COMP3322A Modern Technologies on World Wide Web

Project II – Porting the online cinema ticketing and reviewing system to a system using Express.js and MongoDB

Total 13 points

Due Date: 5 pm December 4, 2018

Overview:

In Project 2, you are going to implement the online cinema ticketing system again. But this time, you need to use Node.js, Express.js, and MongoDB to implement the server-side work. For the client-side, you may have to restructure your code to work with the new server platform. Our goal is to have the new system implements ***the same set of features*** as in Project 1, which allows users to create accounts and supports registered users to:

- buy cinema ticket(s) according to their own preferences and seat availability;
- give comments and view other users' comments on the selected film(s);
- view their own ticketing history.

In this project, you will design the database schema similar to Project 1 using MongoDB instead of MySQL. You are free to use any techniques and technologies which you have learned in this course, except for PHP and MySQL. As you have already implemented all the CSS styling in Project 1, you can keep the ***basic styling settings*** for Project 2 ***except the Responsive Web Design***.

Objectives:

1. A learning activity to support ILO 1 and ILO 2.
2. The goal of this programming assignment is:
   - to get solid experience in using client-side, server-side techniques, and database system to design and implement a web-based application which contains some common features of an online cinema ticketing and reviewing system.

Specifications:

One of the design patterns of Node.js and Express.js is on identifying the routes in handling various client requests.

From Project 1 specification, we can identify the following routes.

GET /index.html	Provide the login and create account features.
GET /createaccount.html	Provide the create account form.
POST /verifyLogin	Perform account checking and session control.
POST /create	Perform account checking and creation.
GET /main.html	Display the main page.
GET /buywelcome	Display the movie list and provide the movie selection feature.
GET /comment	Display the comment page.
GET /history	Display the purchase history of the current user.
GET /logout	Perform log out and clear the session.
GET /seatplantry	Display the seating plan of the selected movie and provide the seat selection feature.
GET /buyticket	Display the seat(s) that the user has selected together with the price options.
POST /confirm	Submit the ticket purchasing request to the server and display the order information.
POST /comment_submit	Submit the comment of the selected movie to the server.

GET /comment_retrieve	Retrieve the comments of the selected movie from the server using AJAX approach.

In Project 2, you have to implement the same set of services/features as in Project 1, but you have the flexibility to reduce the number of routes in rendering the same set of services/features. For example, it is possible to join /seatplantry and /buyticket to form one route as we can render the seating plan and the selected seat(s) with price options in the same location by using JavaScript.

About the database design, you can structure the MongoDB database schema similar to the MySQL database schema as you have adopted in Project 1. However, in RDBMS systems, data is usually spread across a few tables, a JOIN is formed across tables to get the specific view of the data. There is no concept of JOINs in Mongodb; to retrieve data from multiple collections, we can add the $lookup stage to the aggregation pipeline. To improve the performance or simplify the search, you can restructure or redesign some of the schemas. For example, you can have the Broadcast information of a film stored inside the film collection, rather than using two collections.

Computer Platform:

Please install the latest version of Node.js, Express.js, and MongoDB (Community Server) to your computer.

Create a project folder called Project2 and use “npm init” or “express” (express-generator) to initialize the project. Put all your source files (.js, .html. css, .pug and any image files) within this project folder. In addition, create the folder path ‘data’ inside this project folder. Use this folder path as the data path for the MongoDB server (i.e., mongod --dbpath xxxxxxx/Project2/data where xxxxxxx is the directory path to the Project2 folder). By this way, both the database and source code are kept in the same project folder.

Submission:

The deadline of Project 2 is on December 4, 2018 (Tuesday) 17:00.
You are required to:

1. Submit your work to the course Moodle submission page. To do that, use a compression or archive tool to compress the Project2 folder and name the file as Project2-csID.zip (or tgz or other common compression formats). Before creating the project zip file, please make sure all source files and data for the MongoDB are all placed inside the project folder.
2. Upload a text file to:
    a. list the version numbers of the Node.js, Express.js, and MongoDB installed in your platform.
    b. indicate how to start or run your web server.
    c. indicate how much work you have completed for the project. You are recommended to make use of the table shown in the Grading Policy section as the checklist. Using it to show us how much you have accomplished. Please submit this checklist file to the course Moodle submission page too.

Grading Policy:

Total (13 points)

- Implement all features/services related to the route "/index.html" (0.5 points)
- Implement all features/services related to the route "/createaccount.html" (0.5 points)
- Implement all features/services related to the route "/verifyLogin" (1 point)
- Implement all features/services related to the route "/create" (1 point)
- Implement all features/services related to the route "/main.html" (0.5 points)
- Implement all features/services related to the route "/buywelcome" (1 point)
- Implement all features/services related to the route "/seatplantry" (2 points)
- Implement all features/services related to the route "/buyticket" (0.5 points)
- Implement all features/services related to the route "/confirm" (1 point)
- Implement all features/services related to the route "/comment" (1 point)
- Implement all features/services related to the route "/comment_retrieve" (1 point)
- Implement all features/services related to the route "/comment_submit" (1 point)
- Implement all features/services related to the route "/history" (0.5 points)
- Implement all features/services related to the route "/logout" (0.5 points)
- We will not assess the styling in this project phase, but applying basic styling is required (1 point)

Note: Please refer to Project 1 specification on the details of the features/services related to each route.

Plagiarism:

Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it. **Please note that we may request you to explain to us how your program is functioning as well as we may also make use of software tools to detect software plagiarism.**