

CS472 Final Exam

Professor Rujuan Xing

Based on the code below, when user types <http://localhost:3000/users> in browser bar, then hit “Enter”, what’s the output in the console/terminal? Assume all other parts of the application are correct except the code listed below.

```
app.get('/users', (req, res, next) => {
  console.log(1);
  next();
}, (req, res, next) => {
  console.log(2);
  next('route');
}, (req, res, next) => {
  console.log(3);
  next('something');
});

app.get('/users', (req, res, next) => {
  console.log(4);
  res.status(200).send('Success!');
});

app.use((err, req, res, next) => {
  console.log(5);
  res.status(500).send('Try later');
});
```

(4 points) Read the code below and describe what happens in the terminal when run “node app.js”. If no error, list the outputs in the terminal in correct order. If there’s error, describe the error and what cause the error.

NOTE: Inside helloworld folder, there are app.js and mypackage folder. hello.js, index.js and package.json are inside mypackage folder.

<pre>package.json { "name": "mypackage", "version": "1.0.0", "main": "hello.js" }</pre>	
<pre>app.js const person = require('./mypackage'); person.getName();</pre>	
<pre>index.js class Person { constructor(name = 'John') { this.name = name; } getName() { console.log(this.name); } } exports.person = new Person('Bella');</pre>	<pre>hello.js class Person { constructor(name = 'John') { this.name = name; } getName() { console.log(this.name); } } exports.person = new Person('Edward');</pre>

(3 points) Read the code below and describe what will happen when run “node app.js”? If no error, list the outputs in the terminal with correct order. If there’s error, describe the error and what cause the error.

```
app.js
console.log('start');
const result = new Promise(resolve, reject) => {
  console.log(1);
  setTimeout(() => reject('Whoops!'), 1000);
  console.log(2);
})
.then(res => res)
.catch(error => 'Error Happens');

console.log(result);
console.log('end');
```

1. **(14 points)** Use Node.js, Express.js and other necessary JS APIs to implement a RESTful Application which manages students and their courses.

All students are stored in in-memory database, the below is an example:

```
[
  {
    "studentId": "610001",
    "firstname": "John",
    "lastname": "Smith",
    "courses": [
      {
        "courseId": "CS303",
        "coursename": "Web Application Programming in JavaScript",
        "semester": "2021-Spring",
        "grade": 99
      },
      {
        "courseId": "CS445",
        "coursename": "Modern Asynchronous Programming",
        "semester": "2021-Spring",
        "grade": 95
      }
    ]
  },
  {
    "studentId": "610002",
    "firstname": "Edward",
    "lastname": "Jack",
    "courses": [
      {
        "courseId": "CS472",
        "coursename": "Web Application Programming",
        "semester": "2021-Spring",
        "grade": 82
      },
      {
        "courseId": "CS544",
        "coursename": "Web Application Architecture",
        "semester": "2021-Spring",
        "grade": 88
      }
    ]
  }
]
```

You need to implement 2 APIs for this programming question:

- 1) **[7 points]** POST <http://localhost:3000/students/{studentId}/courses> - this API adds a single course to a particular student. For this API, you need to check if student exists, if not exist, throw error with message "Student doesn't exist with ID: {passed in ID}". If student exists, we add the course into student's course list. The course id is manually assigned in Request body, NOT automatically generated on the server side. When save a course, need to check if the course id exists in database. If doesn't exist, save the course directly into database. If already exists in database, replace the existing one with the course sent via request body. In a nutshell, a single course can appear only once in a particular student's course list, and a single course can appear in different students' course list.

Example: Suppose currently the database looks like:

```
[
  {
    "studentId": "610001",
    "firstname": "John",
    "lastname": "Smith",
    "courses": [
      {
        "courseId": "CS303",
        "coursename": "Web Application Programming in JavaScript",
        "semester": "2021-Spring",
        "grade": 99
      },
      {
        "courseId": "CS445",
        "coursename": "Modern Asynchronous Programming",
        "semester": "2021-Spring",
        "grade": 95
      }
    ]
  },
  {
    "studentId": "610002",
    "firstname": "Edward",
    "lastname": "Jack",
    "courses": [
      {
        "courseId": "CS472",
        "coursename": "Web Application Programming",
        "semester": "2021-Spring",
        "grade": 82
      }
    ]
  }
]
```

```

    },
    {
      "courseId": "CS544",
      "coursename": "Web Application Architecture",
      "semester": "2021-Spring",
      "grade": 88
    }
  ]
}
]

```

NOW, make a request with information below:

a) Case 1: student Id isn't found in database

Request: POST <http://localhost:3000/students/619999/courses>

Request Body:

```

{
  "courseId": "CS477",
  "coursename": "Server Side Programming",
  "semester": "2021-Fall",
  "grade": 90
}

```

Response body:

```

{
  "error": "Student doesn't exist with ID: 619999"
}

```

b) Case 2: student id is found in database

Request: POST <http://localhost:3000/students/610001/courses>

Request Body:

```

{
  "courseId": "CS477",
  "coursename": "Server Side Programming",
  "semester": "2021-Fall",
  "grade": 90
}

