Case Study

Meet App

Serverless Progressive Web Application (React)

Mark Francis

Overview

Meet App is a serverless Progressive Web Application (PWA) built with React using a Test-Driven Development (TDD) approach. The application integrates the Google Calendar API to fetch and display upcoming events while providing users with interactive filtering and data visualization features.

Users can search for events by city, specify the number of displayed events, expand and collapse event details, install the app to their home screen, and access cached data while offline. The application is deployed on Vercel and communicates with a serverless authorization backend powered by AWS Lambda.

https://meet-mu-eight.vercel.app/

https://github.com/CreativeMarkus/meet

This project was developed as part of the CareerFoundry Full-Stack Web Development Program to demonstrate mastery of React, serverless architecture, progressive web app standards, OAuth2 authentication, and data visualization techniques.

Purpose

The purpose of this project was to build a fully functional, production-ready React application using modern web development best practices including TDD, serverless functions, and PWA capabilities.

Objective

The primary objective of this project was to:

- Build a scalable React application using TDD
- Implement OAuth2 authentication with Google Calendar API
- Develop serverless authorization using AWS Lambda
- Ensure full offline functionality with service workers
- Pass Lighthouse PWA requirements
- Implement dynamic data visualization
- Deploy a production-ready application

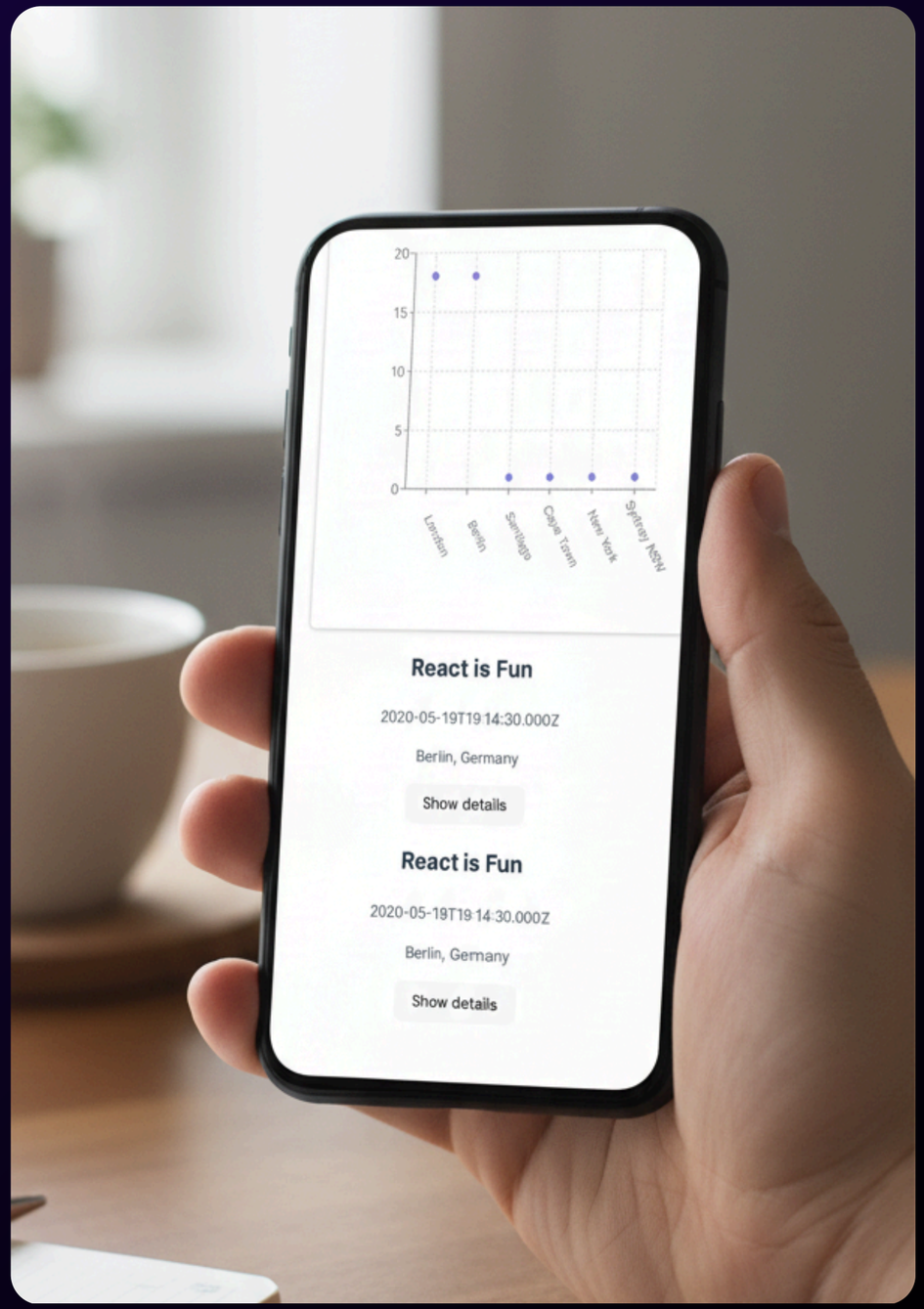
Duration & Credits

The project was developed over several weeks with a strong emphasis on incremental feature delivery using TDD. Each feature was implemented with defined user stories and Gherkin scenarios before development began.
Role: Sole Developer
Responsibility: development, testing, deployment, documentation

