



# Borneo Anfield Stadium Management System

---

A comprehensive full-stack web application for managing football field bookings, loyalty programs, and administrative operations at Borneo Anfield Stadium.



## ✨ Features Overview



### Authentication & Authorization

- **Multi-Role System:** Customer, Admin, Super Admin with role-based access control
- **Secure JWT Authentication:** HTTP-only cookies with bcrypt password hashing
- **Protected Routes:** Middleware-based route protection



### User Management

- **Customer Self-Registration:** Easy account creation for customers
- **Admin Management:** Super admin can create, edit, and delete admin accounts
- **Profile Management:** Users can update personal information
- **Account Status Control:** Active/inactive user management



### Advanced Booking System

- **Real-time Availability:** Live field availability checking
- **Flexible Time Slots:** Book fields for 1-8 hours with custom time ranges
- **Multi-Player Support:** Add multiple players to bookings
- **Booking Lifecycle:** Complete status tracking (Unpaid → Pending → Confirmed → Completed)
- **Conflict Prevention:** Automatic booking overlap detection

## Loyalty & Rewards Program

- **Points System:** Earn 20 points per hour of booking (distributed among players)
- **Reward Catalog:** Customizable rewards with point requirements
- **Digital Redemption:** QR code generation for reward collection
- **Points History:** Complete earning and spending audit trail
- **Expiry Management:** Points expire after 1 year

## Payment & Transaction Management

- **Payment Verification:** Admin approval workflow for uploaded payment proofs
- **Multiple Payment Methods:** Bank Transfer, Cash, QRIS, Credit Card, E-Wallet
- **Transaction Tracking:** Complete financial audit trail
- **Payment Status:** Pending → Paid/Invalid workflow
- **Automatic Booking Updates:** Status changes based on payment verification

## Analytics & Reporting

- **Booking Analytics:** Revenue trends, field utilization, booking patterns
- **Financial Reports:** Payment tracking and revenue analysis
- **User Insights:** Customer behavior and loyalty metrics
- **Field Performance:** Individual field statistics and profitability

## Modern UI/UX

- **Responsive Design:** Mobile-first approach with Tailwind CSS
- **Component Library:** shadcn/ui for consistent, accessible design
- **Real-time Updates:** Live notifications and status updates
- **Image Optimization:** Automatic WebP conversion for uploaded images
- **Form Validation:** Client and server-side validation with Zod

## Technology Stack

PROF

### Frontend

- **Next.js 15** - React framework with App Router
- **TypeScript** - Type-safe development
- **Tailwind CSS** - Utility-first CSS framework
- **shadcn/ui** - Modern, accessible component library
- **React Hook Form** - Performant form handling
- **Zod** - Runtime type validation
- **Lucide React** - Beautiful icon library

### Backend

- **Next.js Server Actions** - Type-safe server functions
- **Prisma ORM** - Type-safe database access
- **PostgreSQL** - Robust relational database

- **JWT (jose)** - Secure authentication tokens
- **bcryptjs** - Password hashing
- **Sharp** - High-performance image processing

## Infrastructure

- **File Upload System** - Organized local storage structure
- **Middleware Protection** - Route-level authentication
- **Database Migrations** - Version-controlled schema changes
- **Seed Data** - Automated initial data setup

## Prerequisites

- **Node.js** 18+
- **PostgreSQL** database
- **pnpm** (recommended) or npm
- **Git**

## Quick Start

### 1. Clone & Install

```
git clone <your-repo-url>
cd borneo-anfield-stadium
pnpm install
```

### 2. Environment Setup

Create **.env** file:

```
copy from .env.example
```

```
# Database
DATABASE_URL="postgresql://username:password@localhost:5432/borneo_anfield?schema=public"

# Security
JWT_SECRET="your-super-secret-jwt-key-minimum-32-characters"

# App
NEXT_PUBLIC_APP_URL="http://localhost:3000"
```

### 3. Database Setup

```
# Generate Prisma client
npx prisma generate

# Run migrations
npx prisma migrate dev --name init

# Seed initial data
npx prisma db seed
```

## 4. Start Development

```
pnpm dev
```

Visit <http://localhost:3000>

### Default Accounts

After seeding, use these accounts:

Role	Email	Username	Password
Super Admin	superadmin@example.com	superadmin	password123
Admin	admin@example.com	admin	password123
Customer	customer@example.com	customer	password123

### Database Schema

#### Core Models

PROF

#### User

```
model User {
  id          String    @id @default(cuid())
  username    String    @unique
  email       String    @unique
  password    String
  fullName    String
  phoneNumber  String?
  role        UserRole  @default(CUSTOMER)
  isActive    Boolean   @default(true)
  // Relations: bookings, userPoints, redemptions, payments...
}
```

#### Booking

```

model Booking {
  id          String      @id @default(cuid())
  userId      String
  fieldId     String
  bookingDate DateTime
  startTime   DateTime
  endTime     DateTime
  duration    Decimal     @db.Decimal(5, 2)
  amount      Decimal     @db.Decimal(10, 2)
  status      BookingStatus @default(UNPAID)
  // Relations: user, field, payments, players...
}

