ASE Athletics Development Challenge

Full Stack Developer Assessment

> **Welcome to your technical assessment!** This document outlines everything you need to build an impressive football analytics platform that showcases your full-stack development skills.

Quick Start Guide

Your Mission: Build a football analytics web application
Timeline: 10 days maximum *(early delivery bonus!)*
Scoring: 280 total points available
Deliverables: GitHub repo + live demo

What Success Looks Like

- A working web application with user authentication
- Complete player management system with database integration
- Interactive dashboard with meaningful charts
- Search and filtering that actually works
- Clean, professional code that we'd be happy to maintain

The Project: Football Analytics Platform

You'll build a digital workspace for football scouts and coaches - think of tools like those used by professional clubs to analyze players and make strategic decisions.

Core Concept

Modern football clubs rely heavily on data analytics. Your application will help football professionals:

- **Manage player databases** with comprehensive statistics
- **Visualize performance trends** through interactive dashboards
- **Compare players** side-by-side for recruitment decisions
- **Generate scouting reports** with structured evaluations
- **Search and filter** large datasets efficiently

Why This Matters at ASE Athletics

This assessment mirrors real projects you'll work on at ASE Athletics. We develop tools that help sports organizations make data-driven decisions, and this challenge reflects the type of problems you'll solve daily.

Your Development Stack

Required Technologies

Frontend Framework *(choose one)*

- React.js with Redux/Context API
- Vue.js with Vuex
- Must include proper routing (React Router/Vue Router)

Backend Framework *(required)*

- Node.js with Express or NestJS
- RESTful API design principles
- JWT-based authentication
- Input validation (Joi, Yup, or similar)

Database *(choose one)*

- PostgreSQL *(recommended for relational data)*
- MongoDB *(good for flexible schemas)*
- MySQL or SQLite *(acceptable alternatives)*

Styling Solution *(choose one)*

- Tailwind CSS *(modern utility-first)*
- Material-UI *(React component library)*
- Bootstrap *(familiar and reliable)*
- Custom CSS *(show your design skills)*

Data Visualization *(choose one)*

- Chart.js *(simple and effective)*
- D3.js *(powerful and flexible)*
- Recharts *(React-specific)*

Architecture Requirements

- **Responsive design** that works on mobile and desktop
- **Clean separation** between frontend and backend
- **Proper error handling** throughout the application
- **Security best practices** for authentication and data validation
- **API documentation** using Swagger/OpenAPI or Postman

Data Assets Provided by ASE Athletics

We've prepared comprehensive datasets to jumpstart your development:

Player Database

Files: `player_statistics_detailed.json` + `players_Data_production.json`

- **100+ professional players** with complete profiles

- **Performance metrics:** goals, assists, appearances, pass accuracy
- **Physical attributes:** pace, shooting, passing, defending ratings
- **Contract details:** salary information, contract end dates
- **Market valuations** and transfer history

Scouting Framework

File: `scout_report.json`

- **Report templates** used by professional scouts
- **Rating system:** standardized 1-10 scale across attributes
- **Evaluation structure:** match context, strengths, weaknesses, recommendations

Design System

File: `ui_guidelines.json`

- **Complete color palette** with hex codes and usage guidelines
- **Typography specifications:** font families, sizes, weights
- **Component standards:** buttons, cards, forms, tables
- **Spacing and layout** guidelines for consistency

All files available through ASE Athletics' GitHub repository or a ZIPFILE provided

---

Feature Specification & Scoring

Core Authentication System (15 points)

Build a secure user management system with:

- **Registration workflow** with email validation
- **Login authentication** using JWT tokens
- **Protected routes** that require authentication
- **Session management** with proper logout
- **Password security** using bcrypt hashing

Keep it simple - basic email/password authentication without complex role systems

Backend API Architecture (25 points)

Develop a RESTful API with these essential endpoints:

Player Management

- `GET /api/players` - Paginated player list with filtering
- `GET /api/players/:id` - Individual player details
- `POST /api/players` - Create new player profile
- `PUT /api/players/:id` - Update existing player
- `DELETE /api/players/:id` - Remove player from system

