COMP308 – Emerging Technologies

Mid-Term Exam

Due: Week 7 (Monday February 25, 2019) @ midnight

Value 20%

Mid-Term Exam **Maximum Mark: 40**

Overview: Using your knowledge of **NodeJS** and **ExpressJS** and the Web App Template provided, complete the **Favourite Book List** web app that you will share on GitHub and deploy to Heroku (or another cloud provider). Your web app already includes basic navigation controls, a **Landing Page,** a **BookList** page and a **BookDetails** page. Your task is to complete the code that is missing from the routing files and the Books List page so that a user can **Add**, **Delete** and **Edit** any Book item from the Database.

Project Setup:

- Rename the Web App Template provided to COMP308-W2019-MidTerm-[YourStudentID]. (e.g. COMP308-W2019-MidTerm-300818557).
- A connection to an **MLab database** has already been included for this project. You will need to change the **`URI`** variable in the db config file (**`config/db.js`**) or add an environment variable to your cloud host for easy switching.

Instructions:

(2 Marks: GUI, 26 Marks: Functionality, 4 Marks: Internal Documentation, 4 Marks Version Control, 4 Marks Cloud Deployment)

1. The **BooksList** page **(`views/books/index.ejs`)** already lists your favourite books. Your job is to fix the **Add Button**, and insert the appropriate code for the **Edit** and **Delete Buttons (2 Marks: GUI, 5 Marks: Functionality):**
    a. Fix the **Add Button** on this page so that it links to the BookDetails page (**`views/books/details.ejs`**). **Hint**: insert the correct route in the **`href`** attribute. (1 Mark: Functionality).
    b. Add an **Edit Button** to each row of the existing Table (the insertion point has been marked for you). Ensure that when the user clicks on the **Edit button**, they are linked to the BookDetails page and the **`_id`** of the book they wish to edit is passed to that page via the URL. **Hint**: the **`href`** attribute requires a reference to the **`_id`** of the book being edited (1 Mark: GUI, 2 Marks: Functionality).
    c. Add a **Delete Button** to each row of the existing Table (the insertion point has been marked for you). Ensure that when the user clicks on the **Delete button**, that the **`_id`** of

the book to be deleted is passed to the router. **Hint**: the **`href`** attribute requires both a link to the **`delete`** route and a reference to the **`_id`** of the book being edited (1 Mark: GUI, 2 Marks: Functionality).

2. The Books Routing File (**`routes/books.js`**) already has a route working to find all the books in the books collection and render your **BooksList** page. Your task for this section is to complete the **logic** for each of the other routes to **render** the book details page when the **Add** or **Edit** Buttons are clicked, process a request to **Add** or **Edit** a Book, and process a request to **Delete** a book (17 Marks: Functionality).:
    a. Complete the **`get('/add')`** router logic that **renders** the **book details page** (**`views/books/details.ejs`**). The form on the book details page will initially be blank. You must pass an appropriate value for the **`title`** property and blank value for the **`books`** property. (3 Marks: Functionality).
    b. Complete the **`post('/add')`** router logic that **processes** the **insertion** of a new book into the database. You need to instantiate an object of the **book model** (*excluding* the **`_id`** property) and pass this object to the **`create`** method of the **book** model to add a new book to the database. **Hint:** the values for the book object will come from the **`name`** attributes of each field of the book details page. Redirect the user back to the BookList page (`'/books'`) when the insertion is completed. (4 Marks: Functionality).
    c. Complete the **`get('/:id')`** router logic that **renders** the **book details page** (**`views/books/details.ejs`**) and uses the **`id`** from the URL to select the book to document to be updated. Declare an **`id`** variable and set its value to the **`id`** property of the **`req`**uest object. Pass this **`id`** to the **book** model's **`findById`** method to **render** the book details view. You must set an appropriate **`title`** property value and set the **`books`** property to the book that was returned from the database as you render the view. (5 Marks: Functionality).
    d. Complete the **`post('/:id')`** router logic that **processes** the **update request** of an existing book by using its **`id`** property. Declare an **`id`** variable and set its value to the **`id`** property of the **`req`**uest object. You need to instantiate an object of the **book model** (*including* the **`_id`** property) and pass this object to the **`update`** method of the **book** model to edit an existing book in the database. **Hint:** the values for the book object will come from the **`name`** attributes of each field of the book details page. Redirect the user back to the BookList page (`'/books'`) when the update is completed. (6 Marks: Functionality).
    e. Complete the **`get('/delete/:id')`** router logic that processes the user's **delete request** and removes an existing book from database by using its **`id`** property. Declare an **`id`** variable and set its value to the **`id`** property of the **`req`**uest object. Pass the id to the book model's **`remove`** method. Redirect the user back to the BookList page (`'/books'`) when the removal is completed. (3 Marks: Functionality).
3. Include **Internal Documentation** for your site **(4 Marks: Internal Documentation):**
    a. Ensure you include a **comment header** for your **JavaScript file** that indicate: The **File name**, **Author's name**, **StudentID**, and **Web App name** (1 Marks: Internal Documentation).

