**Gameplan – Sports Team & Match Scheduling API**

**Project Summary:**

Gameplan is a backend API built using **FastAPI, MongoDB, JWT, Logging, and Custom Exception Handling**. It provides secure access to **user registration, authentication, team management, player management, and match scheduling**. All endpoints are protected with **JWT tokens** and only authorized users can perform certain actions. The system ensures **data consistency**, clear error handling, and activity tracking using structured logs. With this project, sports organizations can easily **manage teams, players, and matches** in a reliable and secure way.

**Features Implemented:**

* User registration and login with **JWT authentication**.
* Create and fetch **teams**.
* Create and fetch **players**, including fetching a player by ID.
* Schedule and fetch **matches**.
* **Role-based access control** (Admin / Coach).
* **Logging** for every request, warning, and error.
* **Custom exception handling** for predictable error messages.

**Technologies Used:**

* **FastAPI** – Web framework (Python)
* **MongoDB (pymongo)** – NoSQL database
* **JWT (python-jose)** – Secure authentication
* **Passlib (bcrypt)** – Password hashing
* **Pydantic** – Request validation and models
* **Logging** – Structured logs for monitoring
* **Custom Exceptions** – For domain-specific errors

**API Endpoints (Implemented):**

| **Method** | **Path** | **Access** | **Description** |
| --- | --- | --- | --- |
| POST | /register | Public | Register a new user |
| POST | /login | Public | Login and get JWT token |
| POST | /create\_team | Admin only | Create a team |
| GET | /all\_teams | Authenticated | Get all teams |
| POST | /add\_player | Admin/Coach only | Add a player |
| GET | /all\_players | Authenticated | Get all players |
| GET | /players/{id} | Authenticated | Get player details |
| POST | /create\_match | Admin only | Schedule a match |
| GET | /all\_matches | Authenticated | Get all matches |

**Project Folder Structure:**

* **main.py** – Entry point (includes routers)
* **app/database.py** – MongoDB connection
* **app/models.py** – Pydantic models
* **app/auth.py** – JWT authentication logic
* **app/routes/** – auth\_routes, team\_routes, player\_routes, match\_routes
* **app/utils/** – helpers, logger, exceptions

**Data Models:**

**User** → username, email, password (hashed), role  
**Team** → name, coach, city  
**Player** → name, age, position, team\_id  
**Match** → home\_team\_id, away\_team\_id, date, location

**Logging & Exception Handling:**

* **Info logs** for successful actions.
* **Warning logs** for invalid or missing resources.
* **Error logs** with stack traces for unexpected failures.
* Custom exceptions like UserAlreadyExistsException, TeamAlreadyExistsException, PlayerNotFoundException.
* Global errors return safe JSON messages (e.g., {"detail": "Internal Server Error"}).

**Key Notes:**

* JWT is required for all protected endpoints.
* Role-based access control ensures only admins/coaches can modify sensitive data.
* Passwords are **securely hashed** (bcrypt).
* MongoDB ObjectIds are converted to strings before returning responses.