🎾 Padel Booking Platform

Project Definition and Scope Document

Document Version	Date	Market
1.0	**January 2025**	**Islamabad, Pakistan**

Table of Contents

Vision: Build a world-class padel court booking web application for a single club in Islamabad that serves as the foundation for Pakistan's first comprehensive padel booking platform

1. Goals and Objectives

Strategic Vision

Transform from a single-club solution to Pakistan's leading padel booking ecosystem, following the proven path of platforms like Playtomic while addressing local market needs.

Phased Objectives

Phase 1: Single Club Launch (Months 1-3)

- ✅ Develop fully-featured web application for pilot club
- ✅ Achieve 80%+ digital booking adoption rate
- ✅ Reduce club's administrative work by 60%
- ✅ Process 1,000+ successful bookings
- ✅ Maintain 99.9% uptime

- ✅ Integrate local payment methods (EasyPaisa, JazzCash, bank transfers)

Phase 2: Multi-Club Expansion (Months 4-9)

- 📈 Onboard 5-10 additional clubs in Islamabad/Rawalpindi
- 📈 Build club self-service portal
- 📈 Reach 5,000+ registered users
- 📈 Implement WhatsApp booking bot
- 📈 Add multi-language support (English/Urdu)
- 📈 Create mobile apps (iOS/Android)

Phase 3: Platform Evolution (Months 10-18)

- 🎯 Expand to Lahore and Karachi
- 🎯 Launch marketplace features (club discovery, comparison)
- 🎯 Introduce player social features and skill matching
- 🎯 Build coaching marketplace
- 🎯 Implement tournament management
- 🎯 Achieve 50,000+ active users

Phase 4: Market Leadership (Months 18-24)

- 🚀 Cover all major Pakistani cities
- 🚀 Process 100,000+ monthly bookings
- 🚀 Introduce AI-powered features
- 🚀 Consider international expansion (UAE, Central Asia)
- 🚀 Evaluate microservices migration based on scale

2. Target Audience

Immediate Target (Phase 1)

Pilot Club Profile

- **Location:** Islamabad (F-6, F-7, E-7, or DHA area)
- **Size:** 4-8 padel courts
- **Current challenges:**
  - Manual booking via phone/WhatsApp
  - Payment collection issues
  - No-shows and last-minute cancellations
  - Difficulty managing peak hours
  - Limited marketing reach

Pilot Club's Players

Demographics

- Professionals and entrepreneurs (60%)
- Expats and diplomats (20%)
- Students from affluent families (20%)

Characteristics

- High smartphone penetration (95%+)
- Comfortable with digital payments
- Active on WhatsApp

- Play during evenings (5-10 PM) and weekends

Expanded Target Audience (Phase 2-4)

Player Segments

Premium Segment (20%)

- C-level executives, business owners
- Play 3-4 times/week
- Book premium slots, hire coaches
- Less price-sensitive

Regular Players (50%)

- Mid-level professionals
- Play 1-2 times/week
- Price-conscious, seek off-peak discounts
- Value convenience and reliability

Casual/Social Players (30%)

- Young professionals and students
- Play occasionally, often in groups
- Highly price-sensitive

- Need player matching features

Geographic Expansion

- **Primary Cities:** Islamabad, Lahore, Karachi
- **Secondary Cities:** Faisalabad, Multan, Peshawar
- **International:** Dubai, Doha (large Pakistani expat communities)

3. Functionality and Features

Core Features (MVP - Phase 1)

Smart Booking System

- Real-time court availability
- 60/90/120-minute slots
- Recurring bookings
- Quick rebooking of favorite slots
- Booking history and upcoming games

Payment Integration

- Online payments (Stripe, local gateways)
- Wallet system for quick payments
- Package deals and memberships
- Digital receipts
- Refund processing

User Experience

- Mobile-responsive design
- Quick registration (phone/email)
- Profile management

Court Management

- Visual court layout dashboard
- Drag-and-drop scheduling
- Block courts for maintenance

- Booking modifications/cancellations
- Push notifications for reminders
- WhatsApp notifications for bookings

- Peak/off-peak pricing configuration
- Special event management

📊 Analytics Dashboard

- Occupancy rates and trends
- Revenue analytics
- Popular time slots
- Customer behavior insights
- No-show tracking

👥 Customer Management

- Member database
- Booking history per customer
- Communication tools
- Loyalty program management
- Blacklist management

Enhanced Features (Phase 2)

