COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT

CSCI 370 Final Report

The Reactionaries V2

Sam Bangapadang
Ethan Leuthauser
Danny Nguyen

Revised December 4, 2024

CSCI 370 Fall 2024

Prof. Kathleen Kelly

Table 1: Revision History

Revision	Date	Comments
New	8/28/24	Completed Sections: I. Introduction II. Functional Requirements III. Non-functional Requirements IV. Risks V. Definition of Done
Rev – 2	9/14/24	Updated and Complected Sections: III. Non-Functional Requirements IV. Risks VI. System Architecture
Rev – 3	10/19/24	Updated and Complected Sections: IV. Risks V. Definition of Done VI. System Architecture VII. Software Test and Quality VIII. Project Ethical Considerations
Rev – 4	11/10/24	Updated and Complected Sections: IX. Project Completion Status X. Future Work XI. Lessons Learned XIII. Team Profile XIV. References XV. Appendix A – Key Terms
Rev – 5	11/19/24	Updated and Complected Sections: I. Introduction VI. System Architecture IX. Project Completion Status X. Future Work XI. Lessons Learned XII. Acknowledgments XIII. Team Profile XV. Appendix A – Key Terms
Rev – 6	11/23/24	Updated and Complected Sections: VI. System Architecture X. Future Work XII. Acknowledgments
Rev – 7	12/4/24	Updated and Complected Sections: VI. System Architecture

Table of Contents

I. Introduction

I.I. Client Information

Qualcomm Incorporated is a U.S.-based multinational company, renowned for its pioneering advancements in semiconductors, software, and wireless technology services. The company is a global leader in mobile technology innovation, with its flagship Snapdragon System-on-Chip (SoC) platforms powering smartphones, tablets, and other devices worldwide. Qualcomm is a driving force behind the development and deployment of modern technology, also spanning Modems, Radio Frequency Transceivers, Artificial Intelligence, Connectivity Chips, Automotive Semiconductor Solutions, and Internet of Things (IoT) Technologies. Ultimately, Qualcomm's development efforts have transformed connectivity across industries.

For this project, our team will collaborate with Qualcomm's highly specialized team based in Boulder, Colorado. This team is at the forefront of designing and engineering the Test Base Station (TBS), a critical component in the validation and optimization of cellular tower base stations deployed worldwide. Cellular base stations act as central hubs, forming the backbone of modern cellular networks. By working with the Boulder team, we aim to contribute to Qualcomm's mission of driving innovation in wireless communication technologies while ensuring the solution meets the needs of its stakeholders, including Customer Engineering and Development teams.

I.II. Project Background

Qualcomm's diverse portfolio of products requires extensive customer support, with over 2,000 systems utilized daily across the organization. This high level of activity generates a significant volume of issues that engineers must address. These issues are tracked using multiple Atlassian Jira projects, a project management and issue tracking tool. While Atlassian Jira is the industry standard for project management, Qualcomm finds this current instance to be cumbersome and lacks the flexibility needed to align with the company's unique and evolving workflow requirements. To accelerate the release of cutting-edge technology, Qualcomm has addressed the need for a solution that extends and enhances Jira's capabilities, rather than replacing it, to better optimize issue tracking and resolution processes.

I.III. High Level Project Description

Our team is dedicated to streamlining Qualcomm's internal workflows by extending and enhancing an existing customer support front-end, enabling seamless visualization, modification, and organization of Jira issues across Customer Engineering and Development teams at Qualcomm. We aim to address pain points by introducing features that cater to Qualcomm's unique workflow practices, such as advanced filtering options and tailored dashboards with custom interactions, enabling faster issue resolution and improved productivity. Through innovative front-end design and optimizations, our solution will play a pivotal role in streaming operations and driving technological advancements within Qualcomm. Once implemented, the software will be maintained by Qualcomm's internal IT and development teams, supported by thorough documentation and knowledge transfer to ensure long-term success.

II. Functional Requirements

As the project evolved, the functional requirements of the project changed drastically to address specific user needs. One of the team's original application requirements, supporting tier escalations, was removed from the scope after an internal reassessment of the project requirements.

Requirement	**Details**	**Priority**	**Testing Criteria**	**Retained in Scope?**
Tier Escalation Tool	Implement a single-click button to escalate issues from Tier 2 to Tier 3.	High	Verify escalation functionality completes with one click and updates issue status in Jira.	

Specialized Task Pages	Create separate pages for Long-Term Support, Escalations, and general use.	Medium	Ensure each page is functional and supports intended user workflows.	X
Advanced Issue Filtering	Enable filtering by priority, type, project, version, status, and resolution.	High	Test filters for accuracy and usability, with the ability to save custom filter presets.	X
Rich Text Input and Issue Highlighting	Support RTF user input and highlight specific Jira labels with icons for roadmap, blocking issues, etc.	Medium	Test RTF input accuracy and verify highlighting is applied correctly to pre-defined Jira labels.	X
Interactive User Interface	Provide visual tools to display dependencies and expandable issue cards with customizable layouts.	High	Confirm issue cards accurately depict issues and that cards are customizable and expandable.	X
Release Management Tools	Integrate Long Term Support (LTS) tools for release approval, analysis, and feature/version tracking.	High	Validate LTS analysis accuracy and the ability to manage trackers and release versions effectively.	X

III. Non-Functional Requirements

Requirement	**Details**	**Priority**	**Testing Criteria**
High-Performance Operations	Ensure efficient handling of issues, filtering, and escalations.	High	Measure API response times, filtering operations under varying workloads, and ticket escalation speed.
Scalability	Maintain consistent performance as the user base and data volume grow.	High	Test system behavior under simulated high workloads and user concurrency.
Security and Data Protection	Adhere to security standards for data handling and internal libraries.	High	Verify secure handling of Jira requests and npm libraries through penetration and compliance testing.
User-Friendly Interface	Provide an intuitive, accessible UI with clear visual indicators for key actions.	Medium	Conduct usability and accessibility testing, ensuring WCAG compliance.
System Reliability	Achieve high system availability with robust error handling.	High	Monitor uptime metrics and test error-handling scenarios to minimize disruptions.
Streamlined Deployment	Implement CI/CD pipelines within a Docker and Kubernetes environment for efficient deployment.	Medium	Validate successful deployments and CI/CD pipeline execution in the target environment.
Maintainable Codebase	Ensure a well-documented, modular, and easily updatable codebase.	Medium	Review code documentation and structure for maintainability during peer reviews.

