SOCCER SCANNER

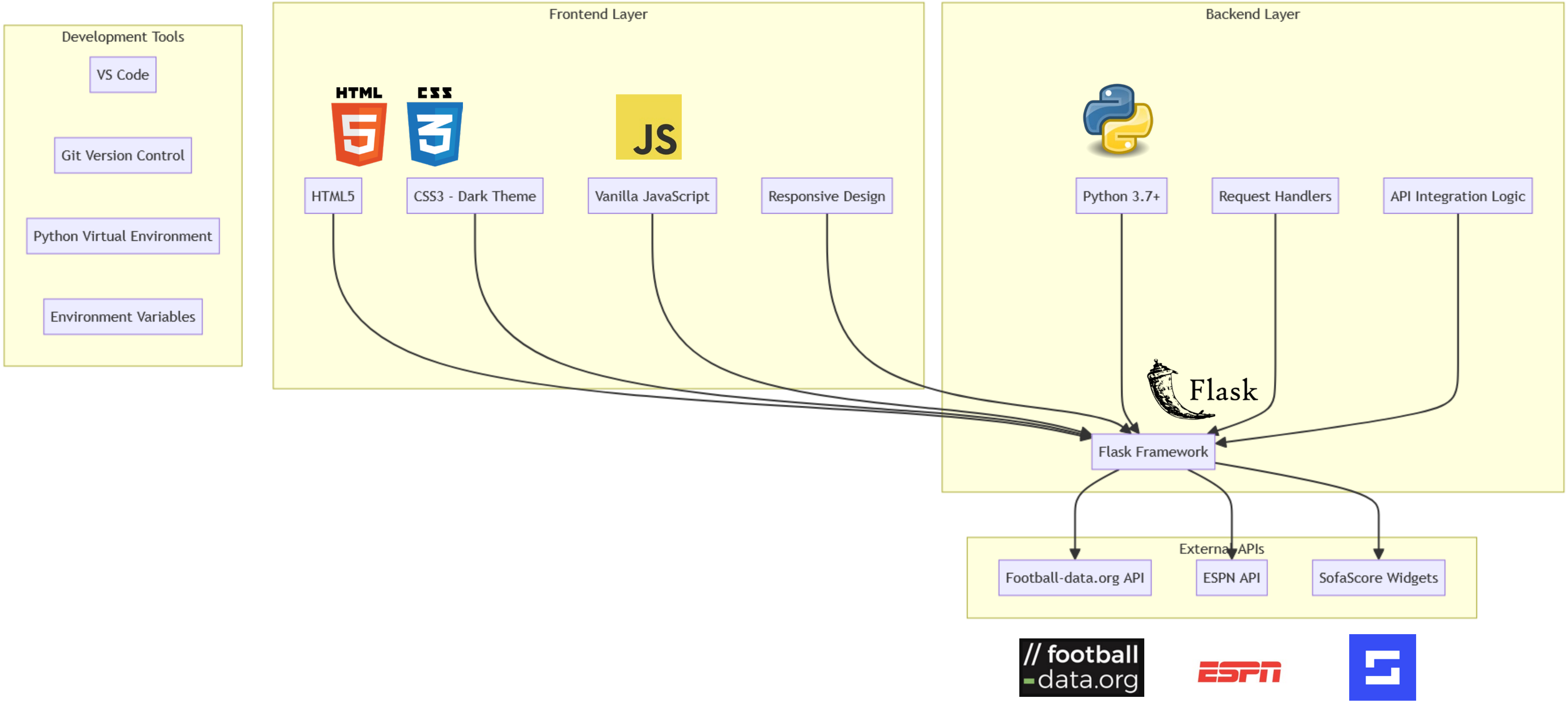
System Diagrams

Author: Leif Heaney

22 August 2025

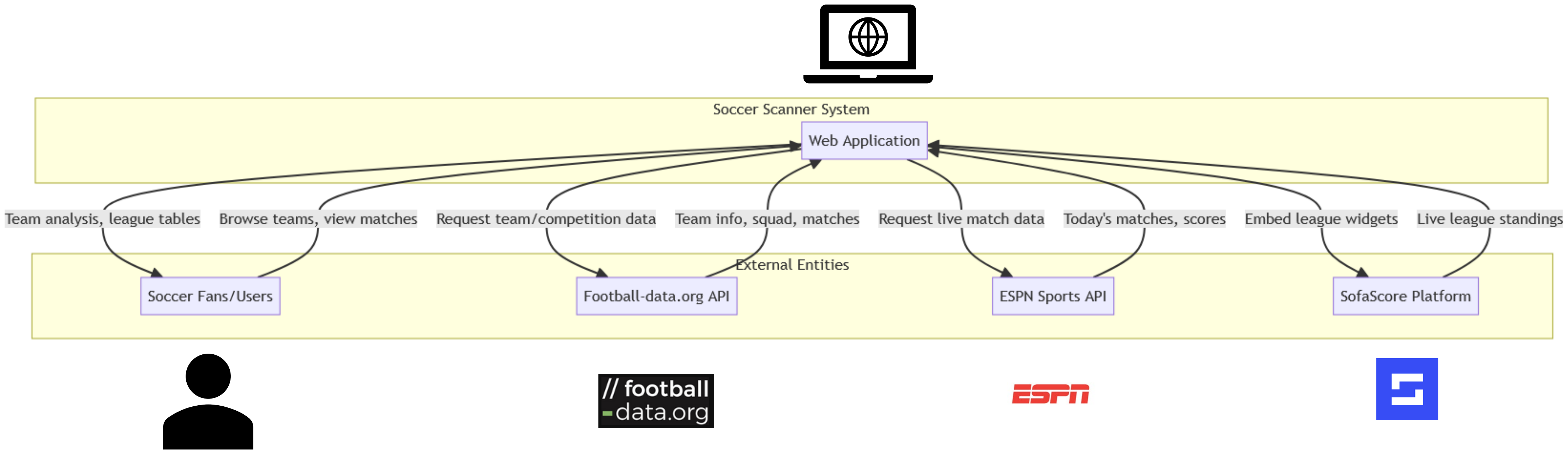
Tech Stack Diagram

Overview of the technology layers and tech stack, including the frontend, backend, external APIs, and development tooling. This diagram depicts how they interact at runtime and clarifies dependencies between client assets, Flask, and third-party services.