- **WhatsApp Bot Integration:** Interactive booking via WhatsApp bot, quick rebooking commands, balance inquiries
- **Payment Splitting:** Automatic payment splitting between players (up to 4), split payment requests
- **Multi-Club Features:** Club discovery map, compare prices and availability, unified user accounts
- **Social Features:** Player matching by skill level (1-7 rating), group creation, game invitations

Advanced Features (Phase 3-4)

- **Tournament Management:** Bracket generation, score tracking, prize management, live standings
- **Coaching Marketplace:** Coach profiles and ratings, lesson booking, video analysis tools
- **AI-Powered Features:** Smart pricing recommendations, demand prediction, personalized recommendations

4. Scope Boundaries

In Scope ✅

Phase 1 Deliverables

- Web application (responsive for mobile/tablet/desktop)
- Admin dashboard for single club
- Payment processing (local + international)
- Basic reporting and analytics
- Email/SMS notifications
- Customer support system
- Data backup and security

Technical Scope

- Microservices architecture (start with 3-4 core services)
- PostgreSQL database (per service)
- Redis caching and messaging
- Real-time updates via WebSocket
- RESTful APIs and event-driven communication
- JWT authentication

- Kubernetes deployment (local or managed)

Integration Scope

- Stripe payment gateway
- Local payment methods (EasyPaisa/JazzCash APIs)
- SMS gateway (Twilio/local provider)
- Email service (SendGrid)
- WhatsApp Business API (notifications only)

Out of Scope ❌

Phase 1 Exclusions

- Native mobile applications (web-responsive only)
- Multiple club support (single club focus)
- Advanced social features
- Tournament management
- Coaching marketplace
- Equipment sales
- Complex loyalty programs
- Multi-language support (English only initially)

Technical Exclusions

- Full microservices suite (start with core services only)
- Service mesh (add in Phase 2)
- Blockchain integration
- VR/AR features
- Live streaming
- Complex AI/ML features (basic analytics only)

Success Criteria

Phase 1 Success Metrics

- ✅ 70%+ of club bookings through platform
- ✅ <5% booking abandonment rate
- ✅ 4.5+ user satisfaction score
- ✅ 50% reduction in club admin time
- ✅ <1% payment failure rate
- ✅ 99.9% uptime

Platform Success Indicators

- 📊 Club retention rate >95%
- 📊 User monthly active rate >60%
- 📊 Organic user growth >20% monthly

- 📊 Positive unit economics by Month 12
- 📊 Market leader position in Pakistan by Month 24

Risk Boundaries

Identified Risks

1. **Market Risks:** Limited padel awareness, slow digital adoption, WhatsApp group competition
2. **Technical Risks:** Internet connectivity issues, payment gateway limitations, scaling challenges
3. **Business Risks:** Club resistance to commission model, seasonal demand, regulatory changes

Mitigation Strategies

- Start with single club to prove concept
- Offer free tier initially
- Heavy focus on user experience
- Local payment method integration
- Offline mode capabilities
- Strong customer support

Part 2: Research and Analysis

Research Objective: Conduct comprehensive market, user, and technical research to validate assumptions, identify opportunities, and ensure successful platform development for the Islamabad padel market

5. Market Research

5.1 Pakistan Padel Market Overview

Market Size & Growth

- **Current Market (2025):** ~15-20 padel clubs nationwide
- **Player Base:** Estimated 10,000-15,000 active players
- **Growth Rate:** 40-50% annually (new clubs opening)
- **Market Value:** PKR 500-750 million annual revenue potential

- **Islamabad/Rawalpindi:** 5-7 clubs, 3,000+ active players

5.2 Competitive Landscape

Competitor Type	Current Solution	Market Share	Weaknesses to Exploit
WhatsApp Groups	Manual coordination, informal bookings	70%	No payment integration, no availability visibility, high admin effort
Phone Bookings	Call reception, manual log	25%	Limited hours, language barriers, no digital record
Walk-ins	First-come-first-served	5%	Uncertainty, wasted trips, no advance planning
International Apps	Playtomic, MATCHi (not in Pakistan)	0%	No local presence, payment methods, or support

5.3 Market Opportunity Analysis

🎯 **Immediate Opportunity**

- First-mover advantage in Pakistan
- No established digital competitor

📈 **Growth Drivers**

- FIFA World Cup 2022 increased racquet sports interest
- Health consciousness post-COVID

- High smartphone penetration (85%+)
- Growing middle class interested in premium sports
- Social media visibility of padel
- Corporate wellness programs adopting padel

5.4 Revenue Model Analysis

💰 Transaction-Based Revenue

- **Booking Commission:** 5-10% per transaction
- **Payment Processing:** 2.9% + PKR 30
- **Estimated per booking:** PKR 100-200
- **Break-even:** ~500 bookings/month

