Databases, Network and the Web

Coursework for Midterm (Oct 2025 Session)

Student Instructions

Event Manager

Introduction

Your task is to create a deployable event manager, which could be used to organize events. For example, a yoga instructor could arrange yoga sessions or an art gallery could put on talks about its artworks.

The event manager will be for one individual or organisation and can be deployed on a web server.

There should be two discrete pages: organiser and attendee. The organiser page is used to set up events and publish them. The attendee page is used to book published events.

Please read the base requirements carefully for the full detail.

Technical specification

All submissions **MUST** use the template project provided with the assignment. Express.js must be used as the basis for the server-side functionality. SQLite must be the basis for the data tier. Client-side pages must be delivered through server-side rendering using Embedded JavaScript Templates. As such the following tools are pre-requisite:

Tool	Description	URL
EJS	Tool for Embedded Javascript Templates. Allows for dynamic rendering of HTML.	https://www.npmjs.com/package/ejs
SQLite3	Query your data tier from within your express app	https://www.npmjs.com/package/sqlite3
ExpressJS	A framework for simplifying building a web server	https://expressjs.com/
NodeJs	Javascript engine for building web servers	https://nodejs.org/en/
SQLite	Portable, single file-based SQL database.	https://www.sqlite.org/index.html

You may make use of NPM addons and front end libraries provided that they are justified and don't fundamentally alter the above specification. Some example packages are listed below:

Tool	Description	URL
Tailwind	A popular CSS library. NB. requires some quite tricky build tools	https://tailwindcss.com/
Bootstrap	An older but still popular CSS library which is easier to use straight off	https://getbootstrap.com/
Axios	A helpful library for making requests from both client and server	https://www.npmjs.com/package/axios
Assert	Internal node module. Useful for testing and basic data validation	https://nodejs.org/api/assert.html
Joi	Really helpful for improving that data validation.	https://www.npmjs.com/package/joi
Express-Validator	A more scalable approach to data validation.	https://express-validator.github.io/docs/
Express Sessions	Persistent and secure storage of user details when they are logged in	https://www.npmjs.com/package/express-session
Date-Fns	Date/Time formatting tool	https://date-fns.org/

- All source code must be human readable
- Do not use bundlers such as Webpack
- You can use CSS precompilers such as Tailwind provided the built CSS is provided with the submission
- We will run 'npm install' followed by 'npm run build-db' and 'npm run start' to run your project. Please ensure that your project runs from these commands alone without need for further compilation or installation of dependencies.
- Ensure your node version is at least 16.x and your npm version is at least 8.x

Data requirements:

- Pages must populate with user data dynamically retrieved from the database on the server. This can be done through templating (as taught in this course), or through other means (eg. JavaScript requests and DOM manipulation).
- Features relying on hard-coded client-side user data will be disregarded
- All user data must be persistently stored in the SQLite database

Code Style Requirements:

- Frontend code comprises template files (.ejs) in the views folder and optionally assets and .js files in the public folder
- Backend code comprises an index.js file with routes implemented as middleware in the routes folder.

- Each route should be preceded by a comment describing its purpose, inputs, and outputs
- Routes should be rationally organised making appropriate use of GET, POST methods.
- Each database interaction should be commented describing the purpose, inputs, and outputs.
- Code should be laid out clearly with consistent indenting.
- Functions and variables should have meaningful names, with a consistent naming style
- All variables should be declared with careful attention to scoping

Template Code

All submissions **MUST** use the template project provided.

Please see the README.md file in the template code for details of how to use it.

Base Requirements:

Implement these basic features fully to achieve a passing grade.

Organiser Home Page:
This is where the Organiser can create, review and edit articles. The minimum requirements for the page are as follows:

- A heading which indicates that this is the **Organiser Home Page**
- It should be accessed through a URL which is distinct from the **Attendee Home Page**
- It should display the event manager name and description. For example “Stretch Yoga” and “Yoga classes for all ages and abilities”.
- It should have a link which points to the **Site Settings Page**
- It should have a “Create New Event” button
    - When pressed this should create a new draft event and redirect to it’s edit page
- It should display a dynamically populated list of *published* events
    - The list should display useful information about the events including the title, date, when they were created and published and the number of each type of ticket for sale
    - For each event the list should display a sharing link which points to the relevant **Attendee Event Page**
    - For each event there should be a delete button. When pressed this should:
        - Remove the event from the database
        - Reload the page to display the updated information
- It should display a dynamically populated list of *draft* events
    - The list should display useful information about the events including the title, date, when they were created and published and the number of each type of ticket for sale
    - Each event in the list should be accompanied by a link which points to its edit page
    - For each event there should be a publish button. When pressed this should:
        - Update the event’s state from draft to published
        - Timestamp the publication date
        - Reload the page to display the updated information
    - For each event there should be a delete button. When pressed this should:
        - Remove the event from the database
        - Reload the page to display the updated information

Site Settings Page:
This is where the organiser can change their site name and description. The minimum requirements for the page are as follows:

- A heading which indicates that this is the site settings page
- A form with text inputs for name and description.
    - The form should be dynamically populated with the current site settings
    - The form should have a submit button which updates the settings with the new values and redirects to the **Organiser Home Page**.
    - Form validation should be used to ensure that all fields have been completed ahead of submission.
- A back button which points to **Organiser Home Page**.

