SODV1201 – Intro to Web Programming
Group Project: Creative Studio Rentals

Case Study

In this project, students will develop a comprehensive web application designed to connect owners of creative studios—such as art studios, recording studios, and rehearsal spaces—with artists, musicians, and creators seeking to rent these spaces for their projects. The application will include user accounts, detailed studio listings, data storage, and responsive web design. Similar to how Uber connects riders with drivers and Airbnb connects tenants with landlords, this application will connect renters with owners of creative studios. This project not only aims to provide a practical solution for the creative community but also offers students hands-on experience in full-stack web development, from front-end design to back-end implementation.

Your task is to build a **Minimum Viable Product (MVP)** for this case study. An MVP is the smallest conceivable list of features that fulfill the primary business goal of a software product. One way to summarize a feature is by a "user story", which is a short sentence describing the feature from the perspective of the user. User stories frequently take the following form: "As a <type of user>, I can <take some action> so I can <some reason>." User stories have been provided to you below to help you create your MVP.

Project Instructions

You will work on this project as a group to develop this web application.

Project restrictions:

- You may only use the frameworks we cover in this course.
- The web application should **not** be hosted online and **not** be made publicly available. It will be implemented on a local server only.
- Do **not** implement passwords for users. Make the user signup and login process as simple as possible. For example, a user can create an account by providing a unique email address. That user can later login by simply entering that unique email address.
- Do **not** implement any form of communication system, booking system or payment processing system. Assume that all communications, bookings and payments happen offline, separate from your web application.

Phase 1: Front-End

Tip: to ensure you have enough time to complete Phase 2, your Phase 1 should be completed by the end of Module 3.

Design and implement a web application for the given case study using HTML, CSS and JavaScript. The web application should have a homepage, and every page should have the same header and footer. For more functionalities refer to the user stories below to determine the pages and controls required to bring your web application to life. Store all application data in temporary memory using JavaScript arrays and/or objects. (In Phase 2 you will shift to using JSON file(s) to store data).

User Stories

With your web application, users should be able to do the following activities:

1. As a new user, I can create an account by providing my name, phone number, and email address so I can be contacted by other users.
2. As a new user, I can select my role as either a renter or studio owner so I can see and use data that is relevant to me.
3. As an existing user, I can log into my existing account by providing my email address so I can use the web application.
4. As an existing user, I can update my name, phone number and email address at any time so I can keep my contact information up to date.
5. As a studio owner, I can add multiple listings of studios for rent so I can attract renters. A studio listing includes the following details:
    a. Name (e.g. "Bow Valley Recording Studio"),
    b. Address (e.g. "345 - 6 Avenue SE, Calgary, AB T2G 4V1"),
    c. Area (in square meters),
    d. Type (e.g. art studio, recording studio, dance studio, rehearsal space, …),
    e. The number of individuals it can accommodate,
    f. Whether it includes parking or not,
    g. Whether it is reachable by public transportation or not,
    h. Whether it is currently available to rent or not,
    i. The rental term (e.g. hourly, daily, weekly, monthly, …),
    j. The price per rental term.
6. As a studio owner, I can view all my existing studio listings so I can make sure my data is correct.
7. As a studio owner, I can edit the details of any of my existing studio listings for rent so I can keep my listings up to date. For example, if a studio becomes rented then I want to mark it as not being available.
8. As a studio owner, I can delete any of my existing studio listings for rent so I can keep my listings up to date. For example, if a studio is sold then I want to remove it from my listings.
9. As a renter, I can view a list of all the studios that are available to rent so I can see all available options.

10. As a renter, I can filter the list of studios that are available to rent using **any** of the listing details (see user story 5) so I can narrow my search.
11. As a renter, I can select a studio from the list so I can view **all** its listing details (see user story 5).
12. As a renter, I can get the contact information of a studio owner (name, phone number and email) so I can contact them to make a booking offline.

Phase 2: Back-End

During this phase of your project, switch to using the local file system to store application data.

- Design and implement a back-end REST API to store data to, and fetch data from, local JSON file(s).
- Your server implementation should use Node.js and Express.
- Use the correct HTTP methods (GET, POST, PUT, and DELETE).
- For each web service and functionality in your API, return a flag that indicates the success or failure of the web request. This is error handling. In the case of any failure, add a detailed error message that clearly lets the user know what to do.

Presentation

On the last day of the course each group will present their web application to the rest of the class. In less than 5 minutes: give a live demonstration of the most important features of your application, display the data stored in your local JSON file(s), and discuss the most important lesson that was learned in this project (for example, if you had to start the project over, what would do differently or the same?). **All group members must participate.**

Tips!

Here are some tips to help you create a successful web application:

- Plan, plan, plan!
- You are building an MVP. Focus on completing the bare minimum features first, then add extras later.
- Speak with your group members to find each other’s strengths, weaknesses, and preferences. Assign duties accordingly.
- Give everyone a task. ***Keep track of who does what by when.***
- Discuss your strategy for how source code will be updated and shared amongst the group members. For example, create a Github repository.
- Think about the user’s journey. Create a flowchart for the “renter” and a flowchart for the “studio owner.” How do you envision them using your web application?
- Create wireframes for every page before you start coding the front-end.
- Your front-end design should have an identity and creative details such as:
    - A brand name,
    - Color palette,
    - Fonts,
    - Images,

    - Interactive elements (CSS, JavaScript, jQuery),
    - Organizing elements (tables, lists, etc.).
- Design your front-end to be responsive to all available viewports (mobile, tablet, laptop, etc.).
- To implement user sessions across the pages of your application consider using the browser's local storage or cookies to store the user ID or email address of the person who is currently logged in. Remove that data when the user logs out.
- Create seed data early. This will help you to understand how the data should be organized.
- Structure your source code files appropriately (e.g. separate CSS, HTML, and JavaScript; separate front-end files from back-end files).
- Contact the instructor early if there are any problems or issues.
- ***Schedule a check-in with the instructor after Phase 1 is complete to see if you are missing any key components and get suggestions for improvement.***

Submission Instructions

- All source files should be logically named and organized in an appropriate file structure.
- Make sure to properly reference any outside resources that you use.
- Place all source files into a zipped (compressed) folder, then upload it to D2L.

Rubric

- 100 points available.
- Partial credit will be given.
- Worth 30% of final grade.
- Your code will be evaluated for correctness (does it achieve the task it is supposed to?)
- Your code will be evaluated for hygiene (is it clear, well-commented, and easy to follow?)
- Use best coding practices:
    - Add intelligent comments that explain your logic and intention.
    - Use sensible variable names that match the purpose of a variable.
    - Use whitespace and indentation to make your code easy to read.

Grading Items	**Points**
Documentation, structure and submission • Correct submission method and format. • Source code files are structured appropriately. • Code is readable and well-organized with adequate whitespace. • Code has detailed comments.	10
Front-End • Interactive, responsive, and styled front-end application.	40

• Properly working forms. • Proper form validation, error handling, and warnings for users. • Front-end serves your back-end server and API. • Front-end meets the requirements of all user stories.	
Back-End and API • Proper incoming data completeness and correctness validation. • Response/confirmation after form submission. • Proper naming, module, package, and API usage. • Working API to fetch the data set based on the CRUD principle. • Back-end meets the requirements of all user stories.	40
Presentation (less than 5 minutes) • All group members must participate. • Live demonstration of your web application: ○ The most important features. ○ Display the data stored in your local JSON file(s). • Lessons learned: ○ Discuss the most important lesson that was learned from this project. ○ If you had to start the project over, what would you make sure to do differently or the same?	10
Total	**100**