Crumbley, Victoria
Southern New Hampshire University
CS 499 Computer Science Capstone
Friday, November 28, 2025

Event Tracker Application Development Proposal

I. 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 Table:** stores user-centric data required for authentication and communication
   (user_id, username, password_hash, email, phone)
2. **Events Table:** stores details about tracked events
   (event_id, user_id, event_name, date, time, location, description, image_url, ticket_url)
3. **Tickets Table:** manages ticket records linked to events and users, crucial for post-event management
   (booking_id, user_id, event_id, purchase_date, quantity, encrypted_price, booking_status)

4. **Payments Table:** manages encrypted transaction details for security and auditing
   (transaction_id, booking_id, encryption_key_id, transaction_status, transaction_date)

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

II. 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, simple tracking, and occasional ticket booking	- Stay organized - Receive timely reminders - Securely track purchased tickets for personal use
Avid Eventgoers	Highly active users who frequently search for, book, and attend events (concerts,	- Discover new events quickly - Manage multiple bookings

	sports games, theatre performances)	- 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 maintaining the application's infrastructure, ensuring data integrity, and providing support	- Ensure data accuracy and 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

III. 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*

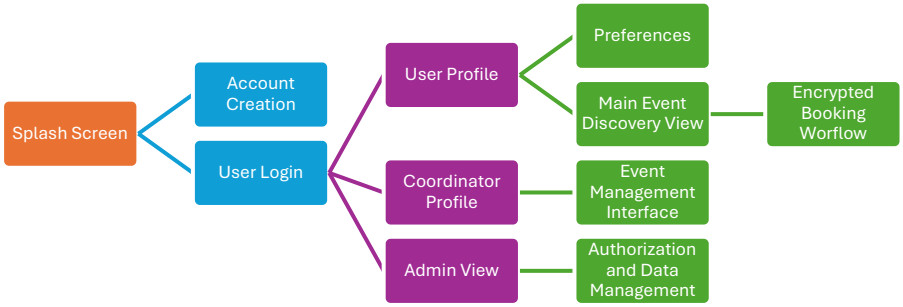
g. **User Profile & Preferences**
   i. Greeting message, list of upcoming events, and past attended events (card/view list)
   ii. Reminder customization and profile editing

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



IV. 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** – NoSQL database; scalable for storage of Users, Events, Tickets, and Payment data
ii. **Backend Framework** – Provides the RESTful API layer handling of all CRUD operations, authentication, and server-side security
iii. **Frontend** – The client-side application responsible for all UI rendering, user interaction, and implementing the MVVM pattern

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**

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

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

ii. **Notification Service Module**

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

Express.js Controllers: logic triggers the backend API

d. **UI/UX Component Inventory**

i. **Core Navigation**

Bottom Navigation Bar: implemented by utilizing Angular routing and component composition

ii. **Key Components**

Event Card View, Encrypted Booking Workflow, and Event Management Interface: built by Angular components