Crumbley, Victoria
Southern New Hampshire University
CS 499 Computer Science Capstone
November 30, 2025

Event Tracker Application Development Proposal

1) Project Goals

a) Overview

The goal is to deliver a user-friendly event tracking service that efficiently manages the entire process, from searching for tickets to post-event management. Key functionalities include managing event ticket searches and bookings, facilitating the payment process, and effectively handling ticket organization. A reliable notification system will keep users informed about their events and ticket status. The ticket purchase process is designed with a heavy focus on encryption, ensuring security and integrity of all transactions.

b) Architecture and Components

The major components comprise a secure backend infrastructure, the database schema, and the user interface, which work together to deliver the application's core service.

i) Secure Database Structure

(1) Users Collection:

Field	Data Type	Description
user_id	**ObjectId**	Unique record ID (primary key)
username	**String**	User's unique identifier
password_hash	**String**	One-way hash
email	**String**	User's email address
phone	**String**	User's phone number

(a) Stores user-centric data required for authentication and communication

(b) Unique identifiers for username, email, and creation of id number; *user_id* wil be indexed for app admin

(c) *Bcrypt* algorithm secures password; runtime no more than 2 seconds; auto-generate a salt and hash on a single function call; used to store *password_hash*

(2) Events Table:

Field	**Data Type**	**Description**
event_id	**ObjectId**	Unique record ID (Primary Key)
user_id	**ObjectId**	User who created/manages this event
event_name	**String**	Title of event
date	**ISODate**	Date of event
time	**String**	Time of event
location	**String**	Venue or location details
description	**String**	Detailed event information
image_url	**String**	URL to event image banner
ticket_url	**String**	URL for purchasing tickets

(a) Stores details about tracked events

(b) No duplications of event name, date, and time to avoid errors**;** event details (*event_id, event_name, date, time, and location)* will be indexed for all users

(3) Tickets Table:

Field	**Data Type**	**Description**
booking_id	**ObjectId**	Unique record ID (Primary Key)
user_id	**ObjectId**	Links booking to purchasing user
event_id	**ObjectId**	Links booking to specific event
purchase_date	**ISODate**	Timestamp of purchase
quantity	**Integer**	Number of tickets purchased
encrypted_price	**String**	Total transaction amount
booking_status	**String**	Status of booking process

(a) Manages ticket records linked to events and users, crucial for post-event management

(b) At-rest encryption in *encrypted_price*

(4) Payments Table:

Field	**Data Type**	**Description**
transaction_id	**ObjectId**	Unique record ID (Primary Key)

booking_id	**ObjectId**	Links payment to specific ticket booking
encryption_key_id	**String**	Identifier for key used in transaction encryption
transaction_status	**String**	Status of payment
transaction_date	**ISODate**	Timestamp of the transaction completion

(a) Manages encrypted transaction details for security and auditing

(b) Implementing a **Key Management System (KMS)**, tied to the *encryption_key_id*, will only allow the database to store encrypted data and a reference ID – never the actual decryption key. Attackers would gain only access to encrypted values, still needing to breach the highly secured KMS to retrieve the decryption keys.

ii) Notification Service Module: a dedicated server-side (backend) module to manage and dispatch reminders via email, SMS, and in-app notifications.

iii) Payment Integration: a secure API integration with a certified payment processor (e.g., Stripe, PayPal) to handle all encrypted financial transactions.

c) Functionalities

These primary features enable a secure and user-friendly experience from sign-up to post-event management.

i) Secure Authentication and Registration

A cohesive “Login/Registration Screen”

(a) Features strong password requirements, email/SMS verification, and robust input validation

(b) Passwords will be stored as one-way hashes

ii) Main Event Discovery Interface

Display for events in a clean grid/card layout with chronological order as the default view

(a) Includes filter and search functionalities based on location, event type, artist/performer, and date range

iii) Personal Event Management (CRUD)

Allows users to Create, Read, Update, and Delete their personal events

(a) Adding new events, modifying existing ones, marking as attended or deleted

iv) Proactive Event Notification System

A system to remind users of upcoming events.

(a) Includes customizable reminder settings and a notification history log

v) Encrypted Booking and Purchase Workflow

A dedicated and secure workflow for ticket purchasing that uses payment integration

(a) Features direct linking for ticket purchases and ensures all payment data is encrypted both in transit and at rest

2) Target Users and Assumptions

a) This table analyzes key user groups for the Event Tracking, their general purpose, and their primary goals.

User Type	**Description**	**Primary Goals**
General Users	Those who utilize the app for personal event organization,	- Stay organized - Receive timely reminders

	simple tracking, and occasional ticket booking	- Securely track purchased tickets for personal use
Avid Eventgoers	Highly active users who frequently search for, book, and attend events (concerts, sports games, theatre performances)	- Discover new events quickly - Manage multiple bookings - Integrate event schedules into their digital calendars
Event Coordinators	Professionals, or dedicated users, responsible for organizing and managing public and private events within the application	- Effectively add, modify, and delete events - Monitor event performance - Utilize notification tools for attendee updates
Artists/Performers	Individuals or groups who use the platform to promote their own scheduled shows, appearances, or tours	- Maximize visibility for upcoming events - Ensure accurate event details are shown - Drive traffic to the booking system
Application Administrators	Internal personnel responsible for	- Ensure data accuracy and

	maintaining the application's infrastructure, ensuring data integrity, and providing support	security compliance - Manage user accounts - Maintain smooth operation of the notification system and the payment integration

b) Assumptions

i) User expects a clean, minimalistic interface with minimal clicks required to complete tasks

ii) The Notification System must have top-notch delivery reliability across multiple channels (in-app, email, SMS)

iii) Users assume that all personal and financial data is handled with industry-standard encryption and compliance protocols

iv) Personal event data is user-generated, but event discovery relies on API integration with third-party ticketing services or public event databases

v) The application will primarily be used on mobile devices, and the system must be designed for cross-platform compatibility

vi) Successful ticket purchasing is entirely dependent on the stability and availability of the external Payment Integration