🔄 Subscription Revenue

- **Club Plans:** PKR 15,000-50,000/month
- **Player Premium:** PKR 500-1,000/month
- **White-label:** PKR 50,000+/month
- **Target:** 70% subscription, 30% transaction

5.5 Market Entry Strategy

Go-to-Market Approach

1. **Pilot Partnership:** Free 3-month trial with leading Islamabad club
2. **Success Story:** Document 50%+ efficiency gains, create case study
3. **Network Effect:** Leverage pilot club's players for organic growth

4. **Referral Program:** Incentivize clubs to recommend to peers
5. **Media Coverage:** Partner with Dawn, Tribune for tech innovation stories

6. User Research

6.1 User Research Methodology

Research Methods Planned

- 📋 Online surveys (n=200 players)
- 🎤 In-depth interviews (n=20)
- 👥 Focus groups (3 sessions)
- 📊 Club data analysis
- 🔍 Competitor app analysis
- 📱 Usability testing (n=15)

Research Timeline

- Week 1-2: Survey design & distribution
- Week 2-3: User interviews
- Week 3-4: Focus groups
- Week 4-5: Data analysis
- Week 5-6: Prototype testing
- Week 6: Research synthesis

6.2 User Personas

👔 Persona 1: "Corporate Ali" (Primary User)

- **Demographics:** 35, Marketing Director, Lives in F-7
- **Behavior:** Books 2-3 games/week, always after 7 PM
- **Pain Points:** Coordinating with friends, finding last-minute partners, payment collection

- **Needs:** Quick booking, reliable partners, corporate billing option
- **Tech Savvy:** High - uses multiple apps daily
- **Quote:** "I waste 30 minutes coordinating each game on WhatsApp"

👩‍🎓 Persona 2: "Social Sara" (Growth Driver)

- **Demographics:** 28, Entrepreneur, Lives in DHA
- **Behavior:** Plays socially 1-2 times/week, organizes group events
- **Pain Points:** Organizing mixed-skill games, splitting payments, finding coaches
- **Needs:** Group booking features, skill matching, social features
- **Tech Savvy:** Very high - early adopter
- **Quote:** "Padel is my networking opportunity, but organizing is a hassle"

🏃‍♂️ Persona 3: "Competitive Ahmed" (Power User)

- **Demographics:** 42, Business Owner, Lives in E-7
- **Behavior:** Plays 4-5 times/week, participates in tournaments
- **Pain Points:** Court quality variance, finding competitive matches, tracking progress
- **Needs:** Advanced booking, player ratings, tournament features

- **Tech Savvy:** Medium - needs simple interface
- **Quote:** "I want to improve my game and find players at my level"

6.3 User Journey Mapping

Current Journey Pain Points

1. **Discovery:** No central place to find available courts (relies on calling)
2. **Booking:** WhatsApp back-and-forth, no confirmation system
3. **Payment:** Cash only, awkward money collection from friends
4. **Day-of:** No reminders, frequent no-shows, locked out if late
5. **Post-game:** No way to rate experience or find same players again

6.4 Feature Prioritization (Based on Research)

Feature	User Demand	Impact	Effort	Priority
One-click rebooking	⭐ ⭐ ⭐ ⭐ ⭐	High	Low	P0 - Must Have
WhatsApp notifications	⭐ ⭐ ⭐ ⭐ ⭐	Very High	Low	P0 - Must Have
WhatsApp booking bot	⭐ ⭐ ⭐ ⭐	High	High	P1 - Should Have

Feature	User Demand	Impact	Effort	Priority
Payment splitting	⭐⭐⭐⭐	High	Medium	P1 - Should Have
Player matching	⭐⭐⭐⭐	Medium	High	P1 - Should Have
Skill ratings	⭐⭐⭐	Medium	Medium	P2 - Nice to Have
Video replay	⭐⭐	Low	Very High	P3 - Future

7. Technical Feasibility

7.1 Technical Requirements Analysis

⚡ Performance Requirements

- **Response Time:** <2s for all operations
- **Concurrent Users:** Support 1,000+ simultaneous
- **Availability:** 99.9% uptime SLA
- **Data Loss:** Zero tolerance, hourly backups
- **Mobile Performance:** 90+ Lighthouse score

🔒 Security Requirements

- **Authentication:** JWT with refresh tokens
- **Data Encryption:** TLS 1.3, AES-256
- **PCI Compliance:** For payment processing
- **GDPR/Local Laws:** Data privacy compliance
- **API Security:** Rate limiting, CORS, CSP

