Capstone Project  
Event Connect  
*Developed by Mitchell Nilsson*

# Introduction

## Purpose

The primary goal of EventConnect is to connect event planners with potential vendors for various event services, such as catering and security. The focus is on simplifying the process of vendor connection through a centralized platform.

#### Problem or Opportunity

Event planners currently lack an efficient method to connect with vendors, often resorting to manual searches through directories.

This presents an opportunity to streamline the process through a centralized platform.

#### Why is this valuable to address?

Developing an application that optimizes communication between event planners and vendors not only saves valuable time for event planners but also provides vendors with the opportunity to effectively showcase and promote their services.

#### Current State & Desired State

Currently, event planners invest considerable time in reaching out to vendors without guaranteed results. Existing solutions typically offer directories, requiring event planners to contact vendors individually, leading to a less streamlined process.

The desired state is to have a user-friendly application that facilitates seamless communication between event planners and vendors, streamlining the vendor connection process.

##### Previous Project Outcomes

Through my research, I have not identified an existing application on the market that provides the specific service EventConnect aims to offer.

Most available apps are tailored for event planning, concentrating on the execution of events rather than the communication between event planners and vendors.

## Industry

#### What is the Industry / Domain

Event Planning

#### Current State of the Industry

Currently, the event planning industry faces challenges related to the fragmented nature of vendor connections. While directories exist, the lack of a unified platform often results in event planners having to engage with vendors separately, introducing complexities and inefficiencies into the planning process

#### Overall Industry Value-Chain

The industry value-chain involves several key stages, including event conceptualization, service selection, vendor engagement, and event execution. The success of each event relies heavily on the seamless integration of these stages.

#### Key Concepts in the Industry

1. Event Coordination: The overarching concept of organizing and coordinating all aspects of an event, including logistical details, creative elements, and communication with various stakeholders.
2. Vendor Collaboration: The main concept EventConnect is built around; the idea that successful events often involve collaboration with a variety of vendors, such as catering services, security providers, venues, and entertainment.

##### Application Integration

EventConnect streamlines event planner / vendor coordination by offering intuitive tools for planners to draft service requirements and broadcast these requirements to potential service providers (vendors).

#### Relevance to other Industries

While the primary focus is on the event planning industry, the application holds significance for vendors across various industries as well. Vendors utilize the platform to showcase their services, effectively turning the application into a valuable advertising tool for them.

## Stakeholders

The two main stakeholders of EventConnect are Event Planners and Vendors.

#### Role

The Event Planner is the core user of EventConnect. They are responsible for conceptualizing, planning, and executing events.

The Vendor performs a service in relation to an event the Event Planner is organising.

#### Interests

The Event Planner’s interest in EventConnect is due to its capability to steamline the vendor connection process.

The Vendor’s interest in EventConnect is due to the opportunities it provides in promoting the vendors services.

