F1 Fantasy League - Komplett Projektspecifikation

1. PROJEKTÖVERSIKT

1.1 Projektbeskrivning

F1 Fantasy League är en fullstack-applikation där användare kan skapa virtuella F1-team, tävla mot vänner och andra användare baserat på verkliga F1-prestationer. Systemet följer Clean Architecture med CQRS/Mediator pattern och Repository pattern.

1.2 Målgrupp

- F1-entusiaster som vill engagera sig mer i sporten
- Fantasy sport-spelare som söker nya utmaningar
- Vängrupper som vill tävla mot varandra

1.3 Teknisk Stack

Backend:

- .NET 8 Web API
- Entity Framework Core
- MediatR (CQRS)
- Repository Pattern
- AutoMapper
- FluentValidation
- JWT Authentication
- SignalR (Real-time updates)

Frontend:

- React 18 med TypeScript
- Tailwind CSS
- React Query/TanStack Query
- React Router
- Zustand (State Management)

Database:

- SQL Server/PostgreSQL
- Redis (Caching)

External APIs:

- Ergast F1 API (Historisk data)
- OpenF1 API (Live data)

---

2. FUNKTIONELLA KRAV

2.1 Användarhantering

- **FR-001:** Användare ska kunna registrera sig med email och lösenord
- **FR-002:** Användare ska kunna logga in och ut
- **FR-003:** Användare ska kunna återställa lösenord
- **FR-004:** Användare ska kunna uppdatera sin profil
- **FR-005:** Användare ska kunna ladda upp profilbild
- **FR-006:** Användare ska kunna ta bort sitt konto

2.2 Team Management

- **FR-007:** Användare ska kunna skapa ett fantasy team
- **FR-008:** Användare ska kunna välja team-namn och logotyp
- **FR-009:** Användare ska kunna välja förare inom budgetgränser
- **FR-010:** Användare ska kunna välja konstruktör inom budgetgränser
- **FR-011:** Användare ska kunna göra transfers mellan race
- **FR-012:** Användare ska ha begränsade transfers per säsong
- **FR-013:** Användare ska kunna sätta kapten (dubbel poäng)

2.3 Liga System

- **FR-014:** Användare ska kunna skapa privata ligor
- **FR-015:** Användare ska kunna gå med i ligor via kod
- **FR-016:** Användare ska kunna se liga-leaderboards
- **FR-017:** Användare ska kunna lämna ligor
- **FR-018:** Liga-ägare ska kunna kicka medlemmar
- **FR-019:** Det ska finnas en global liga för alla användare

2.4 Poängsystem

- **FR-020:** Poäng ska beräknas baserat på race-resultat
- **FR-021:** Bonus-poäng för pole position, snabbaste varv, etc.
- **FR-022:** Poäng ska uppdateras automatiskt efter varje race

- **FR-023:** Historisk poäng ska sparas för varje race
- **FR-024:** Kapten ska få dubbel poäng

2.5 Race Weekend

- **FR-025:** Användare ska kunna se race-schema
- **FR-026:** Användare ska kunna se live race-resultat
- **FR-027:** Team-locking innan varje race weekend
- **FR-028:** Push-notifikationer för viktiga events

2.6 Statistik och Analys

- **FR-029:** Användare ska kunna se detaljerad team-statistik
- **FR-030:** Användare ska kunna se förare/konstruktör-prestationer
- **FR-031:** Användare ska kunna se historiska resultat
- **FR-032:** Leaderboards med olika tidsperioder

---

3. DATAMODELLER

3.1 Core Entities

User

```csharp
public class User
{
    public Guid Id { get; set; }
    public string Email { get; set; }
    public string UserName { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string PasswordHash { get; set; }
    public string? ProfileImageUrl { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }
    public bool IsEmailConfirmed { get; set; }
    public DateTime? LastLoginAt { get; set; }

    // Navigation Properties
    public ICollection<FantasyTeam> FantasyTeams { get; set; }
    public ICollection<UserLeague> UserLeagues { get; set; }
    public ICollection<League> OwnedLeagues { get; set; }
}
```

