**Process, Design, and Data Flow Guide Rohan Kansari**

**Business Overview**

**Live Sports Score Tracker** is a web-based Angular application that fetches live and past cricket match data using the CricAPI. The application is designed to display match lists, real-time scores (simulated via polling), and match details using Angular's reactive programming capabilities (RxJS).

**Why?**

To provide users with real-time (polling-based) updates on cricket matches without the complexity of WebSockets. This supports casual viewers or cricket enthusiasts in tracking scores, team details, and match outcomes conveniently.

**Who?**

* End Users: Cricket fans and enthusiasts
* Admin: Application developers and maintainers
* API Provider: CricAPI service

**Technology**

* Angular (Framework)
* RxJS (Polling, state handling)
* TypeScript (Language)
* CricAPI (Data provider)
* Figma (Design tool)

**Process Flow**

**Module 1: Live Score Dashboard**

**Brief:** Displays a list of current and recent matches fetched from CricAPI. Auto-refreshes data every 10 seconds.

**Features**

**Feature 1: Fetch Matches**

* Uses interval(10000) + switchMap() to poll live data.
* Filters matches based on status (live, completed).

**Task 1:** Build polling mechanism.

* **Acceptance Criteria:**
  + Data should refresh every 10 seconds.
  + No overlapping API calls.
* **Test Cases:**
  + Check if old observable is unsubscribed.
  + Data updates correctly on screen.
* **Technical Details:**
  + Use switchMap() with interval()
  + Service call from MatchService
  + Show loader during API calls
* **Compliances:**
  + Ensure API key security

**Task 2:** Filtering by match status.

* **Acceptance Criteria:**
  + Dropdown must filter matches by status.

**Module 2: Match Details**

**Brief:** Upon selecting a match, user sees detailed stats (teams, score, time).

**Feature 1: Detail Fetching**

* Uses mergeMap() to trigger detail request on match click.
* **Task 1:** Display team and score detail
* **Acceptance Criteria:**
  + Match info should appear without full reload
* **Technical Details:**
  + Use route parameter for match ID
  + Use mergeMap() for sequential calls

**Third Party Integration**

**CricAPI**:

* Used to fetch live and past match data
* Endpoint: https://api.cricapi.com/v1/matches
* Requires apikey via query string

**Security Matrix**

* API Key stored securely using environment.ts
* Use interceptor to attach key to every HTTP request
* Validate response integrity

**Separate excel to be maintained for roles & privileges**

**Notifications Matrix**

* Planned (future scope):
  + Browser alerts for favorite team matches

**Separate excel to be maintained for notification triggers**

**Engines**

**Polling Engine:** Uses interval + switchMap to simulate live updates **Subscription Manager:** Uses takeUntil() to clean up polling

**Data Migration**

* Not required (fully dynamic data from external API)

**Data Design**

**Database Structure**

No local DB. Uses API directly.

**Table Descriptions / Data Dictionary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Example Value** |
| id | Unique Match ID | string | "1234" |
| name | Match title | string | "India vs Aus" |
| status | Match status | string | "Live" |
| score | Team scores | object | {"ind": 250/3} |
| date | Date of match | string | "2024-05-20" |

**ER Diagram not required as we use API only.**

**Project Estimation**

* **Planning:** 1 day
* **Development & Testing:** 1 day

**Milestones:**

* Day 1: UI in Figma + Documentations
* Day 2: Match list module + Testing

**Glossary**

|  |  |
| --- | --- |
| Term | Description |
| RxJS | Reactive Extensions for JavaScript |
| switchMap() | Cancels previous observable, uses latest |
| mergeMap() | Combines observables sequentially |
| interval() | Emits sequential numbers in given intervals |
| takeUntil() | Unsubscribes when notifier emits |
| Interceptor | Middleware to modify HTTP requests |