**TECHNICAL DOCUMENT FOR THE DEVELOPMENT OF AN EVENT VENUE BOOKING SYSTEM**

A Project Proposal Presented to the

Faculty of Datamex College of Saint Adeline, Inc.

In Partial Fulfillment of the Requirements for the  
Degree of Bachelor of Science in Information Technology

By:

Albania, John Maverick B.

October 2025

**TECHNIAL DOCUMENT**

**INTRODUCTION**

The purpose of this document is to provide a clear technical overview of the Event Venue Booking System created for City of Dreams Manila. This document explains the system's goals, functions, and structure to help developers, administrators, and staff understand how it works. It will also serve as a guide for maintenance, updates, and troubleshooting.

The Event Venue Booking System is a web-based system where clients can visit the website to look for available venues. They can check the details of each venue and choose the one they like. After selecting a venue, they will be directed to the booking page where they need to log in using their Google account before submitting the booking form. On the admin side, the admin can see all the booking requests and check if the client’s requirements like sound system or other equipment are available. If everything is complete, the admin will confirm the booking, and the system will send an email to the client. If not, the admin can cancel or delete the booking request. This system helps make the booking process faster and more organized.

This documentation includes the features, functions, and overall design of the Event Venue Booking System. It also explains how the system works on both the client and admin sides, including the Google login and email confirmation process. This document will serve as a reference for developers and administrators who will maintain or improve the system in the future.

**SYSTEM OVERVIEW**

The Event Venue Booking System is a web-based application that has two main sides the client side and the admin side. The client side is where users can browse venues, select the one they like, and make a booking after logging in with their Google account. The admin side is for the management, where the admin can check all the bookings, review the client’s requests, and either confirm or cancel them. Once confirmed, the system automatically sends an email to the client. The system uses a simple client-server setup where data is sent and received between the user’s browser and the web server.

**High-level components and their interactions.**

The main components of the system include the client interface, admin dashboard, database, and email notification module.

* The client interface handles all user interactions like viewing venues and submitting booking forms.
* The admin dashboard is used by the admin to manage bookings and check client details.
* The database stores information such as venue details, booking records, and client data.
* The email notification module sends confirmation messages to clients once their booking is approved.  
  These components work together to make the booking process smooth and automated from start to finish.

**Deployment architecture.**

Currently, the system is deployed on a local server (localhost) for testing and development purposes. It runs using a local environment setup, where both the client and admin sides can access the system through a web browser. Once fully developed and tested, it can be uploaded to an online hosting platform for public access. For now, all files and the database are stored locally on the developer’s computer.

**INSTALLATION GUIDE**

**System requirements**

**Hardware Requirement**

* Processor: Intel Core i3
* Memory (RAM): 4GB minimum (8GB recommended)
* Storage: At least 500GB HDD or 256GB SSD
* Operating System: Windows 10 or higher
* Device: Laptop or Desktop Computer

**Software**

The system will be developed as a web application using HTML, CSS, and JavaScript for the client-side interface. The backend will be built with Node.js for server-side processing, and Microsoft SQL Server will be used as the database management system. This technology stack ensures scalability, maintainability, and compatibility with modern web hosting environments.

**Software and Tools**

* **Code Editor:** Visual Studio Code
* **Version Control:** Git and GitHub
* **Database:** SSMS

**Dependencies**

The system uses the following Node.js dependencies that are listed in the package.json file:

* **bcryptjs** – for password encryption
* **body-parser** – for handling form and JSON data
* **cors** – for enabling cross-origin access
* **dotenv** – for managing environment variables
* **express** – for creating the server and routes
* **google-auth-library** – for Google account login
* **mssql** – for connecting to SQL Server database
* **nodemailer** – for sending email notifications

**Step-by-Step Instructions for Installing the Software**

**Install Node.js**

Download and install Node.js from the official website. After installation, open Command Prompt and type node -v to check if it’s working.

**Install Microsoft SQL Server (SSMS)**

Install SQL Server and SSMS. Open SSMS and create a new database named EventVenueBooking. This will store all the data for the system like users, venues, and bookings.