3) UI Screens and Features

This section outlines the specific user interfaces and the critical features they enable.

a) Splash Screen

i) Displays the branding logo and a short Loading Animation

ii) Transitions immediately to Login/Registration Interface

b) Login/Registration Interface

i) Input fields (username/email, password)

ii) "Login" button that triggers hash validation

iii) Direct button to the Account Creation fields

c) Secure Account Creation

i) Fields for full name, email, phone, and password

ii) Triggers email/SMS verification upon submission

iii) Toggle for push notifications and location access

iv) "Create Account" button, stores password as a hash

d) Main Event Discovery View

i) Events displayed in a grid/card layout, set default chronological

ii) Persistent search bar with filter options: date range, location, event type, or artist(s)/performer(s)

e) Event Details View

i) Large image banner and description

ii) Date, time, venue/location details

iii) "Book Now" button leading to Encrypted Booking Workflow

iv) Display of related or nearby events

f) Encrypted Booking Workflow

i) Quantity and seat selection (if applicable)

ii) Displays ticket pricing, service fees, and total cost

iii) Secure interface for entering payment details

****Option to save encrypted payment information for future use*

g) User Profile & Preferences

i) Greeting message, list of upcoming events, and past attended events (card/view list)

ii) Reminder customization and profile editing

h) Event Management Interface (CRUD)

i) Fields for title, date, time, location, and description

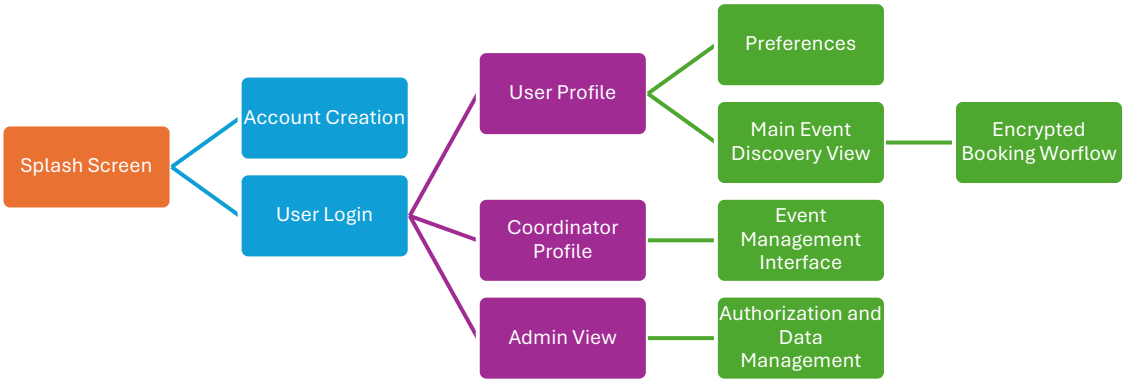
ii) Image uploader tool for the event image banner

iii) "Save/Update", "Publish", and "Delete" buttons

i) Authorization and Data Management

i) Interface for user management and content moderation

ii) Overview of system health, security logs, and user activity



4) Code Design

This application will be built as a MEAN stack application, utilizing MongoDB as the database, Express.js and Node.js for the backend API, and Angular for the frontend client. The Angular frontend will adhere to the Model-View-ViewModel (MVVM) architectural pattern to ensure a clean separation of concerns, enhance testability, and facilitate maintenance

a) Full-Stack Architecture

i) Database

The application will utilize **MongoDB** , which is a **NoSQL database**. MongoDB was selected because it is **scalable** for the storage and growth of user, event, ticket, and payment data. Its **flexible structure** is sufficient to handle various complex data needs, such as storing artist details or managing authorization verification processes.

ii) Backend Framework

The backend is built upon **Node.js** and **Express.js**, which together provide the **RESTful API layer**. This layer is essential for handling all **CRUD (Create, Read, Update, Delete) operations** , managing **user authentication**, and enforcing **server-side security** measures. It acts as the critical bridge between the frontend application and the MongoDB database.

iii) Frontend

The frontend is the **client-side application**, developed using **Angular**. It is responsible for all **UI rendering**, managing user interactions , and ensuring a clean application structure by implementing the **Model-View-ViewModel (MVVM) architectural pattern**. This pattern ensures a clear separation of concerns, which enhances testability and simplifies maintenance.

b) MVVM Architecture

i) Model:

This layer manages data retrieval and submission. It communicates directly with the Express.js API endpoints to fetch or submit data; this is the bridge between UI logic and backend database.

ii) View:

This layer renders the user interfaces and handles all direct user interactions and events

iii) ViewModel:

It contains logic for features like Search/Filtering, Reminder Settings, and Role-Based Routing after login.

c) Security and System Services

i) Backend Security

(a) Authentication: user registration processes hashing of passwords before storage in MongoDB

(b) Data Encryption: sensitive fields in the Payments collection will be encrypted before being written to MongoDB

ii) Notification Service Module

(a) Node.js Backend: responsible for scheduling and dispatching notifications via third-party services

(b) Express.js Controllers: logic triggers the backend API

d) UI/UX Component Inventory

i) Core Navigation

(1) **Bottom Navigation Bar:** implemented by utilizing Angular routing and component composition

ii) Key Components

(1) **Event Card View, Encrypted Booking Workflow, and Event Management Interface:** built by Angular components