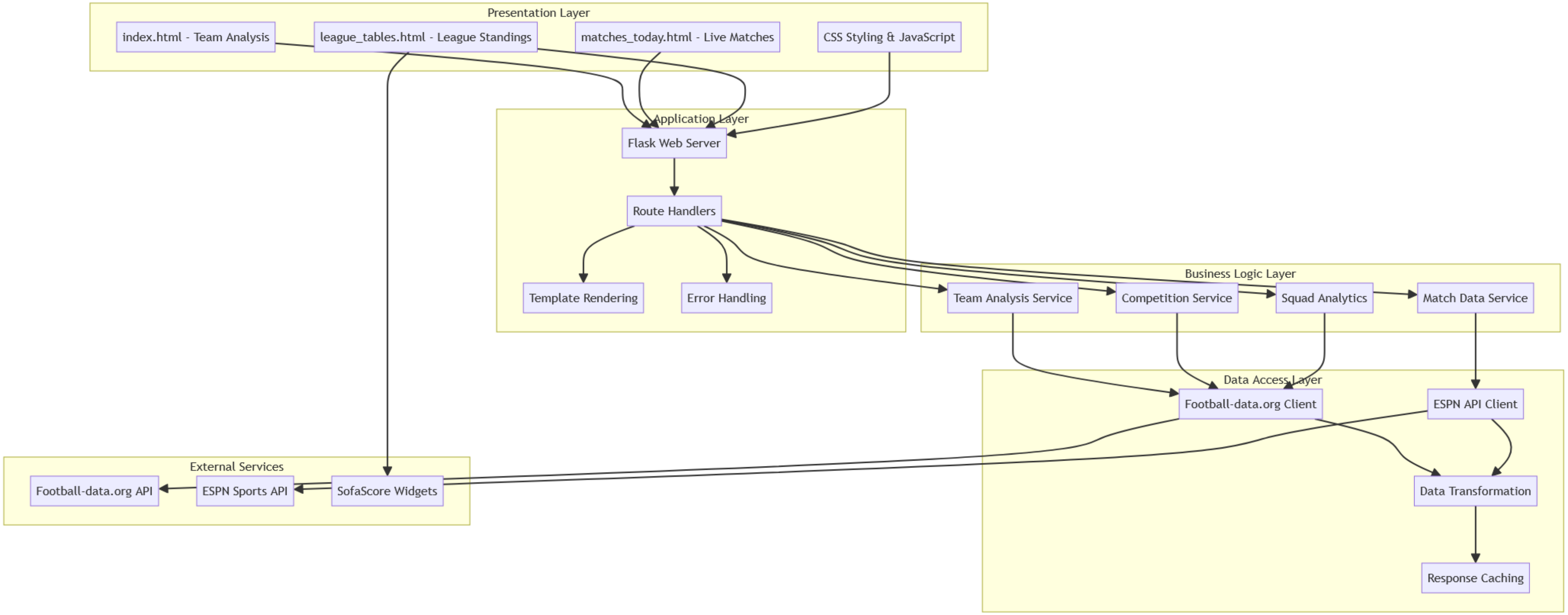
Context Diagram

Defines the system boundaries and depicts how users and external platforms (football-data.org, ESPN, SofaScore) exchange information with the Soccer Scanner web app.



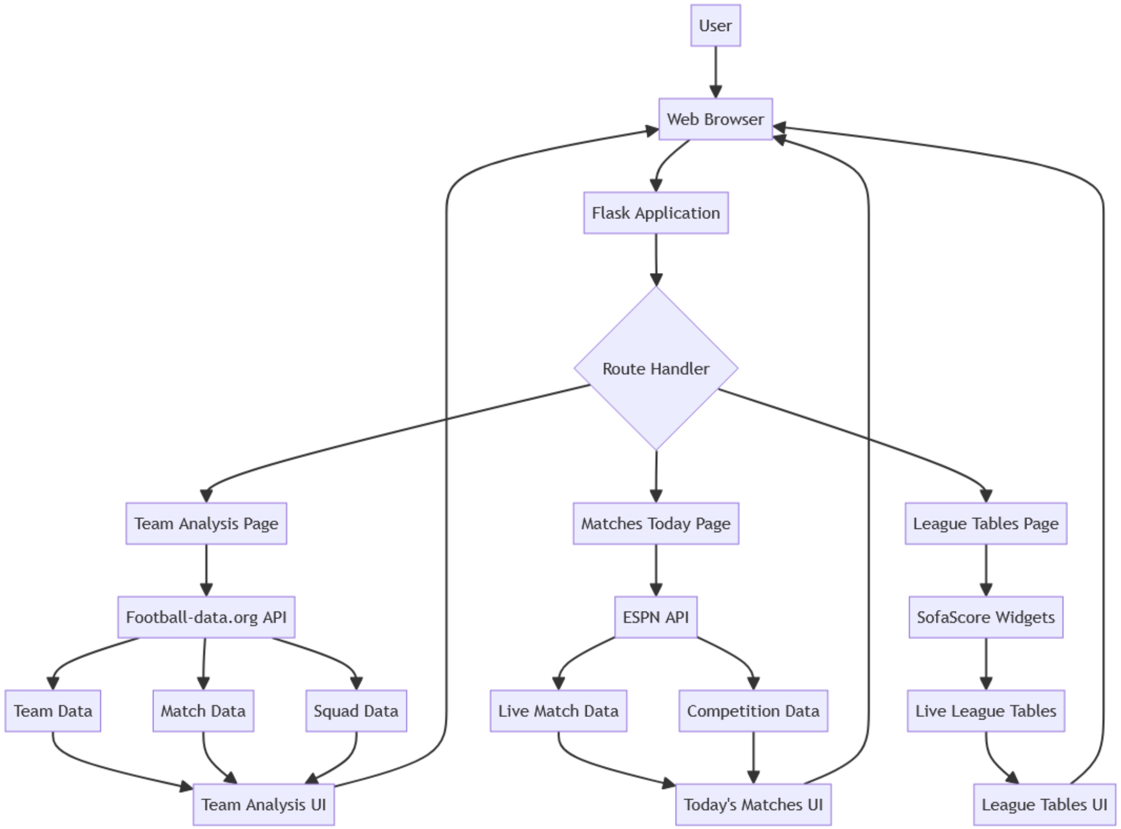
System Architecture Diagram

Highlights the layered architecture of the application: Presentation, Application, Business Logic, Data Access. Integrations with external service are included as well. This diagram emphasizes separation of concerns/handlings, data transformation, and response caching.