b. Ensure you include a **section header** for any **JavaScript functions** (1 Marks: Internal Documentation)
c. Ensure all your code uses **contextual variable names** that help make the files human-readable (1 Marks: Internal Documentation).
d. Ensure you include **inline comments** that describe your GUI Design and Functionality. **Note:** Please avoid "over-commenting" (1 Marks: Internal Documentation)

4. Share your files on **GitHub** to demonstrate Version Control Best Practices and push your site to a cloud host **(4 Marks: Version Control, 4 Marks: Cloud Hosting).**

a. Your repository must include **your code** and be well structured (2 Marks: Version Control).
b. Your repository must include **commits** that demonstrate the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).
c. You must deploy your site to your Cloud Server using **git** (4 Marks: Cloud Hosting).

Optional Site Features (i.e. Potential Bonus Marks).

A. Include a site Security for your web app. Only registered and logged in users should be able to access your Favourite Book List (up to 20 Bonus Marks).
   a. Install the **passport**, **passport-local**, **passport-local-mongoose**, **express-session** and **connect-flash** npm modules and related configuration code in the **`app.js`** file (2 Bonus Marks).
   b. Create a **users** model in the Models folder (**`model/users.js`**) that includes **`username`**, **`email`**, **`displayName`**, **`created`** and **`updated`** properties that points to the users collection (2 Bonus Marks).
   c. Create an **`auth`** subfolder in the **`views`** folder. Add two new views to the folder: **`login.ejs`** and **`register.ejs`**. These views should each display login and register forms respectively (2 Bonus Marks).
   d. Modify the main routing file in the routes folder (**`routes/index.js`**) to include 5 new routes: **`get('/login')`**, **`post('/login')`**, **`get('/register')`**, **`post('/register')`** and **`get('/logout')`** that use the login and register views to login and register a user (6 Bonus Marks).
   e. Modify the books routing file in the routes folder (**`routes/books.js`**) to include a new method, **`requireAuth`** that prevents non-logged in users from accessing the BooksList or Book Details pages (2 Bonus Marks).
   f. Modify the navbar in the header partial file (**`partials/header.ejs`**) to only display the **BookList** page when a user is logged in and display **Login** and **Logout** Links depending on login status (4 Bonus Marks).
   g. Display the logged in user's name in the Nav Bar (2 Bonus Marks).

SUBMITTING YOUR WORK

Your submission should include:

1. A zip archive of your website's **Project files**.
    - Ensure to Name your project files COMP308-W2019-Midterm-[YourStudentID].zip e.g. COMP308-2019-Midterm-300818557.zip
    - Please **do not** create a RAR archive of your project files.
2. A link to your GitHub repository.
    - Ensure to Name your GitHub repo: COMP308-W2019-Midterm-[YourStudentID] e.g. COMP308-W2019-Midterm-300818557)
3. A link to your live site hosted with a Cloud provider (Heroku Recommended).
    - Ensure to name your live site COMP308-2017-Midterm-[YourStudentID] e.g. COMP308-2019Midterm-3008185557.herokuapp.com

Feature	Description	Marks
GUI / Interface Design	Display elements meet requirements. Appropriate spacing, graphics, colour, and typography used.	2
Functionality	Site deliverables are me and site functions are met. No errors, including submission of user inputs.	26
Internal Documentation	File header present, including site & student name & description. Functions and classes include headers describing functionality & scope. Inline comments and descriptive variable names included.	4
Version Control	GitHub commit history demonstrating regular updates. 2 marks for simply pushing your files to GitHub once. An additional 2 marks awarded for using GitHub as you code.	4
Cloud Deployment	Deploy site to Cloud Service.	4
Total		**40**

This exam is weighted **20%** of your total mark for this course.

This is an open-book exam.

Students may use the Internet to view the instructor's GitHub repos and their own work.

Students may also access course PowerPoint presentations or the Textbook outlined in the Course Syllabus.

Use of a search engine is permitted.

However, use of external code is not allowed for this exam.

Please check with your instructor if you are unsure.