

assignment 2

by Sharise Jonathan

Submission date: 05-Jan-2026 12:40AM (UTC+0000)

Submission ID: 272055582

File name: 50061_Sharise_Jonathan_assignment_2_708810_1949335159.docx (664.19K)

Word count: 1773

Character count: 10249



ASSIGNMENT 2: CLOUD BASED WEB SERVICE

Cloud Based Web Services (ACCA6024)

Sharise Jonathan (2306013)
2306013@student.uwtsd.ac.uk

Contents

Abstract 2

Introduction 2

Organisation Description..... 3

System Design and User Interface 3

RESTful API Integration 4

Displaying Item Details 5

Creating, Editing, and Deleting Events 5

Search and Filtering Functionality 9

AJAX Polling and Automatic Updates 10

Creativity and Additional Features..... 10

Setup and Running Instructions 11

Conclusion..... 11

References? **Error! Bookmark not defined.**

Abstract

This project presents the design and implementation of a dynamic web application that communicates with a RESTful Web API using AJAX techniques. The application builds upon a previously developed API and demonstrates real-time data exchange between a client-side interface and a server-side service. The chosen system is an Event Management System, which allows users to view, search, create, edit, and delete event data through a single-page web application.

The client-side application is developed using HTML, CSS, and JavaScript, while the server-side API is implemented using Node.js and the Express framework. Asynchronous communication is achieved using the Fetch API, enabling dynamic updates to the user interface without requiring full page reloads. Additional functionality such as search filtering and AJAX polling is implemented to enhance usability and ensure that displayed data remains up to date.

This project demonstrates the practical application of RESTful principles, client-server architecture, and modern web development techniques, while emphasising usability, responsiveness, and clean system design.

Introduction

Modern web applications increasingly rely on dynamic interaction between the client and server to provide responsive and engaging user experiences. Rather than relying on full page reloads, applications now use asynchronous communication techniques to retrieve and modify data in real time. This project demonstrates the development of a dynamic web application that uses AJAX and a RESTful Web API to exchange data with backend web services.

The aim of this project is to design and implement a single-page web application that allows users to interact with an underlying REST API created in Assignment 1. The application allows users to view, search, create, update, and delete data related to events. The system also includes automatic updates through polling, ensuring that data remains current without manual page refreshes.

The project is implemented using Node.js and Express for the server-side API, while the client-side application is built using HTML, CSS, and JavaScript. Communication between the client and server is handled using the Fetch API, which enables asynchronous HTTP requests and responses in JSON format. Emphasis is placed on creativity, usability, and adherence to RESTful principles.

Organisation Description

The organisation chosen for this project is an event management service. Event management systems are widely used by facilities to organise and manage events efficiently. These systems must support the creation of events, management of availability, updating event details, and removal of outdated information.

In this application, each event represents a single item managed by the organisation. Events include details such as the event name and the number of available seats. The system provides an interface that allows users to interact with these events in a controlled and efficient way. This scenario is realistic and closely mirrors real-world systems, making it well suited for demonstrating RESTful web services and AJAX-based interaction.

System Design and User Interface

The web application is designed as a single page application, meaning all functions are contained within one main page. This approach ensures that user interactions feel smooth and uninterrupted as the page does not need to reload when data changes. Instead, content is updated dynamically using JavaScript and AJAX calls.

The user interface displays a list of available events and includes forms that allow users to add new events and edit existing ones. Users receive immediate feedback when actions are performed, such as successfully creating or deleting an event. Furthermore, Visual styling is applied using CSS to ensure the interface is clear, readable, and visually appealing.

The design prioritises simplicity and clarity. The layout ensures that users can easily understand how to interact with the system, even without technical knowledge. By combining a clean interface with dynamic behaviour, the application demonstrates effective front-end design principles.

Events

Search Events

- Rock Concert (50 seats)
- cinema (30 seats)
- Firework Display (100 seats)
- WinterWonderland (80 seats)

Figure 1: Events retrieved from the REST API using an AJAX GET request

RESTful API Integration

The application integrates with a RESTful Web API that was developed in Assignment 1. The API is built using Node.js and the Express framework and follows REST principles by exposing clearly defined endpoints for interacting with event data.

```
soac-azure@MHUbuntu0ct25:~/event-api$ node app.js
API running at http://localhost:3000
```

Figure 2: Node.js REST API running successfully on localhost

AJAX is used to communicate with the API through asynchronous HTTP requests, enabling the application to update data dynamically without requiring full page reloads. These requests allow the client to retrieve and modify data without interrupting the user experience. All data exchanged between the client and server is formatted as JSON, which is lightweight, readable and widely used in modern web applications.

