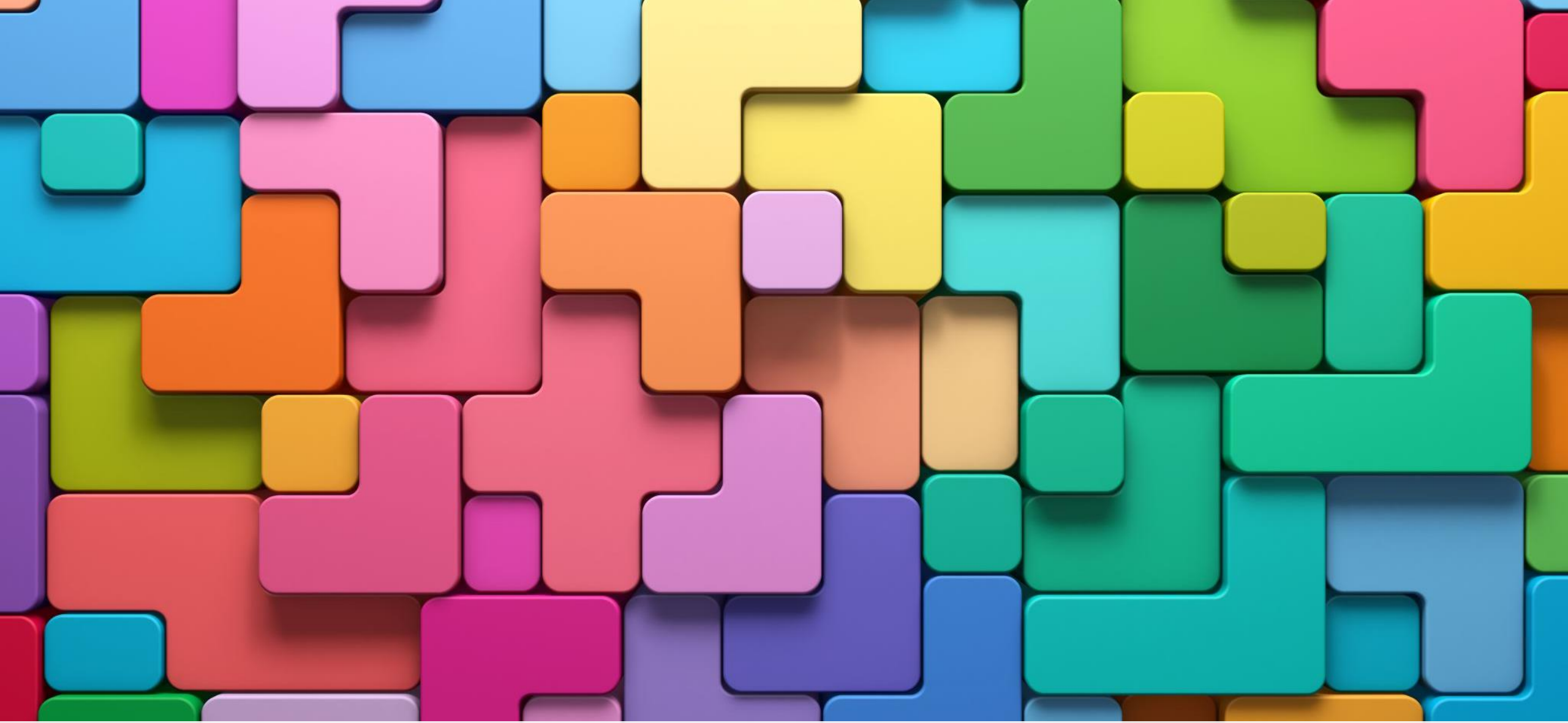


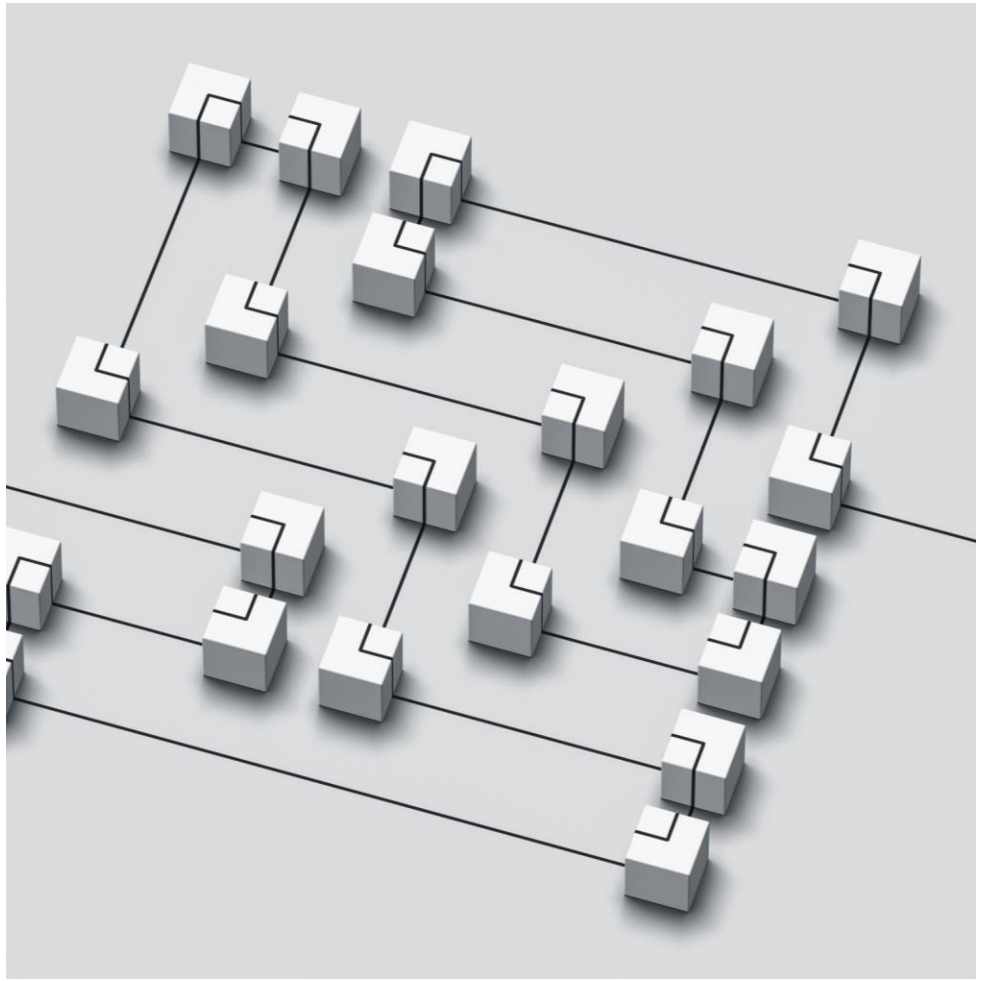
Case Study: Meet App

Overview

- The Meet App is a serverless, progressive web application (PWA) designed for event discovery. Built with React and powered by Google Calendar API, it allows users to browse upcoming events, filter them by city, and customize their event list. The app offers an interactive and responsive experience, including offline access through cached data and the ability to install it as a home screen shortcut. Additionally, Meet App features data visualization, displaying event distribution across cities in an easy-to-read chart.

Purpose

- The Meet App project aims to offer users an intuitive platform for discovering events, with the ability to filter by city, customize the number of events displayed, and view event details interactively. Built with React and utilizing serverless functions for backend operations, the app ensures a seamless, offline-capable experience while also offering data visualization to enhance event exploration. With its responsive design and user-friendly interface, the Meet App makes event discovery accessible and engaging for all users.



Objective

- The objective of the Meet App project is to create a responsive event discovery platform that allows users to filter events by city, toggle event details, and customize the number of events displayed. The app aims to provide an engaging and user-friendly experience by leveraging modern web technologies and serverless functions to ensure seamless access to event information, even when offline.