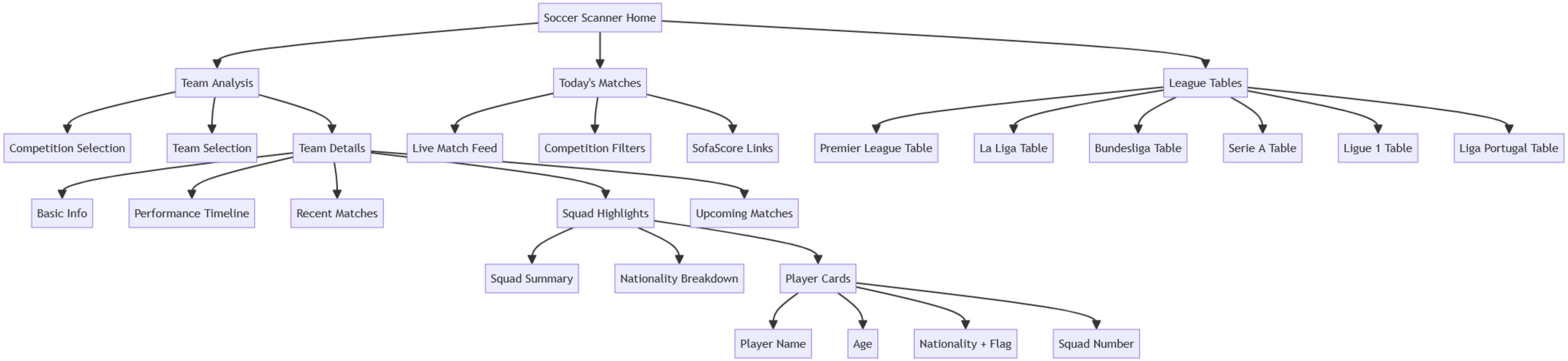
Data Flow Diagram

End-to-end data movement from the user's browser through Flask routes to external APIs, returning processed views for (chosen) Team Analysis, Today's Matches, and League Tables. This diagram depicts how API responses are handled and transformed into UI components.



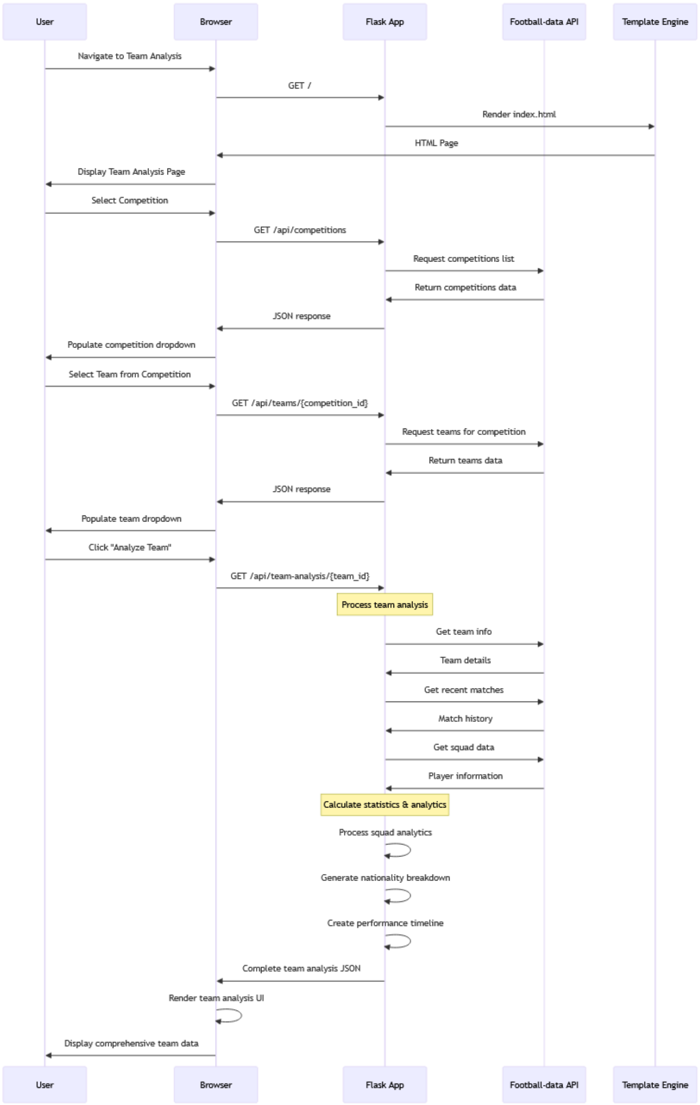
Information Architecture Diagram

Maps the site’s navigation hierarchy from the “Home” to different feature areas (pages) and deep-link content like squad highlights and player cards. Aligns page structure with user tasks.