The API supports retrieving all events, creating new events, updating existing events using PATCH requests, and deleting events. Each client-side action triggers a corresponding API request, ensuring a clear separation of responsibilities between the front end and back end. This separation improves maintainability and reflects industry best practices.

Displaying Item Details

The application allows users to view detailed information about individual events. When an event is displayed in the event list, it can be selected by the user to reveal additional details. This interaction is handled dynamically using JavaScript, without navigating to a new page.

When an event is selected, its details are displayed in a dedicated section of the interface. This may include the event name and the number of available seats. This method of displaying item details ensures that users can quickly access relevant information while remaining on the same page.

- WinterWonderland (80 seats) Delete

Event Details

ID: 1767489653830

Name: WinterWonderland

Available Seats: 80

Status: OPEN

Figure 3: viewing detailed information for a selected event without page refresh

This feature demonstrates effective use of DOM manipulation and event handling as JavaScript is used to update page content dynamically in response to user interactions. It also enhances usability by providing contextual information without disrupting the overall workflow.

Creating, Editing, and Deleting Events

A core requirement is the ability to create, modify, and delete data through interaction with the RESTful API. This application fully supports these operations.

Users can create new events by submitting a form containing the required event details. When the form is submitted, an AJAX POST request is sent to the API. Once the event is created successfully, the event list updates automatically to reflect the new data.

Events

- firework display (70 seats)
- Rock Concert (50 seats)

Add Event

Figure 4: New event created using an AJAX POST request and dynamically displayed

Events

- firework display (70 seats)
- Rock Concert (50 seats)
- Movie Night (30 seats)

Figure 5: New event appears and is displayed

Editing events is achieved using a PATCH request, which allows partial updates to existing data. Users can modify event details, and the updated information is immediately displayed on the page. This approach ensures efficient data transfer and adheres to RESTful principles.

Rock Concert (50 seats)

Delete

cinema (30 seats)

Delete

Ice Skating (100 seats)

Delete

Event Details

ID: 1767493931221

Name: Ice Skating

Available Seats: 100

Status: OPEN

Add Event

Event name

Seats

Add Event

Edit Event

Ice Skating

100

Update Event

Rock Concert (50 seats)	Delete
cinema (30 seats)	Delete
Skating (80 seats)	Delete

Event Details

ID: 1767493931221
Name: Ice Skating
Available Seats: 100
Status: OPEN

Add Event

Add Event

Edit Event

Update Event

Figure 6: Update of an event using a PATCH request

Deleting events is handled using DELETE requests. When a user chooses to remove an event, the request is sent to the API, and the event is removed from the system. The interface updates dynamically to reflect the change, providing immediate feedback to the user.

Events

- firework display (70 seats)
- Rock Concert (50 seats)
- Movie Night (30 seats)

Events

- firework display (70 seats)
- Rock Concert (50 seats)

Figure 7: Event removed using a DELETE request and the interface updated dynamically

Together, these features demonstrate full CRUD functionality and highlight the flexibility of RESTful APIs when combined with AJAX.

Search and Filtering Functionality

The application includes search functionality that allows users to filter events based on their names. This feature improves usability by enabling users to quickly locate specific events within a larger dataset.

As the user types into the search input field, JavaScript dynamically filters the list of events currently loaded from the API. The displayed list updates instantly, without requiring additional server requests or page reloads.

Search Events

Firework Display (100 seats)