Approach

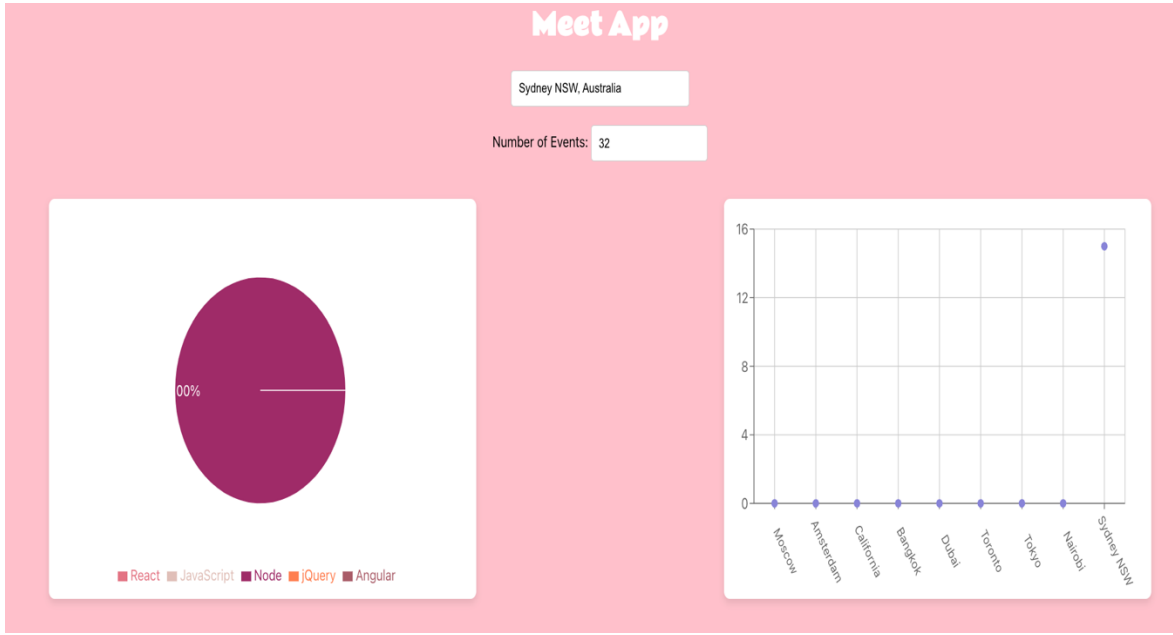
Frontend Development: The app is built using a responsive frontend framework (e.g., React), providing a smooth, interactive interface for users to explore and filter events easily.

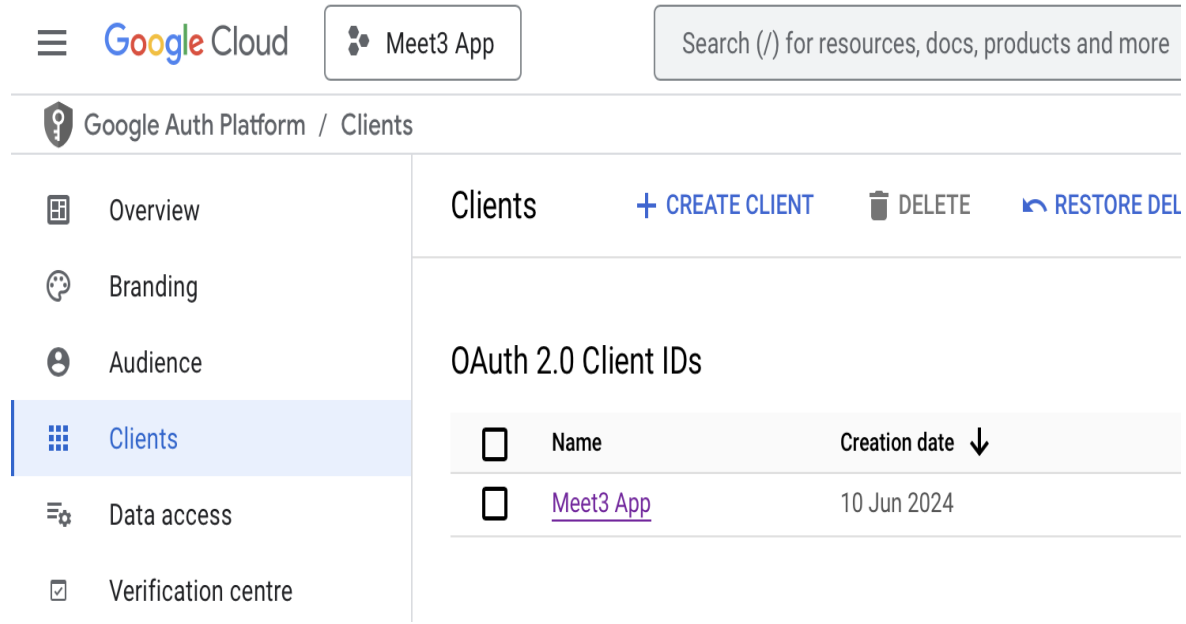
Event Filtering: Users can search for events by city, with suggestions dynamically populated as they type, and toggle event details for a more personalized view of the events.

