Technical Design Document: Management Information System (MIS)

Version: 1.0

Date: September 18, 2025

---

1. Document Overview

Purpose of the Document

This Technical Design Document (TDD) provides a detailed technical blueprint for the development of the Management Information System (MIS). It translates the functional and non-functional requirements specified in the SRS into a concrete architecture and implementation plan. This document is the primary guide for the development team to write, implement, and deploy the software components, interfaces, and data structures required.

Scope of the System

The project's scope is to develop a centralized, real-time web platform to manage Wushu athletes within training centre, replacing current manual processes.

In-Scope Features:

- **User & Profile Management**: Secure registration, login, authentication, and profile management for all user roles (Athlete, Coach, Administrator).
- **Performance Tracking System**: A system for logging daily training data and competition results, with automated performance charting for athletes.
- **Online Payment Processing**: Functionality for athletes to pay admission fees online and for administrators to track these payments.
- **Role-Based Dashboards**: Customized dashboards for each user role to provide relevant, at-a-glance statistics and insights.

Out-of-Scope Features:

- Championship Management.
- A dedicated native mobile application (the system will be a mobile-responsive website).
- Integration with wearable IoT devices or sensors.

References

This TDD is based on the information provided in the following source documents:

- Business Requirements Document: Management Information System (MIS), Version 3.1
- Functional Requirements Document: Management Information System (MIS), Version 1.1
- Non-Functional Requirements Document: Management Information System (MIS), Version 1.1
- Software Requirements Specification for Management Information System (MIS), Version 1.0
- Software Design Document (SDD) for Management Information System (MIS)

Assumptions and Constraints

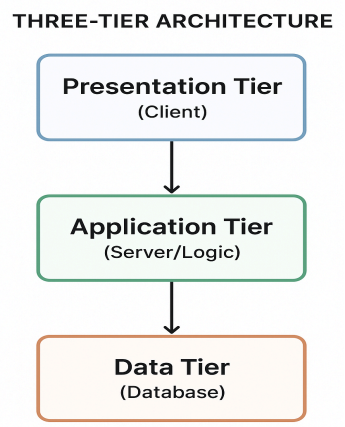
- **Assumptions**:
  - Users possess basic computer and web literacy and can interact with standard web applications.
  - Users have access to a device with a stable internet connection.

- **Constraints**:
  - **Technology**: The project is constrained to using only free and open-source software (FOSS) for development, hosting, and collaboration.
  - **Architecture**: The system architecture must be modular to facilitate future enhancements and ease of maintenance.
  - **Timeline**: The project must be completed within the specified academic timeline, following an Agile methodology with sprint cycles.

2. System Architecture

High-Level Architecture Diagram

The MIS will be built using a **Three-Tier Architecture**, which organizes the application into three logical and physical tiers: a Presentation Tier (Client), an Application Tier (Server/Logic), and a Data Tier (Database). This separation of concerns enhances scalability, maintainability, and security.



Description of Layers and Responsibilities

- **Presentation Tier (Client)**: This is the user-facing layer rendered in the web browser. It is responsible for displaying the user interface, capturing user input, and communicating exclusively with the Application Tier. It will be a fully responsive web interface that functions on desktops, tablets, and mobile phones.
- **Application Tier (Server/Logic)**: This middle tier contains all the business logic. It processes requests from the Presentation Tier, enforces business rules (like Role-Based Access Control), performs calculations, and serves as the secure intermediary between the user and the database.
- **Data Tier (Database)**: This is the back-end tier where all application data is stored and managed in a relational database. It is responsible for data persistence, retrieval, and integrity, and is only accessible through the Application Tier.

Technology Stack

In accordance with the FOSS constraint, the technology stack will consist of:

- **Backend**: A backend framework handling the RESTful API and business logic.
- **Frontend**: A modern frontend framework to build the responsive user interface.
- **Database**: A relational database like MongoDB or MySQL or PostgreSQL to store all system data.
- **API**: A RESTful API will facilitate communication between the frontend and backend.

