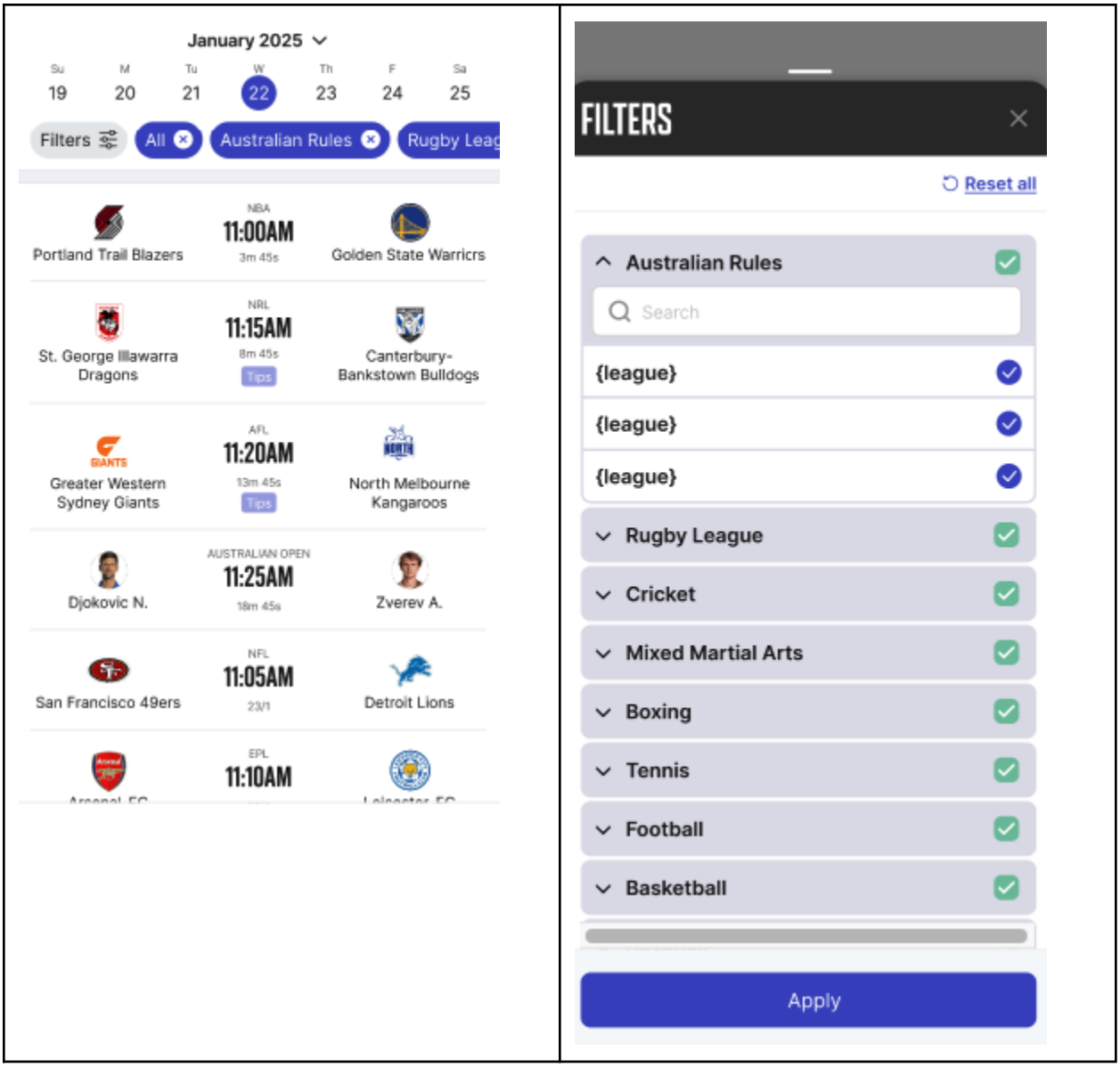
Hello,

As part of our interview process, we would like you to complete the following **mobile frontend assignment** using **React Native**.

This task reflects a real-world scenario and focuses on **UI accuracy, responsiveness, API integration, and clean mobile architecture**.



🎯 Objective

Build a **React Native Match List screen** that closely follows the provided design and integrates with a live API.
The screen must be **pixel-accurate**, **responsive across device sizes**, and provide a **smooth, fast user experience**.

🧩 Scope of Work

You are expected to implement:

1. A **Match List screen** using React Native
2. **Pixel-perfect UI** based on the provided design reference
3. **Infinite scrolling** for loading additional matches
4. Fully working **filters** (tournaments)
5. A **smooth countdown timer** for match start times
6. Optimized animations and interactions
7. Proper handling of loading, empty, and error states

🎨 Design Requirements

- A design reference (from XD) provided as **two images**
- Your implementation should:
    - Match spacing, alignment, typography, and layout as closely as possible
    - Maintain visual consistency across:
        - Small devices

        - Standard devices
        - Large devices
- Lists must scroll smoothly with no layout jumps
- UI should feel **fast, stable, and polished**

The goal is to closely match the provided design while keeping the implementation practical and maintainable.