Serverless Backend: To handle backend tasks like fetching events and managing data, serverless functions are used. This eliminates the need for maintaining complex server infrastructure while ensuring scalability, flexibility, and cost-effectiveness.

Offline Capability: The app is designed to work offline by utilizing cached data, ensuring users can still view event details without an active internet connection.

User Interaction: Features like the ability to customize the number of events displayed and adding the app as a shortcut to the home screen enhance the usability of the app.





Google Calendar API overview

Send feedback

The Google Calendar API is a RESTful API that can be accessed through explicit HTTP calls or using the Google Client Libraries. The API exposes most of the features available in the Google Calendar Web interface.

Following is a list of common terms used in the Google Calendar API:

Event

An event on a calendar containing information such as the title, start and end times, and attendees. Events can be either single events or recurring events. An event is represented by an Event resource.

Calendar

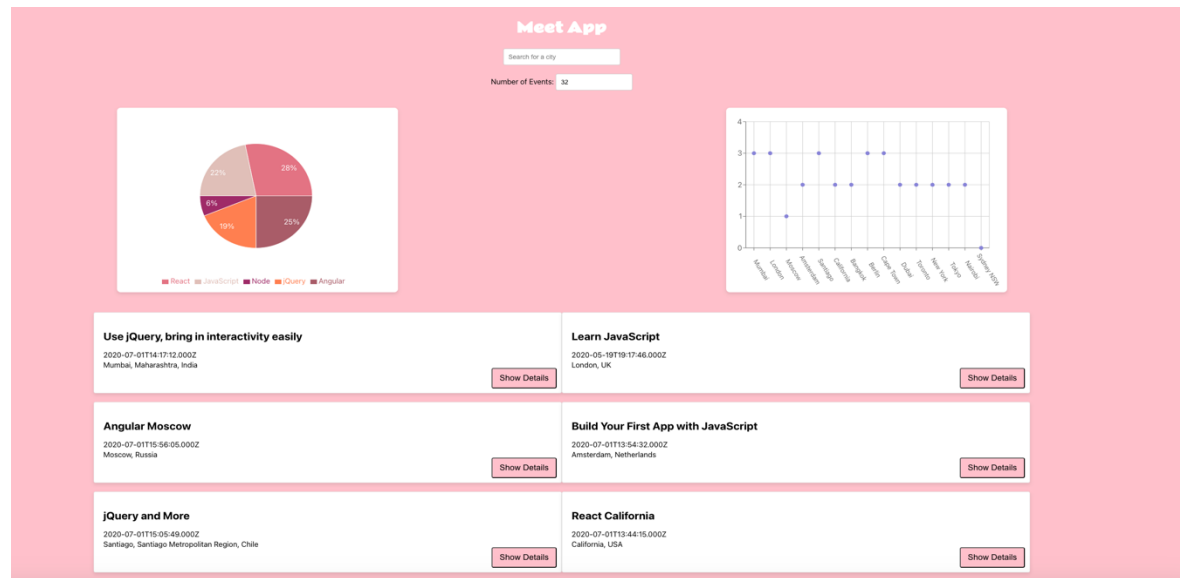
A collection of events. Each calendar has associated metadata, such as calendar description or default calendar time zone. The metadata for a single calendar is represented by a Calendar resource.