IV. Risks

IV.I. Technical/Operational Risks

Risk Description	Impact (0-5)	Likelihood (0-5)	Mitigation Plan
Loss of data disrupting Qualcomm's existing workflow.	5: Destructive to Qualcomm's internal platforms and existing data.	0: Very unlikely	Implement read-only access to critical data and ensure new Jira tickets are created when modifications are needed to avoid accidental data loss.
Security of Jira Requests (i.e., Malicious Requests)	4: Could lead to unauthorized access or manipulation of data.	2: Unlikely	Use secure storage for API keys and tokens, implement request validation, and enforce strict authentication protocols.
Release of confidential information affecting Qualcomm's security internals.	5: Potentially catastrophic, leading to data breaches, legal implications, and loss of trust.	1: Very unlikely	Develop and test in a secure virtual environment, restrict data access, and implement encrypted communications for sensitive operations.
Inadequate testing may lead to bugs in production.	4: Could cause application malfunctions or data corruption.	3: Likely	Develop a comprehensive testing strategy, including unit, integration, and system testing, and enforce thorough code reviews before deployment.
Miscommunication among team members may hinder productivity and lead to misalignment in project goals.	3: Could delay the project and cause misalignment of goals.	2: Unlikely	Use clear team communication protocols, regular check-ins, and project management tools to ensure alignment and consistent collaboration.
Scalability and performance issues appear as the system grows.	3: May cause bottlenecks or system crashes.	5: Very likely	Conduct performance testing, implement load balancing, and optimize API calls to address existing and potential bottlenecks.
Changes in project scope affecting deadlines and final product.	4: Could lead to missed deadlines and unmet objectives.	3: Likely	Establish clear scope agreements at the start, use a change management process for new requests, and prioritize tasks to maintain focus on key objectives.
Refactoring could lead to further broken implementations of previously working systems	3: Fluctuates depending on affected features.	4: Likely	Use version control systems with rollback capabilities, maintain robust test coverage, and conduct regression testing after every refactor.

IV.II. Skills Risks

Skill	Point of Use	Impact (0-5)	Likelihood (0-5)	Risk Mitigation Plan – Avg. Knowledge (1-5)
TypeScript	Front-End: Essential for developing application logic.	5: Essential	3: Likely	**Avg. Knowledge: 4.0** Ensure all team members maintain a strong understanding through regular code reviews and collaborative programming sessions.

React	Front-End: Core framework for building the user interface.	5: Essential	3: Likely	**Avg. Knowledge: 4.0** Provide hands-on workshops and ensure existing UI developers mentor less experienced team members to maintain knowledge parity.
Python	Back-End: Primary language for backend development.	5: Essential	2: Unlikely	**Avg. Knowledge: 4.0** Leverage existing backend expertise; assign complex Python tasks to experienced developers to ensure high-quality implementations.
FastAPI	Back-End: Building APIs with Python.	5: Essential	4: Very Likely	**Avg. Knowledge: 3.3** Conduct targeted training sessions and allocate simpler API tasks to junior members to build proficiency while leveraging team expertise.
Tailwind CSS	Front-End: Utility-oriented CSS framework.	4: Non-Essential	2: Unlikely	**Avg. Knowledge: 3.0** Utilize pre-built templates and limit custom CSS to essential components, reducing reliance on advanced knowledge of the framework.
Mantine	Front-End: React-component library that offers a wide range of pre-built components.	5: Essential	2: Unlikely	**Avg. Knowledge: 4.6** Rely on documentation and pre-built examples to minimize complexity in implementation and maintain existing high proficiency.
Vite	Front-End: Build tool that offers fast development for front-end projects.	5: Essential	2: Unlikely	**Avg. Knowledge: 3.0** Assign setup tasks to experienced developers and provide step-by-step documentation for configuration to simplify adoption.
Jira	Scrum & Front-End: Software used to manage tickets.	5: Essential	3: Likely	**Avg. Knowledge: 3.8** Provide training on best practices for ticket management, ensure adherence to workflows, and use existing documentation as a reference.
Hatch	Back-End: Used for managing Python environments, deployment, and packaging.	5: Essential	2: Unlikely	**Avg. Knowledge: 3.0** Ensure experienced backend developers oversee environment configuration and deployment processes while mentoring team members.
Zustand	Front-End: Used for state-management of React components.	3: Replaceable	2: Unlikely	**Avg. Knowledge: 1.0** Allocate dedicated time to team members to improve Zustand and assign experienced mentors to ensure proper state management practices.

V. Definition of Done

V.I. Minimal Useful Feature Set

The following features are considered the minimum requirements for the project to be deemed complete:

1. Long-Term Support (LTS) Workflow Automation
   a. Automate the Long-Term Support (LTS) process used by Qualcomm engineers to address LTS issues.
   b. Improve performance to align with envisioned speed requirements.
2. Advanced Issue Filtering
   a. Implement filters based on common properties (e.g. priority, type, status, version).
   b. Add preset filters for commonly used queries that can be customized.

3. RTF Support
    a. Enable support for rich text input (RTF) to ensure accurate and flexible user input handling.
4. Task-Specific Dashboards:
    a. Create separate pages or tabs tailored to specific tasks, such as tier escalations or long-term support operations.
5. Intelligent Issue Display:
    a. Implement highlighting and grouping of issues, pointing out current and invalid statuses.

V.II. Client Acceptance Tests

The client will conduct the following tests before accepting the software:

1. Performance Tests:
    a. Measure the efficiency of the LTS workflow automation and ensure it meets performance benchmarks.
    b. Verify that the refactored codebase improvements yield faster and smoother operations.
2. Filter Functionality Tests:
    a. Validate the accuracy and usability of advanced issue filters for common properties and preset queries.
3. RTF Input Validation:
    a. Ensure RTF input is accurately captured and correctly displayed within the system.
4. Dashboard Usability Tests:
    a. Confirm that task-specific dashboards and features function as intended and support specific workflow needs.

