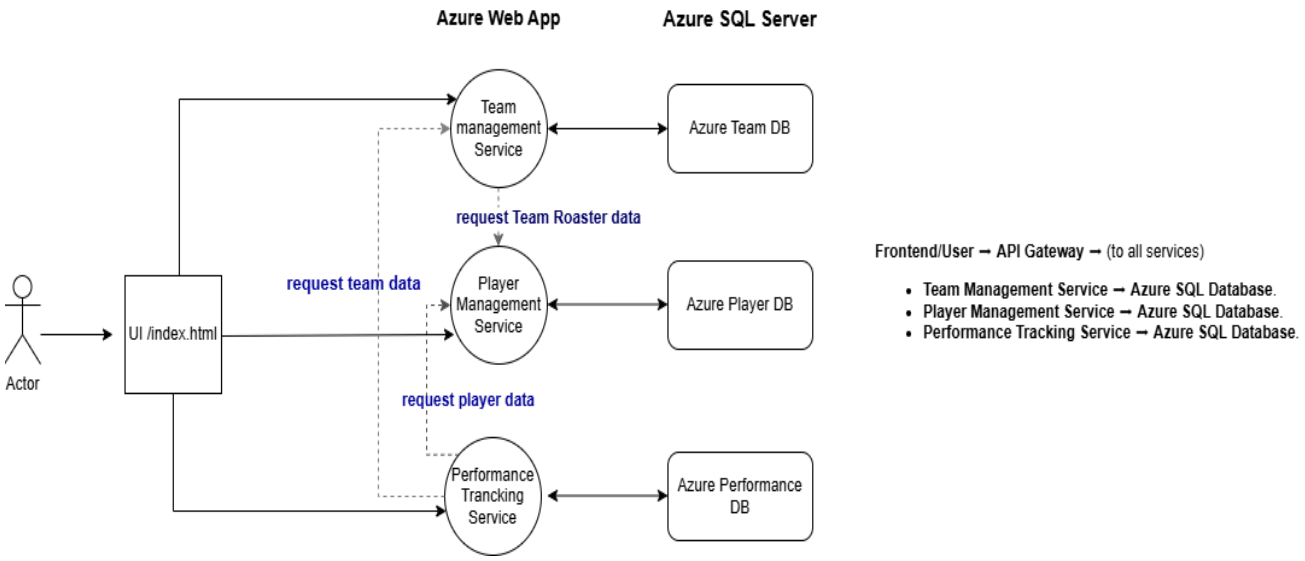
Final Project: Fantasy Sports Team Management System Documentation

DEV422

Team 3: Oscar Moreno, Ying Jiang & Jiwon Oh

1. Architecture Diagram



UI Layer:

- The user interacts with a web interface (/index.html) to perform actions like managing teams, players, and competitions.

Team Management Service:

- Handles team-related operations such as creation, updates, and fetching rosters.
- Stores data in Azure Team DB and fetches player details from the Player Management Service.

Player Management Service:

- Manages player-related operations like drafting and releasing players.
- Stores player data in Azure Player DB and communicates only through its APIs.

Performance Management Service:

- Simulates competitions and updates player performance stats.
- Calls APIs from both the Team Management and Player Management services to retrieve required data and stores stats in Azure Performance DB.

Database Layer:

- Each service has its own database in Azure SQL to maintain modularity. Services do not access each other's databases directly.

2. API Documentation

TEAM_SERVICE_URL: https://fantasysportsteammanagementsystem.azurewebsites.net
PLAYER_SERVICE_URL: https://player-management-service.azurewebsites.net
PERFORMANCE_SERVICE_URL: https://performance-tracking-service.azurewebsites.net

Team Management Service

Base URL**:** https://<team-service-url>/api/team

Method	Endpoint	Description	Request Body	Response
POST	/create	Create a new team.	{ "teamName": "string" }	{ "message": "Team created." }
GET	/	Get all teams.	None	[{ "id": int, "teamName": "string" }]
PUT	/update/{id}	Update team name.	{ "newName": "string" }	{ "id": int, "teamName": "string" }
GET	/{teamId}/roster	Get team roaster in a team.	None	{ "teamName": "string", "players": [] }

Player Management Service

Base URL: https://<player-service-url>/api/player

Method	Endpoint	Description	Request Body	Response
GET	/getTeamRoaster	Get all players in a team.	{“teamId”=int}	["teamPlayers": []]
POST	/draft	Draft a player to a team.	{ "playerId": int, "teamId": int }	{ "message": "Player drafted successfully." }
POST	/release	Release a player from a team.	{ "playerId": int, "teamId": int }	{ "message": "Player released successfully." }

Performance Tracking Service

Base URL: https://<performance-service-url>/api/competitions

Method	Endpoint	Description	Request Body	Response
POST	/simulate	Simulates competition and updates performance stats.	None	[{ "id": int, "teamId": int, "teamName": "string", "playerId": int, "playerName": "string", "points": int, "assists": int, "fouls": int }]
GET	/	Retrieves all performance stats with updated team and player info.	None	[{ "id": int, "teamId": int, "teamName": "string", "playerId": int, "playerName": "string", "position": "string", "points": int, "assists": int, "fouls": int }]
PUT	/{id}	Updates the performance stats for a specific record.	{ "points": int, "assists": int, "fouls": int }	No Content

3. Guide for Testing Services

(Ensure all services are running on Azure App Service and use Azure SQL as the centralized database. Postman for API testing, browser for UI testing, azure SQL Database for database verification.)

- **Testing Services Independently:**
  - **Team management Service (**TEAM_SERVICE_URL**):**
    - **Create Team:** POST /api/team/create by sending a new team name.
    - **Update Team:** PUT /api/team/update/{id} with a new team name.
    - **Retrieve All Teams:** GET /api/team to list all available teams.
    - **Fetch Team Roster:** GET /api/team/{teamId}/roster to retrieve player detail for a team.
  - **Player Management Service (**PLAYER_SERVICE_URL**):**
    - **Draft Player:** POST /api/player/draft by sending a player id and team id.
    - **Release Player:** POST /api/player/release with a player id.
    - **Fetch Team Roaster:** GET /api/player/getTeamRoaster?teamId={teamId} for the roster of a specific team.
  - **Performance Tracking Service (**PERFORMANCE_SERVICE_URL**):**
    - **Simulate Competition:** POST /api/competitions/simulate to update player performance stats.
    - **Retrieve All Stats:** GET /api/competitions for a complete list of performance stats.
    - **Update Stat:** PUT /api/competitions/{id} with updated performance data.
- **Testing Services in Combination:**
  - **Test Team Management Service with Player Management Service:** GET /api/team/{teamId}/roster and verify that player details come from GET /api/player/getTeamRoaster?teamId={teamId}.
  - **Test Performance Tracking Service with Player and Team Services:**

    Check Simulate performance's team names using

    GET /api/team/{teamId}/roster.

    Ensure player name and position integrate correctly using

    GET /api/player/getTeamRoaster?teamId={teamId}.