FantasyTeam

```csharp
public class FantasyTeam
{
    public Guid Id { get; set; }
    public Guid UserId { get; set; }
    public Guid SeasonId { get; set; }
    public string TeamName { get; set; }
    public string? TeamLogoUrl { get; set; }
    public decimal Budget { get; set; }
    public int TransfersRemaining { get; set; }
    public int TotalPoints { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }

    // Navigation Properties
    public User User { get; set; }
    public Season Season { get; set; }
    public ICollection<TeamDriver> TeamDrivers { get; set; }
    public ICollection<TeamConstructor> TeamConstructors { get; set; }
    public ICollection<GameweekTeam> GameweekTeams { get; set; }
}
```

Driver

csharp

```csharp
public class Driver
{
    public Guid Id { get; set; }
    public int DriverNumber { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Code { get; set; } // HAM, VER, etc.
    public string Nationality { get; set; }
    public DateTime DateOfBirth { get; set; }
    public string? ImageUrl { get; set; }
    public decimal CurrentPrice { get; set; }
    public bool IsActive { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }

    // Navigation Properties
    public ICollection<SeasonDriver> SeasonDrivers { get; set; }
    public ICollection<TeamDriver> TeamDrivers { get; set; }
    public ICollection<RaceResult> RaceResults { get; set; }
}
```

Constructor

csharp

```csharp
public class Constructor
{
    public Guid Id { get; set; }
    public string Name { get; set; }
    public string Nationality { get; set; }
    public string? LogoUrl { get; set; }
    public string? Color { get; set; }
    public decimal CurrentPrice { get; set; }
    public bool IsActive { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }

    // Navigation Properties
    public ICollection<SeasonConstructor> SeasonConstructors { get; set; }
    public ICollection<TeamConstructor> TeamConstructors { get; set; }
    public ICollection<ConstructorResult> ConstructorResults { get; set; }
}
```

Season

csharp

```csharp
public class Season
{
    public Guid Id { get; set; }
    public int Year { get; set; }
    public string Name { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public bool IsActive { get; set; }
    public bool IsCompleted { get; set; }
    public DateTime CreatedAt { get; set; }

    // Navigation Properties
    public ICollection<Race> Races { get; set; }
    public ICollection<SeasonDriver> SeasonDrivers { get; set; }
    public ICollection<SeasonConstructor> SeasonConstructors { get; set; }
    public ICollection<FantasyTeam> FantasyTeams { get; set; }
}
```

Race

csharp

```csharp
public class Race
{
    public Guid Id { get; set; }
    public Guid SeasonId { get; set; }
    public string Name { get; set; }
    public string Country { get; set; }
    public string Circuit { get; set; }
    public DateTime RaceDate { get; set; }
    public DateTime? QualifyingDate { get; set; }
    public DateTime? Practice1Date { get; set; }
    public DateTime? Practice2Date { get; set; }
    public DateTime? Practice3Date { get; set; }
    public int Round { get; set; }
    public RaceStatus Status { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }

    // Navigation Properties
    public Season Season { get; set; }
    public ICollection<RaceResult> RaceResults { get; set; }
    public ICollection<ConstructorResult> ConstructorResults { get; set; }
    public ICollection<GameweekTeam> GameweekTeams { get; set; }
}
```

League

```csharp
public class League
{
    public Guid Id { get; set; }
    public Guid OwnerId { get; set; }
    public Guid SeasonId { get; set; }
    public string Name { get; set; }
    public string? Description { get; set; }
    public string InviteCode { get; set; }
    public bool IsPublic { get; set; }
    public int MaxMembers { get; set; }
    public DateTime CreatedAt { get; set; }
    public DateTime UpdatedAt { get; set; }

    // Navigation Properties
    public User Owner { get; set; }
    public Season Season { get; set; }
    public ICollection<UserLeague> UserLeagues { get; set; }
}
```