7.2 Infrastructure Feasibility

🌍 Local Infrastructure Challenges

- Internet reliability: 85% uptime average

✅ Mitigation Strategies

- Progressive Web App for offline capability

- Mobile data speeds: 10-20 Mbps average
- Power outages: Frequent in summer
- Payment gateway limitations

- CDN deployment via CloudFlare
- Multi-region backup servers
- Lightweight, optimized assets

7.3 Technology Stack Validation

Technology	Purpose	Pros	Risks	Decision
Next.js 14	Frontend Framework	SSR, SEO, Fast, React ecosystem	Learning curve for team	✅ Approved
NestJS	Backend Framework	Enterprise-ready, TypeScript, Modular	Overhead for simple features	✅ Approved
PostgreSQL	Primary Database	ACID, JSON support, Proven	Vertical scaling limits	✅ Approved
Redis	Cache & Queue	Fast, Pub/Sub, Reliable	Memory costs	✅ Approved
WhatsApp API	Messaging	95% user adoption	API costs, Approval process	✅ Approved
Stripe	International Payments	Reliable, Well-documented	2.9% + fees	✅ Approved

Technology	Purpose	Pros	Risks	Decision
EasyPaisa/JazzCash	Local Payments	Local adoption 60%+	Integration complexity	✅ Approved

7.4 Integration Feasibility

Third-Party Integration Assessment

WhatsApp Business API	✅ Feasible - 2-3 weeks approval, $500/month
Stripe Pakistan	✅ Available - 2.9% + $0.30 per transaction
EasyPaisa API	⚠️ Complex - Requires business registration, 2-3% fees
JazzCash Business	✅ Feasible - 1.5-2% transaction fees
SMS Gateway (Twilio)	✅ Ready - $0.0075 per SMS
Google Maps API	✅ Ready - $200 free credits monthly

7.5 Scalability Analysis

📈 Scaling Projections

Metric	Phase 1 (1 Club)	Phase 2 (10 Clubs)	Phase 3 (50 Clubs)	Phase 4 (200+ Clubs)
Daily Active Users	100-200	1,000-2,000	5,000-10,000	20,000+

Metric	Phase 1 (1 Club)	Phase 2 (10 Clubs)	Phase 3 (50 Clubs)	Phase 4 (200+ Clubs)
Bookings/Day	20-40	200-400	1,000-2,000	5,000+
Database Size	1-5 GB	10-50 GB	100-500 GB	1+ TB
Infrastructure Cost	$200-300/month	$500-800/month	$2,000-3,000/month	$10,000+/month
Architecture	Microservices (3-4)	Microservices (5-6)	Microservices (8-10)	Microservices (15+)

7.6 Development Feasibility

👥 Team Requirements

- **Phase 1 (MVP):** 3-4 developers (2 backend, 1 frontend, 1 DevOps)
- **Phase 2:** 5-6 developers + 1 DevOps
- **Phase 3:** 8-10 developers + 2 DevOps
- **Skills Needed:** Node.js, React, PostgreSQL, Kubernetes
- **Availability:** Good talent pool in Islamabad

⏱️ Timeline Feasibility

- **MVP Development:** 10-12 weeks (with microservices)
- **Testing & QA:** 2-3 weeks
- **Pilot Launch:** Week 14
- **Iterations:** 2-week sprints
- **Platform Ready:** Month 6

7.7 Risk Assessment Matrix

Risk Category	Probability	Impact	Mitigation Strategy	Status
Payment Gateway Rejection	Medium	High	Multiple gateway integration, COD backup	⚠️ Monitor
WhatsApp API Delay	Low	Medium	SMS fallback, Telegram bot alternative	✅ Mitigated
Slow User Adoption	Medium	High	Incentives, free trial, club partnership	⚠️ Monitor
Technical Debt	Low	Medium	Code reviews, testing, refactoring sprints	✅ Controlled
Scaling Issues	Low	High	Cloud infrastructure, monitoring, caching	✅ Prepared
Competition Entry	Medium	Medium	Fast execution, exclusive contracts	⚠️ Monitor

📊 Research Conclusions

Key Findings

- ✅ **Market Validation:** Clear demand with no digital solution currently available

- ✅ **User Readiness:** High smartphone usage and digital payment adoption among target users
- ✅ **Technical Feasibility:** All core technologies proven and available
- ⚠️ **Primary Challenge:** User behavior change from WhatsApp to platform
- 💡 **Success Factor:** WhatsApp notifications crucial for adoption, full bot in Phase 2
- 🎯 **Recommendation:** Proceed with MVP development focusing on core booking and WhatsApp notifications