V.III. Delivery Process

The product will be delivered incrementally through pull requests submitted to Qualcomm's internal GitHub repository. Each pull request will undergo thorough code review and client testing to ensure quality and alignment with project requirements.

The current state of the GitHub repository employs Continuous Integration (CI) deployment upon merging with the main branch. All deployments are integrated into Qualcomm's internal systems, ensuring seamless updates to staging our production environments. This setup allows the client to validate functionality and performance in real-time as changes are deployed.

Upon completion of all the features and successful client testing, the final version will be approved for deployment and fully operational within Qualcomm's infrastructure.

VI. System Architecture

VI.I. High-Level Architecture Overview

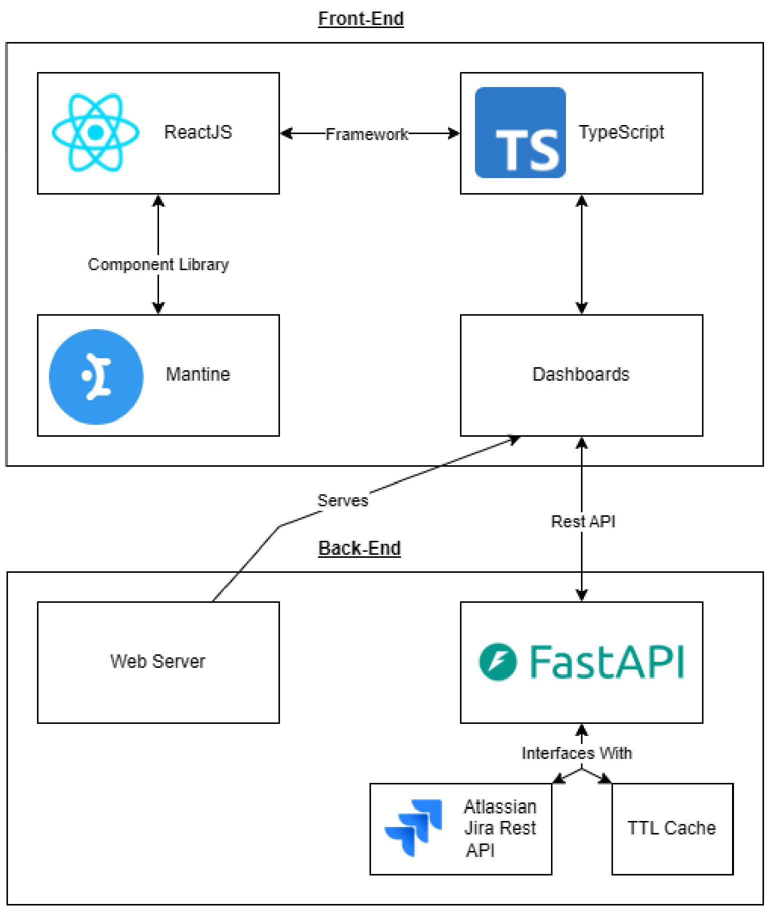


Figure 1: Application Architecture Diagram, Derived from Summer Team [1]

The application consists of a React-based frontend and a Fast-API based back-end, designed to provide a seamless interface for managing Jira tickets and Long-Term Support (LTS) workflows. This architecture ensures a responsive and efficient user experience tailored to Qualcomm's operational requirements. The diagram above illustrates the core components and their interactions.

The front-end leverages ReactJS as the primary framework, combined with TypeScript for type safety and scalable development. It consists of two main components: the general dashboard and LTS dashboard, accessible through selectable tabs. These dashboards allow users to filter and visualize Jira issues while ensuring intuitive navigation and task management. The UI components are built with the Mantine library, which standardizes design and minimizes the need for custom styling. The front-end communicates with the back end through REST API calls, which fetch data and perform actions as needed.

The back-end is powered by FastAPI, which serves as a lightweight and high-performance web server. It processes REST API requests from the front-end, interacts with the Atlassian Jira REST API, and leverages a Time-To-Live (TTL) Cache to optimize data retrieval and system responsiveness.

The TTL Cache is configured to store infrequently changing data, such as projects, priorities, and fix versions, for 12 hours. This interaction reduces latency by minimizing repetitive API calls to Jira, which can be time-intensive, especially for metadata that changes infrequently. By storing this data locally in the cache, the system ensures a more responsive user experience while reducing the load on Jira's API, improving overall performance and scalability.

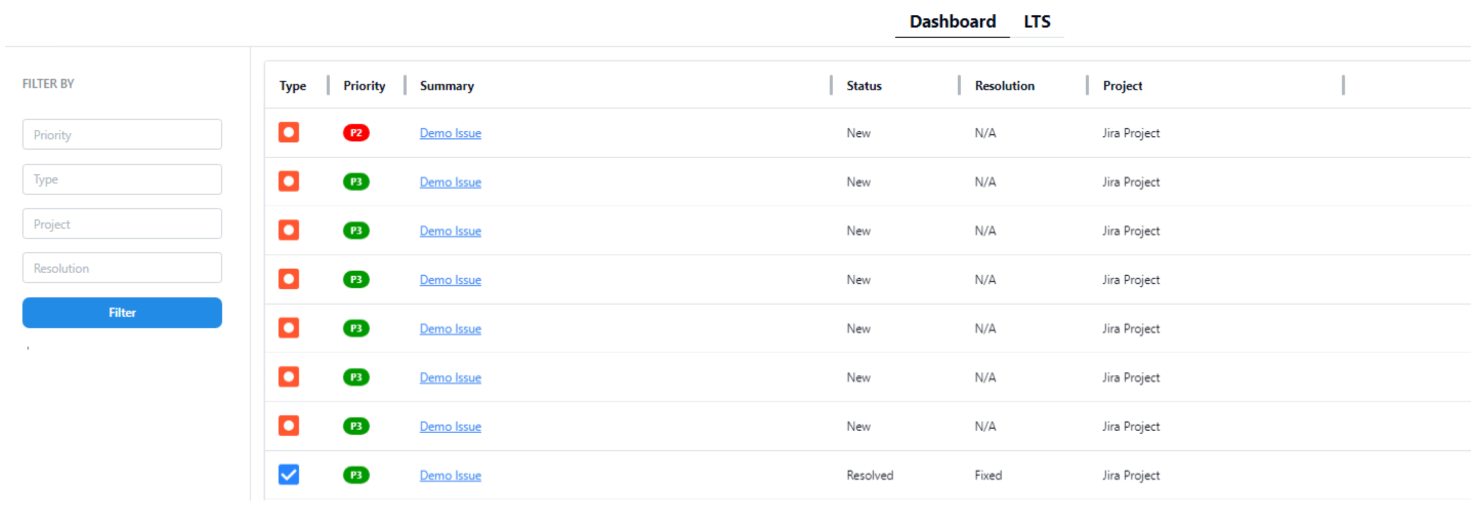
Key third-party elements include:

- Mantine Component Library: Provides pre-built and customizable UI components for consistent design across the application.
- Atlassian JIRA REST API: Handles all interactions with Jira, including fetching, updating, and creating ticket data.

This architecture design ensures for a cohesive integration of technologies, providing users with a responsive interface tailored to Qualcomm's workflows.

VI.II. Low-Level Overview / Technical Design

VI.II.I. General Dashboard Overview



Type	Priority	Summary	Status	Resolution	Project
🟥	P2	Demo Issue	New	N/A	Jira Project
🟥	P3	Demo Issue	New	N/A	Jira Project
🟥	P3	Demo Issue	New	N/A	Jira Project
🟥	P3	Demo Issue	New	N/A	Jira Project
🟥	P3	Demo Issue	New	N/A	Jira Project
🟥	P3	Demo Issue	New	N/A	Jira Project
🟥	P3	Demo Issue	New	N/A	Jira Project
✓	P3	Demo Issue	Resolved	Fixed	Jira Project

Figure 2: Refactored General Dashboard

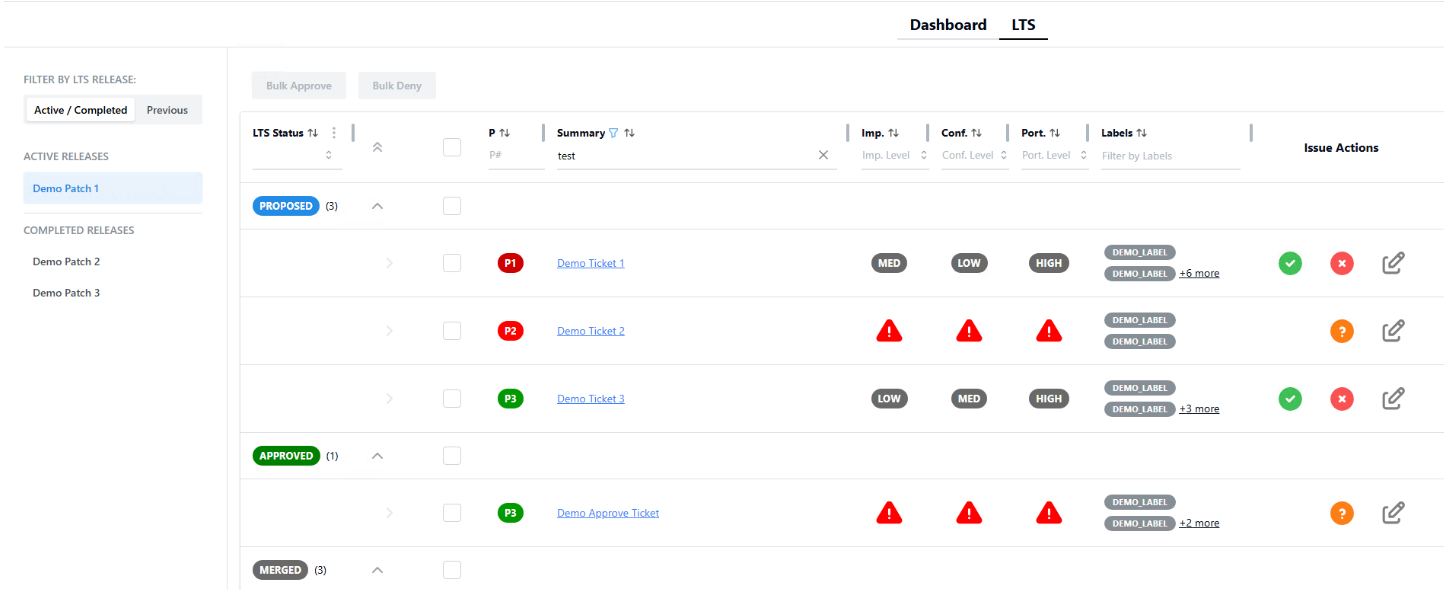
The General Dashboard serves a critical front-end component for managing and filtering Jira tickets across all projects. Initially, the dashboard was set up to integrate automation for escalation and handover processes. However, these functionalities were later removed from scope. This refactor establishes a foundation for integrating these processes in the future, while delivering significant performance, usability, and maintainability improvements.

During the development phase, the dashboard was refactored into a table-based structure utilizing the Mantine React Table (MRT) library. This decision was made to maintain design and functional consistency with the newly developed LTS Dashboard. This significant refactor led to the below improvements:

1. Performance Enhancements
    a. By integrating API caching and removing the unnecessary custom fields in the back-end (discussed in a later section), page load times were significantly reduced.
    b. Previously, the dashboard took approximately 8000-9000 milliseconds to load. After optimization, the load time was reduced to under 3000 milliseconds.
2. Standardized Component Library
    a. The General Dashboard was migrated away from custom Radix components in favor of the Mantine UI library.
    b. This transition eliminated the need for custom UI components, streamlining development and improving maintainability.
3. Simplified UI and Usability
    a. The new table structure enables for better organization and navigation of ticket data, with intuitive filtering options for common metadata such as ticket type, priority, status, resolution, and parent project.
    b. The consistent design ensures that users experience a cohesive interface across both the General and LTS dashboards.