3.2 Junction/Relationship Entities

TeamDriver

```csharp
public class TeamDriver
{
    public Guid Id { get; set; }
    public Guid FantasyTeamId { get; set; }
    public Guid DriverId { get; set; }
    public bool IsCaptain { get; set; }
    public decimal PurchasePrice { get; set; }
    public DateTime AddedAt { get; set; }
    public DateTime? RemovedAt { get; set; }

    // Navigation Properties
    public FantasyTeam FantasyTeam { get; set; }
    public Driver Driver { get; set; }
}
```

TeamConstructor

csharp

```csharp
public class TeamConstructor
{
    public Guid Id { get; set; }
    public Guid FantasyTeamId { get; set; }
    public Guid ConstructorId { get; set; }
    public decimal PurchasePrice { get; set; }
    public DateTime AddedAt { get; set; }
    public DateTime? RemovedAt { get; set; }

    // Navigation Properties
    public FantasyTeam FantasyTeam { get; set; }
    public Constructor Constructor { get; set; }
}
```

UserLeague

csharp

```csharp
public class UserLeague
{
    public Guid Id { get; set; }
    public Guid UserId { get; set; }
    public Guid LeagueId { get; set; }
    public DateTime JoinedAt { get; set; }
    public DateTime? LeftAt { get; set; }
    public bool IsActive { get; set; }

    // Navigation Properties
    public User User { get; set; }
    public League League { get; set; }
}
```

SeasonDriver

csharp

```csharp
public class SeasonDriver
{
    public Guid Id { get; set; }
    public Guid SeasonId { get; set; }
    public Guid DriverId { get; set; }
    public Guid? ConstructorId { get; set; }
    public decimal StartingPrice { get; set; }
    public decimal CurrentPrice { get; set; }
    public int TotalPoints { get; set; }

    // Navigation Properties
    public Season Season { get; set; }
    public Driver Driver { get; set; }
    public Constructor? Constructor { get; set; }
}
```

SeasonConstructor

csharp

```csharp
public class SeasonConstructor
{
    public Guid Id { get; set; }
    public Guid SeasonId { get; set; }
    public Guid ConstructorId { get; set; }
    public decimal StartingPrice { get; set; }
    public decimal CurrentPrice { get; set; }
    public int TotalPoints { get; set; }

    // Navigation Properties
    public Season Season { get; set; }
    public Constructor Constructor { get; set; }
}
```

3.3 Result Entities

RaceResult

csharp

```csharp
public class RaceResult
{
    public Guid Id { get; set; }
    public Guid RaceId { get; set; }
    public Guid DriverId { get; set; }
    public int Position { get; set; }
    public int? GridPosition { get; set; }
    public int Points { get; set; }
    public TimeSpan? Time { get; set; }
    public bool FastestLap { get; set; }
    public bool PolePosition { get; set; }
    public bool DidNotFinish { get; set; }
    public bool DidNotStart { get; set; }
    public bool Disqualified { get; set; }
    public int FantasyPoints { get; set; }
    public DateTime CreatedAt { get; set; }

    // Navigation Properties
    public Race Race { get; set; }
    public Driver Driver { get; set; }
}
```

ConstructorResult

csharp

```csharp
public class ConstructorResult
{
    public Guid Id { get; set; }
    public Guid RaceId { get; set; }
    public Guid ConstructorId { get; set; }
    public int Position { get; set; }
    public int Points { get; set; }
    public int FantasyPoints { get; set; }
    public DateTime CreatedAt { get; set; }

    // Navigation Properties
    public Race Race { get; set; }
    public Constructor Constructor { get; set; }
}
```

GameweekTeam

csharp

```csharp
public class GameweekTeam
{
    public Guid Id { get; set; }
    public Guid FantasyTeamId { get; set; }
    public Guid RaceId { get; set; }
    public int Points { get; set; }
    public int TransfersUsed { get; set; }
    public int TransferCost { get; set; }
    public DateTime TeamLockTime { get; set; }
    public DateTime CreatedAt { get; set; }

    // Navigation Properties
    public FantasyTeam FantasyTeam { get; set; }
    public Race Race { get; set; }
    public ICollection<GameweekDriver> GameweekDrivers { get; set; }
    public ICollection<GameweekConstructor> GameweekConstructors { get; set; }
}
```