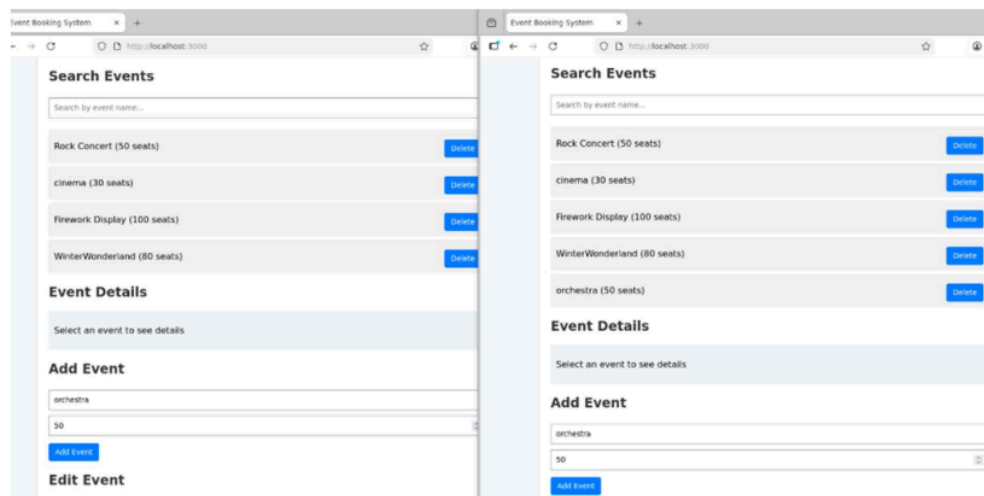
Figure 8: Client-side search functionality filtering events in real time

This client-side filtering approach ensures fast performance and demonstrates effective use of JavaScript to manipulate and display data dynamically. It also enhances the overall user experience by making the application more interactive and responsive.

AJAX Polling and Automatic Updates

To ensure that the displayed data remains up to date, the application implements AJAX polling. Polling involves sending periodic requests to the API to retrieve the most recent data.

A timed function is used to fetch updated event data at regular intervals. When new data is received, the interface updates automatically to reflect any changes. This ensures that users always see the latest information without manually refreshing the page.



+: Periodic polling implemented to automatically update event data using AJAX

This technique is particularly useful in applications where data may change frequently. It demonstrates an advanced use of AJAX and highlights how web applications can remain current while maintaining performance and efficiency.

Creativity and Additional Features

Beyond the core requirements, additional effort was made to improve the usability and presentation of the application. Custom CSS styling was used to enhance visual clarity and improve layout. User feedback is provided through visible interface updates, such

as immediate changes to the event list when actions are performed, helping users understand the results of their interactions and reducing confusion. The application structure clearly separates server-side logic from client-side functionality. This separation improves readability and maintainability, making the system easier to extend in the future.

These design choices demonstrate creative thinking and attention to detail, contributing to a polished and professional final product.

Setup and Running Instructions

To run the application locally, Node.js must be installed on the system. After navigating to the project directory in the terminal, the required dependencies can be installed using npm. Once installed, the server can be started by running the application file.

After the server starts, the application can be accessed through a web browser by navigating to the local host address. The interface allows full interaction with the event management system, including viewing, searching, and modifying events. Additionally, Nodemon was used to automatically restart the server when changes were made to the source code, improving development efficiency during development.

Clear setup instructions ensure that the application can be easily run and tested by others.

Conclusion

This project successfully demonstrates the development of a dynamic web application that uses AJAX and a RESTful Web API to exchange data with backend web services. The Event Management System provides users with the ability to view, search, create, edit, and delete event data in real time.

By further developing the API created in Assignment 1, the project showcases a complete client-server system that follows RESTful principles. The use of asynchronous communication, dynamic updates, and polling ensures a responsive and modern user experience.

Overall, the application meets requirements and demonstrates a strong understanding of web development concepts, RESTful API integration, and AJAX-based interaction. The project reflects both technical components and efficient design, making it a successful and effective solution.



assignment 2

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

60/100

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills		
2. Knowledge and Understanding Skills		
3. Cognitive and Intellectual Skills	Assignment 2: 40	
Design and documentation		25
Test documentation and testing		
Well presented report of implementation. More documentation would have been desirable here.		

Good understanding demonstrated of the functionality of the SPA, and API. UI testing with evidence would have been beneficial. Instructions as part of the documentation were desirable here too. Some good use of screenshots.		
4. Practical Application Skills	Assignment 2: 50	
AJAX implementation REST API A nice clean file and code structure. Detailed instructions would have helped with installation and running. CORS should have been used within the app.js file as this stopped the functionality locally for me.		30
5. Transferable Skills for Life and Professional Practice	Assignment 2: 10	
Presentation Reasonable understanding of the code and what		5

it does. A little more detail would have been beneficial, such as knowing what the functions do.		
Nice demonstration of a simple and clean dynamic and functioning UI.		

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7
PAGE 8
PAGE 9
PAGE 10
PAGE 11
PAGE 12
PAGE 13