By focusing on performance, usability, and maintainability, the General Dashboard now provides an efficient and consistent user experience while reducing the complexity of the codebase. Additionally, the groundwork laid in this refactor supports the potential for future integration of automated processes.

VI.II.II. LTS Dashboard Overview



LTS Status		P	Summary	Imp.	Conf.	Port.	Labels	Issue Actions
		P#	test	Imp. Level	Conf. Level	Port. Level	Filter by Labels	
PROPOSED (3)								
		P1	Demo Ticket 1	MED	LOW	HIGH	DEMO_LABEL DEMO_LABEL +6 more	
		P2	Demo Ticket 2				DEMO_LABEL DEMO_LABEL	
		P3	Demo Ticket 3	LOW	MED	HIGH	DEMO_LABEL DEMO_LABEL +3 more	
APPROVED (1)								
		P3	Demo Approve Ticket				DEMO_LABEL DEMO_LABEL +2 more	
MERGED (3)								

Figure 3: LTS Dashboard

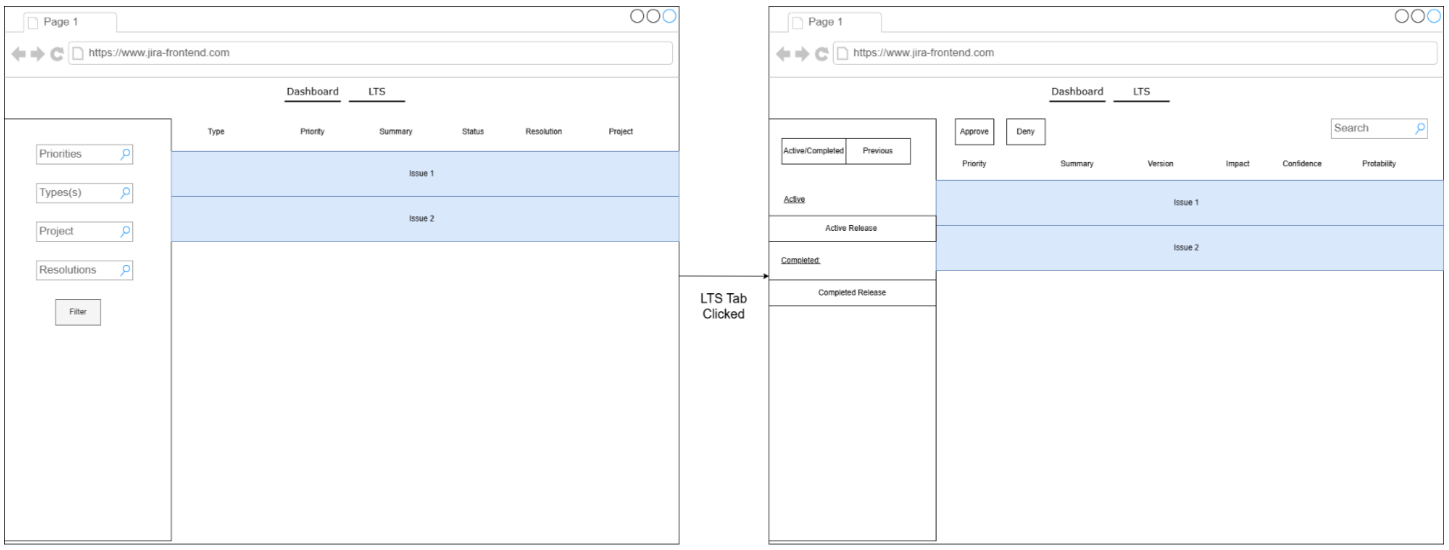


Figure 4: Tab Navigation & LTS Design Wireframe

The Long-Term Support (LTS) Dashboard is a specialized interface designed to streamline actions and workflows related to LTS processes. This dashboard provides advanced filtering and analytical capabilities that help users manage and assess LTS tickets more effectively. Key features and functionality include:

1. Version-Based Filtering
    a. Users can filter issues based on release types, specifically “Active”, “Completed”, and “Previous” versions.
    b. Selecting a release type in the navigation bar automatically filters the relevant issues to be displayed in the table. This filtered data can be further refined using the column-based filter menus below the header of every column.
2. LTS Status and Label Grouping

    a. Tickets are grouped dynamically by their LTS Status (e.g. Proposed, Approved, Merged), providing a clear visual organization of tickets.
    b. Additional grouping is available by Labels, allowing users to categorize tickets based on relevant workflow labels.
    c. This grouping structure enhances usability by enabling users to quickly identify and prioritize tickets within their workflow.
    d. Several groups are established to identify edge cases or invalid statuses, functioning as a utility to ensure the integrity of the ticket merging process.
3. LTS Analysis Columns:
    a. The dashboard introduces three dedicated columns: Impact, Confidence, and Portability, which provide critical metrics for LTS ticket evaluation.
    b. These metrics, entered by ticket reporters, help assess the viability of merging the ticket with the active release.
        i. Impact: Evaluates the importance or severity of the ticket.
        ii. Confidence: Indicates the certainty or reliability of the analysis
        iii. Portability: Reflects how easily the solution can be implemented across different versions.
    c. The columns also serve as a tool to identify tickets missing an LTS analysis.
4. Navigation and Usability:
    a. The LTS Dashboard is accessible via a tab-based navigation system located at the top of the application.
    b. This design allows users to easily switch between the General Dashboard and the LTS Dashboard, ensuring a seamless workflow.
5. Visualization and Design Enhancements
    a. The layout and user experience of the LTS Dashboard are informed by design wireframes (as shown in Figure 4). These wireframes were iteratively refined to maximize usability and align with Qualcomm's operational needs.