**Open the Project Folder in VS Code**

Open Visual Studio Code and load the project folder of the Event Venue Booking System.

**Install Dependencies**

Open the terminal in VS Code and type: npm install

This will install all the required dependencies from the package.json file.

**Set Up the Database Connection**

Go to the configuration file (for example,.env) and update the connection details based on your SQL Server setup

* DB\_USER=admin
* DB\_PASSWORD=admin123
* DB\_SERVER=Your Server Name
* DB\_NAME=EventVenueBooking
* JWT\_SECRET=this is your JWT Secret key.
* PORT=3000
* GOOGLE\_CLIENT\_ID= paste your google key in there.

**Start the Server**

In the termina, type:node server.js

When the message appears saying “Server running on port 300 and Connected to SQL server” the system is ready to run.

**Run the System**

Open your browser and type: http://localhost:3000.The system will open where you can view venues, make bookings, and log in using your Google account.

**Configuration Settings and Options**

* Make sure the SQL Server service is running before starting the Node.js server.
* Update your database connection details (server name, username, password) in your configuration file.
* In the .env file, set the correct values for:
* Database credentials
* Port number (example: 3000)
* Email and app password for Nodemailer
* Google API credentials for login
* When testing the email and Google login feature, make sure your device has internet connection.
* If you plan to deploy the system online later, change the base URL from localhost to your actual domain name.

**CONFIGURATION GUIDE**

**Detailed instructions for configuring the software**

After installing the system, make sure all configuration files are properly set up. Open the .env or configuration file in your project folder and check if the values match your setup. The configuration mainly includes the database connection, email settings, and port number. If you are using Microsoft SQL Server, make sure the server name and credentials are correct so the system can connect properly. You also need to check if your Node.js server is running before testing the system.

**Configuration File Formats and Parameters**

The system uses a .env file to store important configuration values. Below is an example format of the file and what each parameter means:

* DB\_USER=your user here
* DB\_PASSWORD= there is password
* DB\_SERVER=your server name
* DB\_NAME=EventVenueBooking
* JWT\_SECRET=your secret key here
* PORT=3000
* GOOGLE\_CLIENT\_ID= paste your key here

**Description of parameters:**

* DB\_SERVER - the name of your SQL Server (default is localhost)
* DB\_USER - your SQL Server username
* DB\_PASSWORD - your SQL Server password
* DB\_NAME - name of the database used by the system
* PORT - the port number where Node.js runs (default is 3000)
* GOOGLE\_CLIENT\_ID - used for Google account login integration

Make sure to keep this file private and never upload it publicly because it contains sensitive credentials.

**Best Practices for Customization**

* Always keep a backup of your .env file before making any changes.
* Use environment variables instead of hardcoding your credentials in the code.
* If you want to change the port number, make sure it is not already used by another service.
* When changing database names or email settings, update them in both the .env file and related code files.
* Regularly check for updates to your Node.js dependencies to avoid compatibility issues.

**API DOCUMENTATION**

**List of APIs Exposed by the System**

The Event Venue Booking System has several APIs that connect the frontend (HTML, CSS, JS) to the backend (Node.js and SQL Server). These APIs handle different operations such as viewing venues, submitting bookings, logging in with Google, and sending email confirmations.

**Public APIs (no login required)**

* POST /api/login - used for admin or staff login.
* GET /api/venues - fetches the list of available venues.
* POST /api/venues/availability - checks if a venue is available on a specific date and time.
* POST /api/bookings - used by clients to submit a booking form with Google account authentication.

**Admin APIs (login with token required)**

* GET /api/bookings - retrieves all bookings (can be filtered by date, venue, or status).
* GET /api/bookings/:id - retrieves details of a specific booking.
* PUT /api/bookings/:id/status - updates the booking status (Pending, Confirmed, or Cancelled).
* PUT /api/bookings/:id - updates or edits booking details.
* GET /api/calendar - retrieves booking data for the calendar view.
* GET /api/reports - generates reports (venue utilization or event types).

**Endpoint URLs, request/response formats, and parameters**

The system provides several API endpoints used for user authentication, venue booking, and admin management. Each endpoint accepts and returns data in JSON format. Client requests include parameters such as user credentials, booking details, or venue information, while responses contain confirmation messages, booking statuses, or error details.