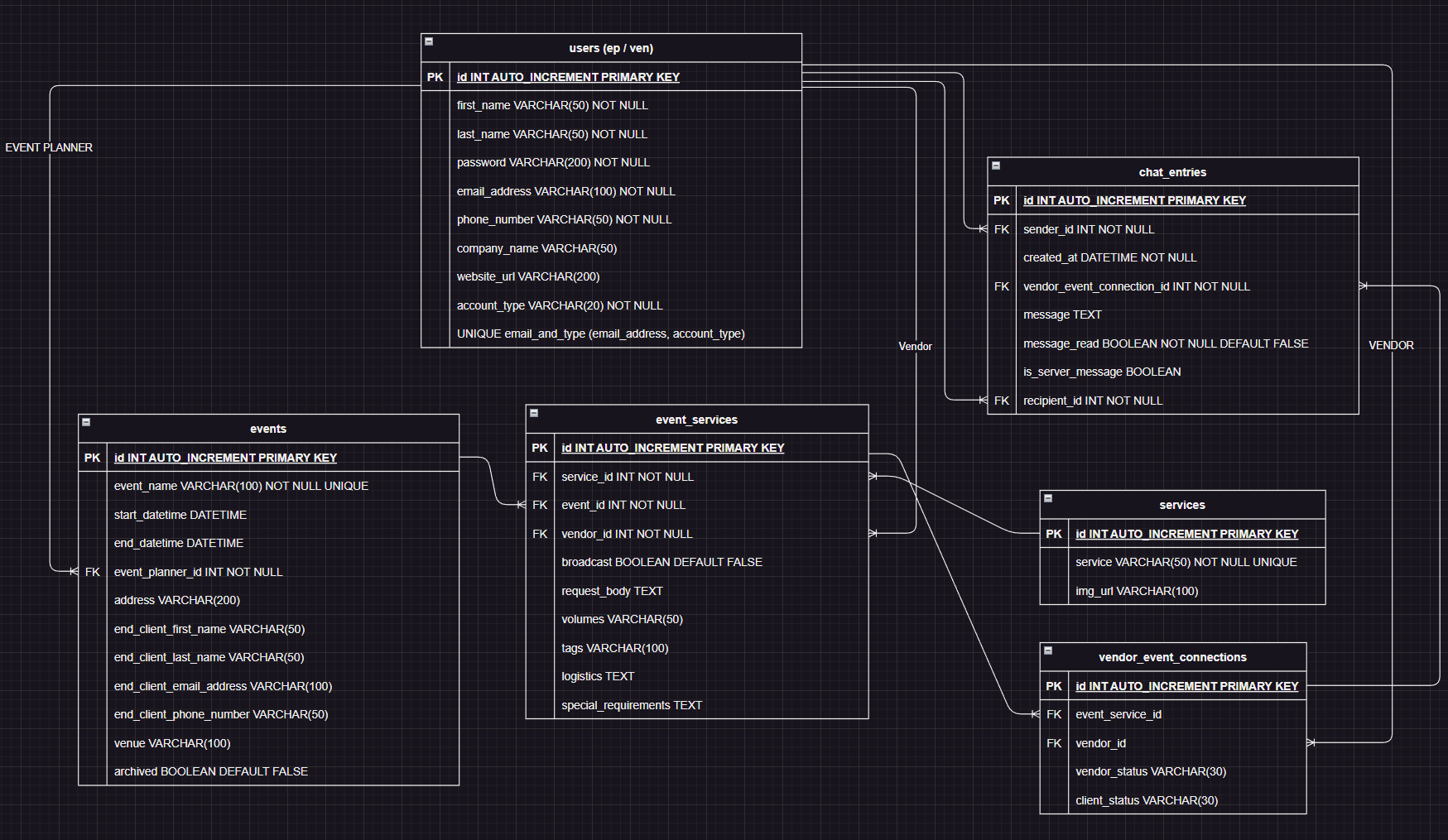
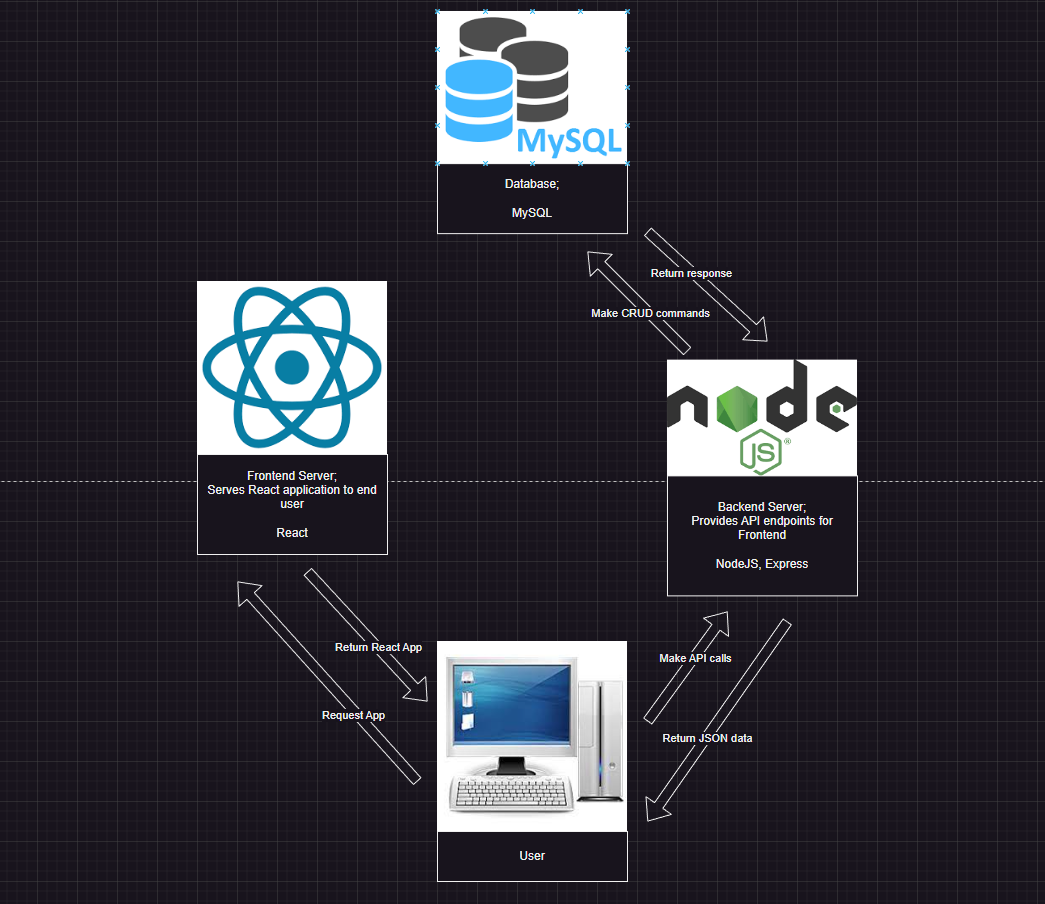
#### Expectations