VI.II.III. Results of Performance Enhancements

Performance Metric	Before	After
Network Performance (General Dashboard)	~8,000-10,000 ms	~2,000-3,000 ms
Network Performance (LTS Dashboard)	N/A	~4,000-5,000 ms
API Calls per Scroll (n) (General Dashboard)	(1 + n) Calls	1 Call

VII. Software Test and Quality Assurance

VII.I. Purpose of Testing

The purpose of the team's testing efforts is to ensure code clarity, maintainability, and functionality throughout the development of the application. Testing guarantees that incremental changes of the codebase meet both client expectations and system requirements. Through manual user-interface and regression testing, the team ensures that the previous functionality remains functional as new features are being implemented.

VII.II. Description of Testing

The team employs frequent code reviews and manual testing as part of the quality assurance process.

1. Code Reviews:
    a. Allow the client to review the team's work regularly and to provide detailed revisions as necessary for the sake of code clarity and functionality.

    b. Code reviews have taken the form of a pull request (PR), following the GitHub flow model [2], which each new change committed to a branch.
    c. PRs are reviewed by Martin Bakiev, a Qualcomm Engineer assigned to our project, ensuring code clarity, encapsulation, and maintainability. An automated test suite runs to confirm the code builds successfully in both front-end and back-end environments.
    d. This iterative process allows for continuous client feedback, promotes concise code syntax, and identifies areas needing improved documentation for future teams.

2. Manual User Interface (UI) Testing
    a. Ensure that the application behaves correctly from the end-user's perspective. Verifies that key interactions, such as filtering issues, switching tabs, and interactions with Jira issues work as intended.
    b. Each test case is executed by hand, validating UI visibility, interaction flow, data accuracy, and layout responsiveness to meet client specifications.
2. Regression Testing
    a. Ensures that previously working functionality remains intact after changes, including new feature additions or code refactoring. The team focuses on maintaining system stability across incremental updates.
    b. The following behaviors are tested:
        i. Positive behavior: Adds data to the system (e.g. filter selection)
        ii. Negative behavior: Removes data (e.g. filter deselection)
        iii. Neutral behavior: Does not alter data, ensures smooth interaction (e.g. tab switching)
    c. The suite could be automated with tools like Cypress so that it can be run on a more consistent basis.

VII.III Testing Plan

Test Name	Priority	Expected Behavior	Issue Type	Environment	Action	Expected Result	Test Type
Column Sorting: Sorting Function	Medium	Positive	User Input	LTS Dashboard: LTSList	Click a column header's sorting function button, issues should be reordered based on the column values in ascending or descending order.	Issues should be reordered based on the column values in ascending order.	UI Test
Bulk Action: Approve Tickets	High	Positive	User Input	LTS Dashboard: Bulk Actions	Select multiple tickets and perform the bulk "Approve" action.	All selected tickets should have their status updated to "Approved."	Functional Test
Bulk Action: Invalid Bulk Operation	Medium	Negative	Edge Case	LTS Dashboard: Bulk Actions	Attempt to perform a bulk action (e.g., "Deny") on tickets missing required fields.	The system should prevent the operation and display an error message for incomplete tickets.	Validation Test
Page Reload After Bulk Action	Medium	Neutral	Regression	LTS Dashboard: Bulk Actions	Perform a bulk action (e.g., Approve).	The changes from the bulk action should persist and update Jira accordingly.	Regression Test
Empty Result After Filter Application	Medium	Neutral	User Input	General Dashboard: FilterMenu	Apply filters that do not match any issues in the system.	The issue table should display a message indicating that no results were found.	UI Test
Submit Filter Button: Submit Filter Options	High	Neutral	User Input	General Dashboard: FilterMenu	The "Submit Filter" Button is Selected from the FilterMenu	When inspecting the selected filter options object, the currently selected results should appear in the corresponding returned JSON for each type.	Regression Test
Submit Filter Button: Resubmission	Medium	Neutral	Edge Case	General Dashboard: FilterMenu	The "Submit Filter" button is resubmitted, after an Initial submit, with an added & removed option	When inspecting the selected filter options object, the selected filters should be available upon the first submission but modified after. The page skeleton should appear, and the page should be reloaded with updated results.	Regression Test

Submit Filter Button: Remove Filter	Medium	Negative	User Input	General Dashboard: FilterMenu	The "Submit Filter" Button is Submitted After a Filter Removal	When Inspecting the Selected Filter Options Object, The Selected Filters Should NOT Contain the Removed Filter.	Regression Test
Grouping of LTS Issues	High	Positive	Regression	LTS Dashboard: LTSList	Verify that issues are correctly grouped.	When inspecting the labels	
IssueList: Scroll Load 30 Issues	High	Positive	User Input	General Dashboard: Issue List	Scroll to the bottom of the issue list page	30 additional issues are fetched and seamlessly loaded into the issue list.	UI Test
IssueList: Load Enough Issues to Fill 1920x1080 page	High	Positive	User Input	General Dashboard: Issue List	Load the general dashboard with a 1920x1080 screen resolution.	Enough issues are fetched and rendered to fully occupy the vertical space without empty gaps.	UI Test

VII.IV. Tools Utilized for Testing

1. GitHub & Git Bash
   a. For version control, pull request management, and code reviews.
2. Spreadsheets
   a. To track and manage test cases (can be replaced with a specialized tool like Qase).
3. Manual Testing
   a. Executed by the team, focusing on UI functionality.

VII.V. Threshold for Acceptability

1. Code Reviews
   a. All PRs must be built successfully in both front-end and back-end environments and approved by our Qualcomm contact, Martin Bakiev.
   b. Through this process, there are typically requests for additional revisions until the branch can be merged with the main branch.
3. UI & Regression Testing
   a. All critical interactions and functionalities must behave as intended, with accurate data display and consistent performance throughout the application.

VII.VI. Edge Cases

1. Massive Pull Requests
   a. The previous team submitted large PRs, leading to delayed reviews that slowed down productivity. The current team aims to avoid this by using smaller, well-scoped PRs.
2. File Structure Issues
   a. Incorrect dependency installation ~3000 unintended files to be added to a PR. This issue was identified and resolved early, preventing further delays.