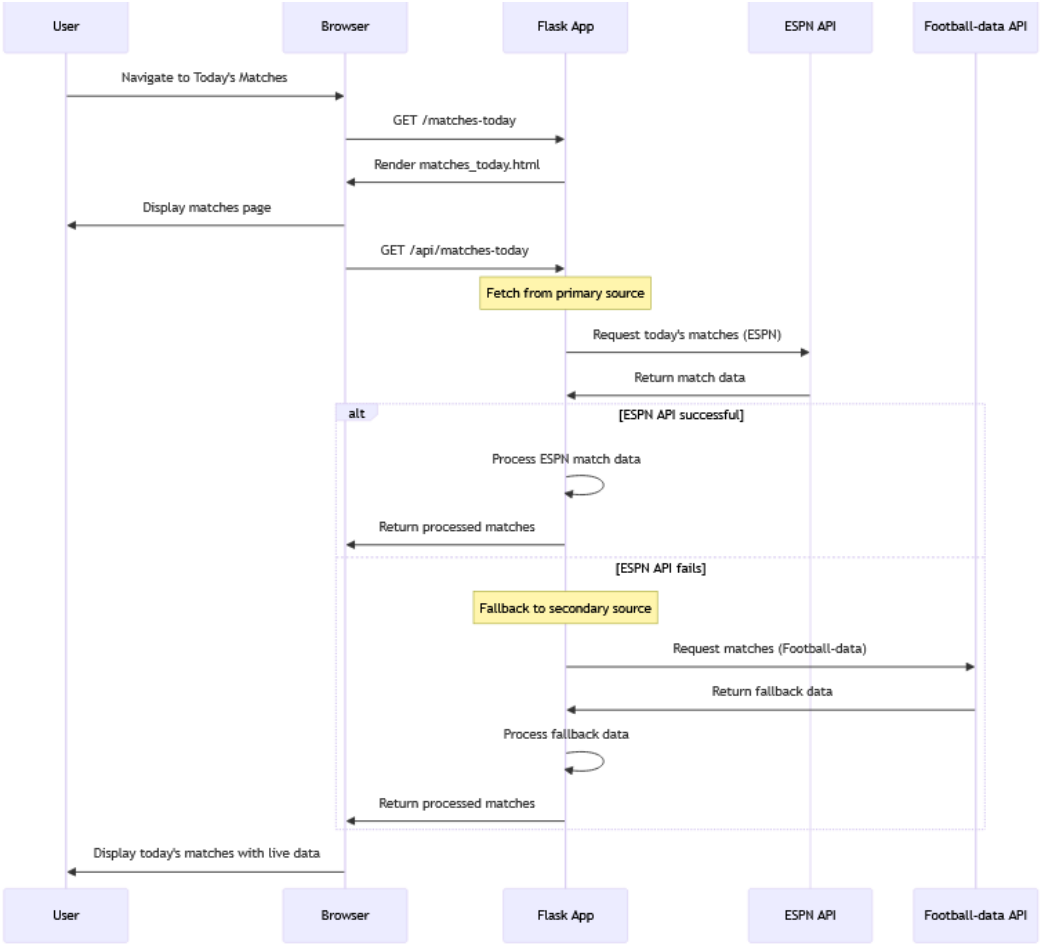
UML Sequence Diagram
Team Analysis Flow

Step-by-step flow from page load through competition/team selection to generating analytics and rendering results. Captures API requests, server-side processing, and UI updates.