Both main stakeholders expect an application that is easy to use and efficiently connects them to one another.

# Product Description

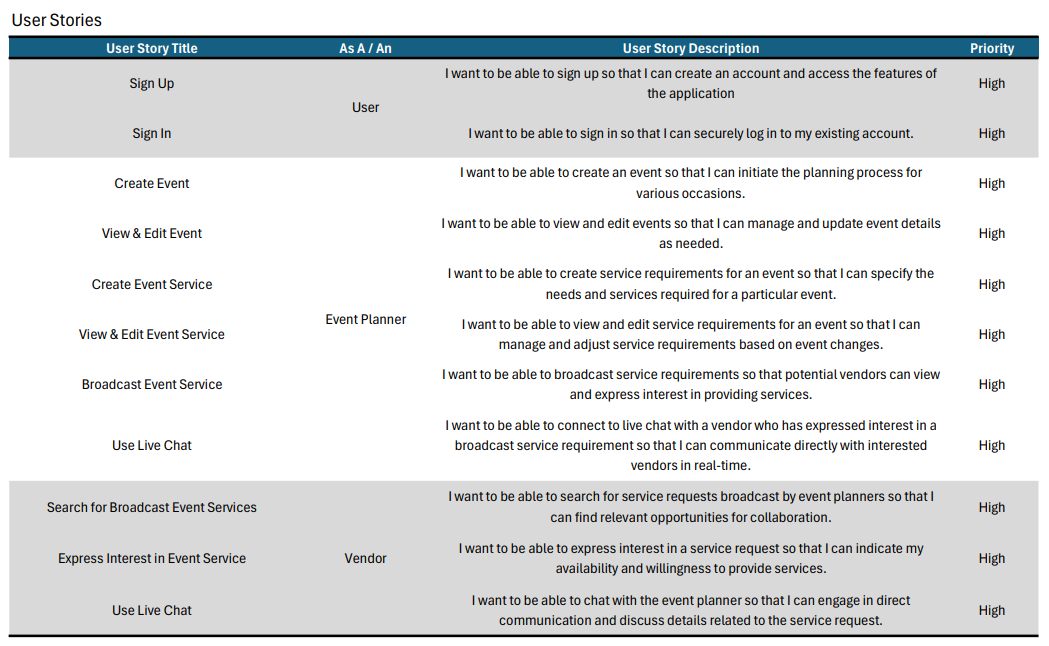
## Architecture Diagram

The Architecture diagram and DB logical model provided below



## User Stories

Some of the main user stories are provided below.



## User Flow

Some of the main user flow chats are provided below. More are available in the Capstone\_Documentation git repository. Due to time constraints, main user flows were focused on.

|  |  |
| --- | --- |
|  |  |
| User onboarding | Create Event |

|  |
| --- |
|  |
| View / Edit & Delete Event |

## Figma Designs

Due to time constraints, the main pages required for the Event Planner side of the application were designed. Other pages are very similar in design to these.

For Figma designs, see the Capstone\_Documentation repository, a Figma file is provided.

## Open Questions / Out of Scope

#### Security Features

* **HTTPS / Encryption**  
  Due to current knowledge limitations, the implementation of HTTPS and encryption is out of scope for the initial release.

#### Additional Features

* **Calendar View**  
  An advanced calendar displaying events within a calendar grid is a desired feature but is currently out of scope for the initial release.
* **Basic User Version**  
  A simplified version of the application designed for general users (non-professional event planners) with limited functionality is a future consideration but is not included in the initial release.
* **End User View**  
  A read-only version of the event for the client of an event planner is a valuable feature, planned for a future iteration beyond the initial release.
* **User Reviews**  
  The capability to leave reviews on event planners and vendors after an event has concluded means future users can make more informed decisions about the vendor or event planner they may work with if they express interest in one another.