Deployment Architecture

- The system will be hosted on a cloud platform that offers a free tier for development and initial deployment.
- The entire application and its dependencies will be containerized using **Docker** to ensure consistent and easy deployment across different environments (development, staging, production).

3. Detailed Design

Module Design

The system is decomposed into the following major software components, each with specific responsibilities.

3.1 User & Authentication Component

- **Purpose**: Handles the entire user lifecycle, including registration, authentication, security, and management of core user data stored in the users table.
- **Functions**:
  - registerUser(phoneNumber, password, email): Handles new user registration via phone OTP verification.
  - loginUser(phoneNumber, password): Authenticates a user and establishes a secure session.
  - recoverPassword(phoneNumber): Manages the password recovery workflow via OTP.
  - changePassword(userId, oldPassword, newPassword): Allows a logged-in user to change their password.
  - updateProfile(userId, profileData): Allows a user to update their own profile information.
  - setTrainingCenter(coachId, centerId): Handles a new coach selecting their training center on first login.

3.2 Athlete Profile Component

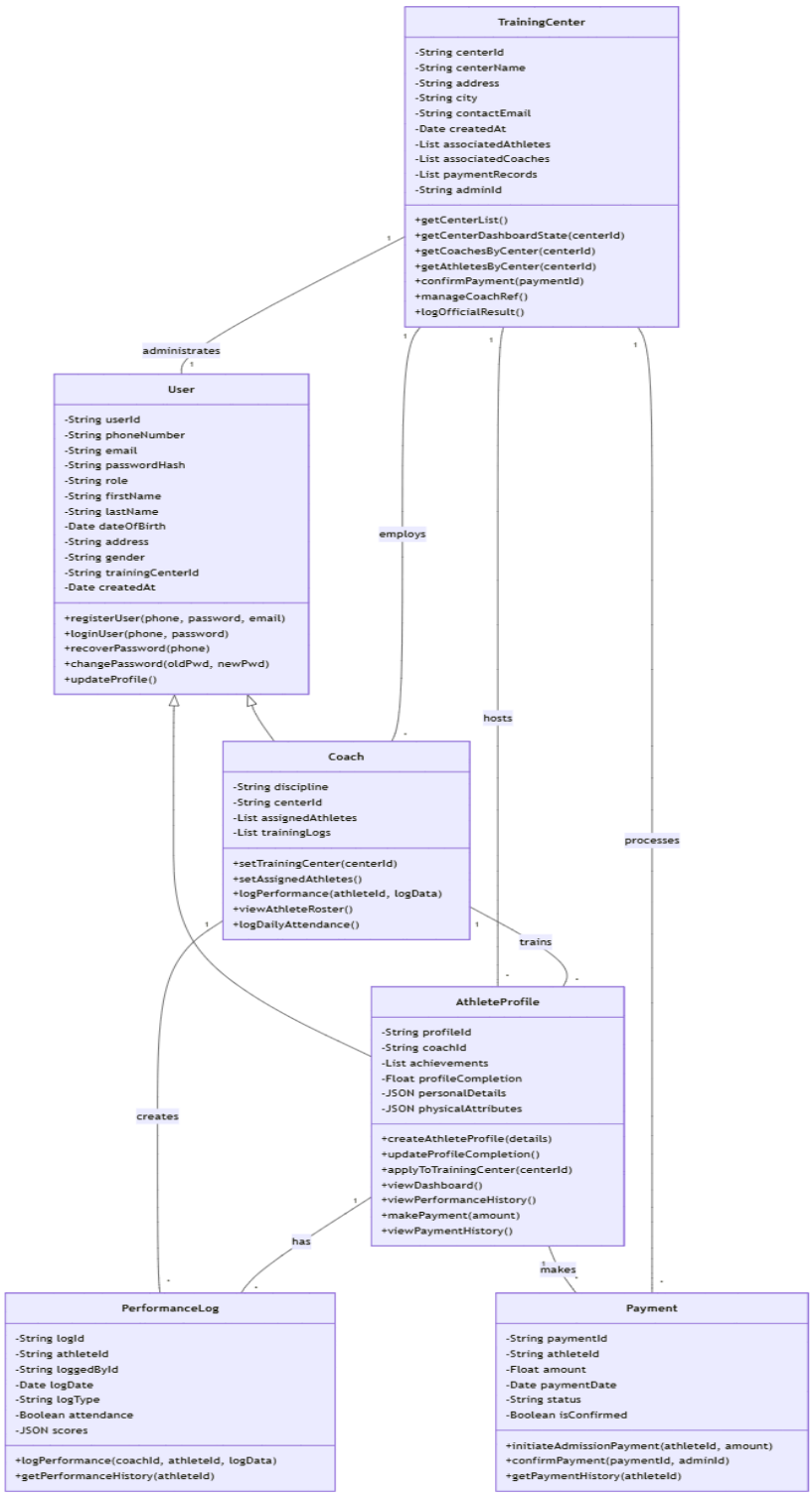
- **Purpose**: Manages all detailed information specific to athletes, primarily interacting with the athlete_profiles table.
- **Functions**:
  - createAthleteProfile(userId, firstName, lastName, dateOfBirth): Creates the initial profile for a new athlete.
  - updateProfileCompletion(userId): Calculates and updates the profile completion percentage.
  - applyToTrainingCenter(athleteId, centerId): Links an athlete's account to a specific training center.
  - getAssignedAthletes(coachId): Retrieves the roster of athletes assigned to a coach.
  - getAthletesByCenter(centerId): Retrieves all athletes registered at a specific center for an admin's view.

