BEARS-1273 : Update the competition filter to the same component as seen on the Daily List

Sarath Ramachandran

Table of Contents

Apprenticeship Activity Report

Introduction

In this project, I was assigned the BEARS-1273 ticket as part of our Front-End Platform team's ongoing effort to improve usability and maintainability across our sports betting platform. The business goal was to refactor and standardise the competition filter dropdown on the **Outrights** page by replacing a legacy component with the reusable **CompetitionFilter** already used on the Daily List view. This change was part of a wider initiative to improve user experience consistency, accessibility, and developer efficiency.

The Outrights page allows users to view and filter outright betting events across different sports and competitions. However, the filter dropdown had outdated logic and inconsistent design compared to other parts of the platform. My task was to integrate the new shared CompetitionFilter component, ensure its functionality remained intact, and resolve any visual or logic mismatches during the transition.

I collaborated with product owners, front-end developers, and QA testers in an agile team setting. The work was tracked using Jira (under the BEARS-1273 ticket), with regular updates shared in stand-ups and code reviews. Development was version-controlled via GitLab and tested via both manual UI testing and automated unit test coverage.

This report outlines the full lifecycle of this feature—from understanding the requirements, planning and design, implementation and debugging, to testing and deployment—following standard SDLC practices and reflecting on the impact and learning outcomes achieved through this work.

Requirement Gathering

The requirement for this task was documented in Jira under the ticket BEARS-1273. The business objective was to replace the existing competition dropdown filter on the Outrights page with the shared CompetitionFilter component already used on the Daily List page. This was part of a platform-wide effort to consolidate UI components for better consistency, accessibility, and maintainability.

The key requirements were :

- Integrate the reusable CompetitionFilter component in place of the legacy dropdown.
- Ensure correct filtering of events based on selected competition.
- Ensure "All Competitions" selection shows all relevant events.
- Maintain parity in styling, scroll behaviour, and dropdown interaction.
- Preserve existing query param syncing for regionId, competitionId, and marketTypeGroupId.

Tools & Sources Used :

- Jira for tracking the BEARS-1273 ticket and subtasks.
- GitLab Merge Request and comments for peer review and feedback.
- DailyList implementation of CompetitionFilter as reference for integration.
- Browser dev tools and Redux store inspection to debug event rendering and dropdown state.
- React DevTools, component test files, and shared helper libraries for understanding logic.

Stakeholders Involved :

- SETL – clarified intended shared usage of CompetitionFilter.
- UX Designers – confirmed alignment with Daily List design.
- QA Engineers – highlighted edge cases (e.g., American Football's NCAA filter issue).
- Product Owner – defined scope and confirmed visual/functional parity expectations.

We worked in a Kanban-based Agile workflow. The ticket moved through stages :
To Do → In Progress → Code Review → QA → Done, with team input gathered during stand-ups, pair debugging sessions, and MR reviews. Additional feedback was raised during feature testing to address a regression where "All Competitions" did not restore events after applying and clearing individual filters.