GameweekDriver

csharp

```csharp
public class GameweekDriver
{
    public Guid Id { get; set; }
    public Guid GameweekTeamId { get; set; }
    public Guid DriverId { get; set; }
    public bool IsCaptain { get; set; }
    public int Points { get; set; }
    public decimal Price { get; set; }

    // Navigation Properties
    public GameweekTeam GameweekTeam { get; set; }
    public Driver Driver { get; set; }
}
```

GameweekConstructor

csharp

```csharp
public class GameweekConstructor
{
    public Guid Id { get; set; }
    public Guid GameweekTeamId { get; set; }
    public Guid ConstructorId { get; set; }
    public int Points { get; set; }
    public decimal Price { get; set; }

    // Navigation Properties
    public GameweekTeam GameweekTeam { get; set; }
    public Constructor Constructor { get; set; }
}
```

3.4 Enums

csharp

```csharp
public enum RaceStatus
{
    Scheduled = 0,
    InProgress = 1,
    Completed = 2,
    Cancelled = 3,
    Postponed = 4
}

public enum TransferType
{
    Purchase = 0,
    Sale = 1,
    WildCard = 2
}

public enum NotificationType
{
    RaceStart = 0,
    RaceEnd = 1,
    TeamLocked = 2,
    LeagueInvite = 3,
    PriceChange = 4
}
```

---

4. API ENDPOINTS

4.1 Authentication Endpoints

POST /api/auth/register

```
json

Request:
{
    "email": "user@example.com",
    "userName": "username",
    "firstName": "John",
    "lastName": "Doe",
    "password": "SecurePassword123!",
    "confirmPassword": "SecurePassword123!"
}

Response:
{
    "token": "jwt_token_here",
    "user": {
        "id": "guid",
        "email": "user@example.com",
        "userName": "username",
        "firstName": "John",
        "lastName": "Doe"
    }
}
```

POST /api/auth/login

```
json

Request:
{
    "email": "user@example.com",
    "password": "SecurePassword123!"
}

Response:
{
    "token": "jwt_token_here",
    "user": {
        "id": "guid",
        "email": "user@example.com",
        "userName": "username",
        "firstName": "John",
        "lastName": "Doe"
    }
}
```

POST /api/auth/refresh-token

POST /api/auth/forgot-password

POST /api/auth/reset-password

POST /api/auth/change-password

4.2 User Management Endpoints

GET /api/users/profile

PUT /api/users/profile

POST /api/users/upload-avatar

DELETE /api/users/account

4.3 Season Endpoints

GET /api/seasons

```
json

Response:
[
    {
        "id": "guid",
        "year": 2024,
        "name": "2024 Formula 1 World Championship",
        "startDate": "2024-03-01T00:00:00Z",
        "endDate": "2024-12-01T00:00:00Z",
        "isActive": true,
        "isCompleted": false
    }
]
```

GET /api/seasons/{id}

GET /api/seasons/current

GET /api/seasons/{id}/races

GET /api/seasons/{id}/drivers

GET /api/seasons/{id}/constructors

4.4 Driver Endpoints

GET /api/drivers

```
json

Response:
[
    {
        "id": "guid",
        "driverNumber": 44,
        "firstName": "Lewis",
        "lastName": "Hamilton",
        "code": "HAM",
        "nationality": "British",
        "dateOfBirth": "1985-01-07T00:00:00Z",
        "imageUrl": "https://example.com/drivers/hamilton.jpg",
        "currentPrice": 12.5,
        "isActive": true,
        "constructor": {
            "id": "guid",
            "name": "Mercedes",
            "color": "#00D2BE"
        }
    }
]
```

GET /api/drivers/{id}

GET /api/drivers/{id}/stats

GET /api/drivers/search?query={query}

4.5 Constructor Endpoints

GET /api/constructors

GET /api/constructors/{id}

GET /api/constructors/{id}/stats

GET /api/constructors/search?query={query}

4.6 Fantasy Team Endpoints

POST /api/fantasy-teams

```
json

Request:
{
    "teamName": "Speed Demons",
    "seasonId": "guid",
    "drivers": [
        {
            "driverId": "guid",
            "isCaptain": true
        },
        {
            "driverId": "guid",
            "isCaptain": false
        }
    ],
    "constructors": [
        {
            "constructorId": "guid"
        }
    ]
}

Response:
{
    "id": "guid",
    "teamName": "Speed Demons",
    "budget": 87.5,
    "totalPoints": 0,
    "transfersRemaining": 5
}
```