VII.VII. Results of Testing

1. Improved Clarity
   a. Incremental PRs with iterative reviews have enhanced code maintainability and progress.
2. Early Error Detection
   a. Issues like dependency misplacement were caught early, avoiding further complications with continuous deployment.
3. Continuous Client Collaboration
   a. Regular feedback has resulted in better documentation and streamlined code.

VIII. Project Ethical Considerations

Since this project will be primarily used by Qualcomm engineers to improve customer service workflows, there are minimal ethical concerns associated with it that could potentially harm the public. However, it is important to consider relevant ACM/IEEE principles to ensure ethical standards are upheld during the project's development.

VIII.I. Relevant ACM Principles:

ACM 1.7 Honor confidentiality.

- This project involves Qualcomm's internal database, making it essential for all team members to maintain strict confidentiality. No confidential information about Qualcomm or this project should be shared publicly.
- This confidentiality requirement will impact on the group's ability to seek external assistance during development. For instance, team members cannot share code on public platforms like Stack Overflow to resolve errors, as all development must remain within Qualcomm's secure network.

ACM 2.4 Accept and provide appropriate professional review.

- Given the limited timeframe for this project, it is unlikely that all aspects of the application will be fully developed. To ensure the code is efficient and functions as intended, it is crucial for the team to seek professional reviews from Qualcomm software engineers as well as conduct internal peer reviews.
- Feedback from these reviews will be incorporated into the project to improve its overall quality and functionality.

ACM 3.6 Use care when modifying or retiring systems.

- Since the project aims to enhance and extend on Qualcomm's current Jira interface, the team has developed a more efficient and user-friendly design while preserving some aspects of the current Jira interface that do not require modification.
- When removing, modifying, or updating any features of the current Jira interface, the team has taken care to ensure that these changes do not disrupt the workflows of Qualcomm engineers or compromise the quality of their work.

VII.II. Relevant IEEE Principles:

IEEE 3.08. Ensure that specifications for software on which they work have been well documented satisfy the users' requirements and have the appropriate approvals.

- To facilitate future development, it is necessary to maintain comprehensive documentation. This ensures that future teams can easily understand the codebase and continue development.
- The team regularly collaborates with Qualcomm stakeholders to confirm that the implemented features align with the specified requirements.
- All changes to the project undertake client review and approval to guarantee they meet user's needs and expectations.

IEEE 7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files and other confidential information, and security measures in general.

- With access to Qualcomm's confidential files and work accounts, the team prioritizes adherence to data privacy policies to protect sensitive information and credentials.
- Team members actively support one another in protecting Qualcomm's CCI (Commercially Confidential Information), ensuring strict compliance and minimizing risks of unauthorized disclosure.

IEEE 8.02. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.

- This project provided an opportunity for the team to learn and apply new technologies, enhancing their ability to deliver high-quality software within the project timeline.
- By refining initial requirements and prioritizing key deliverables, the team ensured that project goals were met on schedule while allotting time to develop the necessary skills for producing a reliable Jira application.

IX. Project Completion Status

IX.I. Complete Features and Summary of Feature Performance

The project has successfully met most of the goals set by the client, with key features implemented and significant performance improvements achieved. One of the major accomplishments is the addition of preset filters for Long-Term Support (LTS) queries, which include filtering for the three most recent active releases. These filters are integrated into the LTS release management dashboard, enabling users to efficiently manage and organize LTS tickets. The dashboard also supports bulk modifications to LTS tickets, allowing users to perform actions such as approvals, denials, and data editing. Importantly, these bulk actions adhere to RTF formatting standards, ensuring data consistency and usability. However, it should be noted that bulk actions are currently limited to the LTS dashboard and are not currently supported on the general ticket page.

Another completed feature is the addition of warning icons on the LTS dashboard. These icons highlight tickets missing required fields, improving workflow visibility and helping users address incomplete data. Backend optimizations have also significantly improved page loading time. When the project was initially received, the general dashboard took around 8000-9000 milliseconds to load. After extensive refactoring, both the general and the LTS dashboards now load in less than 3000 milliseconds in the browser window, demonstrating a drastic improvement in responsiveness and efficiency.

In terms of quality assurance, the team has thoroughly tested the user interface to ensure that all features function as intended. The testing process assisted in identifying and resolving bugs, ensuring a seamless user experience. Basic test cases have been implemented to validate the integration between the front-end and back-end, although there is room to expand the test suite for more comprehensive test coverage.

IX.II Incomplete Features

While the project has made significant progress, some features are incomplete. The general dashboard was initially meant to set up automation for the handover and escalation workflows, but this functionality was pushed out of scope early in the project due to the lack of a feasible way to automate these processes at the time. Consequently, the dashboard currently lacks the necessary implementation to support actions related to these workflows.

If these workflows were to be implemented in the future, they would require:

1. Workflow Definition and Automation Feasibility
    a. Clearly defined business rules for handover and escalation processes.
    b. Integration with external systems to automate escalations and streamline handovers.
2. System-Level Changes
    a. Enhancements to the API and back-end logic to support creation and execution of work-flow related actions.
    b. UI updates to enable users to interact with these workflows intuitively, such as adding buttons or prompts for escalation and handover actions.
3. Testing and Validation
    a. A robust test suite that includes edge cases and complex workflows to ensure reliability.

While significant refactoring has improved the dashboard's performance and maintainability, additional development will be necessary to fully integrate these workflows. Despite these limitations, the project has delivered substantial enhancements, providing a solid foundation for potential expansion to include automated handover and escalation workflows.

X. Future Work

While significant progress was made during this semester, there are opportunities for future field session groups to enhance and expand onto this project. Below is a list of recommended future work.