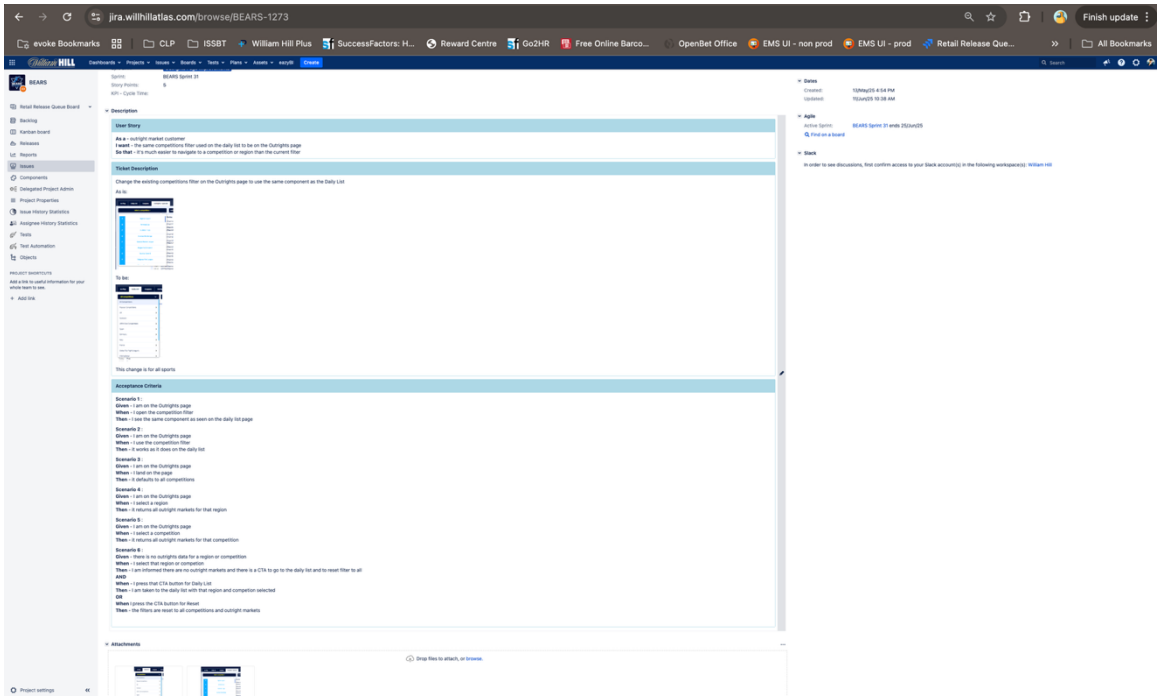


Figure 1 : Story ticket details

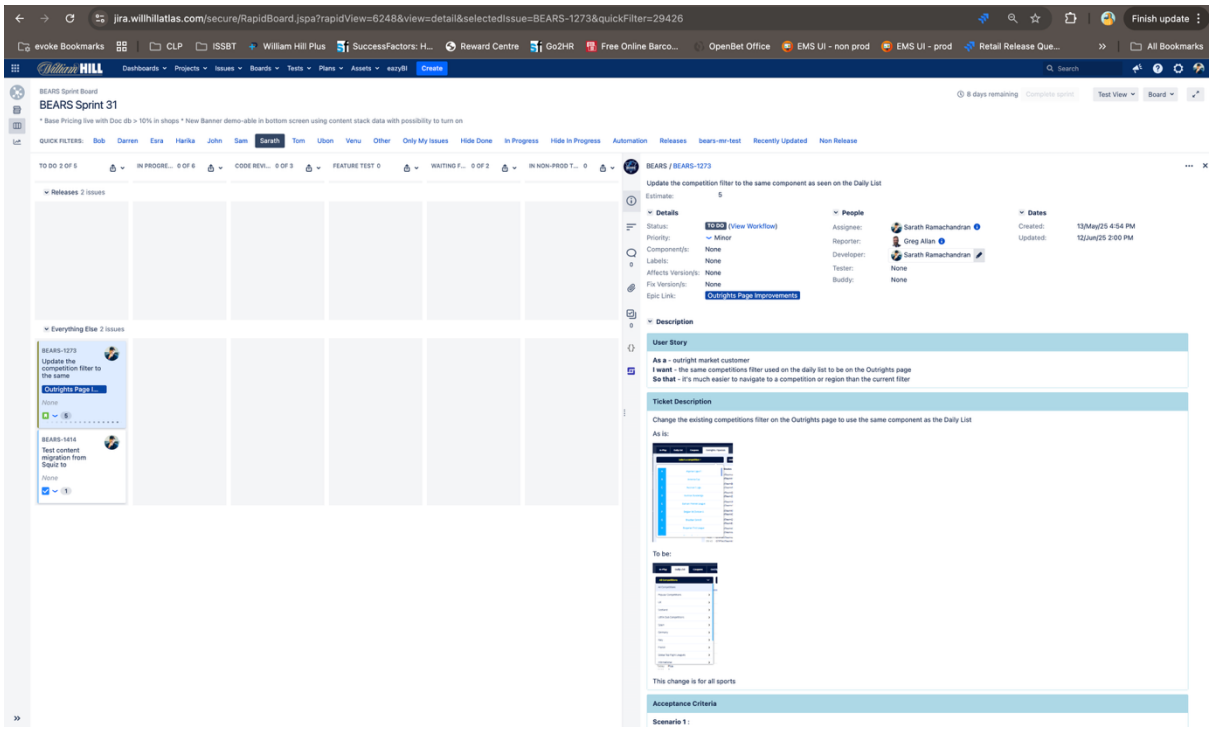


Figure 2 : Sprint board

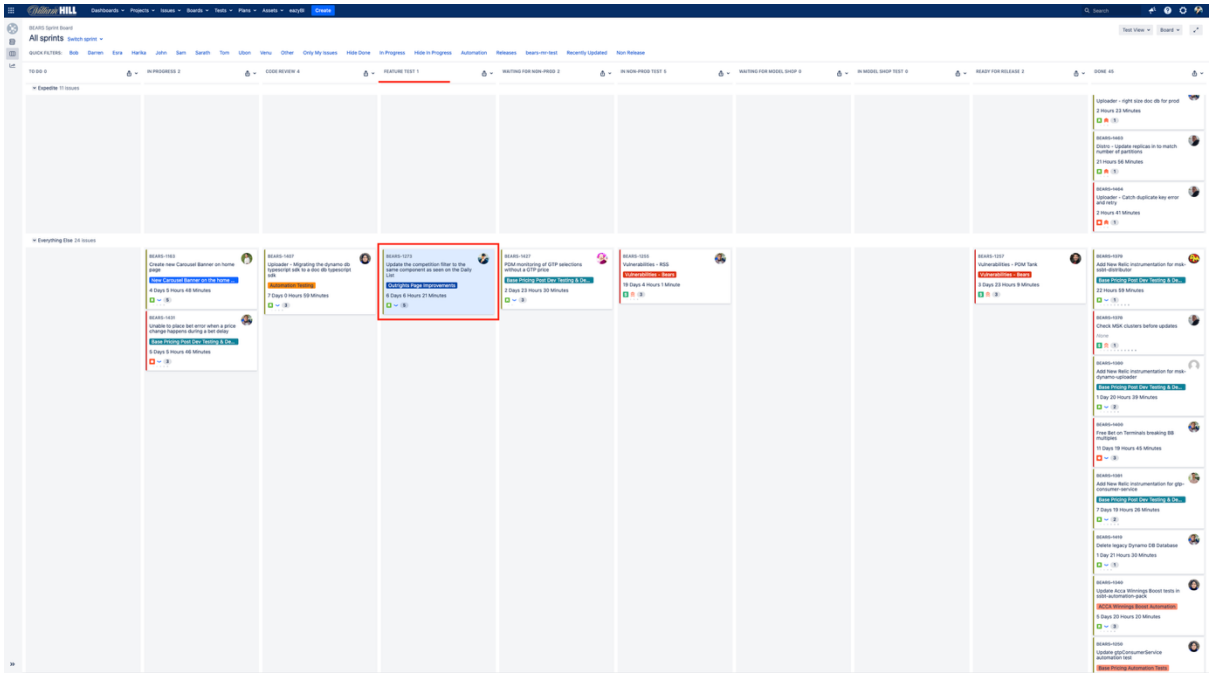


Figure 3 : Sprint board (Ticket in Feature test)

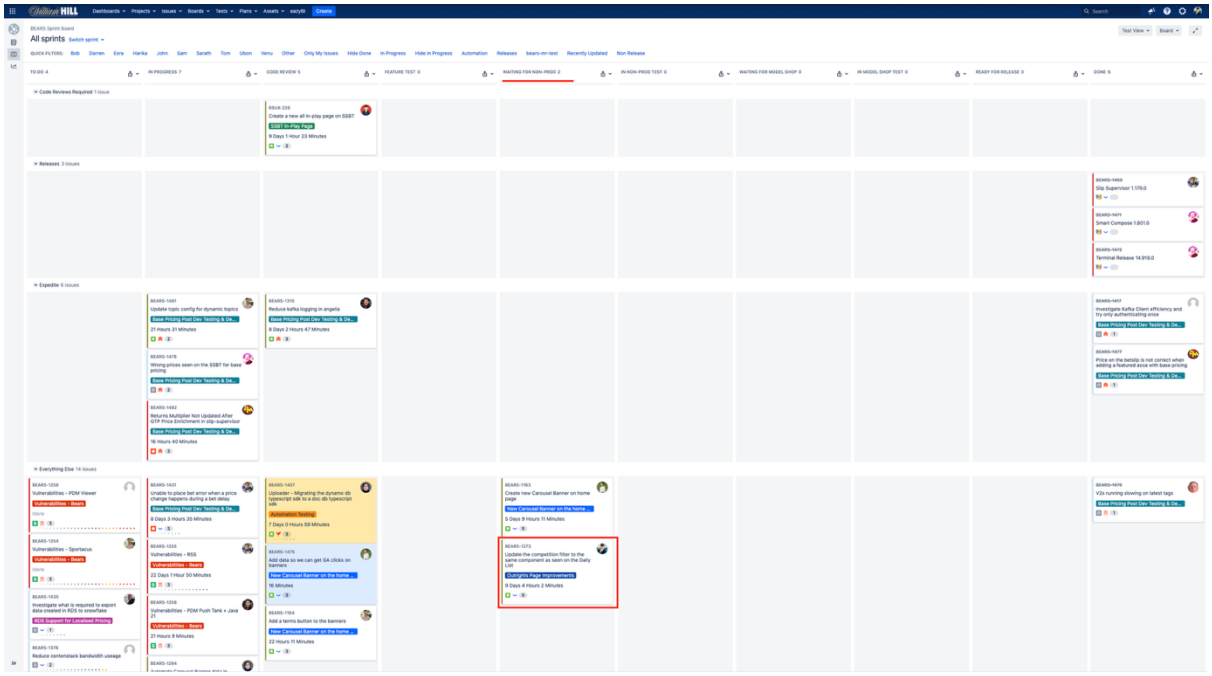


Figure 4 : Ticket in Waiting for non prod stage

Analysis

To understand how to integrate the shared CompetitionFilter component into the Outrights page without breaking functionality, I began by reviewing both the current Outrights implementation and the Daily List page, where the filter was already working correctly.

Steps involved:

- Reviewed the CompetitionsContainer.js logic to understand how event filtering and query params were handled.
- Traced the data flow from URL search params → state → render logic, especially around how marketTypeGroupId, regionId, and competitionId were applied.
- Analysed the structure of events, ancestors, and existing filtering logic inside the component.
- Debugged scenarios were switching back to “All Competitions” would result in blank events, despite the data being available.
- Compared getCompetitionFilterUrl logic on DailyList and Outrights to identify differences in how competition filters were reset.

Key Observations & Decisions:

- The competitionId was being passed as "null" in some URLs, breaking filtering logic.
- The Outrights page didn’t consistently update the URL on “All Competitions” selection.
- Decided to unify the filter URL logic using the pattern from DailyList for consistency.

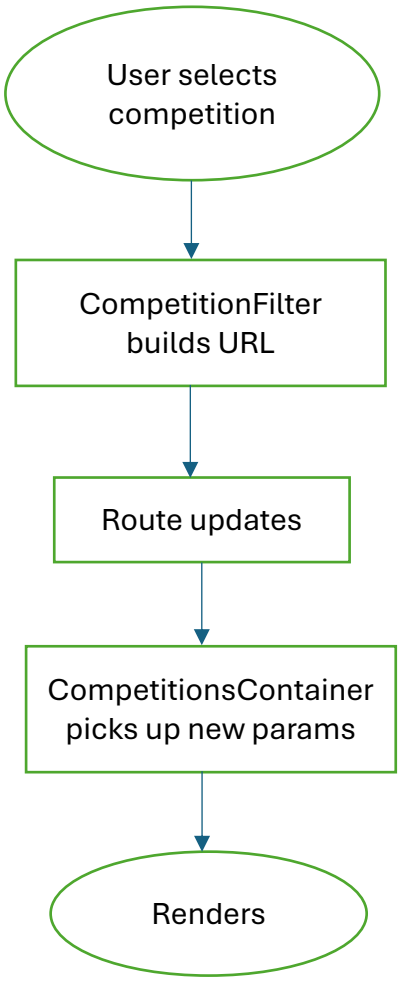
- Avoid redundant filtering by removing fragile checks like market.ancestors.marketTypeGroupId.