GET /api/fantasy-teams/my-teams

GET /api/fantasy-teams/{id}

PUT /api/fantasy-teams/{id}

DELETE /api/fantasy-teams/{id}

GET /api/fantasy-teams/{id}/stats

GET /api/fantasy-teams/{id}/history

4.7 Transfer Endpoints

POST /api/transfers

```
json

Request:
{
    "fantasyTeamId": "guid",
    "transfers": [
        {
            "type": "driver",
            "action": "sell",
            "playerId": "guid"
        },
        {
            "type": "driver",
            "action": "buy",
            "playerId": "guid",
            "isCaptain": false
        }
    ]
}

Response:
{
    "success": true,
    "newBudget": 85.2,
    "transfersRemaining": 4,
    "transferCost": 0
}
```

GET /api/transfers/validate

GET /api/transfers/history/{fantasyTeamId}

4.8 League Endpoints

POST /api/leagues

```
json

Request:
{
    "name": "Friends League",
    "description": "League for close friends",
    "isPublic": false,
    "maxMembers": 20,
    "seasonId": "guid"
}

Response:
{
    "id": "guid",
    "name": "Friends League",
    "inviteCode": "ABC123",
    "memberCount": 1
}
```

GET /api/leagues/my-leagues

GET /api/leagues/{id}

PUT /api/leagues/{id}

DELETE /api/leagues/{id}

POST /api/leagues/join

POST /api/leagues/{id}/leave

GET /api/leagues/{id}/members

GET /api/leagues/{id}/leaderboard

POST /api/leagues/{id}/kick-member

4.9 Race Endpoints

GET /api/races/current

GET /api/races/upcoming

GET /api/races/{id}

GET /api/races/{id}/results

GET /api/races/{id}/live-timing

4.10 Leaderboard Endpoints

GET /api/leaderboards/global

GET /api/leaderboards/league/{leagueId}

GET /api/leaderboards/gameweek/{raceId}

4.11 Statistics Endpoints

GET /api/stats/overview

GET /api/stats/driver/{driverId}

GET /api/stats/constructor/{constructorId}

GET /api/stats/team/{fantasyTeamId}

---

5. USER STORIES

5.1 Epic: Användarregistrering och Autentisering

US-001: Registrera nytt konto

Som en ny användare
Vill jag kunna registrera ett konto med email och lösenord
Så att jag kan använda applikationen

Acceptanskriterier:

- Användaren kan ange email, användarnamn, förnamn, efternamn och lösenord
- Email måste vara valid och unik
- Lösenord måste uppfylla säkerhetskrav (minst 8 tecken, stor/liten bokstav, siffra, specialtecken)
- Användaren får en bekräftelse-email
- Användaren loggas in automatiskt efter registrering

US-002: Logga in i systemet

Som en registrerad användare
Vill jag kunna logga in med mina uppgifter
Så att jag kan komma åt min profil och mina fantasy teams

Acceptanskriterier:

- Användaren kan logga in med email/lösenord
- Felmeddelande visas vid felaktiga uppgifter

- JWT token genereras vid lyckad inloggning
- Användaren omdirigeras till dashboard

5.2 Epic: Fantasy Team Management

US-003: Skapa fantasy team

Som en inloggad användare
Vill jag kunna skapa ett fantasy team för aktuell säsong
Så att jag kan börja tävla

Acceptanskriterier:

- Användaren kan välja team-namn
- Användaren kan välja 2 förare inom budget
- Användaren kan välja 1 konstruktör inom budget
- Total budget får ej överskridas (100.0 miljoner)
- En förare måste väljas som kapten
- Team sparas i databasen