Methodologies & Tools

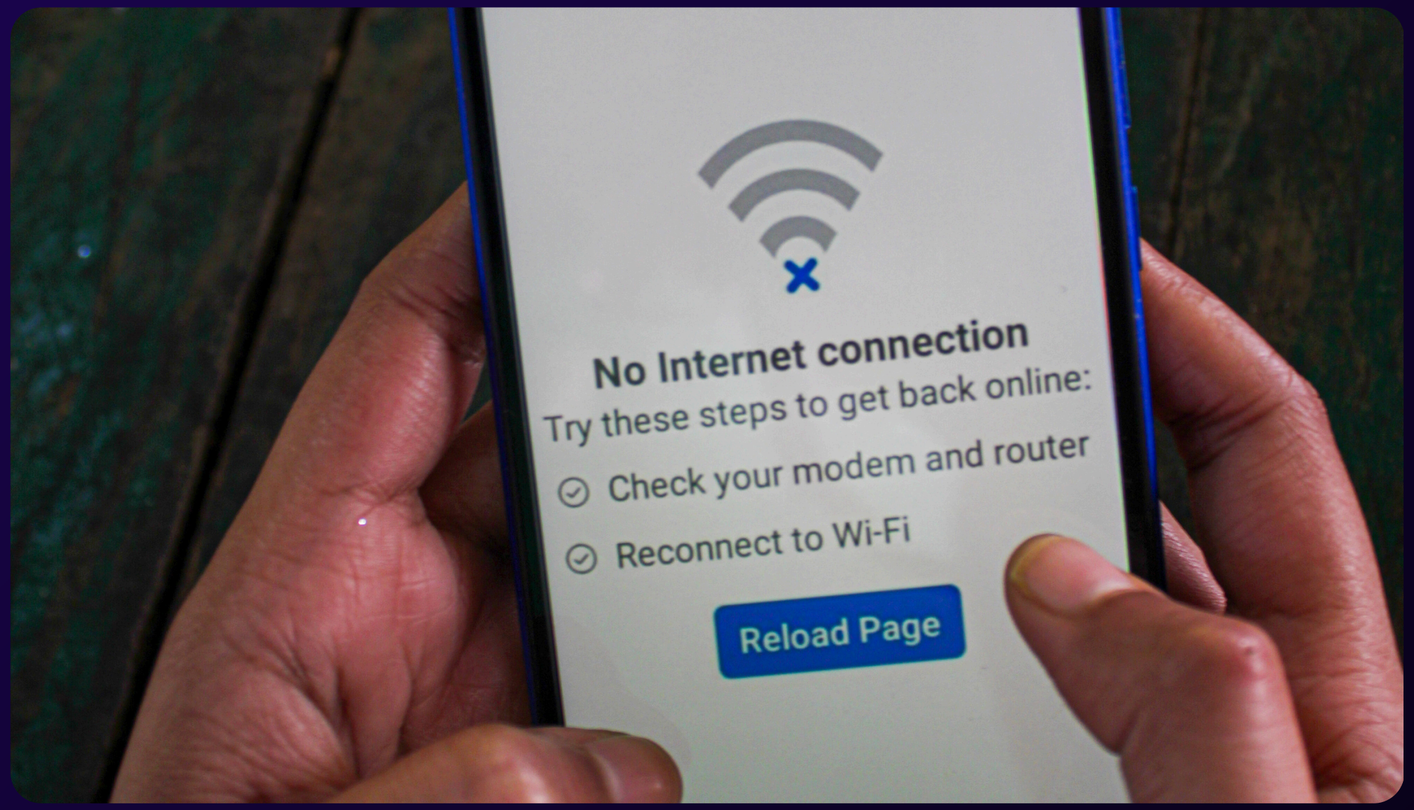
- React
- Vite
- Google Calendar API
- OAuth2 Authentication
- AWS Lambda (Serverless backend)
- Jest & Enzyme (Testing)
- Recharts (Data Visualization)
- Service Workers
- Vercel (Deployment)
- GitHub (Version control)

Meet App



Server-Side Integration

The Meet App connects to the Google Calendar API using OAuth2 authentication. A serverless authorization server built with AWS Lambda securely handles token exchange.
All data is retrieved in JSON format and follows RESTful principles.