Design

The design aimed to make the Outrights competition filter behave identically to Daily List, reusing the shared CompetitionFilter dropdown component while ensuring correct event rendering on all filter changes.

Steps Planned:

- Replace the existing dropdown on Outrights with <CompetitionFilter /> from MatchList.
- Define a custom getCompetitionFilterUrl function to build URLs cleanly, omitting undefined or "null" params.
- Safeguard event filtering in CompetitionsContainer.js by checking if ancestor values exist before filtering.
- Adjust navigation to update the route using createSearchParams() only when needed.
- Ensure correct publishing of params to the store and listeners via PubSub.



Key Design Improvements:

- Centralized URL-building logic with fallback for “All Competitions”.
- Preserved scroll context, loading state, and param syncing.
- Adopted the reusable component without affecting existing Outrights structure.

Implementation

The goal of this phase was to integrate a reusable, accessible, and consistent dropdown filter component (CompetitionFilter) into the Outrights page, replacing the older CompetitionsMenu. This aligns with broader goals of improving maintainability, UI/UX consistency, and code reuse across the application.

To achieve this, the following structured implementation steps were carried out:

1. Wrapping Routes with CompetitionGroupsProvider

In order to allow CompetitionFilter to access grouped competitions via React context, I updated CompetitionsRoute.js to wrap the route with <CompetitionGroupsProvider>. This provided a shared competitionGroups context accessible deep within the component tree.

```
<CompetitionGroupsProvider>
  <CompetitionsContainer ...props />
</CompetitionGroupsProvider>
```

This ensured we could remove prop-drilling and rely on a clean provider-consumer relationship using the useCompetitionGroups() hook inside the CompetitionFilter.

2. Passing Redux-Fetched Group Data into the Container

To ensure the dropdown component reflects correct and current competition groups, I added logic in the route container to fetch competitionGroups from the Redux store via a selector. This was passed into the CompetitionGroupsProvider as a value.

This made sure any changes in global state were automatically reflected across the app without hardcoded or static group data.

3. Replacing UI Dropdown in CompetitionsMenu with CompetitionFilter

The old CompetitionsMenu was a custom dropdown with inconsistent styling and limited reusability. It was replaced with the already shared CompetitionFilter component, which is also used on the DailyList page.

We ensured:

- Correct props (sportId, initialCompetitionId, and getCompetitionFilterUrl) were passed
- The filter reflects and controls the URL
- Re-selection of “All Competitions” resets filters

This made the UI/UX consistent across pages and aligned with design standards.

4. CSS & Visual Alignment

The layout was adjusted to visually align with DailyList’s design. This included:

- Padding adjustments in .scss files
- Refactoring component spacing within the Competitions/index.js view
- Ensuring the new dropdown fits into the flex layout and behaves correctly on scroll

5. Accessibility & UX Improvements

- CompetitionFilter already had aria-* labels and keyboard navigation, ensuring accessibility compliance.
- Consistent dropdown placement and design across pages reinforced predictability and usability.
- “All Competitions” selection now triggers a fallback state that shows events or informative messages if empty, instead of rendering a blank view.

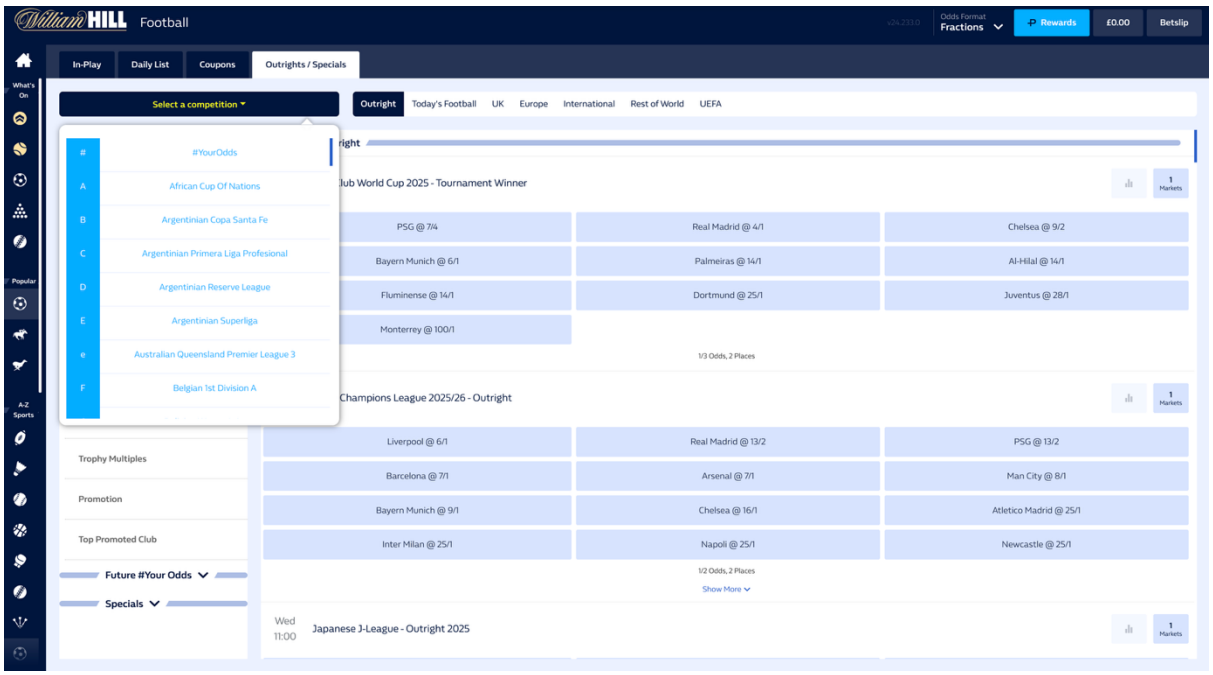


Figure 5 : Old outrights page filter

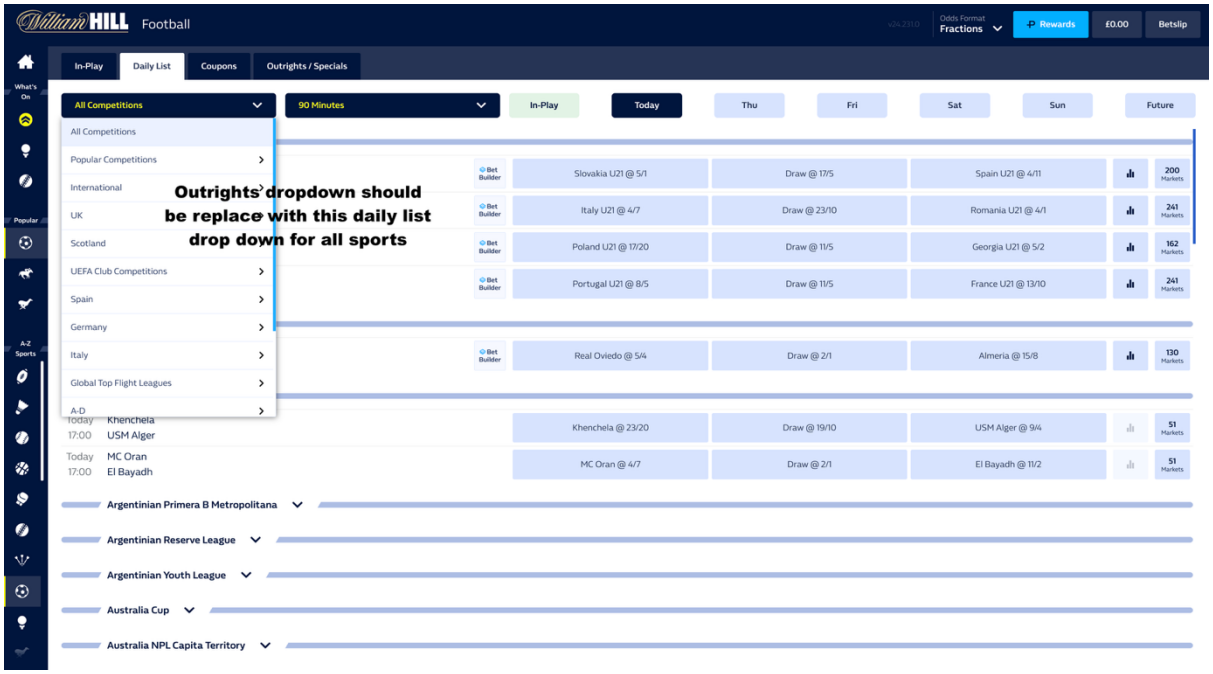


Figure 6 : Dailylist page

Development

All work for this project was carried out under the feature branch feature/BEARS-1273, which was continuously rebased against develop to avoid merge conflicts and stay up to date with upstream changes.

