OPUS: Optimistic Proactive Update System for Real-Time User Experience in SolidJS Applications

Technical Documentation Team
Reynard Project

September 8, 2025

Abstract

We present OPUS (Optimistic Proactive Update System), a comprehensive framework for implementing optimistic UI updates in SolidJS applications, with a focus on the YipYap image annotation platform. OPUS leverages SolidJS's fine-grained reactivity to deliver instant feedback for user actions, improving perceived performance and user satisfaction. We analyze current optimistic update patterns in YipYap, identify areas for further enhancement, and provide best-practice guidelines for robust, resilient optimistic UI.

1 Introduction

Modern web applications must deliver instant feedback to users, even when network latency is unavoidable. Optimistic UI updates—where the interface is updated immediately in anticipation of a successful server response—are a proven strategy for achieving this. In SolidJS, fine-grained reactivity enables efficient, low-overhead optimistic updates. OPUS is designed to systematize and extend these patterns in the YipYap platform.

2 SolidJS Reactivity and Optimistic Updates

SolidJS's reactivity model is based on signals, stores, and resources. Optimistic updates are implemented by updating local state immediately upon user action, initiating the server request in the background, and rolling back the UI if the server fails. SolidStart's `useSubmissions` hook provides a higher-level abstraction for optimistic UI with server actions, but the pattern is easily implemented in classic SolidJS using signals and stores.

3 OPUS Algorithm

3.1 Optimistic Update Flow

Let S be the current state, S' the optimistically updated state, and R the server response. The OPUS flow is as follows. When a user initiates an action, the UI state is immediately updated to S'. An asynchronous request is sent to the server. If the server responds with success, S' is retained. If the server fails, the UI is rolled back to S and the user is notified. This approach ensures a highly responsive user experience while maintaining data integrity.

3.2 Error Handling

Rollback is performed by restoring the previous state. The user is notified of the error via the notification system. Optionally, retry logic or conflict resolution can be implemented for advanced scenarios.

4 System Architecture

OPUS is integrated into the YipYap frontend as a set of composable patterns and best practices. The architecture leverages SolidJS's fine-grained reactivity, the notification system, and the app context for global state management. Optimistic updates are applied at the component or context level, with rollback logic and error notifications handled consistently across the application.

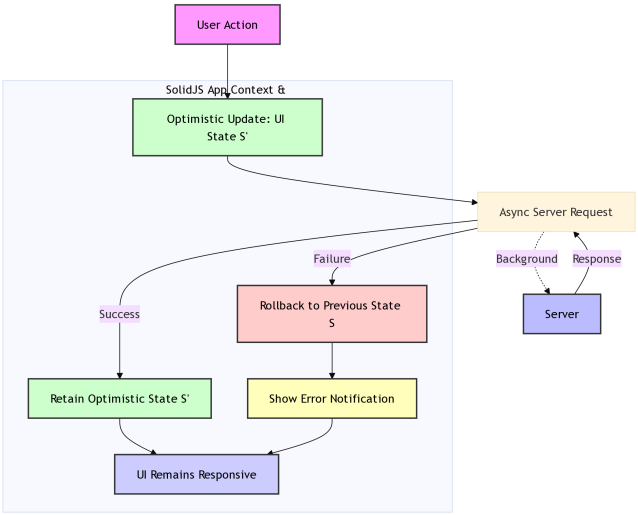

Figure 1: OPUS System Architecture and Data Flow

5 Current Optimistic Update Implementations in YipYap

5.1 Gallery File Uploads

After upload, new items are added to the gallery UI before server confirmation. If the server fails, the UI is reverted and an error is shown.

5.2 Caption Editing

Caption changes are reflected in the UI immediately, with rollback on error.

5.3 Favorites

Favorite state is toggled in the UI before server confirmation, providing instant feedback.

5.4 Tag Management

Tag add, edit, and remove operations are performed optimistically, updating the UI instantly.

5.5 Bounding Box Editing

Add, update, and delete operations for bounding boxes are performed optimistically, ensuring a smooth annotation experience.

6 Opportunities for Further Optimistic Updates

6.1 Git Operations

Currently, the UI waits for server response before updating git status, branches, and related state. OPUS proposes optimistically updating git state in the UI, with rollback on error.

6.2 Batch Operations

Batch deletes and multi-item operations may not update the UI until server confirmation. OPUS recommends removing items from the UI immediately, with rollback if the server fails.

6.3 Settings and Preferences

Some settings changes are only reflected after server confirmation. OPUS suggests updating UI state immediately, with rollback on error.

6.4 Gallery Deletion

Deletion of images and folders may not be optimistic. OPUS proposes removing items from the UI immediately, with rollback if the server fails.

6.5 Candidate Files for OPUS Integration

The following files are prime candidates for further OPUS integration:

- `src/composables/useGitManager.ts`
- `src/contexts/gallery.ts`
- `src/contexts/selection.ts`
- `src/components/Gallery/Gallery.tsx`
- `src/components/Settings/MemorySettings.tsx`

7 Best Practices for Optimistic Updates in SolidJS

OPUS recommends keeping update functions pure, handling errors gracefully with rollback and user notification, leveraging fine-grained reactivity for efficient updates, batching updates to minimize reactivity cycles, and providing clear feedback through the notification system. Thorough testing is essential to ensure rollback logic is robust and covers all edge cases.

8 Conclusion

OPUS demonstrates that optimistic UI updates, when combined with SolidJS's fine-grained reactivity, can deliver a highly responsive user experience. By expanding optimistic update patterns to more areas of YipYap—such as git operations, batch actions, and settings—the platform can further enhance user satisfaction and perceived performance.

9 References

References

[1] Optimistic UI With SolidStart. Available: `https://www.brenelz.com/posts/optimistic-ui-with-solid-start/`

[2] Understanding optimistic UI and React's useOptimistic Hook. Available: `https://blog.logrocket.com/understanding-optimistic-ui-react-useoptimistic-hook/`

[3] OPTIMUS v2.1: Performance-Optimized Direct API Integration with Progressive Loading for Large-Scale Data Selection in SolidJS Applications. Reynard Project Documentation.

[4] SolidJS Documentation. Available: `https://www.solidjs.com/docs`