- Cached event data is stored for offline usage
- Service workers enable offline functionality
- Error handling notifies users when settings cannot be updated offline

Client-Side Architecture

After API integration, the focus shifted to building a clean and modular React architecture. The application is structured around:

Core Components

- Event list rendering
- City search filtering
- Event count selection
- Charts and visualizations

Feature Components

- Event detail expansion/collapse
- City suggestions dropdown
- Data visualization components

Services

- API communication
- Authorization handling
- Local caching management

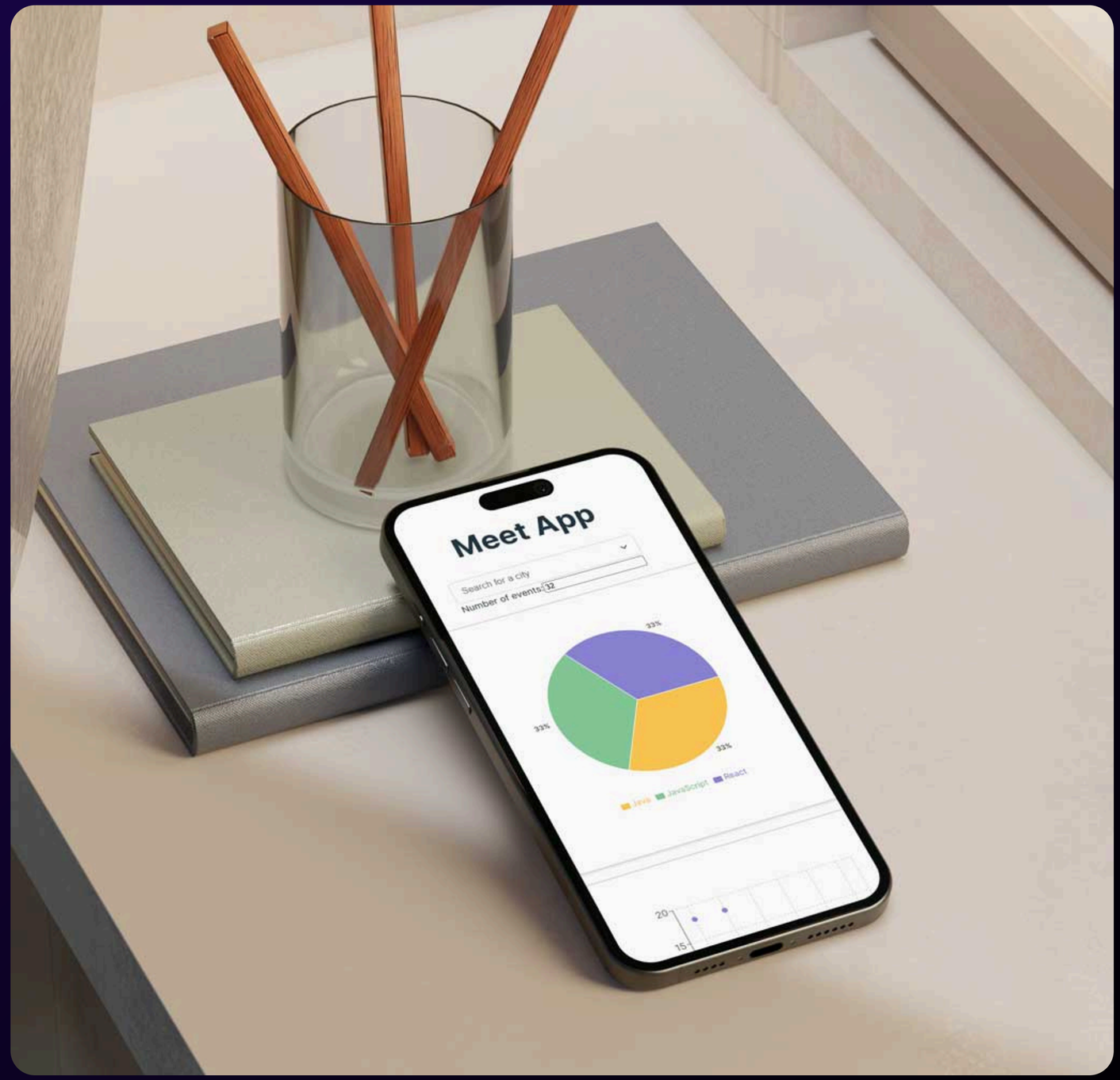
Progressive Web App Features

- Installable on supported devices
- Offline data caching
- Service worker integration
- Lighthouse PWA compliance
- Responsive design across devices

Testing Strategy (TDD)

The application was developed using Test-Driven Development.
Each feature began with:

- User story definition
- Gherkin scenarios
- Unit tests with Jest
- Integration tests



Retrospective

What Went Well

The TDD approach significantly improved development confidence and reduced debugging time. Serverless integration with AWS Lambda was efficient and scalable. Data visualization added strong analytical value to the application.

Challenges

OAuth2 authentication flow required careful token validation logic. Managing offline state updates and service worker caching required additional debugging. Ensuring consistent PWA compliance across browsers took refinement.

Future Steps

- Add advanced event filtering
- Improve UI animations
- Enhance accessibility features
- Expand test coverage
- Add user personalization features

Final Thoughts

This project strengthened my understanding of React architecture, serverless computing, authentication workflows, and progressive web applications. Building a production-ready PWA with data visualization and full offline support provided valuable real-world frontend experience.