Part 3: System Architecture and Technology Stack

Architecture Vision: Build a scalable, cloud-native microservices architecture from day one, starting with 3-4 core services for the MVP and expanding as the platform grows

8. System Architecture Design

8.1 Architecture Overview

Microservices Architecture Principles (MVP Approach)

- **Start Small:** Begin with 3-4 essential services for MVP
- **Domain-Driven Design:** Services aligned with business domains
- **Database per Service:** Each service owns its data from the start
- **Event-Driven Communication:** Asynchronous messaging between services
- **API Gateway Pattern:** Single entry point for all clients
- **Container-First:** All services containerized with Docker

- **Cloud-Native:** Ready for Kubernetes orchestration
- **Progressive Complexity:** Add services and infrastructure as needed

8.2 High-Level Architecture

System Components Overview

Client Layer

- 🌐 **Web Application:** Next.js 14 Progressive Web App
- 📱 **Mobile Applications:** React Native (iOS/Android)
- 💬 **WhatsApp Bot:** Node.js webhook-based bot
- 🤖 **Admin Dashboard:** React-based SPA

API Layer

- 🚪 **API Gateway:** Kong/AWS API Gateway
- ⚖️ **Load Balancer:** NGINX/AWS ALB
- 🔐 **Authentication:** OAuth 2.0 / JWT
- 📊 **Rate Limiting:** Redis-based throttling

Service Mesh Layer

- 🔄 **Service Discovery:** Consul/Kubernetes DNS
- 🔒 **mTLS:** Service-to-service encryption

- 📈 **Circuit Breakers:** Hystrix patterns
- 🔍 **Distributed Tracing:** Jaeger/Zipkin

8.3 Microservices Breakdown (Phased Approach)

🚀 MVP Services (Phase 1 - Core 4 Services)

- **Auth Service:** Authentication, JWT, Session Management
- **User Service:** User profiles, Preferences
- **Booking Service:** Court bookings, Availability, Club management
- **Notification Service:** Email, SMS, WhatsApp notifications

Service	Responsibilities	Technology	Database	MVP Phase
Auth Service	Authentication, JWT, OAuth, Session Management	Node.js + Express	Redis + PostgreSQL	✅ Phase 1
User Service	User profiles, Preferences, Player ratings	Node.js + NestJS	PostgreSQL	✅ Phase 1

Service	Responsibilities	Technology	Database	MVP Phase
Booking Service	Court bookings, Availability, Scheduling, Clubs, Courts	Node.js + NestJS	PostgreSQL + Redis	✅ Phase 1
Notification Service	Email, SMS, Push, WhatsApp notifications	Node.js + BullMQ	Redis Queue	✅ Phase 1
Payment Service	Payment processing, Wallets, Refunds	Python + FastAPI	PostgreSQL	Phase 2
Club Service	Multi-club management, Pricing tiers	Node.js + NestJS	PostgreSQL	Phase 2
Analytics Service	Reports, Metrics, ML predictions	Python + FastAPI	ClickHouse	Phase 3
WhatsApp Bot Service	Interactive bot, Message handling	Node.js + Webhooks	Redis Queue	Phase 2
Search Service	Club search, Player matching	Node.js + Elasticsearch	Elasticsearch	Phase 3
Media Service	Image upload, Processing, CDN	Node.js + Sharp	S3 + CloudFront	Phase 2

8.4 Inter-Service Communication

🔄 Synchronous Communication

- **Protocol:** REST over HTTP/2
- **Format:** JSON
- **Use Cases:**
  - Auth verification
  - Real-time availability checks
  - Payment processing
- **Timeout:** 3 seconds default
- **Retry:** Exponential backoff

📬 Asynchronous Communication

- **Message Broker:** RabbitMQ/Kafka
- **Pattern:** Event-driven
- **Use Cases:**
  - Booking confirmations
  - Payment notifications
  - Analytics events
- **Delivery:** At-least-once
- **Format:** CloudEvents spec

8.5 Event-Driven Architecture

Core Events Flow

Event	Producer	Consumers	Actions Triggered
booking.created	Booking Service	Payment, Notification, Analytics	Process payment, Send confirmation, Update stats

Event	Producer	Consumers	Actions Triggered
payment.completed	Payment Service	Booking, Notification, User	Confirm booking, Send receipt, Update wallet
user.registered	User Service	Notification, Analytics, Club	Welcome email, Track acquisition, Notify club
booking.cancelled	Booking Service	Payment, Notification, Analytics	Process refund, Send notice, Update availability
court.maintenance	Club Service	Booking, Notification	Block slots, Notify affected users