3.3 Performance Logging Component

- **Purpose**: Handles the recording and retrieval of athlete performance data, interacting with the performance_logs table.
- **Functions**:
  - logPerformance(coachId, athleteId, logData): Allows an authorized coach to log training data for an assigned athlete after an authorization check.
  - getPerformanceHistory(athleteId): Retrieves all performance logs for a specific athlete for dashboard display.

3.4 Payment Management Component

- **Purpose**: Manages all financial transactions, interacting with the payments table.
- **Functions**:
  - initiateAdmissionPayment(athleteId, amount): Creates a new payment record when an athlete starts the admission process.
  - confirmPayment(paymentId, adminId): Allows an administrator to confirm receipt of a payment.
  - getPaymentHistory(athleteId): Retrieves a full history of an athlete's transactions.



4. Data Design

4.1 Data Description

The information domain of the MIS will be transformed into a set of related data structures stored in a relational database. The database will serve as the central repository for all system entities. The conceptual data model consists of the following primary entities identified in the SRS:

- **Users:** A central entity representing any individual who can log in. It stores credentials and role information.
- **Profiles:** Stores personal information specific to each user role (e.g., physical attributes for an athlete). This is linked one-to-one with the Users entity.
- **Training Centers:** The organizational unit to which users belong.
- **Performance Logs:** Records of an athlete's training or competition results, logged by a coach or administrator.
- **Payments:** Records of all financial transactions, such as admission fees paid by athletes.

Key relationships include:

- a User belongs to one TrainingCenter.
- a Coach (User) is assigned many Athletes (Users).
- an Athlete has many PerformanceLogs and many Payments.

4.2 Data Dictionary

The following tables define the structure for the primary database entities.

Table: users

This table stores the core information for any individual who can log into the system. It manages credentials, roles, and links to a training center.

Column Name	Data Type	Constraints	Description
user_id	SERIAL	PRIMARY KEY	Unique identifier for the user.
phone_number	VARCHAR(15)	UNIQUE, NOT NULL	User's registered phone number, used for login and

			OTP.
email	VARCHAR(255)	UNIQUE	User's optional email address.
password_hash	VARCHAR(255)	NOT NULL	Securely hashed user password (using bcrypt/Argon2).
role	ENUM('Athlete', 'Coach', 'Admin')	NOT NULL	The role assigned to the user.
training_center_id	INTEGER	FOREIGN KEY	Links the user to their training center.
created_at	TIMESTAMP	DEFAULT NOW()	Timestamp of account creation.
gender	VARCHAR(50)		User's gender.

Table: athlete_profiles

This table contains detailed personal and athletic information specific to users with the '**Athlete**' role.