Search & Discovery

- `GET /api/players/search?q={term}` - Text search across player data
- `GET /api/players?position={pos}&team={team}` - Multi-parameter filtering

Analytics Support

- `GET /api/dashboard/stats` - Aggregated metrics for dashboard

Focus on solid CRUD operations with proper validation and error responses

Database Design & Management (20 points)

Create a well-structured database schema:

Essential Tables

```
-- User authentication
users: id, email, password_hash, name, created_at

-- Core player data
players: id, name, position, age, team, nationality,
         height, weight, goals, assists, appearances,
         contract_salary, contract_end, market_value,
         created_at, updated_at

-- Player attributes (optional separate table)
player_attributes: player_id, pace, shooting, passing,
                   defending, dribbling, physicality

-- Scouting reports
scout_reports: id, player_id, scout_id, match_date,
               overall_rating, strengths, weaknesses,
               recommendation, created_at
```

Database Requirements

- **Schema design** with appropriate relationships
- **Data seeding scripts** to load provided JSON files
- **Basic indexing** for search performance
- **Migration files** for database setup

Player Management Interface (25 points)

Build a comprehensive player management system:

Player Directory

- **Grid/table layout** showing essential player information
- **Pagination controls** (20-30 players per page)
- **Sortable columns** by name, age, position, team, market value
- **Player profile images** (placeholder functionality acceptable)

Detailed Player Profiles

- **Complete player information** with statistics and attributes
- **Performance visualization** using progress bars or mini-charts
- **Contract and market value information**
- **Edit capabilities** for updating player data

Player Creation & Editing

- **Form-based interface** with proper validation
- **All required fields** with appropriate input types
- **Error handling** and success feedback
- **Image upload placeholder** (actual upload is bonus)

Safe Deletion

- **Confirmation dialogs** before removing players
- **Cascade handling** for related data (reports, etc.)

Analytics Dashboard (25 points)

Create an interactive dashboard with meaningful insights:

Key Performance Indicators

- **Summary cards** showing total players, average age, top performers
- **Market insights** like most valuable players, contract expirations
- **Team distributions** and position breakdowns

Interactive Visualizations

- **Goals/assists distribution** by position (bar charts)
- **Age demographics** across teams (pie/donut charts)
- **Market value trends** over time (line charts)
- **Player attribute comparisons** (radar charts)

Dynamic Filtering

- **Real-time chart updates** based on selected filters
- **Team/position/age range** filter controls
- **Clear filter functionality** to reset views
- **Export capabilities** for chart data (bonus)

Search & Filtering System (25 points)

Implement comprehensive search and discovery features:

Global Search

- **Instant search** across player names, teams, nationalities
- **Search result highlighting** to show matched terms
- **Search suggestions** or autocomplete (bonus)

Advanced Filtering Options

- **Position filter** with multi-select capabilities
- **Age range slider** for demographic filtering
- **Team selection** with dropdown or multi-select
- **Nationality filter** for international analysis
- **Market value range** filtering
- **Performance statistics** range filters (goals, assists)

Filter Management

- **Combine multiple filters** simultaneously
- **Active filter indicators** showing current selections
- **Quick clear options** for individual or all filters
- **Filter state persistence** during session (bonus)

Player Comparison Tool (25 points)

Build a side-by-side player analysis system:

Player Selection Interface

- **Multi-select functionality** for 2-4 players
- **Drag & drop** or checkbox-based selection
- **Search within comparison** tool for quick player finding

Comparison Visualizations

- **Side-by-side statistics table** with key metrics
- **Radar chart overlay** showing attribute comparisons
- **Performance trend charts** for goals, assists over time
- **Market value comparison** with historical data

Interactive Features

- **Toggle between metric categories** (performance, attributes, market)
- **Export comparison** as image or PDF (bonus)
- **Shareable comparison URLs** (bonus)

Scouting Reports System (10 points)

Implement basic report creation and management:

Report Creation

- **Player selection** from dropdown or search
- **Match context** input (date, opponent, venue)
- **Attribute rating system** with 1-10 sliders or star ratings
- **Text areas** for strengths, weaknesses, overall assessment
- **Overall recommendation** (sign, monitor, pass)