```

## Field

```

model Field {
  id          String      @id @default(cuid())
  name        String
  description  String?
  capacity    Int
  hourlyRate  Decimal     @db.Decimal(10, 2)
  isAvailable Boolean     @default(true)
  // Relations: bookings
}

```

## Database Commands

```

# View data in Prisma Studio
npx prisma studio

# Reset database (⚠ Destructive)
npx prisma migrate reset

# Create new migration
npx prisma migrate dev --name migration_name

# Deploy to production
npx prisma migrate deploy

```

## API Reference

### Server Actions

#### Authentication (**app/actions/auth.ts**)

```
// Register new customer
registerUser(formData: FormData)

// Login user
loginUser(formData: FormData)

// Logout user
logoutUser()

// Get current user
getUser()
```

### Booking Management (**app/actions/booking.ts**)

```
// Create new booking
createBooking(formData: FormData)

// Upload payment proof
uploadPaymentProof(formData: FormData)

// Cancel booking
cancelBooking(bookingId: string)

// Get user bookings
getUserBookings(user: User)

// Set players for booking
setPlayersToBooking(formData: FormData)

// Complete booking (admin only)
completeBooking(bookingId: string)
```

---

PROF

### Admin Operations (**app/actions/admin.ts**)

```
// Payment verification
confirmPayment(formData: FormData)
invalidPayment(formData: FormData)

// Admin management (super admin only)
createAdmin(data: AdminSchema)
updateAdmin(adminId: string, data: AdminSchema)
deleteAdmin(adminId: string)

// Field management
createField(data: FieldSchema)
updateField(fieldId: string, data: FieldSchema)
```



# User Guide

For Customers

## Making a Booking

1. **Login** to your account
2. **Select Field** from available options
3. **Choose Date & Time** (check availability)
4. **Add Players** (optional, can be done later)
5. **Confirm Booking** and note the booking ID
6. **Upload Payment Proof** with booking ID
7. **Wait for Verification** from admin

## Loyalty Program

- **Earn Points:** 20 points per hour booked (shared among players)
- **View Rewards:** Browse available rewards in loyalty section
- **Redeem:** Exchange points for rewards
- **QR Code:** Show generated QR to staff for collection

For Administrators

## Booking Management

1. **View All Bookings** in admin dashboard
2. **Filter by Status** (Unpaid, Pending, Confirmed, Completed)
3. **Verify Payments** by reviewing uploaded proofs
4. **Update Status** (Confirm/Reject payments)
5. **Complete Bookings** after field usage

PROF

## Field Management

1. **Add New Fields** with pricing and capacity
2. **Edit Field Details** (rates, availability, description)
3. **Monitor Utilization** through analytics

For Super Administrators

## System Management

1. **Create Admin Accounts** for staff
2. **Monitor System Activity** across all modules
3. **View Comprehensive Reports** and analytics
4. **Manage Admin Permissions** and access



## Project Structure

```

app/
├── actions/                # Server actions for data operations
│   ├── auth.ts            # Authentication operations
│   ├── booking.ts         # Booking management
│   ├── admin.ts           # Admin operations
│   └── loyalty.ts         # Loyalty program
├── admin/                  # Admin dashboard
│   ├── bookings/          # Booking management
│   ├── fields/            # Field management
│   ├── loyalty/           # Loyalty program admin
│   └── transactions/      # Payment verification
├── (auth)/                # Authentication pages
│   ├── login/
│   ├── register/
│   └── forgot-password/
├── pengguna/              # Customer dashboard
│   ├── booking/           # Booking interface
│   ├── loyalty/           # Loyalty program
│   ├── profil/            # Profile management
│   └── transactions/      # Transaction history
├── super-admin/           # Super admin interface
│   ├── add-admin/
│   └── edit-admin/
└── api/                   # API routes

components/ui/             # Reusable UI components
lib/                       # Utilities and configurations
prisma/                    # Database schema and migrations
public/                    # Static assets and uploads

```

## Deployment

### Environment Variables for Production

```

DATABASE_URL="your-production-database-url"
JWT_SECRET="your-production-jwt-secret"
NEXT_PUBLIC_APP_URL="https://your-domain.com"
NODE_ENV="production"

```

### Build & Deploy

```

# Build for production
pnpm build

# Start production server
pnpm start

```



```
# Deploy migrations
npx prisma migrate deploy
```

## 📖 Available Scripts

```
# Development
pnpm dev           # Start development server
pnpm build         # Build for production
pnpm start        # Start production server
pnpm lint         # Run ESLint

# Database
npx prisma db seed # seed database table

npx prisma migrate dev --name init # create table for database

npx prisma migrate reset # delete all table, create table, seed database
table
```

## 🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open Pull Request

## 📄 License

This project is licensed under the MIT License.

PROF

## 🆘 Support

- **Issues:** [GitHub Issues](#)
- **Documentation:** [Project Wiki](#)
- **Email:** [support@borneoanfield.com](mailto:support@borneoanfield.com)

---

Built with ❤️ for Borneo Anfield Stadium Management