Column Name	Data Type	Constraints	Description
profile_id	SERIAL	PRIMARY KEY	Unique identifier for the profile.
user_id	INTEGER	FOREIGN KEY (users), UNIQUE	Links to the corresponding user account.
first_name	VARCHAR(100)	NOT NULL	Athlete's first name.
last_name	VARCHAR(100)	NOT NULL	Athlete's last name.

date_of_birth	DATE	NOT NULL	Athlete's date of birth.
coach_id	INTEGER	FOREIGN KEY (users)	The user_id of the coach assigned to this athlete.
achievements	TEXT		A summary of the athlete's achievements.
profile_completion	INTEGER	DEFAULT 0	Percentage of profile completion, tracked for athletes.
physical_attributes	JSONB		Flexible field for storing physical data (e.g., height, weight).

Table: Coach

Column Name	Data Type	Constraints	Description
profile_id	SERIAL	PRIMARY KEY	Unique identifier for the profile.
user_id	INTEGER	FOREIGN KEY (users), UNIQUE	Links to the corresponding user account.
discipline	VARCHAR(100)	NOT NULL	The coach's primary Wushu discipline (e.g., Taolu, Sanda).

Table: training_centers

This table stores information about the individual training centers that use the MIS. Every user in the system is associated with one of these centers.

Column Name	Data Type	Constraints	Description
center_id	SERIAL	PRIMARY KEY	Unique identifier for the training center.
center_name	VARCHAR(255)	NOT NULL, UNIQUE	The official name of the training center.
address	TEXT		The physical address of the center.
city	VARCHAR(100)		The city where the center is located.
contact_email	VARCHAR(255)		A primary contact email for the center.
created_at	TIMESTAMP	DEFAULT NOW()	Timestamp of when the center was added to the system.

The training_center_id foreign key in the users table references the training_centers table, establishing the relationship between a user and their designated center.

Table: performance_logs

This table stores records of an athlete's performance, whether from daily training or official competitions.

Column Name	Data Type	Constraints	Description
log_id	SERIAL	PRIMARY KEY	Unique identifier for the performance entry.
athlete_id	INTEGER	FOREIGN KEY (users)	The athlete for whom the data is being logged.

logged_by_id	INTEGER	FOREIGN KEY (users)	The coach or admin who logged the data.
log_date	DATE	NOT NULL	The date of the training or competition.
log_type	ENUM('Training', 'Official')	NOT NULL	Differentiates between practice and official results.
attendance	BOOLEAN		Daily attendance status logged by a coach.
scores	JSONB		Flexible field to store different score types (e.g., Taolu, Sanda).