Report Management

- **List view** of all reports with filtering by player/scout
- **Edit existing reports** with form pre-population
- **Delete reports** with confirmation
- **Export to PDF** functionality (bonus)

Responsive Design Implementation (10 points)

Ensure your application works seamlessly across devices:

Mobile-First Approach

- **Touch-friendly interfaces** with appropriate button sizes
- **Collapsible navigation** for mobile screens
- **Responsive tables** that work on small screens
- **Optimized chart displays** for mobile viewing

Cross-Platform Consistency

- **Consistent spacing** and typography across breakpoints
- **Readable text** at all screen sizes
- **Accessible color contrast** meeting WCAG guidelines
- **Fast loading times** with optimized assets

Code Quality Standards

Clean Architecture (60 points total)

Code Organization (20 points)

- **Logical file structure** with clear separation of concerns
- **Reusable components** that follow DRY principles
- **Consistent naming conventions** for files, functions, variables
- **Modular design** that's easy to maintain and extend

Error Handling (15 points)

- **User-friendly error messages** that guide users
- **API error handling** with appropriate HTTP status codes
- **Graceful fallbacks** when data is unavailable
- **Loading states** for better user experience

Code Documentation (10 points)

- **Clear README** with setup and running instructions
- **API documentation** with endpoint descriptions
- **Code comments** explaining complex logic
- **Environment setup** guide with examples

Clean Code Practices (15 points)

- **Readable code** that follows language conventions

- **Appropriate abstraction** levels throughout the application
- **Consistent formatting** using linters (ESLint, Prettier)
- **Security best practices** for authentication and data handling

---

Technical Implementation Excellence

Database Integration (15 points)

- **Proper ORM/ODM usage** (Sequelize, Mongoose, etc.)
- **Efficient queries** that don't cause performance issues
- **Data validation** at the database level
- **Connection pooling** and error handling

API Design Quality (15 points)

- **RESTful principles** with appropriate HTTP methods
- **Consistent response formats** across all endpoints
- **Proper status codes** for different scenarios
- **Request validation** with meaningful error messages

Frontend-Backend Integration (15 points)

- **Smooth data flow** between client and server
- **Proper state management** on the frontend
- **API service layers** that handle errors gracefully
- **Real-time updates** where appropriate

Bonus Opportunities (30 points)

Testing Implementation (10 points)

- **Unit tests** for critical business logic
- **API endpoint testing** with proper test data
- **Frontend component testing** for key features
- **Integration tests** demonstrating full workflows

Advanced Features (10 points)

- **Data export** functionality (CSV, Excel, PDF)
- **Advanced search** with filters and sorting
- **Real-time updates** using WebSockets
- **Performance optimization** with caching

Deployment Excellence (10 points)

- **Production deployment** of both frontend and backend
- **Environment configuration** with proper secrets management
- **Database hosting** with connection management

- **CDN setup** for static assets (bonus)

Project Structure Recommendation

```
ase-athletics-assessment/
├── frontend/                    # React/Vue application
│   ├── public/
│   ├── src/
│   │   ├── components/          # Reusable UI components
│   │   │   ├── common/          # Buttons, inputs, layouts
│   │   │   ├── charts/          # Chart components
│   │   │   └── forms/           # Form components
│   │   ├── pages/               # Main application pages
│   │   │   ├── Dashboard/       # Analytics dashboard
│   │   │   ├── Players/         # Player management
│   │   │   ├── Comparison/      # Player comparison
│   │   │   └── Reports/         # Scouting reports
│   │   ├── services/            # API communication
│   │   ├── store/               # State management
│   │   ├── utils/               # Helper functions
│   │   └── styles/              # Global styles
│   ├── package.json
│   └── .env.example
├── backend/                     # Node.js API server
│   ├── src/
│   │   ├── controllers/         # Route handlers
│   │   ├── models/              # Database models
│   │   ├── routes/              # API route definitions
│   │   ├── middleware/          # Auth, validation, etc.
│   │   ├── services/            # Business logic
│   │   └── utils/               # Helper functions
│   ├── migrations/              # Database migrations
│   ├── seeds/                   # Data seeding scripts
│   ├── tests/                   # API tests
│   ├── package.json
│   └── .env.example
├── data/                        # Provided JSON files
│   ├── player_statistics_detailed.json
│   ├── players_Data_production.json
│   ├── scout_report.json
│   └── ui_guidelines.json
├── docs/                        # Project documentation
│   ├── api-documentation.md
│   ├── database-schema.md
│   └── deployment-guide.md
├── docker-compose.yml           # Local development setup
├── README.md                    # Main project documentation
└── .gitignore
```