#### Payment Requirements

* Payment Requirements  
  The implementation of payment requirements is not included in the initial release. While this is a potential future consideration, it will be assessed based on market feedback and the app's performance after launch.

## Non-functional Requirements

#### Security Requirements

* The application must ensure the secure handling of user data during transmission and storage.
* Due to current knowledge limitations, HTTPS will not be implemented in the initial release. Alternative secure practices, such as careful handling of sensitive data in HTTP requests, should be followed. HTTPS will be considered for future iterations of the application.
* Password hashing will be implemented to enhance database security.

#### Performance Requirements

* The application should be designed to scale and handle increased user loads during peak times, ensuring optimal response times for both basic tasks and transactions.
* For basic tasks, where the application retrieves data from the backend, it should aim to achieve response times of within 500ms.
* In scenarios where the application is processing larger datasets or complex transactional operations, a response time of a around 1000ms is permissible.

#### Usability Requirements

* The user interface must be intuitive, requiring minimal training for both event planners and vendors.
* The application should follow industry best practices for user experience design.

#### Technical Standards

* **Architectural Structure**  
  The backend follows the Model-View-Controller (MVC) architecture to separate concerns, promoting a modular and maintainable codebase.
* **Frontend Structure**The frontend, developed with React, adopts a functional component structure for efficient organization and maintenance.
* **Coding Practices**  
  Standard coding practices are followed, including consistent naming conventions, camelCase used in JavaScript and PascalCase in JavaScript classes.  
  Comprehensive commenting within the codebase ensures clarity and facilitates understanding for future developers.
* **Data Handling**

For frequently accessed data, a strategy of locally updating the frontend state without triggering a full re-fetch will be employed, optimizing performance by minimizing unnecessary data retrieval operations.

* **Version Control**

Separate repositories are maintained for the frontend and backend, each with its dedicated development branch. This approach supports parallel development and ensures clear ownership for both components.

# Project Planning

TRACKING CHART

# Testing Strategy

#### Steps Undertaken to Achieve Product Quality:

To guarantee the application's quality, a third party conducted thorough testing to spot any faults. Furthermore, I cross-referenced the implementation with the initial design specifications in Figma to ensure alignment with intended features and the user interface. While the designs have had minor alterations during the development process, the design adheres to the main principles of the Figma design

#### Testing Each Feature:

Each feature underwent distinct testing methods. For backend functionalities, ThunderClient within Visual Studio Code provided comprehensive assessments. This ensured the server-side components were in optimal condition. Frontend features were individually tested in a controlled environment. Manual inputs were provided, and outputs were observed to ensure correct functionality.

#### Handling Edge Cases:

For edge cases, the use of console.logs allowed me to capture and evaluate data at various stages of the faulty functions, serving as a practical means of identifying potential issues. This proved to be very effective.

# Implementation

#### Considerations for Deploying the Software:

Initially, the plan was to host everything on AWS. However, due to time constraints and a priority on emphasizing the application's functionality, the decision was made to run the front end, backend, and database from a local machine. This approach allows for a more concentrated focus on refining the core features of the application.

# End-to-end Solution

#### Meeting Objectives:

The application has proven successful. While some functionality is out of scope due to time constraints, the core functionality of Event Connect is fully implemented. Event Connect is a fully functional application that enables event planners to draft service requests and seamlessly connect with potential vendors. Beyond meeting its primary objectives, the application shows significant potential for further growth and enhancements.

# References

## Code Repository:

The code for this project is available on github links below;

* **Front End:** <https://github.com/MitchellN89/Capstone_Frontend.git>
* **Back End:** <https://github.com/MitchellN89/Capstone_Backend.git>
* **Documentation:** https://github.com/MitchellN89/Capstone\_Documentation.git

## Resources Used:

Frontend

* React for application design
* Material UI for styling
* Axios for API calls
* Dayjs for date handling
* Dotenv for environment variables
* React Router DOM for managing React routes
* Socket.IO for live chat functionality
* use-places-autocomplete for address search
* react-google-maps/api for Google Maps integration

Backend

* Bcrypt for password hashing
* CORS to allow frontend access to the backend
* Dayjs for date handling
* Dotenv for environment variables
* Express for handling HTTP server
* Jsonwebtoken for authentication tokens
* MySQL2 in conjunction with Sequelize for database handling
* Socket.IO for live chat functionality

Database

* MySQL for database management