9. Database Design

9.1 Database Strategy

📊 Data Distribution Strategy

- Service-specific databases
- No shared database access
- Event-based data synchronization
- CQRS for read-heavy operations
- Event sourcing for audit trails

🔄 Data Consistency Approach

- Eventually consistent by default
- Strong consistency for payments
- Saga pattern for distributed transactions
- Compensating transactions for rollbacks
- Idempotency keys for deduplication

9.2 Service-Specific Database Schemas

User Service Database Schema

```
Tables:
├── users
│   ├── id (UUID, PK)
│   ├── email (VARCHAR, UNIQUE)
│   ├── phone (VARCHAR, UNIQUE)
│   ├── password_hash (VARCHAR)
│   ├── first_name (VARCHAR)
```

```
│   ├── last_name (VARCHAR)
│   ├── avatar_url (VARCHAR)
│   ├── skill_level (INT 1-7)
│   ├── preferred_position (ENUM)
│   ├── status (ENUM: active, suspended, deleted)
│   ├── created_at (TIMESTAMP)
│   └── updated_at (TIMESTAMP)
│
├── user_preferences
│   ├── user_id (UUID, FK)
│   ├── notification_email (BOOLEAN)
│   ├── notification_sms (BOOLEAN)
│   ├── notification_whatsapp (BOOLEAN)
│   ├── preferred_clubs (JSONB)
│   ├── preferred_times (JSONB)
│   └── language (VARCHAR)
│
├── user_connections
│   ├── user_id (UUID, FK)
│   ├── connected_user_id (UUID, FK)
│   ├── connection_type (ENUM: friend, blocked)
│   └── created_at (TIMESTAMP)
│
└── player_ratings
    ├── id (UUID, PK)
    ├── rated_user_id (UUID, FK)
    ├── rater_user_id (UUID, FK)
    ├── skill_rating (INT 1-5)
    ├── sportsmanship_rating (INT 1-5)
    ├── booking_id (UUID)
    └── created_at (TIMESTAMP)
```

Indexes:

- users_email_idx (email)

```
- users_phone_idx (phone)
- users_skill_level_idx (skill_level)
- connections_user_idx (user_id, connection_type)
```

Booking Service Database Schema

```
Tables:
├── bookings
│   ├── id (UUID, PK)
│   ├── court_id (UUID)
│   ├── user_id (UUID)
│   ├── start_time (TIMESTAMP)
│   ├── end_time (TIMESTAMP)
│   ├── duration_minutes (INT)
│   ├── status (ENUM: pending, confirmed, cancelled, completed, no_show)
│   ├── total_amount (DECIMAL)
│   ├── payment_status (ENUM)
│   ├── players (JSONB) [{user_id, name, status}]
│   ├── booking_type (ENUM: single, recurring)
│   ├── parent_booking_id (UUID, nullable)
│   ├── cancellation_reason (TEXT)
│   ├── created_at (TIMESTAMP)
│   └── updated_at (TIMESTAMP)
│
├── court_availability
│   ├── court_id (UUID)
│   ├── date (DATE)
│   ├── available_slots (JSONB)
│   ├── blocked_slots (JSONB)
│   └── last_updated (TIMESTAMP)
```

```
│
├── recurring_bookings
│   ├── id (UUID, PK)
│   ├── user_id (UUID)
│   ├── court_id (UUID)
│   ├── day_of_week (INT 0-6)
│   ├── time_slot (TIME)
│   ├── duration_minutes (INT)
│   ├── start_date (DATE)
│   ├── end_date (DATE)
│   ├── status (ENUM)
│   └── created_at (TIMESTAMP)
│
└── booking_waitlist
    ├── id (UUID, PK)
    ├── user_id (UUID)
    ├── court_id (UUID)
    ├── preferred_date (DATE)
    ├── preferred_time (TIME)
    ├── flexibility_hours (INT)
    ├── status (ENUM)
    └── created_at (TIMESTAMP)
```

Indexes:

- bookings_court_date_idx (court_id, start_time)
- bookings_user_idx (user_id, status)
- bookings_status_idx (status, start_time)
- availability_court_date_idx (court_id, date)

MVP Payment Service Approach (Simplified)