Tools and Stack

- React 18
- Redux
- React Router DOM
- Immutable.js
- Jest & React Testing Library
- GitLab (source control + CI/CD)
- IntelliJ IDEA (IDE)

Key Files Modified

File	Purpose
Competitions/index.js	View integration of reusable CompetitionFilter
CompetitionsContainer.js	Updated logic to support dynamic filter changes and context
CompetitionsMenu/index.js	Fully removed and replaced by CompetitionFilter
CompetitionGroupsProvider.js	Provides grouped data context to child components
useCompetitionGroups.js	Hook to consume grouped competition data

File	Purpose
CompetitionsRoute.js	Wrapped route in provider to enable context
index.spec.js	Updated unit tests to handle new context and URL logic

Key Implementations

1. Replaced CompetitionsMenu with CompetitionFilter

   - Passed props: initialCompetitionId, sportId, getCompetitionFilterUrl
   - Ensured consistent UX with DailyList

2. Context Setup

   - Created and wrapped <CompetitionGroupsProvider> in route
   - Fetched grouped competitions via Redux and injected via provider

3. Dynamic URL Handling

   - Refactored getCompetitionFilterUrl to conditionally include competitionId
   - Prevented broken links like ?competitionId=null

```
const getCompetitionFilterUrl = (competitionId) => {
 const queryParams = {
  marketTypeGroupId: params.get('marketTypeGroupId'),
  marketTypeId: params.get('marketTypeId'),
  regionId: params.get('regionId'),
  ...(competitionId && { competitionId }),
 };
 const queryString = createSearchParams(queryParams);
 return `/sports/${params.get('sportId')}/competitions?${queryString}`;
};
```

4. Dropdown Selection Sync

   - Clicking “All Competitions” now clears the filter and resets the page
   - Hooked into routing logic via createSearchParams and React Router

5. State Computation and Filtering in CompetitionsContainer

   - Computed filteredEvents based on competitionId, regionId, and store data
   - Safely handled missing fields using Immutable.Map().get()
   - Maintained compatibility with “All Competitions” fallback behaviour

Testing and Debugging

The BEARS-1273 feature underwent thorough testing at multiple levels:

Unit & Component Testing

- Updated Jest unit tests and snapshots in Competitions/index.spec.js to align with the new filter structure and context usage.
- Fixed broken test cases by ensuring conditional URL generation (competitionId was excluded when not needed).
- Used mocks for CompetitionGroupsContext, PubSub, and useScroll to isolate UI logic.

Automation Testing

- Before merging into develop, the full automation regression suite was executed.
- These tests cover event filtering, dropdown interactions, and route updates — ensuring that existing functionality was not broken.

Manual QA Testing

- Manual testers verified the behaviour in non-production and production environments:
  - Confirmed dropdown accessibility and selection behaviour
  - Checked rendering of events on selecting individual competitions
- Ensured switching back to “All Competitions” correctly repopulates events
- Manual testing included tag validation, visual checks, and cross-device UX testing.

These layers of validation ensured confidence before go-live and helped prevent regressions in the event display or competition filtering.

Challenges Faced

1. **Blank Page on “All Competitions”**

   Initial issues arose when selecting “All Competitions” from a filtered state. Events disappeared due to over-filtering in CompetitionsContainer.

   Fix: We relaxed filtering conditions and ensured correct fallback when competitionId is not present.

2. **Context Setup Complexity**

   Wrapping the right components in CompetitionGroupsProvider was critical. Missing provider led to null data and rendering failures.

3. **Snapshot Test Failures**

   Caused by href="/sports/undefined/competitions?competitionId=null"—fixed by omitting null competitionId entirely in URLs.

The code was committed in stages and reviewed via GitLab Merge Requests. DangerJS and lint warnings were addressed, and local regression testing was completed before final approval.

Code changes and MR snippets

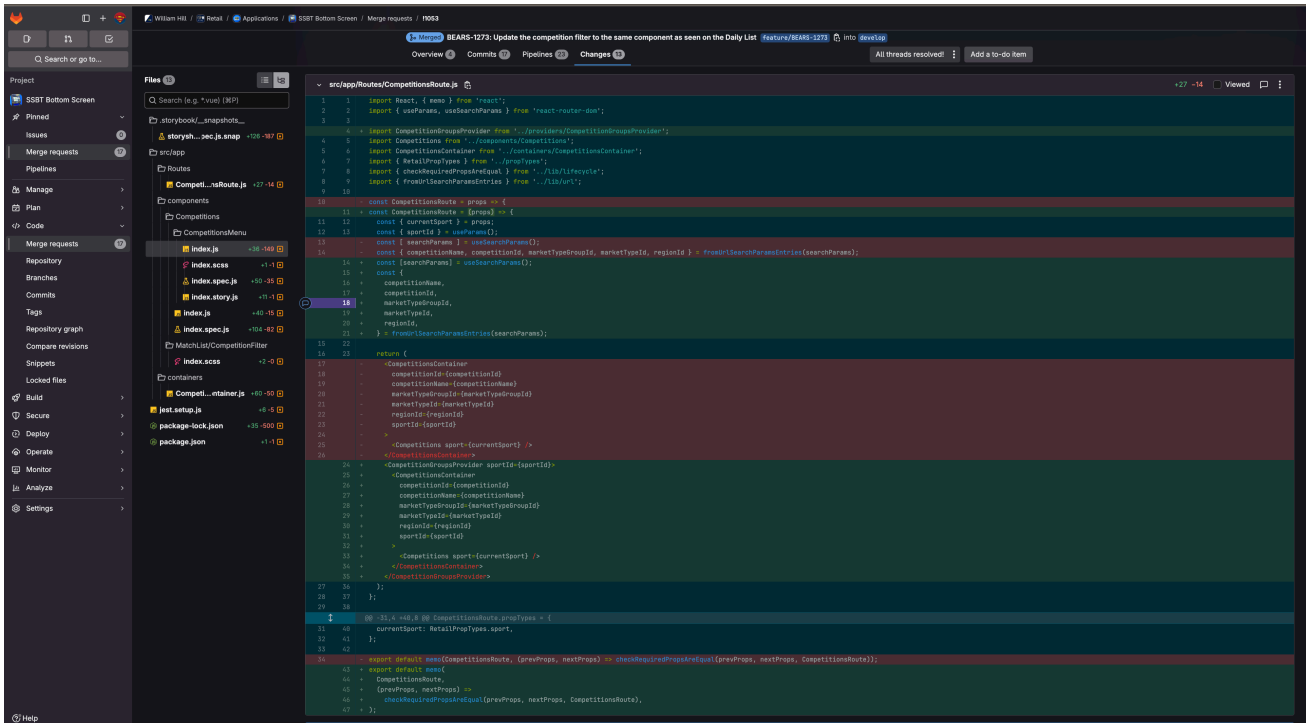
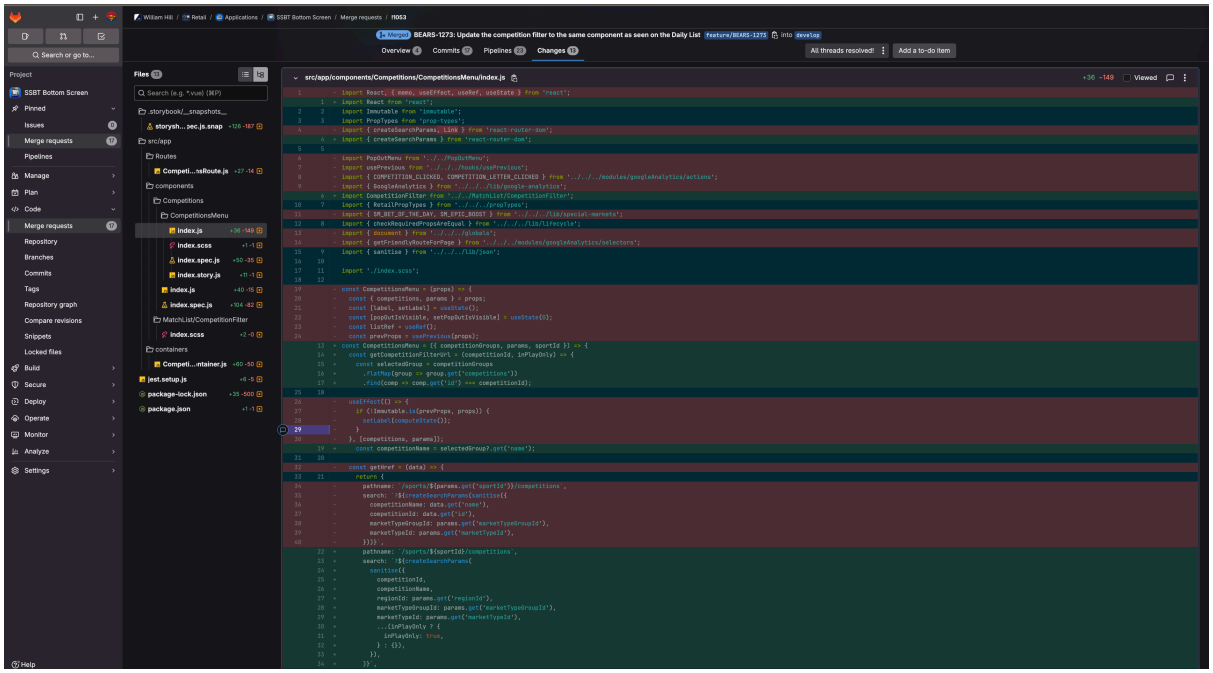