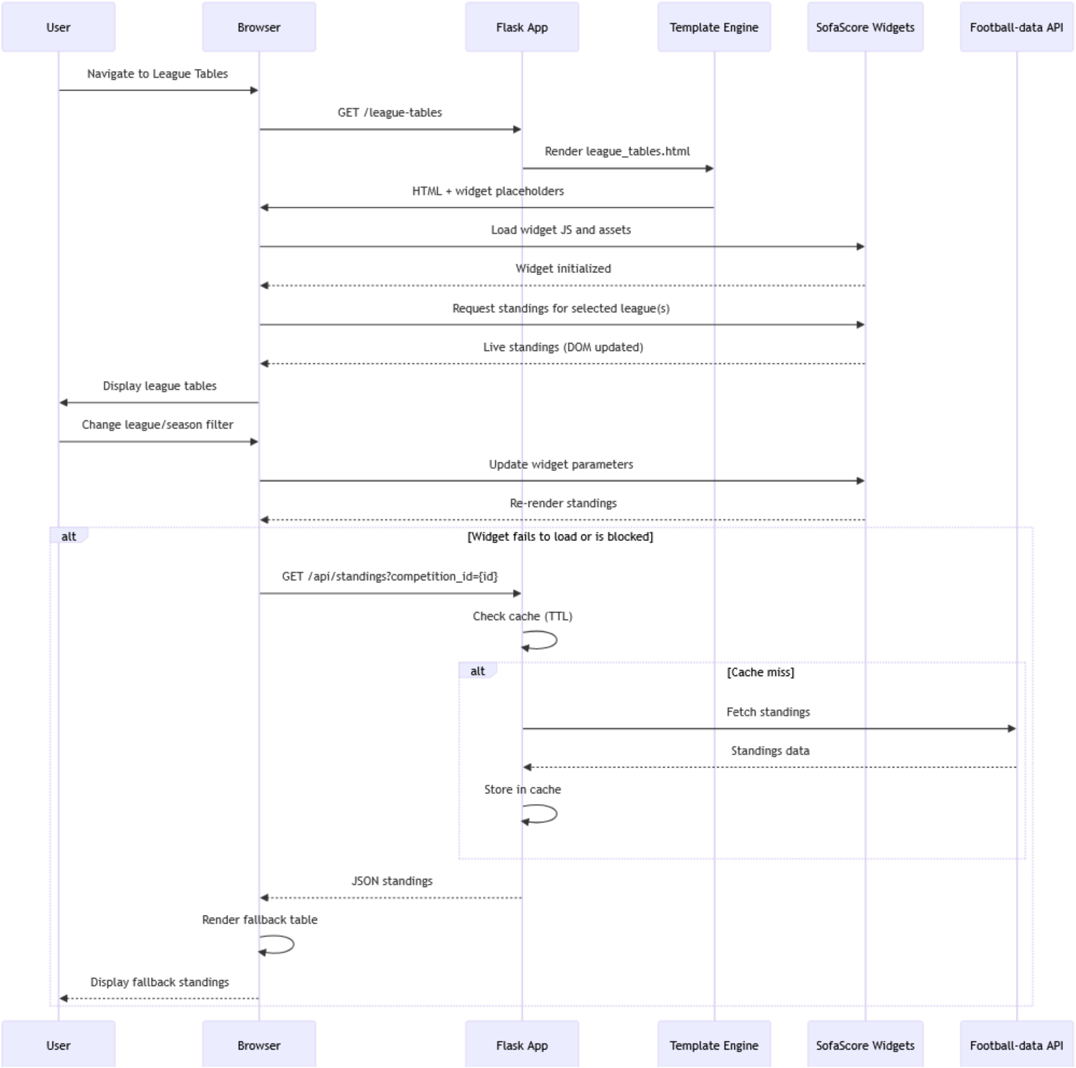
UML Sequence Diagram
Today's Matches Flow

Illustrates fetching live matches with ESPN as the primary source and Football-data.org as fallback. Includes error-handling paths and return of processed data to the UI.



UML Sequence Diagram
Today's Matches Flow

Illustrates the process of fetching live matches with ESPN as the primary source and Football-data.org as fallback. Includes error-handling paths and return of processed data to the UI.



User
Browser
Flask App
Template Engine
SofaScore Widgets
Football-data API
Navigate to League Tables
GET /league-tables
Render league_tables.html
HTML + widget placeholders
Load widget JS and assets
Widget initialized
Request standings for selected league(s)
Live standings (DOM updated)
Display league tables
Change league/season filter
Update widget parameters
Re-render standings
alt
[Widget fails to load or is blocked]
GET /api/standings?competition_id={id}
Check cache (TTL)
alt
[Cache miss]
Fetch standings
Standings data
Store in cache
JSON standings
Render fallback table
Display fallback standings
User
Browser
Flask App
Template Engine
SofaScore Widgets
Football-data API