> **Note:** For MVP with single club, payment processing will be handled as a module within the Booking Service to reduce complexity. A dedicated Payment Service will be extracted in Phase 2 when adding payment splitting and multi-club support.

```
Tables (within Booking Service DB for MVP):
├── payments
│   ├── id (UUID, PK)
│   ├── booking_id (UUID)
│   ├── user_id (UUID)
│   ├── amount (DECIMAL)
│   ├── currency (VARCHAR)
│   ├── status (ENUM: pending, completed, failed, refunded)
│   ├── payment_method (ENUM: card, easypaisa, jazzcash)
│   ├── gateway_transaction_id (VARCHAR)
│   ├── created_at (TIMESTAMP)
│   └── updated_at (TIMESTAMP)
│
└── refunds
    ├── id (UUID, PK)
    ├── payment_id (UUID, FK)
    ├── amount (DECIMAL)
    ├── reason (TEXT)
    ├── status (ENUM)
    └── created_at (TIMESTAMP)
```

9.3 Data Management Patterns

🔄 SAGA Pattern Implementation

Booking Creation Saga:

1. Create booking (Booking Service)
2. Reserve payment (Payment Service)
3. Validate user (User Service)
4. Process payment (Payment Service)
5. Confirm booking (Booking Service)
6. Send notifications (Notification Service)

Compensation on Failure:

- Release booking slot
- Refund payment
- Notify user of failure

📊 CQRS Implementation

Command Side:

- Write to primary PostgreSQL
- Emit domain events
- Ensure consistency

Query Side:

- Read from replicas/cache
- Elasticsearch for search
- Redis for hot data
- ClickHouse for analytics

10. Technology Stack

10.1 Frontend Technologies

Layer	Technology	Purpose	Justification
Web Framework	Next.js 14	SSR, SEO, Performance	Best-in-class React framework, excellent PWA support
Mobile Apps	React Native + Expo	Cross-platform mobile	Code reuse from web, faster development
Language	TypeScript	Type safety	Reduces bugs, better IDE support
Styling	Tailwind CSS + Shadcn/ui	Rapid UI development	Consistent design system, responsive by default
State Management	Zustand + React Query	Client state & server state	Lightweight, excellent caching
Forms	React Hook Form + Zod	Form handling & validation	Performance, schema validation
Real-time	Socket.io Client	WebSocket communication	Reliable, fallback support

Layer	Technology	Purpose	Justification
PWA	Workbox	Offline support	Google's PWA toolkit

10.2 Backend Technologies

Component	Technology	Purpose	Justification
Primary Language	Node.js (v20 LTS)	Main services	JavaScript everywhere, large ecosystem
Secondary Language	Python 3.11+	Analytics, ML, Payments	Better for data processing, ML libraries
Node Framework	NestJS	Service architecture	Enterprise-grade, modular, TypeScript-first
Python Framework	FastAPI	High-performance APIs	Modern, async, automatic OpenAPI
API Gateway	Kong	Request routing	Plugin ecosystem, Kubernetes-native
Message Broker	RabbitMQ	Event bus	Reliable, battle-tested, good tooling
Cache	Redis 7+	Caching & sessions	Fast, versatile, pub/sub support

Component	Technology	Purpose	Justification
Search	Elasticsearch 8	Full-text search	Powerful search, analytics capabilities

10.3 Data Layer Technologies

🗄️ Primary Databases

- **PostgreSQL 15:** OLTP, ACID compliance
- **MongoDB:** Notifications, flexible schemas
- **Redis:** Cache, sessions, queues
- **ClickHouse:** Analytics, time-series data
- **Elasticsearch:** Search, logs

🔧 Data Tools

- **Prisma:** TypeScript ORM
- **TypeORM:** Alternative ORM
- **Alembic:** Python migrations
- **Debezium:** CDC for event streaming
- **Apache Kafka:** Event streaming (Phase 3+)

10.4 Infrastructure & DevOps (MVP Focused)

🚀 MVP Infrastructure Strategy

- **Simplified Kubernetes:** Use managed K8s (EKS/GKE/DO) to reduce operational overhead

- **Basic Service Mesh:** Start with simple Kubernetes networking, add Istio in Phase 2
- **Minimal Services:** 3-4 core services only, expand gradually
- **Managed Services:** Use managed databases and caching to focus on business logic

Category	Technology	Purpose	Alternative
Container	Docker	Containerization	Podman
Orchestration	Kubernetes	Container orchestration	Docker Swarm
Service Mesh	Istio	Traffic management	Linkerd
CI/CD	GitLab CI	Automation	GitHub Actions
Monitoring	Prometheus + Grafana	Metrics & visualization	DataDog
Logging	ELK Stack	Log aggregation	Fluentd + CloudWatch
Tracing	Jaeger	Distributed tracing	Zipkin
Secrets	HashiCorp Vault	Secret management	AWS Secrets Manager

