the user clicks on the grade button, the correct answer is highlighted green and, if selected, the wrong answer is highlighted red. A checkmark is rendered next to the question if it was answered correctly and an X is rendered otherwise. Use the wireframes as a guideline.

Create a Multiple Choice Question

Create a component called multiple-choice-question that renders a multiple choice question. Use the wireframes below as a guideline



The component should receive a question object as an input and use the object to render the question as shown in the wireframes above. The question should be rendered at the top of the component, then a list of radio buttons representing each of the choices, then the answer selected, and then a grade button. The radio buttons should be mutually exclusive and independent from other questions. This can be achieved by using the unique ID of the question as the name for the radio buttons. When the user makes a selection, the answer is shown. When the user clicks on the grade button, the correct answer is highlighted green and, if selected, the wrong answer is highlighted red. A checkmark is rendered next to the question if it was answered correctly and an X is rendered otherwise. Use the wireframes as a guideline.

Deliverables

As a deliverable, provide a link to the github repository for the Node.js server and React application. TAs will clone and build your project to run it locally.