1. Introduction

Sports Booking Platform is a smart, event-driven **sports venue discovery and slot-booking system** built with **.NET**, enabling users to discover venues, lock and book slots concurrently, create/join games, pay via wallet, receive refunds, and submit ratings. It emphasizes **dynamic slot pricing**, **distributed locking**, **background processing**, **analytics**, and **secure multi-role authorization**.

2. Problem Statement

Design and implement a modular, scalable **Sports Booking Platform** where:

- **Venue Owners** register venues/courts, configure base prices, operating hours, and discounts
- **Admins** approve venues, oversee analytics, and enforce governance
- **Users** discover venues, view dynamically priced slots, lock slots, confirm bookings with wallet payments, cancel bookings with refunds, create/join games, use waitlists, and submit ratings
- The system runs **background services** to expire locks, auto-cancel games, process refunds, expire discounts, and compute historical demand signals.

3. Functional Requirements

Using the application, the user should be able to register/login, browse venues and slots with dynamic pricing, lock and confirm bookings safely under concurrency, manage wallet payments/refunds, create and manage games, use waitlists, and submit ratings based on assigned roles.

3.1 Venue & Court Management

Every venue has an owner and must be **approved by Admin** before it can accept bookings.

Venue Attributes (minimum):

- VenueId, Name, Address,SportsSupported
- OwnerId
- ApprovalStatus (Pending/Approved/Rejected)
- CreatedAt

Court/Field Attributes (minimum):

- CourtId, VenueId, Name, SportType
- SlotDurationMinutes (e.g., 30/60)
- BasePrice
- OperatingHours (open/close per day)
- IsActive / Availability flag

Discount Attributes (minimum):

- DiscountId, Scope (Venue/Court), VenueId, CourtId
- PercentOff (or DiscountFactor), ValidFrom, ValidTo
- IsActive

Constraints / Validations:

- Venue must be **Approved** before any slot can be booked.
- Courts cannot be deleted if **future bookings** exist.
- Discount validity period must be valid (ValidFrom < ValidTo); expired discounts cannot be applied.

API Endpoints (sample):

Method	Endpoint	Role	Decription
POST	/api/venues	VenueOwner	Register a venue
PUT	/api/venues/{id}/approve	Admin	Approve/reject venue
GET	/api/venues	Any	List venues (filters optional)
POST	/api/venues/{venueId}/courts	VenueOwner	Create court/field
PUT	/api/courts/{id}	VenueOwner	Update court settings
DELETE	/api/courts/{id}	VenueOwner	Delete court (if allowed)
POST	/api/discounts	VenueOwner	Create discount (venue/court scoped)
GET	/api/discounts	VenueOwner/Admin	List discounts

3.2 Slot Availability & Dynamic Pricing Engine

Objective

Adjust slot prices in real-time based on **demand**, **time-to-slot**, **historical popularity**, and **discounts**.

Formula

final_price = base_price × demand_multiplier × time_based_multiplier × historical_multiplier × discount_factor

Multiplier Definitions

Demand Multiplier (from real-time viewers)

- 1.0 if no viewing
- 1.2 if 2–5 users viewing
- 1.5 if >5 users viewing

Time-based Multiplier

- 1.0 if >24 hrs away
- 1.2 if 6–24 hrs away
- 1.5 if <6 hrs away

Historical Popularity Multiplier

- 1.0 for low-demand slots (rating 1–2)
- 1.2 for medium-demand slots (rating 3)
- 1.5 for high-demand slots (rating 4–5)

Discount factor

- Derived from configured discounts; must be validated for scope and time window.

Implementation Notes

- Use **Caching** to track real-time slot (or game/slot page) views.
- Historical multiplier computed via background job over past bookings + ratings.
- **Price locked for 5 minutes** post computation.
- Final price must be shown before payment and must be revalidated on confirm.

API Endpoints

Method	Endpoint	Role	Description
GET	/api/slots/available	Any	Returns available slots + dynamic price
POST	/api/bookings/lock-slot	User	Locks slot + locks computed price (5 mins)

3.3 Slot-Based Booking System (Concurrency-Safe)

Booking Statuses

- Pending
- Confirmed
- Cancelled
- Expired
- Locked

Booking Flow

1. User requests available slots → system computes dynamic price (incl. discounts)
2. User locks slot → system stores locked price and lock expiry (5 minutes)
3. User confirms booking → wallet is debited and booking becomes Confirmed
4. If lock expires or payment fails → booking/lock released

Requirements / Constraints

- Prevent **double-booking** under concurrent requests.
- Slot locking must use **Caching/distributed lock** with expiry-based release.

API Endpoints

Method	Endpoint	Role	Description
POST	/api/bookings/lock-slot	User	Lock a slot + price for 5 mins
POST	/api/bookings/confirm	User	Confirm booking; validates lock and price
PUT	/api/bookings/{id}/cancel	User	Cancel booking; triggers refund rules
GET	/api/bookings	Admin/User	List bookings (filters optional)

3.4 Wallet Management, Payments & Refunds

Every user has a wallet upon registration.

Wallet Features

- Add funds (mocked; idempotent)
- Get balance
- Transaction history (audit)

Transaction Types

- CREDIT: add funds, refunds
- DEBIT: booking payments, game fees

Constraints

- Wallet balance cannot go negative.
- All booking payments must use wallet.
- Booking confirmation + wallet debit must be **ACID** (single transactional unit).
- Idempotency required for add-funds and refund processing.