10.5 Cloud Services

☁️ **AWS Services (Primary)**

- **EKS:** Kubernetes hosting

🔄 **Alternative Cloud (Backup)**

- **DigitalOcean:** Cost-effective VPS

- **RDS:** Managed PostgreSQL
- **ElastiCache:** Managed Redis
- **S3:** Object storage
- **CloudFront:** CDN
- **SQS/SNS:** Queue/Notifications
- **Lambda:** Serverless functions

- **Kubernetes:** DO Managed K8s
- **Spaces:** S3-compatible storage
- **Managed Databases:** PostgreSQL, Redis
- **Load Balancers:** Regional LBs
- **Cloudflare:** CDN & DDoS protection

10.6 Third-Party Integrations

External Service Integrations

Service	Provider	Purpose	SDK/Library
International Payments	Stripe	Card processing	@stripe/stripe-js
Local Payments	EasyPaisa + JazzCash	Mobile wallets	Custom REST API
WhatsApp	WhatsApp Business API	Messaging bot	whatsapp-web.js
SMS	Twilio	SMS notifications	twilio-node
Email	SendGrid	Transactional email	@sendgrid/mail
Maps	Google Maps	Location services	@googlemaps/js-api-loader
Analytics	Mixpanel	Product analytics	mixpanel-browser
Error Tracking	Sentry	Error monitoring	@sentry/node

10.7 Security Stack

🔒 Security Technologies

Application Security:

- JWT with RS256 signing
- OAuth 2.0 + OpenID Connect
- Helmet.js for headers
- Rate limiting with Redis
- CORS configuration
- Input validation (Joi/Zod)

Infrastructure Security:

- mTLS between services
- Network policies in K8s
- WAF with CloudFlare
- Secrets in Vault
- Container scanning
- SIEM with ELK

10.8 Development Tools

🛠️ Development Environment

- **IDE:** VS Code with extensions
- **API Testing:** Postman/Insomnia
- **DB Client:** DBeaver/pgAdmin
- **Container:** Docker Desktop
- **K8s Local:** Minikube/Kind

✅ Testing Stack

- **Unit Tests:** Jest + Testing Library
- **Integration:** Supertest
- **E2E:** Playwright/Cypress
- **Load Testing:** K6/Artillery
- **API Mocking:** MSW

- **Git Flow:** GitLab/GitHub

- **Coverage:** Istanbul/NYC

10.9 Migration Strategy

📈 Progressive Migration Path

Phase	Architecture	Services Count	Complexity	When to Migrate
MVP	Core Microservices	3-4 services	Medium	1 club, <1k users
Growth	Expanded Microservices	5-7 services	Medium-High	10 clubs, <10k users
Scale	Full Microservices	8-12 services	High	50+ clubs, <100k users
Enterprise	Complete Microservices	15+ services	Very High	200+ clubs, 1M+ users

Migration Triggers:

- Team size exceeds 15 developers
- Deployment frequency > 10/day needed
- Different scaling requirements per domain
- Compliance requires service isolation
- Performance bottlenecks in monolith

10.10 Architecture Decision Records (ADRs)

Key Architecture Decisions

ADR-001: Microservices from MVP

- **Decision:** Start with microservices architecture in MVP (3-4 core services)
- **Rationale:** Build for scale from day one, avoid costly migration later
- **Trade-offs:** Higher initial complexity, more infrastructure setup
- **Mitigation:** Start with only essential services, use managed Kubernetes, simple service mesh

ADR-002: Event-Driven Communication

- **Decision:** Use events for inter-service communication
- **Rationale:** Loose coupling, scalability, resilience
- **Trade-offs:** Eventual consistency, debugging complexity
- **Mitigation:** Comprehensive logging, distributed tracing

ADR-003: Database per Service

- **Decision:** Each service owns its database
- **Rationale:** Independence, technology flexibility
- **Trade-offs:** Data duplication, consistency challenges
- **Mitigation:** SAGA pattern, event sourcing

📊 Architecture Summary

Key Characteristics

- ✅ **Scalable:** Horizontal scaling per service based on load
- ✅ **Resilient:** Service isolation, circuit breakers, retries
- ✅ **Flexible:** Technology choice per service domain
- ✅ **Observable:** Comprehensive monitoring, logging, tracing
- ✅ **Secure:** Defense in depth, zero-trust networking
- ✅ **Progressive:** Can start simple, evolve with growth

This document outlines the strategic vision and scope for Pakistan's first comprehensive padel booking platform.

Starting with a single club in Islamabad, we aim to transform the padel ecosystem nationwide.

Confidential - Not for Distribution