Submission Requirements

GitHub Repository Checklist

- [ ] **Complete source code** for both frontend and backend
- [ ] **Comprehensive README** with setup instructions
- [ ] **Environment configuration** examples (.env.example files)
- [ ] **Database setup** instructions and migration files
- [ ] **API documentation** (Swagger UI or detailed markdown)
- [ ] **Clean commit history** showing development progression
- [ ] **Proper .gitignore** excluding sensitive files

Live Demo Requirements

- [ ] **Deployed frontend** application (Netlify, Vercel, etc.)
- [ ] **Deployed backend** API (Heroku, Railway, DigitalOcean)
- [ ] **Hosted database** (PostgreSQL, MongoDB Atlas, etc.)
- [ ] **All features functional** in production environment
- [ ] **Demo credentials** provided for testing

Documentation Standards

- [ ] **Local development setup** with step-by-step instructions
- [ ] **Database schema documentation** with relationships explained
- [ ] **API endpoint documentation** with request/response examples
- [ ] **Technology stack justification** explaining your choices
- [ ] **Feature overview** with screenshots or GIFs
- [ ] **Deployment process** documentation

README Template for Your Submission

```
# ASE Athletics - Football Analytics Platform

## Live Demo
- **Frontend Application:** [Your deployed frontend URL]
- **Backend API:** [Your deployed backend URL]
- **API Documentation:** [Swagger UI or documentation URL]

## Project Overview
[Brief description of your implementation and key features]

## Technology Stack

### Frontend
- Framework: React.js / Vue.js
- State Management: Redux / Context API / Vuex
- Styling: Tailwind CSS / Material-UI / Bootstrap
- Charts: Chart.js / D3.js / Recharts
- Additional libraries: [List key dependencies]
```

```
### Backend
- Runtime: Node.js
- Framework: Express / NestJS
- Database: PostgreSQL / MongoDB / MySQL
- Authentication: JWT with bcrypt
- Validation: Joi / Yup
- Documentation: Swagger / OpenAPI

### DevOps & Deployment
- Frontend Host: Netlify / Vercel
- Backend Host: Heroku / Railway
- Database Host: [Your database hosting solution]
- Version Control: Git with GitHub

## Local Development Setup

### Prerequisites
```bash
Node.js (v16 or higher)
PostgreSQL / MongoDB
Git
```

Backend Setup

```
# Clone and navigate
git clone [your-repo-url]
cd backend

# Install dependencies
npm install

# Environment setup
cp .env.example .env
# Edit .env with your database credentials

# Database setup
npm run db:create
npm run db:migrate
npm run db:seed

# Start development server
npm run dev
# Server runs on http://localhost:5000
```

Frontend Setup

```
# Navigate to frontend
cd frontend
```
```

```
# Install dependencies
npm install

# Environment setup
cp .env.example .env
# Set REACT_APP_API_URL=http://localhost:5000

# Start development server
npm start
# Application runs on http://localhost:3000
```

Database Schema

[Include a visual diagram or detailed text description]

API Endpoints

Authentication

- `POST /api/auth/register` - User registration
- `POST /api/auth/login` - User login
- `POST /api/auth/logout` - User logout

Players

- `GET /api/players` - Get players (with pagination/filtering)
- `GET /api/players/:id` - Get specific player
- `POST /api/players` - Create new player
- `PUT /api/players/:id` - Update player
- `DELETE /api/players/:id` - Delete player

Dashboard

- `GET /api/dashboard/stats` - Dashboard metrics

Reports

- `GET /api/reports` - Get scouting reports
- `POST /api/reports` - Create report
- `PUT /api/reports/:id` - Update report
- `DELETE /api/reports/:id` - Delete report