US-004: Göra transfers

Som en team-ägare
Vill jag kunna byta ut förare/konstruktörer mellan race
Så att jag kan optimera mitt team

Acceptanskriterier:

- Användaren kan sälja befintliga förare/konstruktörer
- Användaren kan köpa nya inom budget
- Transfers räknas av från tillgängliga transfers
- Kostnad kan tillkomma efter gratis-transfers är slut
- Changes låses innan race-start

US-005: Sätta kapten

Som en team-ägare
Vill jag kunna välja vilken förare som ska vara kapten
Så att jag får dubbel poäng från den föraren

Acceptanskriterier:

- Endast en förare kan vara kapten åt gången
- Kapten-val kan ändras innan team-lock

- Kapten får 2x poäng för race-resultat

5.3 Epic: Liga-system

US-006: Skapa privat liga

Som en användare
Vill jag kunna skapa en privat liga
Så att jag kan tävla mot mina vänner

Acceptanskriterier:

- Användaren kan ange liga-namn och beskrivning
- System genererar unik invite-kod
- Liga-skaparen blir automatiskt administrator
- Max antal medlemmar kan sättas

US-007: Gå med i liga

Som en användare
Vill jag kunna gå med i en liga via invite-kod
Så att jag kan tävla mot andra

Acceptanskriterier:

- Användaren kan ange invite-kod
- System validerar kod och lägger till användaren
- Användaren ser liga i sin lista
- Användaren kan se andra medlemmar

5.4 Epic: Poäng och Resultat

US-008: Automatisk poängberäkning

Som systemet
Vill jag automatiskt beräkna poäng efter varje race
Så att användarna får uppdaterade resultat

Acceptanskriterier:

- Poäng beräknas baserat på race-resultat
- Bonus-poäng för pole position, snabbaste varv etc.
- Kapten får dubbel poäng
- Poäng läggs till användarens totala poäng

- Leaderboards uppdateras

US-009: Se race-resultat

Som en användare
Vill jag kunna se resultat från senaste race
Så att jag förstår hur jag fick mina poäng

Acceptanskriterier:

- Användaren ser sitt teams prestationer
- Poäng-breakdown visas per förare/konstruktör
- Jämförelse med liga-medlemmar
- Historiska resultat kan visas

5.5 Epic: Live Race Tracking

US-010: Live race-uppdateringar

Som en användare
Vill jag kunna följa race live
Så att jag kan se hur mitt team presterar i realtid

Acceptanskriterier:

- Real-time positioner visas
- Poäng uppdateras live
- Push-notifikationer för viktiga events
- Liga-position uppdateras live

5.6 Epic: Statistik och Analys

US-011: Se team-statistik

Som en team-ägare
Vill jag kunna se detaljerad statistik för mitt team
Så att jag kan förbättra mina val

Acceptanskriterier:

- Total poäng över säsongen
- Poäng per race
- Bästa/sämsta race
- Förare-prestationer

- Transfer-historik

US-012: Jämföra med andra

Som en användare
Vill jag kunna jämföra mitt team med andra
Så att jag kan se var jag står

Acceptanskriterier:

- Liga-leaderboards
- Global leaderboard
- Ranking-förändringar
- Poäng-gap till topp/nästa position

---

6. USE CASES

6.1 UC-001: Registrera och skapa första team

Primär aktör: Ny användare
Syfte: Användaren ska kunna registrera sig och skapa sitt första fantasy team
Förkrav: Användaren har tillgång till applikationen

Huvudflöde:

1. Användaren navigerar till registreringssidan
2. Användaren fyller i registreringsformulär (email, användarnamn, namn, lösenord)
3. Systemet validerar uppgifterna
4. Systemet skapar användarkonto och skickar bekräftelse-email
5. Användaren loggas in automatiskt
6. Systemet visar team-skapande guide
7. Användaren väljer team-namn
8. Användaren väljer 2 förare inom budget
9. Användaren väljer 1 konstruktör inom budget
10. Användaren sätter en förare som kapten
11. Systemet validerar team-sammansättning och budget
12. Systemet sparar fantasy team
13. Användaren omdirigeras till dashboard

**Alternativa fl