1. Support for Intelligent Linking of Issues and Dependency Diagrams
    a. Description: Implement functionality to visualize issue relationships in formats such as graphs or trees.
    b. Resources: Dependency mapping libraries, React components for visualization, existing project infrastructure.
    c. Knowledge/Skills: React, JavaScript/TypeScript, state management, data visualization.
    d. Estimated Time: 2-3 weeks.
2. Single-Click Tier 2 to Tier 3 Escalation
    a. Description: Add automation for single-click escalation workflows, integrating it with the existing project.
    b. Resources: Jira REST API documentation, current project infrastructure.
    c. Knowledge/Skills: React, FastAPI, familiarity with Jira workflows.
    d. Estimated Time: 3-5 weeks.
3. Comprehensive Backend Test Cases
    a. Description: Expand the current suite of test cases to include advanced scenarios and edge cases.
    b. Resources: Testing tools (e.g. pytest), existing back-end codebase.
    c. Knowledge/Skills: FastAPI, Python, testing frameworks.
    d. Estimated Time: 2-3 weeks.
4. Issue Handover Workflow Automation
    a. Description: Automate the issue handover process between team tiers, allowing for smoother and faster ticket transitions.
    b. Resources: Jira REST API documentation, existing project infrastructure.
    c. Knowledge/Skills: React, FastAPI, familiarity with Jira workflows.
    d. Estimated Time: 3-5 weeks.
5. User Authentication (internal work)
    a. Description: Implement secure user authentication methods to integrate with Qualcomm's internal infrastructure.
    b. Resources: Qualcomm's internal authentication systems.
    c. Knowledge/Skills: Secure authentication methods, React, system integration.
    d. Estimated Time: To be handled by Qualcomm engineers.

XI. Lessons Learned

XI.I. Challenges with Development Environment

The team did all of their development within a Windows Subsystem for Linux (WSL) on a Windows 365 Cloud PC which introduced performance issues. Intermittent unresponsiveness of the development environment impacted productivity at times, as the environment did not perform as expected. However, optimizing settings, managing resources, and diagnosing issues within these tools allowed for performance improvements and workarounds for some limitations. Navigating these challenges taught the team valuable troubleshooting skills specific to WSL and cloud-based environments.

XI.II. New Technologies and Tools

At the beginning stages of the project, the team struggled to progress as we familiarized ourselves with new technologies and tools that expanded the team's technical skillset. Knowledge of full-stack development has largely improved, with tools such as React and Mantine UI for front-end development, FastAPI and httpx for back-end development, and Jira for sprint planning and task management. As we gained familiarity with these tools, our efficiency and workflow improved significantly. In the future, these experiences will allow the team to onboard new tools and frameworks more quickly.

XI.III. Time Management and Team Collaboration

Effective time management and collaboration were essential throughout the project. Using GitHub Flow as a version control strategy taught the team the best practices for coordinating work with team members, managing pull requests, and ensuring high quality code. The team learned the importance of having a clear and consistent method of communication and adhering to daily meetings and standups to maintain a cohesive workflow throughout the length of the project.

XII. Acknowledgments

The team extends its gratitude to our clients, Kevin Wolver and Martin Bakiev, along with their team, for their support and guidance throughout this project. We deeply appreciate their willingness to dedicate time to help us familiarize ourselves with the project, address challenges, and provide continued advice and feedback. Their insights were instrumental in keeping us aligned with the project's goals and overall vision. Additionally, we are grateful for the opportunity to visit Qualcomm's Boulder campus, which was both a fun and enlightening experience.

We would also like to thank our advisor, Kathleen Kelly, for her firm support and mentorship. Her guidance was critical in helping us navigate challenges, make consistent progress, and stay on track with the project's timeline. The sprint meetings were particularly helpful in enabling the team to plan effectively and meet our objectives.

Thank you all for your contributions to the success of this project.

XIII. Team Profile

Sam Bangapadang
Junior
B.S in Computer Science, Focus: Data Science
Hometown: Aurora, CO
Work Experience: Applied Math & Science IT Admin
Hobbies: Powerlifting, Video games, Cycling

Danny Nguyen
Junior
B.S in Computer Science, Focus: Data Science
Hometown: Littleton, CO
Work Experience: Undergraduate Researcher, CSCI 128 Teacher Assistant
Hobbies: Hiking, Video games, Weightlifting

Ethan Leuthauser
Junior
B.S in Computer Science, Focus: Data Science
Hometown: Commerce City, CO
Work Experience: Next Gen Transport, Inc. (Information Technology Intern)
Hobbies: Weightlifting, Music Production, Video Games

XIV. References

[1] D. Coventry et al. "CSCI 370 Final Report" Colorado School of Mines and Qualcomm, Colorado, United States, 2024. Accessed: Sept. 14, 2024. [Online]. Available: https://cs-courses.mines.edu/csci370/FS2024S/FinalReport/FinalReport_Qualcomm.pdf.

[2] "GitHub Flow." GitHub Docs. Accessed: Oct. 20, 2024. [Online]. Available: https://docs.github.com/en/get-started/using-github/github-flow.

[3] E. Anderson et al. "ACM Code of Ethics and Professional Conduct." Association for Computing Machinery. Accessed: Oct. 20, 2024. [Online]. Available: https://www.acm.org/code-of-ethics.

[4] P. Barnes et al. "Code of ethics." Institute of Electrical and Electronics Engineers Computer Society. Accessed: Oct. 20, 2024. [Online]. Available: https://www.computer.org/education/code-of-ethics.

XV. Appendix A – Key Terms

Term	**Definition**
Escalation	*Process for transferring tickets between tier 2 and tier 3 teams.*
FastAPI	*Python library for building Web APIs.*
httpx	*Python based HTTP client library.*
Jira	*Agile management tool developed by Atlassian.*
Long Term Support (LTS)	*A version of software or a product that receives extended maintenance and support over a specified time period.*
Long Term Support Analysis	*Quantifications of the Long-Term Support Process. Consists of Impact, Confidence and Portability metrics.*
Mantine/Mantine UI	*Pre-styled React component library.*
npm	*Node Package Manager.*
Rich Text Format (RTF)	*Standardized text format.*
Test Base Station (TBS)	*Qualcomm's internal custom integrated cellular call flow test box.*
Tier 2	*Bug ticket response.*
Tier 3	*Engineering team ticket response.*
Windows Subsystem for Linux	*Windows Terminal extension which allows for running Linux environments without a virtual machine.*
Windows 365 Cloud PC	*Remote Windows desktop.*