Calendar List

A list of all calendars on a user's calendar list in the Calendar UI. The metadata for a single calendar that appears on the calendar list is represented by a CalendarListEntry resource. This metadata includes user-specific properties of the calendar, such as its color or notifications for new events.

Setting

A user preference from the Calendar UI, such as the user's time zone. A single user preference is represented by a Setting Resource.



Challenges

Event Data Management: Ensuring accurate and up-to-date event data can be a challenge, especially when dealing with multiple cities and varying event details. The app needs to pull from reliable data sources, and keeping this information consistent is crucial for the user experience.

Offline Functionality: Implementing offline support for the app can be difficult, especially when dealing with live data. Ensuring that users can access cached event data without issues, and handling scenarios where the cached data may be outdated, requires careful planning and optimization.

Serverless Function Limitations: While serverless functions offer scalability and cost-effectiveness, they can also have limitations in terms of execution time, request handling, and resource constraints. Managing high volumes of data requests or heavy user interaction could present performance issues.

Event Filtering Accuracy: Providing accurate and relevant event suggestions when the user types a city can be challenging, as it requires efficient search algorithms to suggest cities and filter the events based on user input.

Cross-Browser Compatibility: Ensuring the app works seamlessly across different browsers and devices is always a challenge, especially when integrating interactive features like toggling event details and offline functionality.

User Authentication & Security: Managing user accounts, securing personal data, and providing a secure way for users to access their event history or preferences are important aspects of the app, and require effective security practices.

Scalability of Data: As the app grows and more events and cities are added, ensuring the system can handle large amounts of data while maintaining a smooth user experience could be a challenge, especially with serverless backend architecture.

UI/UX Design: Striking the right balance between usability and aesthetic design is a challenge when trying to make the app both functional and user-friendly, especially for diverse users who may interact with the app in different ways.

Duration

- **Week 1**: Initial planning and project setup, including backend API development using serverless functions and designing the database structure for event data.
- **Week 2**: Frontend development with React, focusing on building the event listing page, implementing city search functionality, and designing the UI for event filtering and toggling details.
- **Week 3**: Integrating the event filtering system, allowing users to view events by city and specifying the number of events displayed. Implementing offline functionality and caching to ensure data is accessible without an internet connection.
- **Week 4**: Implementing additional features, such as adding app shortcuts to the home screen and visualizing event data with charts. Conducting cross-browser testing and fixing any bugs related to UI/UX and data handling.
- **Week 5**: Final testing, debugging, and optimizing performance for scalability, followed by deployment and final review of the app.

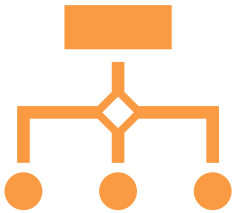
Retrospective

The Meet App project has been an enriching experience that allowed me to blend my web development skills with my passion for creating user-centric applications. Using React for the frontend and serverless functions for the backend challenged me to focus on both performance and scalability, especially when working with real-time data and offline functionality.

Implementing features like filtering events by city, showing and hiding event details, and providing offline access helped me think critically about user needs and optimize the app for a smooth experience. I also learned the value of testing and debugging to ensure the app is fully functional across different scenarios.

This project not only strengthened my technical skills, particularly in React and serverless architectures, but also reinforced the importance of clear and accessible user interfaces. It has fueled my drive to build more intuitive and engaging applications, solidifying my path toward becoming a frontend web developer.

Credits



Developer: Haley Tolar

Tutor: Adam Pagels

Mentor: Shawn Rice