Figure 7 : Code changes - Routes/CompetitionsRoute.js



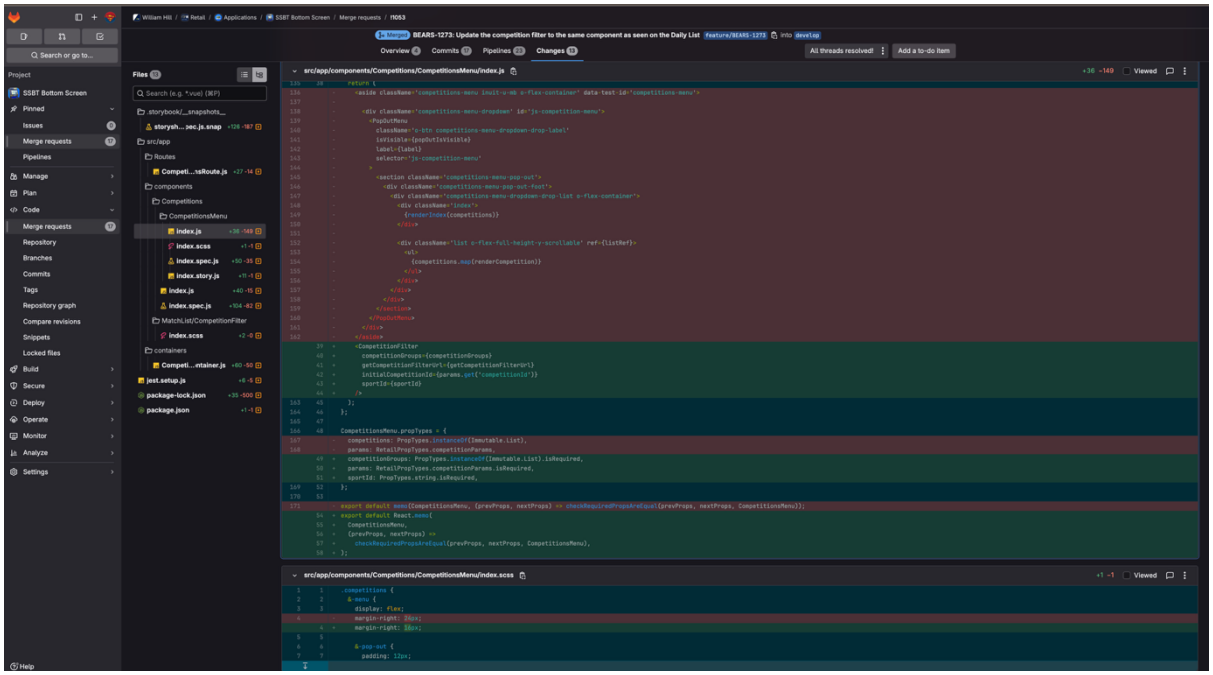


Figure 8 : Code changes - components/Competitions/CompetitionsMenu/ index.js & index.scss