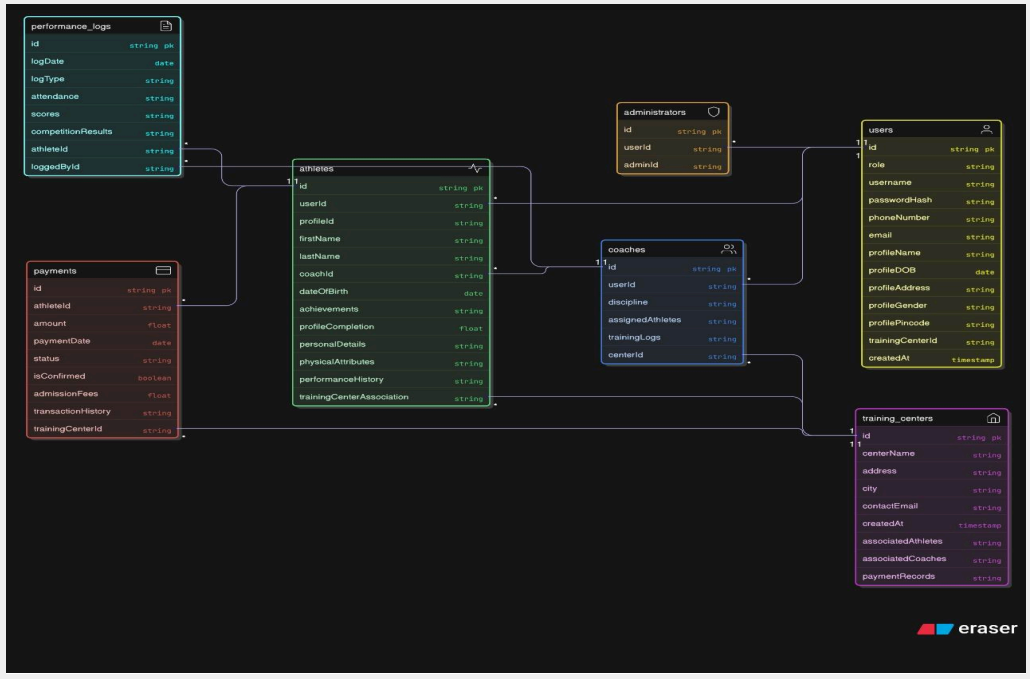
Table: payments

This table tracks all financial transactions within the system, such as admission fees.

Column Name	Data Type	Constraints	Description
payment_id	SERIAL	PRIMARY KEY	Unique identifier for the transaction.
athlete_id	INTEGER	FOREIGN KEY (users)	The athlete who made the payment.
amount	DECIMAL(10, 2)	NOT NULL	The amount paid.
payment_date	TIMESTAMP	DEFAULT NOW()	Timestamp of the transaction.
status	ENUM('Pending', 'Completed', 'Failed')	NOT NULL	The status of the payment gateway transaction.

is_confirmed	BOOLEAN	DEFAULT FALSE	Flag set by an Administrator upon confirming payment receipt.

ER Diagram:



Data Structures

- Flexible data storage will be achieved using the **JSONB** data type for fields like scores in performance_logs and physical_attributes in athlete_profiles. This allows for storing varied data structures without altering the database schema.

API Design

The system shall provide a **RESTful API** for all communication between the frontend client and the backend server.

- **Endpoints**: Endpoints will be designed logically around the system's resources. For example:
  - POST /api/auth/register - Corresponds to registerUser()
  - POST /api/auth/login - Corresponds to loginUser()
  - GET /api/athletes/{athleteId}/performance - Corresponds to getPerformanceHistory()
  - POST /api/coaches/log-performance - Corresponds to logPerformance()
- **Request/Response Format**: All data will be exchanged in **JSON** format.
- **Authentication**: The API will be secured using session tokens generated upon successful

login, which must be included in the headers of subsequent requests.

Algorithm Design

- **User Registration & OTP Verification**:
  1. User submits a phone number.
  2. The system checks if the phone number is unique in the `users` table.
  3. A 6-digit OTP is generated and sent via an SMS gateway.
  4. The OTP is temporarily stored with a short expiry time (e.g., 10 minutes).
  5. The user submits the OTP; if it matches the stored value, the account is created.
- **Authorization Check for Performance Logging**:
  1. A coach initiates a logPerformance request for a specific athlete.
  2. The Application Tier verifies that the athleteId is officially assigned to the coachId by querying the athlete_profiles table.
  3. If the association is valid, the data is written to the performance_logs table. If not, an "Unauthorized" error is returned.

4. User Interface Design

Wireframes / Mockups

Conceptual mockups define the layout of key system screens. The UI will be clean, modern, intuitive, and fully responsive.

- [Insert Login & Registration Screen Mockup]
- [Insert Athlete Dashboard Mockup]
- [Insert Coach's Athlete Roster Mockup]
- [Insert Administrator's Payment Confirmation Page Mockup]

Navigation Structure

A consistent navigation structure will be used across the application to provide a predictable user experience. A standard navigation bar will contain links to key areas like "Dashboard," "My Profile," and a "Logout" button.

UI Components and Interaction Behavior

- **Athlete Dashboard**:
  - **Profile Completion Widget**: A visual progress bar showing profile completeness (e.g., "75% complete") with a link to the profile editing page.
  - **Performance History Chart**: An interactive line graph visualizing progress over time. Hovering over a data point will display a tooltip with the exact date and score. A

dropdown will allow filtering by different metrics.

5. Security Design

Authentication and Authorization

- **Authentication**: User authentication is handled via a registered phone number and a securely hashed password. Password recovery and contact updates require OTP verification sent to the user's registered phone or email.