Refund Rules

- 24 hrs before slot: 100% refund
- 6–24 hrs: 50% refund
- <6 hrs: 0% refund
- If venue/court marked unavailable by owner: **full refund** regardless of window

API Endpoints

Method	Endpoint	Role	Description
POST	/api/wallet/add-funds	User	Add funds (idempotent)
GET	/api/wallet/balance	User	Get wallet balance
GET	/api/wallet/transactions	User/Admin	Transaction history
POST	/api/bookings/confirm	User	Debit wallet and confirm booking

3.5 Game Management System

Game Features

- Create public/private games
- Associate game with venue + slot (must align to a booking/slot)
- Join/leave
- Enforce min/max players

Constraints

- Cannot exceed max players
- Auto-cancel if min players not met before start time

API Endpoints

Method	Endpoint	Role	Description
POST	/api/games	User/GameOwner	Create a game (assign GameOwner if first)
PUT	/api/games/{id}/join	User	Join game (capacity enforced)
GET	/api/games/{id}/leave	User	Leave game
PUT	/api/games	Any	List games (filters optional)

3.6 Waitlist System

Features

- Join waitlist when game is full

- Waitlist capped (e.g., 10) (configurable)
- Sorted by player rating
- Game owner can invite from waitlist
- Notifications hook (can be mocked/logged)

Constraints

- Waitlisted users removed once game starts or invited

API Endpoints

Method	Endpoint	Role	Description
POST	/api/games/{id}/waitlist	User	Join waitlist
GET	/api/games/{id}/waitlist	GameOwner	View waitlist
POST	/api/games/{id}/invite	GameOwner	Invite waitlisted player

3.7 Rating, Review & User Profile

Ratings only after game completion.

Rating Targets

- Venue (1–5)
- Court (1–5)
- Player (1–5)

Constraints

- Only after game status = COMPLETED
- One rating per user per game per entity

User Profile

- Aggregated player rating
- Number of games played
- Preferred sports
- Recent reviews

API Endpoints (sample):

Method	Endpoint	Role	Description
GET	/api/users/{id}/profile	Any	View player profile
POST	/api/ratings/venue	User	Rate a venue
POST	/api/ratings/court	User	Rate a court
POST	/api/ratings/player	User	Rate a player
GET	/api/venues/{id}/ratings	Any	Venue rating summary

GET	/api/courts/{id}/ratings	Any	Court rating summary
GET	/api/users/{id}/ratings	Any	Player rating summary

3.8 Roles and Permissions

Roles:

1. **Admin**
    - Approve venues
    - View analytics/dashboards
    - Access all bookings, wallet transactions, refunds, ratings
2. **Venue Owner**
    - Manage own venues/courts/operating hours/base price
    - Manage discounts and availability flags
3. **Game Owner**
    - Create games, manage participants, invite from waitlist
4. **Normal User**
    - Book slots, pay via wallet, cancel and receive refunds
    - Join games/waitlists
    - Submit ratings after completion

Authorization Rules

- Token required for all POST/PUT/DELETE operations.
- VenueOwner can only modify their own venues/courts/discounts.
- GameOwner can only invite/manage their own games.
- Admin override capabilities for governance tasks.

4. Milestones

Milestone – 1 Core Setup & User/Roles

1. Setup boilerplate with .NET 8; run migrations
2. Implement authentication (Register/Login with token)
3. Implement role model and assignment (Admin, Venue Owner, Game Owner, Normal user)
4. Add role-based authorization middleware for protected endpoints.
5. Swagger integration + documentation
6. Proper git commits per feature

Milestone – 2 – Venue, Courts, Game and Discounts management

1. Venue registration by owners and admin approval workflow.
2. Court/field creation per venue with operating hours, slot duration, base price.
3. Discount model and APIs (create, list, apply) with validity periods and scope (venue/court).
4. Game management: create public/private games, associate with venue/slot, join/leave, capacity rules, auto-cancel if min players unmet.
5. Game auto-cancel service.

Milestone – 3 Slot Booking & Dynamic Pricing

1. Slot model and availability API: GET /slots/available.Dynamic pricing engine (caching demand + time + historical + discount)
2. Implement dynamic pricing engine (demand/time/historical/discount multipliers).Slot lock expiry background service
3. Slot locking with 5-minute price lock.
4. Enforce Background serive like - Slot-lock expiry service.
5. Booking lifecycle: Available,Pending, Confirmed, Cancelled, Expired, Completed,Locked; prevent double booking under concurrency.

Milestone – 4 Wallet, Payments & Refunds

1. Wallet creation per user; wallet balance and transaction history endpoints.
2. Add-funds (mock gateway, idempotent), debit for bookings, credit for refunds.
3. ACID transaction handling for booking + wallet deduction.
4. Refund rules implementation and APIs: 100% / 50% / 0% based on time window; POST /bookings/confirm, cancellation + refund.

5. Enforce Background service like - Async refund processor,Discount expiry service.
6. Enforce constraints: If any venue/court is marked as not available by venue owner then full refund will be issued to user in wallet.

Milestone – 5 Waitlist & Rating System

1. Waitlist system with capped size, rating-sorted list, invite API, and notification hooks.
2. Rating & review system for venues, courts, and players.
3. Player profiles with aggregated ratings, games played, preferences, recent reviews.
4. Rating processor after game completion.

5.Non-Functional Requirements

Code Quality

- Follow C# standards; SOLID principles; DI; service layer + repository pattern
- Async/await for I/O; comprehensive exception handling
- Unit tests with **minimum 80% coverage**

Documentation

- Swagger (API Documentation)
- Database schema documentation

Security

- Token-based authentication; secure password hashing
- Input validation on all endpoints
- Injection prevention via parameterized queries
- Strict authorization for owner/admin actions

Performance

- Efficient queries + indexing (slots availability, bookings, wallet transactions)
- caching for hot-path counters/locks
- Background services for long-running/offline computation

Maintainability

- Enums/constants for statuses
- Model validation via data annotations/FluentValidation
- Extension methods for reusable logic
- Clear separation of pricing/booking/wallet/rating domains