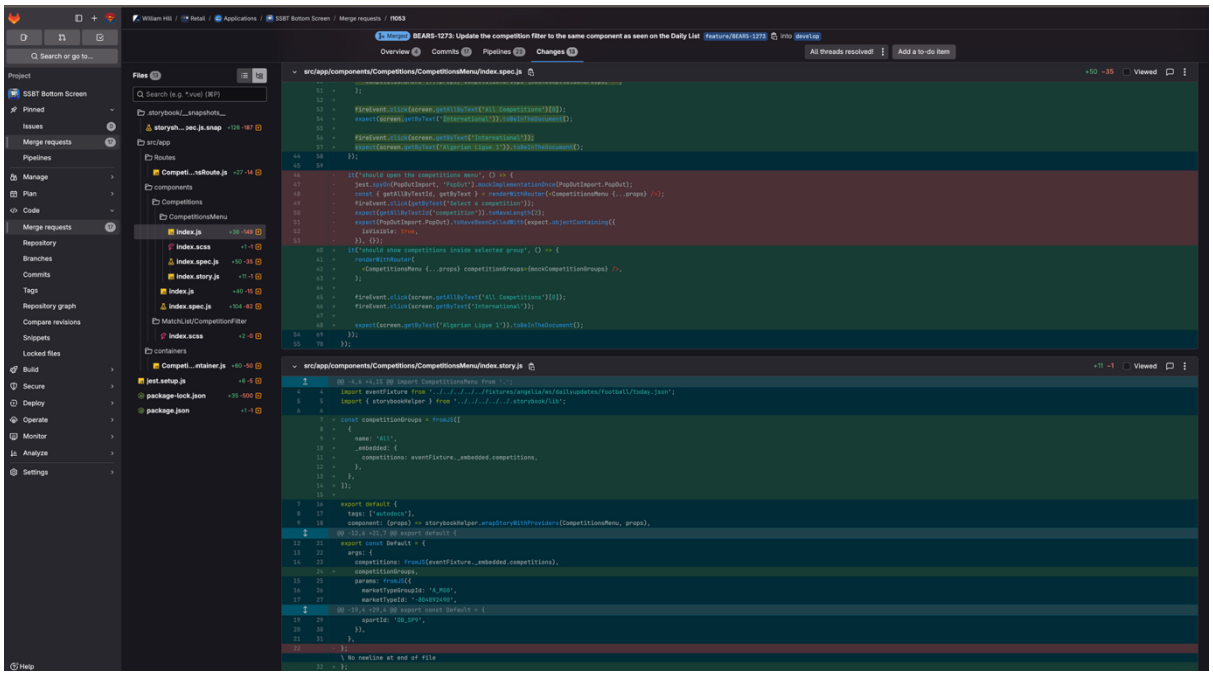


Figure 9 : Code changes

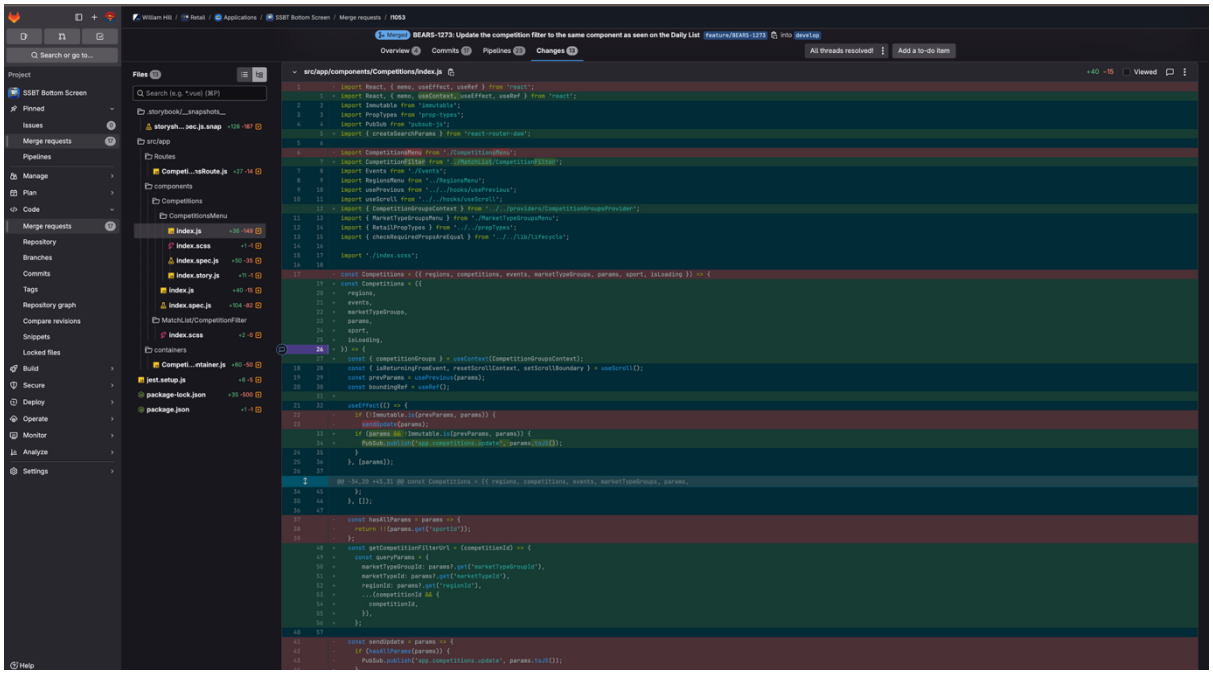


Figure 10 : Code changes

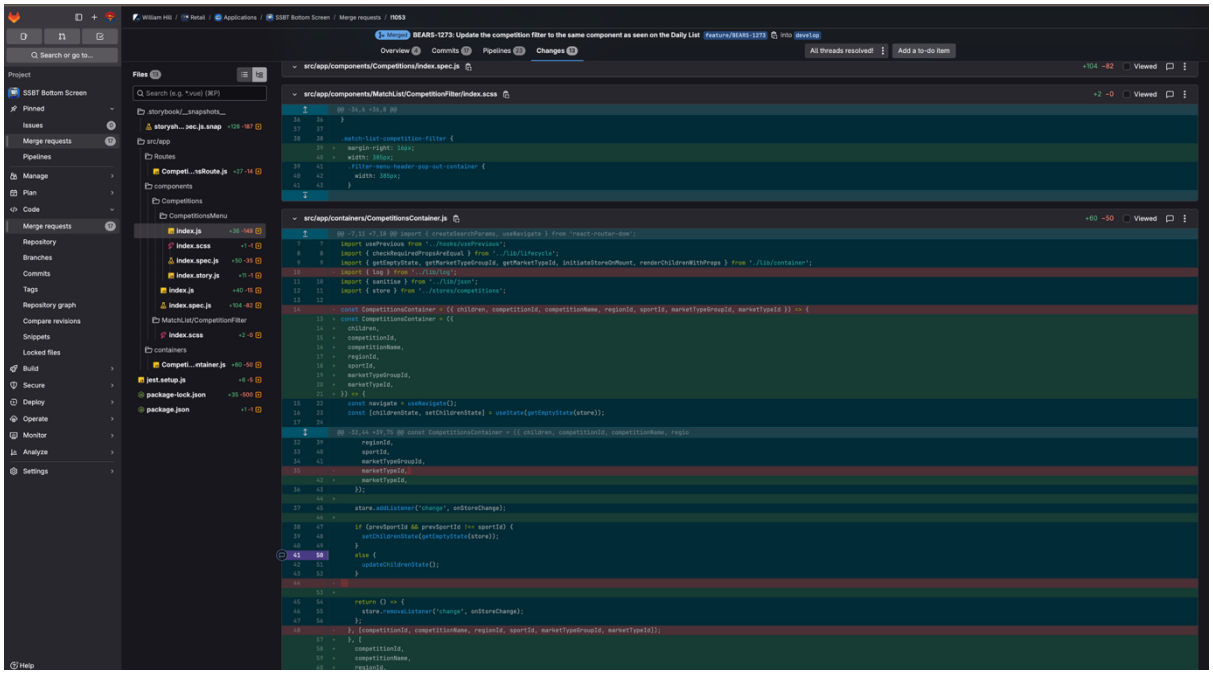


Figure 11 : Code changes

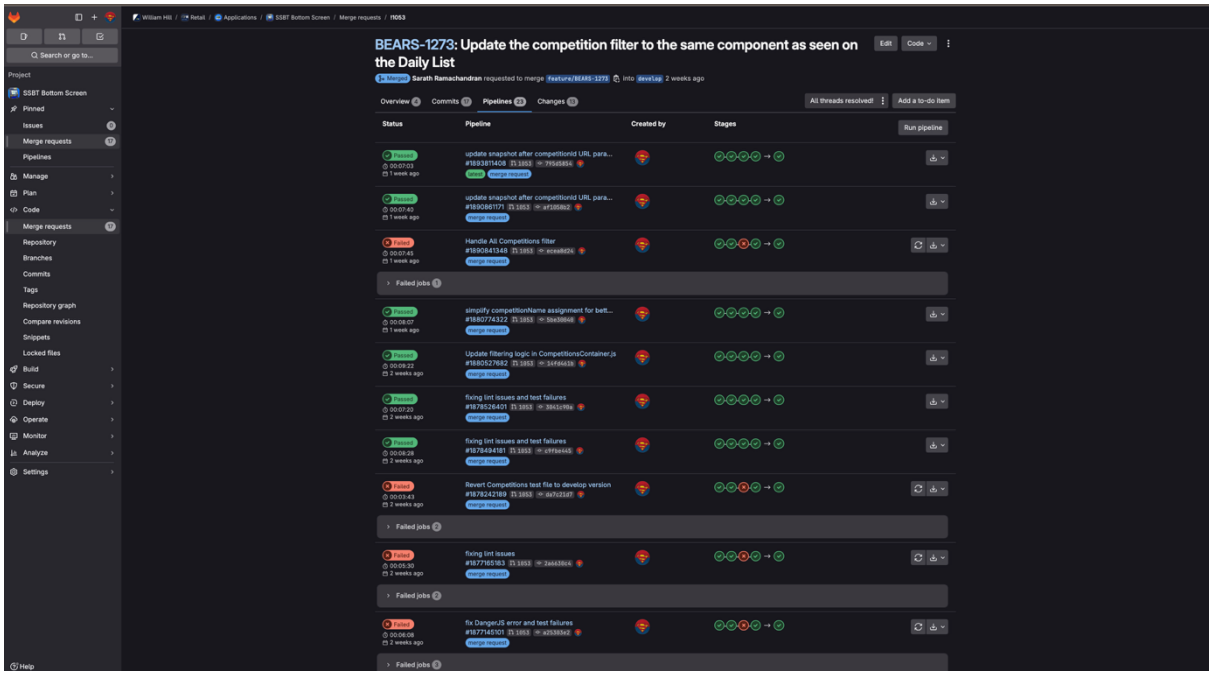


Figure 12 : Pipelines

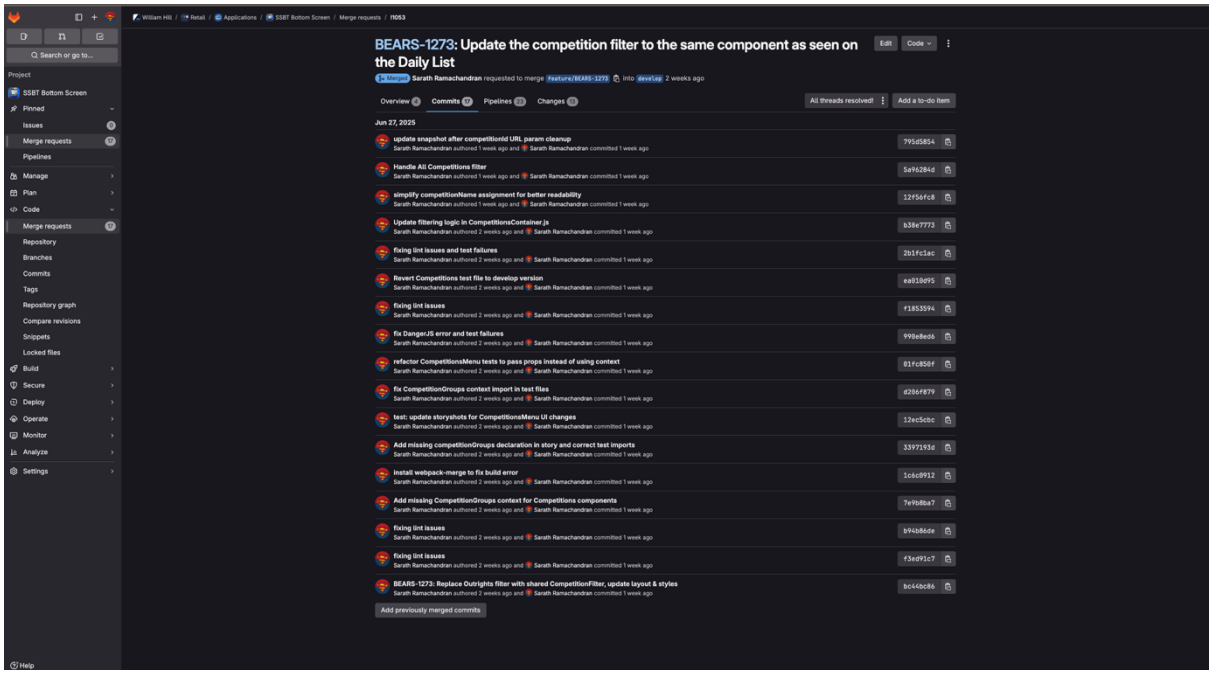


Figure 13 : Commit history

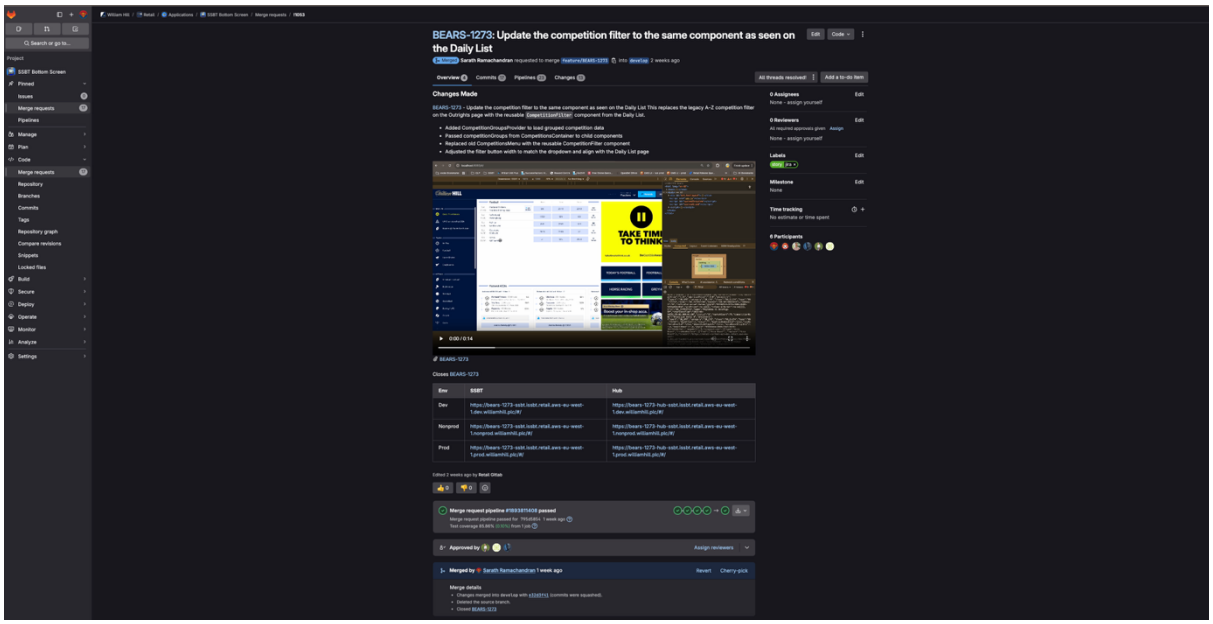


Figure 14 : MR overview

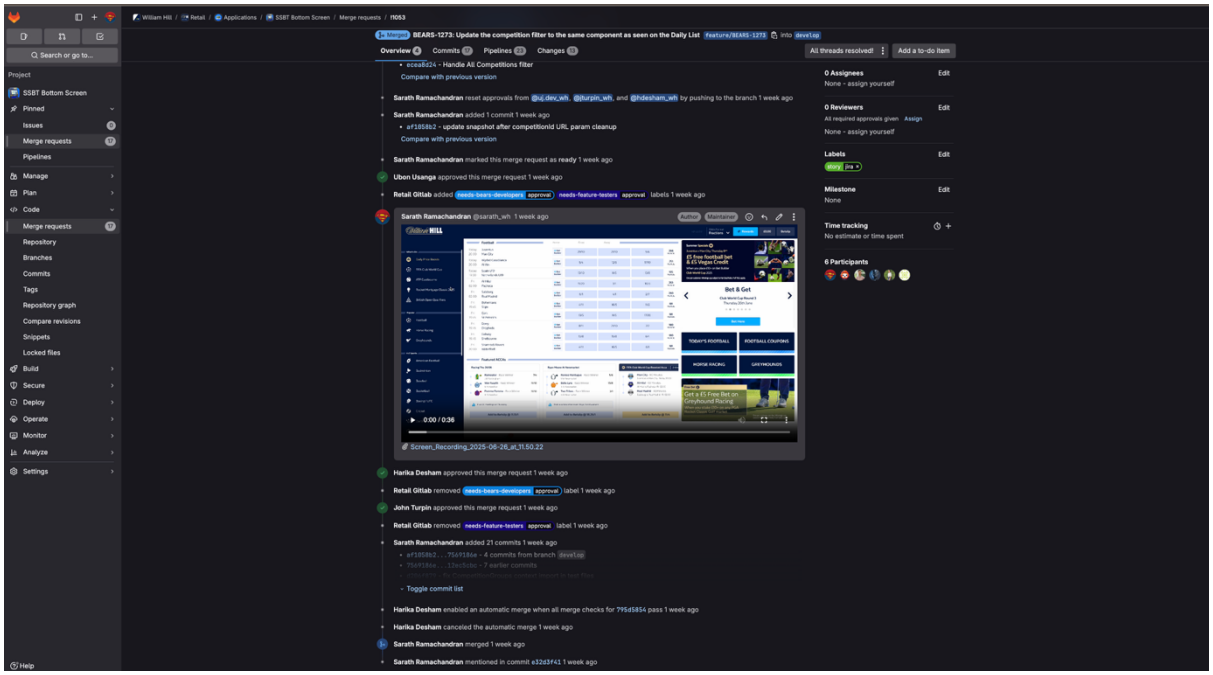


Figure 15 : MR overview

Communication and Collaboration

Throughout the BEARS-1273 implementation, I worked closely with my team to ensure alignment, quality, and smooth delivery:

- Daily Standups: Shared blockers and progress updates, particularly around testing issues and inconsistent filter behaviour.
- Slack Conversations: Used for quick clarifications with the designer and lead developer about dropdown structure, styling tweaks, and expected behaviours when switching filters.
- Merge Request Reviews: Received feedback on code quality and suggested changes such as simplifying the logic for competition Name and conditionally generating URLs. Addressed comments promptly.
- Pair Debugging: Collaborated with a senior developer to investigate why switching back to “All Competitions” resulted in a blank state. This involved debugging state updates, PubSub events, and route transitions.