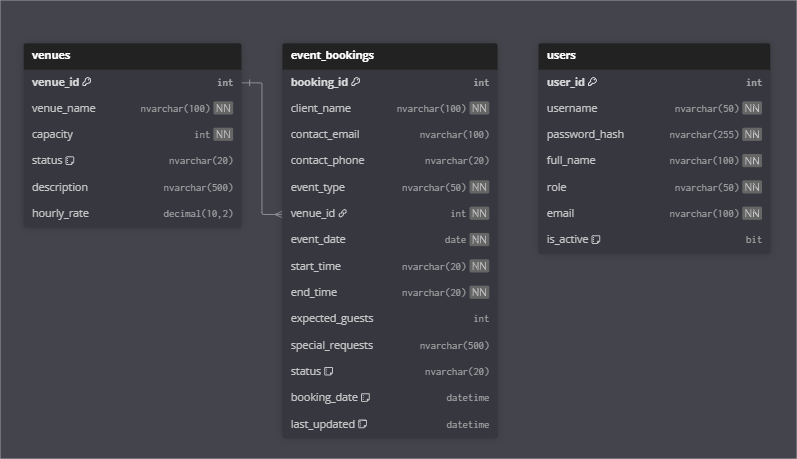
**Authentication and Authorization Requirements**

* Public APIs (such as /api/venues, /api/bookings, and /api/login) do not require authentication.
* Admin APIs require a JWT token, which is obtained from the /api/login endpoint.
* The token must be included in the request header: Authorization: jwt\_token\_here.
* Google authentication is required when clients create a booking (/api/bookings) to verify their email address.

**DATABASE DOCUMENTATION**

The system uses Microsoft SQL Server (SSMS) as its main database for storing and managing data. The database contains several related tables that handle client bookings, venue details, and admin records.

**Entity-relationship diagram**

****

**Image 1:** ERD image

**Description of database tables**

|  |  |
| --- | --- |
| **Table Name** | **Description** |
| users | Stores all registered admin and client accounts. It includes fields such as user\_id, username, password\_hash, full\_name, role, email, and is\_active. |
| venues | Contains all available venue information. Fields include venue\_id, venue\_name, capacity, status, description, and hourly\_rate. |
| event\_bookings | Keeps track of all booking transactions made by clients. It includes details such as booking\_id, client\_name, contact\_email, contact\_phone, event\_type, venue\_id, event\_date, start\_time, end\_time, expected\_guests, special\_requests, status, booking\_date, and last\_updated. |

**Table 1: Database Table**

**Data Relationships**

* The event\_bookings table links to the venues table through the venue\_id field.
* Each record in event\_bookings represents a specific venue booked by a client for a chosen date and time.
* The users table stores information for system users (admins and possibly clients), although client details are also stored in event\_bookings for convenience and record tracking.

**Data Migration and Backup Procedures**

Backups can be performed manually using SQL Server Management Studio by exporting the database into a .bak file. To restore or migrate data, this backup file can be imported on another machine using the restore feature in SSMS. Regular database backups are recommended to maintain data integrity and prevent loss of booking or user information.

**USER MANUAL**

The system is simple to use and can be accessed through a web browser.

**How to Use the System**

1. **Visit the Website:**

Open the system in your browser using the provided local URL (for example, http://localhost:3000)

1. **Browse Venues:**

Go to the Venues page to view all available event venues, including details such as capacity, rate, and description.

1. **Select a Venue:**

Click on the venue you want to book. This will open the *Booking Page*.

1. **Sign In with Google:**

Before submitting a booking, the user needs to log in using their Google account.

1. **Fill Out the Booking Form:**

Enter the required booking details such as event type, date, time, expected guests, and any special requests.

1. **Submit the Booking:**

After filling out the form, click *Submit*. The request will be sent to the admin for review.

1. **Admin Review and Confirmation:**

On the admin side, the admin will check the request. If the venue and equipment are available, the booking will be confirmed; otherwise, it may be canceled or deleted.