Features Implemention checklist

- [ ] User Authentication & Session Management
- [ ] Complete Player CRUD Operations
- [ ] Interactive Analytics Dashboard
- [ ] Advanced Search & Filtering
- [ ] Player Comparison Tool

- [ ] Scouting Reports System
- [ ] Responsive Design
- [ ] Data Export Functionality

Testing

```
# Backend tests
cd backend && npm test

# Frontend tests
cd frontend && npm test
```

Deployment Notes

[Explain your deployment process and any specific configuration]

Demo Credentials

- Username: demo@ase-athletics.com
- Password: demo123

Performance Considerations

[Discuss any optimization decisions you made]

Future Enhancements

[Ideas for additional features or improvements]

Development Strategy & Timeline

Week 1: Foundation (Days 1-3)

Day 1: Project Setup

- Initialize both frontend and backend repositories
- Set up development environment and dependencies
- Configure database and basic project structure
- Implement basic authentication system

Day 2: Core Backend

- Design and implement database schema
- Create data seeding scripts for provided JSON files
- Build essential API endpoints for player CRUD
- Implement JWT authentication middleware

Day 3: Basic Frontend

- Set up frontend framework and routing
- Create authentication pages (login/register)
- Build basic player list and detail pages
- Implement API service layer

Week 2: Features (Days 4-7)

Day 4: Player Management

- Complete player CRUD interface
- Implement search and filtering functionality
- Add form validation and error handling
- Create responsive design for mobile

Day 5: Analytics Dashboard

- Build dashboard with key metrics
- Implement chart components
- Add interactive filtering for charts
- Ensure data visualization is meaningful

Day 6: Advanced Features

- Build player comparison tool
- Implement scouting reports system
- Add data export functionality
- Polish user interface and interactions

Day 7: Testing & Deployment

- Test all features thoroughly
- Deploy frontend and backend to production
- Verify all functionality works in production
- Create comprehensive documentation

Final Days (8-10): Polish & Documentation

- **Code review** and refactoring
- **Performance optimization** where needed
- **Documentation completion** with screenshots
- **Final testing** and bug fixes

Success Metrics & Evaluation

What Makes a Winning Submission

Technical Excellence

- All core features work reliably
- Clean, maintainable code structure

- Proper error handling throughout
- Responsive design that works on mobile

User Experience

- Intuitive navigation and interface
- Fast loading times and smooth interactions
- Meaningful data visualizations
- Professional appearance following design guidelines

Code Quality

- Well-organized project structure
- Consistent coding standards
- Proper separation of concerns
- Clear documentation and setup instructions

Deployment Success

- Both frontend and backend working in production
- Database properly configured and seeded
- All features functional in live environment
- Demo credentials provided for easy testing

Common Pitfalls to Avoid

Technical Issues

- Incomplete authentication implementation
- Missing error handling for API failures
- Poor database design with no relationships
- Non-responsive design that breaks on mobile

Code Quality Problems

- Inconsistent file/folder organization
- Hardcoded values instead of environment variables
- No input validation on forms
- Missing or unclear documentation

Deployment Failures

- Frontend deployed but backend not working
- Database not properly seeded with test data
- Environment variables not configured correctly
- API endpoints returning errors in production

Support & Questions

Technical Support

If you encounter issues with the provided data files or have questions about technical requirements, please reach out to our development team. We want you to succeed and are here to help clarify any confusion.

Evaluation Process

1. **Initial Review:** We'll test your live demo and review your code
2. **Technical Assessment:** Evaluation against the scoring criteria
3. **Code Quality Review:** Assessment of structure and best practices
4. **Final Selection:** Top performers will be invited for interviews

Next Steps

Selected candidates will move to final interviews where we'll discuss:

- Your technical decisions and trade-offs
- How you approached problem-solving
- Your experience building the application
- How you'd fit with the ASE Athletics team

Final Notes

This assessment represents the type of work you'll do at **ASE Athletics**. We're looking for developers who can build practical, user-friendly applications that solve real business problems in the sports analytics space.

Remember: We value working software over perfect code. Focus on building something functional and well-documented rather than trying to implement every possible feature.

Good luck! We're excited to see your implementation and potentially welcome you to the ASE Athletics development team.

ASE Athletics Development Team
Technical Assessment v2.1 - July 2025