Organiser Edit Event Page:
This is where the organiser creates or amends events. The minimum requirements for the page are as follows:

- A heading which indicates that this is the **Organiser Edit Event Page**
- The event creation date
- A form containing the following input fields and controls:
    - Event title
    - Event description
    - The number and price of full-price tickets
    - The number and price of concession-priced tickets
    - A submit changes button
- When editing an existing event, the form should be populated with the current event data
- When changes are submitted the event's last modified date should be changed
- A back button which points to **Organiser Home Page**.

Attendee Home Page:
This is the front page which users can see what events there are and book them. The minimum requirements for the page are as follows:

- A heading which indicates that this is the **Attendee Home Page**
- It should be accessed through a URL which is distinct from the **Organiser Home Page**
- It should display the site name and description
- A list of *published* events
    - The event title and date should be visible for each item
    - Events should be ordered by event date with the next event appearing at the top
    - Clicking on an item in the list should take the user to the **Attendee Event Page** for that particular event

Attendee Event Page:
This is where the attendee can view the details of events and make bookings. The minimum requirements for the page are as follows:

- A heading which indicates that this is the **Attendee Event Page**
- It should display a single event determined by the url
- It should display information about the event including title, description, date, ticket types and ticket prices
- The user should be able to select the number of each type of ticket they want to purchase
- The user should enter their name
- There should be a Book button which will book the tickets
- Attendees should only be able to book as many tickets as are available
- There should be a back button which redirects the user to the **Attendee Home Page**.

Main Home Page:
This should be the default home page of the application, accessible from http://localhost:3000/. It should simply contain the following:

- A link to the Organiser Home Page
- A link to the Attendee Home Page

Extension:

You should also enhance your project by completing an extension. You can add whatever extension you want but ensure you read the guidelines below.

- For the extension, consider how it demonstrates your understanding of the course content and how it enhances the functionality and usefulness of the application from different perspectives.
- Focus on demonstrating skills in developing server-side functionality, rather than client-side improvements
- Use the video screencast to show and explain your extension, ensuring you adhere to the time limit for the video
- Use the documentation to guide the marker through your extension. Include urls of the pages you want us to examine and show screenshots of what we should be seeing.
- Ensure any additional libraries are common and quickly and easily installed. Any issues with installing the libraries will limit your grade. In your documentation, explain the purpose of any additional library you use.

Deliverables

1. **Source Code**
   Please upload the complete source code for your web application in zip format.

   - Make sure node_modules is removed from your project
   - Make sure package.json is included and that all additional packages are listed in your package.json file
     - HINT: make sure you used ```npm install –save package_name``` when you installed them
   - Do not include your SQLite database.db file. We will build this by running ```npm build-database```
   - A README.md
     - remove the existing contents that we have provided and replace with just the *simple* instructions on how to set up the application
     - include a list of any additional libraries you are using

2. **Report in PDF Format**
   Please upload your report in PDF format (max 1000 words).

   - A high level schematic diagram for your website demonstrating all three tiers of your architecture, the end points that connect client to server. Highlight the additional elements that your extension adds.
   - A high level diagram showing your data model. Draw an ER diagram showing the tables and how they are related to each other. Use “crows feet” notation, as shown in the course. Highlight the additional elements that your extension adds.
   - Extension description
     - Explain the extension you implemented.
     - Discuss what you did to implement it.
     - Explain how it applies techniques covered in the course content
     - Explain if and how it does beyond the course content
     - Paste relevant parts of your code into the document to explain how you implemented the extension. Refer to the filenames and line numbers where the code can be found.

   You can use a free tool such as Figma (https://figma.com) or Lucid Chart (https://lucid.app/) to produce diagrams

3. **Video Screencast**
   Please upload a video screencast of your project working.

   - This should not be longer than 2.5 minutes (we will not watch beyond 2.5 minutes).
   - It should demonstrate all of the functionality, including your extension
   - We recommend that you talk through what you are showing on the video
   - We recommend that you capture the video in mp4 format using software such as OBS

4. **Code in PDF Format**

   Please upload your code in **text format (not screenshots)** in a PDF file here.

   Please clearly label with start and end comments exactly which sections of code you personally wrote without assistance. This will allow us to carry out the necessary plagiarism checks.

   You will receive a zero mark if you don't submit your code in this way.

Tips for Success

- Read the requirements above carefully and ensure you implement them exactly as specified.
- Add good comments to your code. Comment each route end-point and each function. Include its purpose and list the inputs and outputs.
- Before submitting your work, test all your application functionality.
- Before submitting your work, test your deployment process. Delete the node_modules directory and then run: ‘npm install’, ‘npm run build-db’ (or ‘npm run build-db-win’) and ‘npm run start’.
- Pay particular attention to ensuring that the database script db_schema.sql creates your database without any errors or issues

Review Criteria

You must provide the PDF of your source code as requested. Failure to do so will result in a zero mark for your entire project.

- Functionality (25 marks)
  - Main Home Page
  - Organiser - Homepage functionality
  - Organiser - Settings page
  - Organiser - Edit Article
  - Attendee - Home page
  - Attendee - Article page
- Quality of Implementation (25 marks)
  - Frontend usability and styling
  - Backend code quality
  - Database design
  - README and comments
- Documentation (10 marks)
  - Architecture diagram
  - Database diagram
- Extension (40 marks)
  - Extension ambition
  - Extension functionality
  - Extension quality of implementation
  - Extension video and documentation