Reviewer feedback directly influenced improvements to:

- Filter URL generation logic
- Default value handling for missing parameters
- Snapshot and test reliability

This collaborative process ensured cleaner code, better UX, and a shared understanding across the team.

Result

The final feature was merged into the develop branch following code review approval and successful regression testing in both automation and manual testing environments.

- Automated Tests : Unit and integration tests validated the component behaviour, including dropdown rendering, state updates, and URL changes. Snapshot tests were updated accordingly.
- Manual QA : Testers verified the dropdown interaction on the Outrights page in both non-prod and production-like environments. A full regression pack was executed before go-live.
- Version Control : The implementation was tracked in a dedicated feature/BEARS-1273 branch, regularly rebased with develop. Merge requests were used for peer review.
- Visibility : The completed feature was demoed to stakeholders in sprint review, showcasing improved dropdown functionality, consistent UX across pages, and improved accessibility compliance.

The change successfully resolved the issue of inconsistent behavior in the Outrights page's competition filter, bringing it in line with the Daily List experience. The ticket was marked complete in JIRA with all acceptance criteria met.

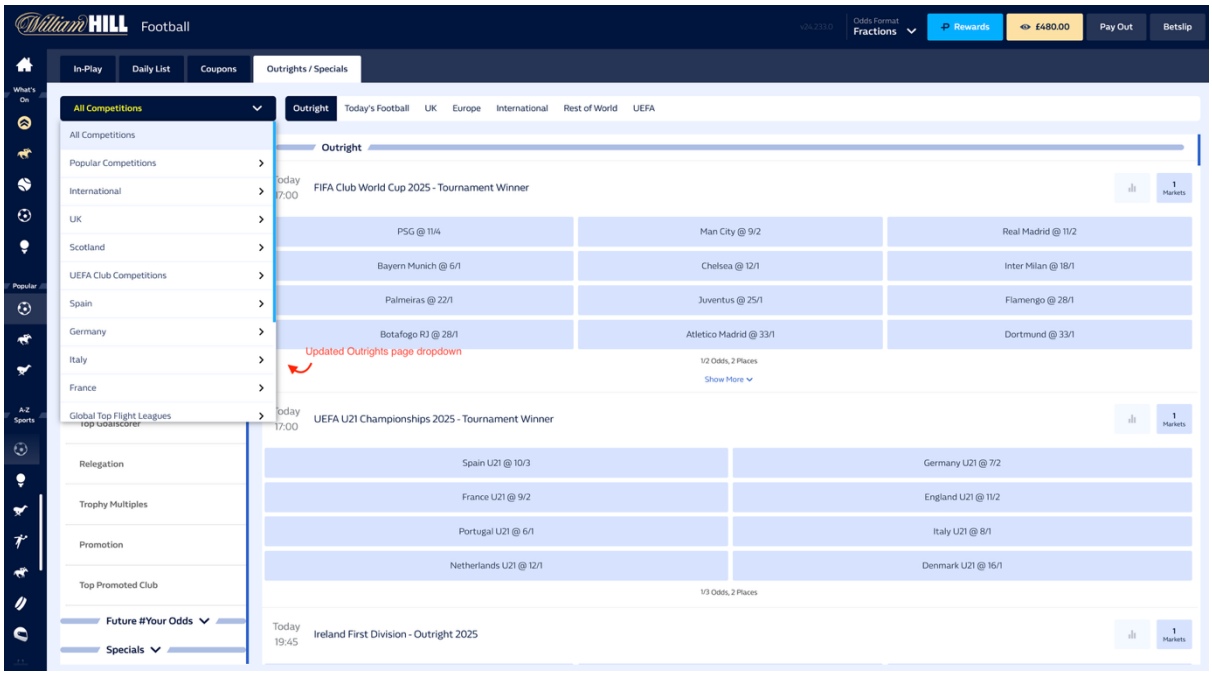


Figure 16 : Updated Outrights page

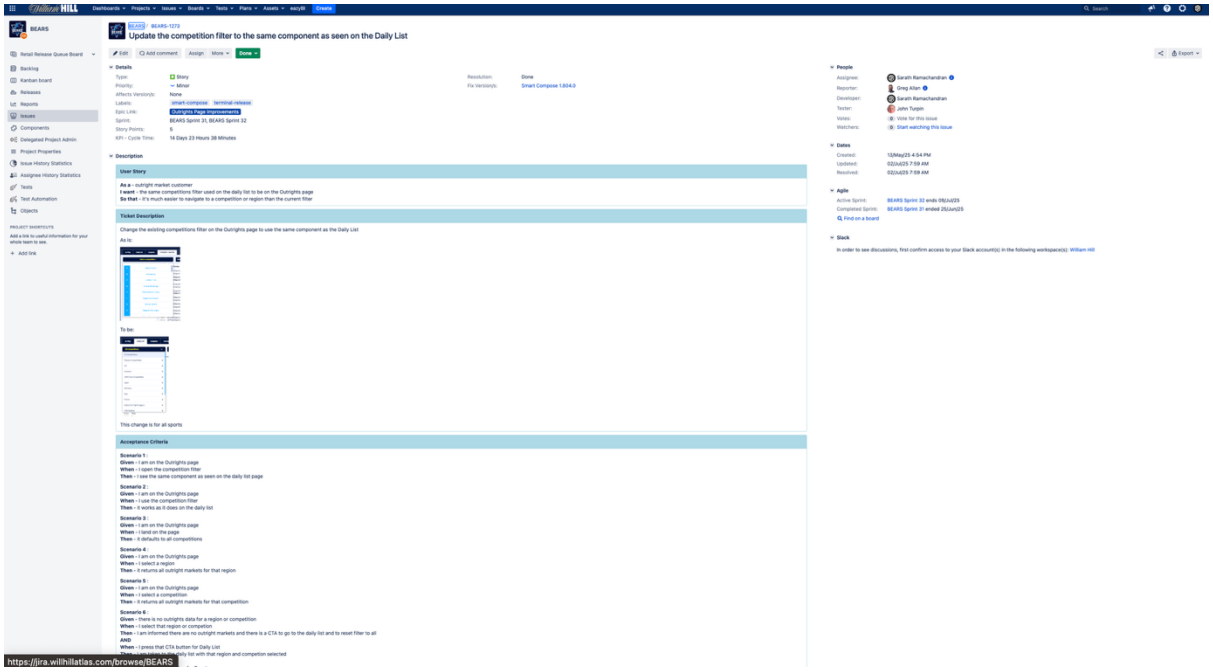


Figure 17 : Ticket status

Reflection

Wins :

- Successfully refactored and reused the existing CompetitionFilter component across pages, improving consistency.
- Learned how to properly integrate and scope React Context providers to avoid prop-drilling and state issues.
- Enhanced understanding of writing clean, isolated unit and snapshot tests for components relying on context and routing.

Challenges :

- Ensuring the filter behaved identically on both Outrights and Daily List pages, especially when selecting and reverting to “All Competitions”.
- Addressing DangerJS MR warnings, linting issues, and ESLint import order rules post-merge.
- Debugging snapshot mismatches and route param inconsistencies—particularly with competitionId=null appearing in URLs.

Lessons Learned :

- Wrapping routes in correctly scoped context providers is essential when reusing UI logic.
- Consistent UX patterns across similar views help reduce cognitive load and improve user trust.
- Maintaining test isolation and snapshot discipline is critical during larger refactors, saving time during CI and peer review.

Overall, this ticket strengthened my ability to contribute clean, maintainable code and deepened my confidence in tackling cross-page refactors, improving accessibility and developer experience.