Entertainia – Documentation

---

1. Project Planning

1.1 Overview

Entertainia is a centralized digital platform designed to gather all types of events — entertainment, cultural, sports, and more — in one place.
The platform enables users to **browse, explore, and book events** happening across Egypt, making event discovery and ticket booking simple and accessible.
Entertainia aims to become the go-to destination for event lovers by providing up-to-date event listings, location-based browsing, and secure online bookings.

1.2 Objectives

- To create a **unified events platform** that aggregates multiple event categories.
- To enable users to **book and pay** for event tickets securely online.
- To support **venue organizers** in publishing and managing their events efficiently.
- To improve **accessibility and visibility** of cultural, sports, and entertainment activities across Egypt.
- To design a **scalable architecture** capable of future regional or global expansion.

1.3 Scope

The current scope of the project includes:

- Developing a **web-based platform** using ASP.NET Core MVC.
- Implementing a **secure authentication and authorization system** for users and organizers.
- Building **event management features**, including event creation, editing, and cancellation.
- Providing a **booking system** for users with integrated payment and refund options.
- Supporting **notifications** and **email confirmations** for successful bookings and payments.
- The platform is currently focused on **Egyptian users**, but structured for scalability.

1.4 Features Summary

Feature	Description
Event Discovery	**Users can browse and search events by category, date, or location.**
Event Management	**Organizers can add, update, and manage their events easily.**
Online Booking	**Users can book tickets through an interactive booking system.**
Payment Integration	**Secure payment and refund system using Microsoft SQL and EF Core.**
Email Notifications	**Automated confirmations and reminders for users and organizers.**
Role-Based Access	**Admins manage users, events, and approvals through a dashboard.**
Scalability	**Designed to expand to regional and global audiences.**

1.5 Technologies & Tools

Category	Tools / Technologies
Backend Framework	ASP.NET Core 8.0 (MVC)
Database	Microsoft SQL Server with Entity Framework Core
Frontend	HTML, CSS, JavaScript, Bootstrap
Email & Notifications	SendGrid API, SignalR
IDE & Version Control	Visual Studio 2022, GitHub
Architecture Pattern	Clean Architecture (Core, Infrastructure, Presentation Layers)

1.6 Development Phases

Phase	Description	Status
Phase 1: Requirements Gathering	Define features, user roles, and data flow.	Completed
Phase 2: System Design	Define architecture, database design, and UI flow.	Completed
Phase 3: Backend Implementation	Develop models, repositories, and services.	In Progress
Phase 4: Frontend Integration	Connect backend APIs to frontend views.	In Progress
Phase 5: Testing & Deployment	Perform testing and deploy MVP version.	Upcoming

2. Stakeholder Analysis

2.1 Key Stakeholders

Stakeholder	Role	Interest / Responsibility
Platform Administrators	Manage the system, approve event organizers, handle issues, and monitor performance.	High
Event Organizers	Create and manage events, set prices, and monitor ticket sales.	High
End Users (Attendees)	Discover, book, and pay for events securely.	High
Payment Gateway Provider	Ensures secure transaction processing and refund operations.	Medium
Developers	Responsible for maintaining and enhancing the platform.	Medium
Government / Cultural Authorities	Potential data validation or promotional collaborations.	Low–Medium

2.2 Roles and Responsibilities

- **Administrators: Oversee all operations, user roles, and platform integrity.**
- **Organizers: Post events, manage bookings, and view analytics.**
- **Users: Browse and book events based on interests and availability.**
- **Developers: Maintain system reliability, add new features, and fix bugs.**

2.3 Communication and Decision Flow

1. **Users and organizers interact primarily through the Entertainia platform.**
2. **Administrators handle event approvals, refund requests, and conflict resolution.**
3. **Developers coordinate with administrators for updates and system enhancements.**

---

3. Database Design

3.1 Overview

The Entertainia database is built using Microsoft SQL Server, with Entity Framework Core managing the ORM layer.
The design follows a normalized relational structure that ensures data integrity, scalability, and efficient query performance.

3.2 Main Entities

Table	Description	Key Fields
Users	**Stores information about registered users and organizers.**	**UserId, FullName, Email, PasswordHash, Role, AccountStatus**
Venues	**Contains details about event venues.**	**VenueId, Name, Location, Capacity**
Events	**Represents all types of events (entertainment, cultural, sports, theater, tourism).**	**EventId, Title, Description, Category, StartDate, EndDate, VenueId, OrganizerId**
EventTicketTiers	**Defines ticket categories (Regular, VIP, etc.) with pricing and availability.**	**TierId, EventId, TierName, Price, Capacity**
Bookings	**Records users' event bookings.**	**BookingId, UserId, EventId, TierId, BookingDate, Status**
Payments	**Manages payment transactions for bookings.**	**PaymentId, BookingId, Amount, Currency, PaymentStatus, PaymentDate**
Refunds	**Stores refund requests and statuses.**	**RefundId, PaymentId, Reason, RefundStatus, RequestedDate**
Notifications	**Tracks user notifications and system alerts.**	**NotificationId, UserId, Message, DateSent, IsRead**

3.3 Relationships

- **One-to-Many:**
  - **One User → Many Bookings**
  - **One Event → Many Ticket Tiers**
  - **One Event → Many Bookings**
  - **One Venue → Many Events**
- **One-to-One:**
  - **One Booking ↔ One Payment**
  - **One Payment ↔ One Refund (optional)**
- **Many-to-Many (implicit through Bookings):**
  - **Users ↔ Events**

3.4 Data Flow Summary

1. **Organizers create events and define ticket tiers.**
2. **Users browse events and initiate bookings.**
3. **The system processes payments and generates notifications.**
4. **Admins manage events, refunds, and platform activities through a dashboard.**