🔄 Infinite Scroll Behavior

- The match list must support **infinite scrolling**
- Data should load incrementally using:
    - `limit`
    - `offset`
- When the user reaches the bottom:
    - Fetch the next page of matches
    - Append results smoothly
- Ensure:
    - No duplicate items
    - No unnecessary re-renders
    - Loading indicators appear naturally

⏱ Countdown Timer Requirements

Each match item should display a **live countdown timer** until the match start time.

- Countdown should:
    - Update smoothly
    - Be accurate across timezones
    - Stop or change state once the match starts
- Timer updates should not cause performance issues or excessive re-renders

---

🎞️ Animations & Performance

- Interactions should feel **snappy and responsive**
- Avoid heavy or blocking animations
- Ensure:
    - Smooth list scrolling
    - Stable layouts during data updates
    - Efficient rendering for large lists
- Use native performance best practices where appropriate

🔗 Match List API

Media Base URL (use for images)

`GET https://media.smartb.com.au/`

Endpoint

```
GET https://au.testing.smartb.com.au/api/sports/matchList
```

⚙️ Supported Sports

The API returns matches only for the following sports:

- Cricket (sportId: 4)
- Soccer (sportId: 8)
- Australian Rules (sportId: 9)
- Basketball (sportId: 10)
- Rugby League (sportId: 12)

No additional sports need to be handled.

🔍 Query Parameters

Parameter	Required	Description
`timezone`	Yes	Example: `Australia/Sydney`
`status`	No	`all`
`todate`	No	Format: `YYYY-MM-DD`
`tournament_ids`	No	Comma-separated tournament IDs

`limit`	No	Page size for infinite scroll
`offset`	No	Offset for pagination

🔗 Sports & Tournaments (For Filters)

Endpoint

```
GET https://au.testing.smartb.com.au/api/sports/AllSportsAndLeagues
```

Query Parameters

Parameter	Required	Example	Description
`search`	No	`4[india test],8[premier]`	Sport-based tournament search
`limit`	No	`10`	Tournaments per sport
`offset`	No	`0`	Pagination per sport

`search` Parameter Format

The `search` parameter allows filtering tournaments **per sport**.

```
sportId[searchText]
```

Example:

```
search=4[india test],8[premier]
```

This returns:

- Cricket tournaments matching “india test”
- Soccer tournaments matching “premier”

Example API Call

```
https://au.testing.smartb.com.au/api/sports/AllSportsAndLeagues
?search=4[india test],8[premier]
&limit=10
&offset=0
```

📦 Response Structure (Simplified)

```
[
  {
    "id": 4,
    "sportName": "Cricket",
    "tournaments": [
```

```
      {
        "id": 12,
        "name": "India Test Series"
      }
    ]
  }
]
```

🧠 Filter Flow (Expected Usage)

1. Load tournaments using **AllSportsAndLeagues API**
2. Display tournaments in the bottom-sheet filter UI
3. Allow multi-select tournaments
4. On apply:
    - Collect selected tournament IDs

Pass them to Match List API as:

```
tournament_ids=12,18,25
```

5. Refresh match list and reset pagination

🧠 Filters Behavior

- Filters should open in a bottom sheet or modal

- Users can:
    - Select multiple tournaments
    - Apply or reset filters
- On applying filters:
    - Selected tournament IDs must be passed via `tournament_ids`
    - Match list should refresh and reset pagination
- Filter interactions should feel instant and smooth

🧪 Example API Call

```
https://au.testing.smartb.com.au/api/sports/matchList

?timezone=Australia/Sydney

&status=all

&tournament_ids=12,18

&limit=20

&offset=0
```

📦 Data Handling Expectations

- Display a loading state while fetching data
- Show a clear empty state when no matches are available
- Handle API errors gracefully without crashing

- Ensure smooth UI updates when:
    - Filters change
    - New pages load
    - Timers update

🧰 Technical Expectations

- React Native (TypeScript preferred)
- Functional components and hooks
- Clean, modular component structure
- Optimized list rendering (`FlatList` / `SectionList/FlashList`)
- Well-managed state and side effects
- Production-ready code quality

📤 Submission

Please provide:

- A GitHub repository link **OR**
- A zip file containing the project
- An APK File

Include a short **README** describing:

- How to run the app
- Any assumptions or decisions made

- Any trade-offs worth noting

⏱ Time Expectation

Please keep the scope reasonable.
Focus on **correctness, UI accuracy, smooth UX, and clean implementation** rather than over-engineering.

🙌 Final Notes

This assignment reflects real production requirements.
If anything is unclear, make a reasonable assumption and document it briefly in the README.

We look forward to reviewing your work.

Best regards,

Digiground