```

Response body:

```

{
  "studentId": "610001",
  "firstname": "John",
  "lastname": "Smith",
}

```

```

    "courses": [{
      "courseId": "CS303",
      "courseName": "Web Application Programming in JavaScript",
      "semester": "2021-Spring",
      "grade": 99
    },
    {
      "courseId": "CS445",
      "courseName": "Modern Asynchronous Programming",
      "semester": "2021-Spring",
      "grade": 95
    },
    {
      "courseId": "CS477",
      "courseName": "Server Side Programming",
      "semester": "2021-Fall",
      "grade": 90
    }
  ]
}

```

- 2) **[7 points]** GET <http://localhost:3000/students/{studentId}/getAverage> - this API returns the average grade of a student with given student Id. This API can optionally add a query string which used to filter based on semester like this:
<http://localhost:3000/students/{studentId}/getAverage?semester=2021-Spring>

Suppose currently for student 610001, he has 3 courses like below:

```

{
  "studentId": "610001",
  "firstname": "John",
  "lastname": "Smith",
  "courses": [
    {
      "courseId": "CS303",
      "courseName": "Web Application Programming in JavaScript",
      "semester": "2021-Spring",
      "grade": 99
    },
    {
      "courseId": "CS445",
      "courseName": "Modern Asynchronous Programming",
      "semester": "2021-Spring",
      "grade": 95
    }
  ]
}

```

```

    },
    {
      "courseId": "CS477",
      "coursename": "Server Side Programming",
      "semester": "2021-Fall",
      "grade": 90
    }
  ]
}

```

a) Case 1:

Request: GET <http://localhost:3000/students/610001/getAverage> - it returns the average of all courses for student 610001. Based on the sample data above, it returns the average of 3 courses (CS303, CS445, CS477) which is $(99+95+90)/3 = 94.66666666666667$

Response JSON:

```

{
  "average": 94.66666666666667
}

```

b) Case 2:

Request: GET <http://localhost:3000/students/610001/getAverage?semester=2021-Spring> – it returns the average grade of courses under “2021-Spring” semester belongs to student 610001. Based on the sample data above, it returns the average of 2 courses (CS303, CS445) which is $(99+95)/2 = 97$

Response JSON:

```

{
  "average": 97
}

```

c) Case 3

Request: GET <http://localhost:3000/students/610001/getAverage?semester=2021-Fall> – it returns the average grade of courses under “2021-Fall” semester. Based on the sample data above, it returns the average of 1 course (CS477) which is 90.

Response JSON:

```

{
  "average": 90
}

```


d) Case 4

Request: GET <http://localhost:3000/students/610001/getAverage?semester=2099-Fall> – There's no course under "2099-Fall" semester for this student. Return message "couldn't find semester 2099-Fall"

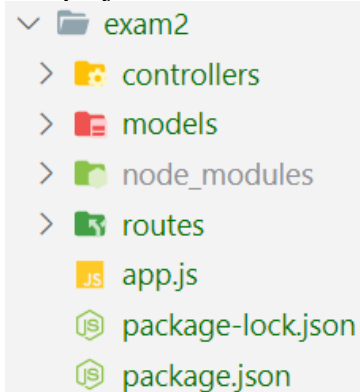
Response JSON:

```
{
  "average": "couldn't find semester 2099-Fall"
}
```

For the API 2) , your code MUST fulfill a), b), c), d) four cases above. The value of semester in query string should be changeable. The above cases are just examples.

NOTE for entire programming question:

- I. You need to design the domain model by yourself. It's your own choice to have a single domain model or multiple domain models.
- II. Fill in the necessary code below. You also need to name the js files in each folder and connect/import modules by yourself.
- III. The project folder structure is:



```
exam2
├── controllers
├── models
├── node_modules
├── routes
├── app.js
├── package-lock.json
└── package.json
```

app.js

```
const express = require('express');  
const app = express();  
app.use(express.json());
```

//place your code below

```
app.listen(3000, () => console.log('listening to 3000...'));
```

routes/_____.js

```
const express = require('express');  
const router = express.Router();  
//place your code below
```

```
module.exports = router;
```

controllers/_____.js

//place your code below

models/_____.js

[7 points] Use HTML and JavaScript to implement a client side for the feature: Provide a UI for user retrieve student's average grade based on semester they input. The server side API is the Question 10's 2nd API. (You're NOT allowed to use 3rd party JS libraries)

Enter Student ID:

Enter Semester:

Case 1: When user doesn't enter value for student ID, click "Submit" button, you need to check and alert error message: Please provide student ID.

Case 2: When user enters student Id and semester, click "Submit" button, you need to make a request to server side API, then display returned message on the page.

In the screenshot below: if user enters studentId: 610001, semester: 2021-Spring, then click "Submit" button, it'll display Average 97 on the page without refresh the page.

Enter Student ID:

Enter Semester:

Average: 97

All your javascript code and HTML code are in the `index.html` file below. Your JS code goes inside `<script>` tag, your HTML code goes inside `<body>` tag.

```
index.html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home</title>
  <script>
    //Place your JavaScript Code here
```

```
</script>
</head>

<body>
  <!-- Place your HTML code here-->


</body>
</html>
```