- **Authorization**: The system implements strict **Role-Based Access Control (RBAC)**. Access is restricted based on the user's role, which is stored in the users table. This is enforced in the Application Tier's business logic to ensure users can only perform actions appropriate to their role (e.g., a coach can only log data for their assigned athletes).

Data Encryption and Protection

- **Data in Transit**: All data transmitted between the client and server will be encrypted using **HTTPS (TLS 1.2 or higher)**.

- **Data at Rest**: All user passwords will be protected using a strong, one-way hashing algorithm (e.g., bcrypt, Argon2) with a unique salt for each user. The hashed value is stored in the password_hash column of the users table.

Logging and Auditing

A logging mechanism will be implemented to record key security events, such as login attempts and profile changes, to support auditing and troubleshooting.

6. Performance & Scalability

Performance Targets

The system is designed to meet the following performance benchmarks:

- **Load Time**: All user-facing pages must load in **under 3 seconds** on a standard mobile network.

- **API Response Time**: 95% of all standard API GET requests must be completed in **under 500 milliseconds**.

- **Concurrency**: The system must handle a peak load of **100 concurrent users** without performance degradation exceeding 20% of baseline response times.

Scalability Approach

- The **Three-Tier Architecture** was chosen specifically to enhance scalability.

- The system is architected to scale efficiently to handle a growing number of users and a large volume of historical data without requiring significant re-engineering.

7. Integration Design

Interaction with Third-Party Systems

The system is dependent on the following external integrations:

- **SMS Gateway**: Integration with a third-party SMS gateway service is required to send OTPs for user registration and account recovery.

- **Payment Gateway**: To fulfill the online payment requirements, the system will need to integrate with a payment gateway (e.g., PhonePe, Razorpay) in the future.

8. Error Handling & Recovery

Exception Handling Strategy

- The system will provide clear, immediate, and user-friendly feedback for all actions.
- For invalid input, descriptive error messages will be displayed to guide the user (e.g., "User not found" during login).

Fault Tolerance and Failover Design

- **Availability**: The system is required to achieve **99.5% uptime**.
- **Recovery**: Automated **daily database backups** will be performed to prevent data loss. A documented disaster recovery plan will be in place to restore service within 4 hours of a critical failure.

9. Testing & Validation Approach

Testing will be aligned with the requirements to ensure the system is built correctly. The **Requirements Matrix** in the SDD will be the primary tool for tracing test cases back to specific requirements.

- **Unit Testing**: Each function within the Component Design (e.g., loginUser, confirmPayment) will be tested individually to validate its internal logic.

- **Integration Testing**: Tests will be designed to verify interactions between components, such as ensuring that applyToTrainingCenter() correctly triggers initiateAdmissionPayment().
- **System Testing**: End-to-end test cases will validate entire user workflows, such as the complete athlete registration and payment process.
- **Performance Testing**: The system will be tested against the specified performance targets (e.g., load time, concurrency) to ensure compliance with NFRs.

10. Deployment & Maintenance

Environment Setup

The system will be hosted on a cloud platform, utilizing its free tier for the initial development and deployment phases.

CI/CD Pipeline

The use of **Docker** for containerization is a foundational element that enables a modern CI/CD pipeline. This will ensure that deployments are automated, consistent, and reliable across all environments.

Rollback and Recovery Strategy

The recovery strategy is based on the ability to restore the system from automated daily backups, with a target recovery time of under 4 hours.

11. Appendix

Glossary of Terms

- **API**: Application Programming Interface.
- **BRD**: Business Requirements Document.
- **FOSS**: Free and Open-Source Software.
- **MIS**: Management Information System, the official name of the project.
- **OTP**: One-Time Password, a temporary code for user verification.
- **RBAC**: Role-Based Access Control, a security model restricting access based on user roles.

- **Sanda**: The full-contact, sparring discipline of Wushu.
- **Taolu**: The non-combat, forms-based discipline of Wushu.

Diagrams

This section is a placeholder for detailed analysis and design diagrams.

- [Insert Use Case Diagram]
- Sequence Diagram: