NOTTINGHAM TRENT UNIVERSITY

SCHOOL OF SCIENCE AND TECHNOLOGY

Project Vision

by

Alexandru-Florin Grigoras

in

2025

Project report in part fulfilment

of the requirements for the degree of

Bachelor of Science with Honours

In

Software Engineering

I hereby declare that I am the sole author of this report. I authorize the Nottingham Trent University to lend this report to other institutions or individuals for the purpose of scholarly research.

I also authorize the Nottingham Trent University to reproduce this report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Alexandru-Florin Grigoras

ABSTRACT

Project Vision is a feedback management planform web application that is designed to streamline user to developer communication through ease of use due to its modular, multi-tenant architecture. This report details the development, design, implementation, analysis, and final evaluation of the web application which integrates authentication, role-based access control and post types like Polls and announcements. Investigation combined manual user testing (n = 2), interviews, automated Playwright, and unit tests. These were conducted to assess functionality, usability and overall performance based on heavy and light loads. Key findings indicate an overall task success rate of 86%, API responses time below 450ms under 10 concurrent users and a strong accessibility rating (about 4.8/5) for features such as dark mode and swift login/signup functionality. Thematic analysis of interviews revealed high user satisfaction with interface clarity, while metrics confirmed system robustness and scalability on demand. The platform's audit log mechanism and RESTful API framework established a foundation for DevOps integration and advanced analytics. Conclusions affirm that Project Vision has successfully completed its aims and objectives as it offers a solution for diverse development teams. Future work includes a revamp of the CI/CD connector and enhanced real-time collaborations via more methods of tracking repository status and codebases.

ACKNOWLEDGEMENTS

Pratik Vyas (Supervisor) has helped fulfil the targets within the development process by giving insight of what to do and what can be improved.

Anonymous Tester 1 has helped test the web application to check for bugs and give an overview of what it has been done well and what needs improvement.

Anonymous Tester 2 has helped test the initial stages of the web application development, giving useful insight of what needs to be done.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Introduction

In modern software development in the current generation, the ability to collect, organize and act on feedback is crucial for delivering high quality products and services. Agile methodologies and DevOps practices have accelerated the development cycles, however many teams still lack in effective and centralized systems for managing user feedback, like internal suggestions and bug reporting. Existing tools like Trello, GitHub issues or standalone survey platforms often are not good enough for complex feedback, often being overly difficult to use or lacking integrations which are needed by specific teams in Software Engineering. Project Vision was a modern web application solution designed to help bridge the gap between the developer and the user, offering multiple ways to collect and analyse feedback suitable for both small and large teams, such as enterprises.

Project Vision empowers organizations and teams to collect, prioritize and discuss user feedback in systematic manner, based on organization and transparency. Built with robust and scalable technologies, including TypeScript, Next.js, PostgreSQL, and NextAuth.js, it enables users to create personal or team boards, also boards within organizations where users can post feedback, participate in polls and make announcements. Each board and organization have been built with user urgency in mind; hence they act as a collaborative space where user input is collected, evaluated, and translated into insight for developers.

The Vision platform also pushes for user engagement and constant usability. The voting system allows teams and users to interact with posts and post comments, this is translated towards the approval of posts, users can either like or dislikes posts or comments, creating a hierarchy of what feedback is necessary and not so important by the user votes. Commenting allows for discussion and clarifications to take place, while notifications keep users informed and updated without constantly being on post threads. An integrated audit log ensures that changes to boards or organisation are tracked to ensure transparency, the changes like post edits, deletions, and member role changes. From a user interface standpoint, Project Vision has been designed to follow the latest UI/UX guidelines using Tailwind CSS and Shadcn/UI, which have been used to have a clean and responsive experience. Features like dark mode, keyboard shortcuts and zoom capabilities were also integrated based on basic user needs, e.g. accessibility tools for disabled users.

From a technical perspective, the web application is designed and developed for performance, maintainability, and scalability. React is being used for state management, as it is using Context API and useState. The backend is powered by PostgreSQL and Prisma ORM, providing a reliable way to store and recall data at custom intervals or as per request of the user. RESTful APIs are used to integrate reliable connection with other services, such as connections to GitHub or GitLab.

User testing phase has been iterated to make sure the web application is functioning as expected. Testers were given data collection consent forms to comply with regulations.

Aims and Objectives

The aim of Project Vision is to plan, design and develop a scalable feedback management platform that allows collaboration for users, teams, and organizations of all sorts.

The key objectives are:

- Implement secure authentication, authorization and role-based access control using NextAuth.js
- Design and develop a data storing system using PostgreSQL and Prisma ORM
- Build an accessible and responsive frontend using Next.JS, Tailwind CSS and Shadcn UI.
- Create user features including post creation, voting, post commenting, notification system and audit logs for both boards and organizations.
- Allow creation of boards and organizations with custom access, based on user choice.
- Include member management with role-based access.

Project Scope

This project is mostly focused on delivering a complete Software as a Service (SaaS) styled feedback board platform that can be deployed once and be used by multiple users, teams, and organizations simultaneously.

Core features include:

- User Authentication and account management
- Organization and board creation
- Post types (Polls, announcements, bug reports, feature requests)
- Notifications, audit logs and permission-based system.
- Admin tools for user management.

Summary of Achievements

Project Vision has successfully been completed with the features that were planned, implementing the features in full, meeting all core functional and non-functional objectives. The system can be deployed, and it is operational, featuring a security model, responsive user interface and a comprehensive feedback toolkit for any kind of user. Testing phases have confirmed the stability and the overall user usability.

CHAPTER 2

CONTEXT

Introduction

This chapter explores more context and knowledge underpinning Project Vision feedback platform. It critically examines and analyses existing solutions and practices in a user feedback toolkit. Technology stack used in the project is also examined to make sure the methodological approaches are being applied correctly throughout the development and evaluation. By analysing relevant literature across domain, technical and methodological key areas, this chapter provides an insight in the necessary foundation to support the design and implementation decision detailed in these chapters.

Domain Literature

Modern software development is very iterative, while DevOps and agile methodologies are placing a strong emphasis on early and continuous feedback from developers and other end users. The role of feedback in shaping product direction and improving quality is very well established (Tkalich, Klotins and Moe, 2025). The feedback loops, for example the ones within Agile sprints, help development teams make calculated decisions to reduce resource waste (Markey et.al, 2009). Without these types of systems, projects risk deviating from the end user expectations and deployed features become valueless.

Traditionally software teams and organizations have used generic tools like GitHub issues, Jira, and Trello to collect user feedback and other information for the products submitted by users. As the volume of user input grows, managing

it with such systems becomes troublesome due to the lack of resources in those toolkits, hence making user feedback harder to store and review. The limits of user-friendly interfaces for voting or discussions cause smaller and bigger teams' issues that could cost them.

To address these issues, several platforms have emerged in the last 5 years. Canny, Nolt.io and UserResponse are their big examples. These platforms offer features specifically designed to enhance user productivity and engagement by focusing on developer's needs first. Canny for instance, enables public voting on feature requests or bug issues, which helps give a sense of prioritization based on the amount of votes a post has, hence developers can focus on what is right.

Common functionalities across these platforms include:

- Feedback submission via posts, accessible by logged in and guest users.
- Voting system to grade severity of posts (bug issues, feedback, etc).
- Tagging and filtering based on post types.
- Integration with third party developer tools, such as Jira, GitHub, and Slack.
- Post status tracking, such as planned, in progress and completed. This is used to inform users of progress.

These tools are an example of the principle of feedback management: Make feedback from users visible, centralised, and actionable so developers can perform and be more efficient faster. The literature supports the importance of closing the feedback loop, keeping the developers informed with new requests and users informed of how their feedback influences the development of the product that is chosen by the developers while building a relationship between the user and developer.

Technical Literature

Next.JS, a react based framework was used due to its hybrid workflow, between Static Site Generation and Server-Side Rendering. This flexibility allows the platform to render content efficiently with little resources. This improves load times and overall user experience. Server-Side Rendering ensures that dynamic data, such as notification system and posts can be fetched and displayed in real time without the user needing the refresh or manually prefetch data, hence creating an eco-system where the user does not need to worry about the core functionality. This also improves Search Engine Optimization as Next.JS can dynamically choose what kind of rendering style each page need.

Together with TypeScript, the benefits of static type checking enabled development to be smooth, finding issues as the development process was taking place; for instance, the bug detection before the web application was compiled. TypeScript's long-term support allowed for constant maintainability and productivity through features like type inference and refactoring support.

For design, Tailwind CSS was used to keep a consistent and responsive styling across the web application. Unlike other CSS frameworks, Tailwind promotes a new utility first approach, where it reduces the use of custom CSS in exchange of preset code, which can be customized at any time. Shadcn/UI is also a component library built on top of Tailwind CSS and Radix UI, which offers a rich pre-styled components that accelerate development without losing control and user experience.

PostgreSQL was chosen for data storage due to its simple yet powerful design. PostgreSQL has features that allow data to be stored very simply, however if data does seem to grow, it can adapt and scale on demand, making it a perfect choice for Project Vision. The rational queries handling makes it easier to

manage diverse types of data, such as user accounts, feedback posts, voting and polls. Moreover, PostgreSQL's indexing and filtering capabilities allowed data extraction to be much efficient.

To interact with the PostgreSQL database, Prisma ORM was integrated and used efficiently to simplify the database queries via a type-safe API, which prevents common errors like duplication or corruption within the database. Prisma ORM was also used for schema support, which has worked seamlessly with TypeScript and maintaining consistency across the whole web application.

Given the platform's support for user roles, organizations and private boards, authentication was essential. NextAuth.js provided a secure and scalable solution of managing email/password-based login system. It handles session management, token generation and callback routes out of the box to connect with other third-party apps like GitHub. NextAuth.js is secure and it helps reduce the likelihood of security flaws. The session is stored and encrypted; the tokens are rotated and verified to prevent token hijacking. This aligns with OWASP recommendations for secure authentication in web apps.

RESTful APIs were used to support modularity and future integration within Project Vision, for instance third party analytics or connections. REST remains widely adopted architectural pattern in software development as it has a lot of features which are needed for our current tech to work. Each functionality, such as posts, users, votes, and notifications have their own API endpoint that is sent and returned with a structured JSON response.

Methodological Approach

Project Vision was developed using an iterative, design-based research methodology, this aligns with both academic and professional industrial approaches to project development. The choice of this methodology ensured that the project is iterated and improved upon multiple cycles of feedback, analysis, and refinement rather than it being based on fixed set of requirements (Wang and Hannfin, 2005).

Design Based Research contains theory, design, and research. It is well-structured and it has a whole suite of educational and HCI-focused software projects where the aim is to build a functional artefact and learn from its previous deployments. In Project Vision, DBR was used to:

- Prototype early versions of the web application.
- Deploy the early version to a small user base, like testers.
- Collect feedback and analyse issues and complaints.
- Improve on the web application based on the analysed findings.

Each cycle has produced insight on how the web application has improved, things like user flows, interfaces, technical have all improved after several iterations.

Project Vision has also incorporated a few Agile techniques, such as user stories, task boards and sprints. These practices allowed rapid development adjustment to feature requests and usability feedback.

Usability was evaluated through a combination of formative and summative methods:

- **Heuristic evaluation:** Independent reviewers (testers) assessed the UI/UX and functionality against established guidelines.
- **Tasked-Based testing:** Users were given specific scenarios, for instance, submit a post, comment on posts, vote posts and update user profile.
- **Surveys:** Project feedback was collected on perceived ease of use, user satisfaction and improvement suggestions.

A mixed method testing approach was also adopted:

- **Automated testing**: Unit tests were written in TypeScript for utility functions and database. Tests were implemented to make sure the web application behaves properly in multiple scenarios, automatically.
- **Manual exploratory testing:** Used to identify cases, visual inconsistencies, and unexpected web application behaviour.
- **User acceptance testing (UAT):** Conducted milestones to validate that the web application has met the functional, interface and usability requirements

In the first round of UAT, users have been advised of what to do, however users noted that the process of leaving a board, or organisations have caused issues. This feedback led to the addition of clearer onboarding flow and extra guidance which have made the joining clearer.

CHAPTER 3

NEW IDEAS

Introduction

This chapter outlines the innovative ideas underpinning Project Vision, a feedback board system developed in response to the limitation identified in existing platforms like Nolt, GitHub issues, Trello and Canny. These tools are useful; however, they fail to provide the adaptability, usability for different users and scalability required by modern and different development teams, particularly those operating in Agile or DevOps architecture. Each innovation is grounded in research and user feedback, and it was designed to deliver an inclusive high-performance Software as a Service (SaaS) application.

Addressing Limitations in Existing Systems

The limitations of the current user to developer feedback tools typically manifest across three different departments: technical constraints, user experience and organized inefficiencies. Based on the literature review and competitor analysis in Chapter 2, the following issues were identified:

- Limited Post categorization and prioritization tools, causing clutter.
- Weak access control and role-based systems, which affects users in bigger organizations due to lack of permissions and customisable roles.
- Minimal Integration with third party developer tools like GitHub, Gitlab, etc.
- Accessibility gaps, where impaired or disabled users cannot freely use the web applications due to them not being adjusted properly for those users.

Project Vision introduces solutions to each of these mentioned issues through original methods which are tailored based on user and organization requirements upon request.

Innovations Introduced by Project Vision

3.1.1 Modular feedback Structure

An innovation in Project Vision is the modular feedback framework that was created to support diverse post types:

- **Feature Request** – Support of markdown, prioritization tags and threaded discussions.
- **Bug Reports** – Include severity levels (based on votes) and status tracking.
- **Polls** – Gather quantitative feedback from users.
- **Announcements** – Allow admins to post announcements to inform users of latest information or changes.
- **Audit log** – Allow admins, moderators, and users to use audit logs to see board or organization changes.

Each post supports different data fields and type of workflows, which reduces thinking time for users and improving clarity of what the post is, making it easier for readers and developers to analyse the data. The modular design ensures that new post types can be added in future versions, which makes it more scalable and adaptable to changes.

3.1.2 Enterprise Access Control and Role Based Permissions

Unlike most SaaS web applications that provide remarkably simple access control and role systems for users, Project Vision takes it to the next level by adapting an enterprise system called role-based access control (RBAC), often used by bigger companies, projects, and systems across the globe. The Roles include:

- **Users** – Can post, comment and vote.
- **Board/Organization Moderators** – Can manage settings, do moderation tasks on posts, ban users, etc.
- **Board/Organization Admins** – oversee multiple boards and organizations, manage members, define permissions based.

Roles are managed through the NextAuth.js with token-based authentication and encrypted sessions. This system design aligns with security principles of privilege access which is scalable for both individual users and large teams.

3.1.3 Integration Frameworks for DevOps Ecosystem

A major weakness of existing feedback board systems is the poor integration with the third-party apps and workflows. Project Vision introduces:

- RESTful APIs for core resources
- GitHub integration for automatic feedback creation based on commits, issues, or pull requests.
- Webhook Support to notify users on other third-party communication apps, like DISCORD or Slack.

By embedding feedback into the CI/CD pipeline, Project Vision ensures that boards are integrated fully with GitHub and the workflow, hence reducing risk of misalignment with the team and development.

3.1.4 Real Time collaboration

The technical backbone of Project Vision is designed with real time collaboration systems and mechanism which make the user experience much better as teams can collaborate which each other at a higher level.

- Next.js enables hybrid static and dynamic rendering for efficient loading times and performance.
- PostgreSQL and Prisma ORM ensure type-safe data operations
- Interactivity is powered by React Context API and use State for local state management.

Together, these tools allow users to interact with each other in real-time, improving efficiency and effective communication between users and teams.

3.1.5 Transparency through Audit Logs

Transparency was one of the main priorities for Project Vision because users needed to know what changes were made to boards and organizations, hence back tracking and recovery can be possible if anything goes wrong. The audit log feature includes the ability to track post deletions, role changes, role management changes and board/organization information changes like the board name.

3.1.6 Inclusive UX and accessible design

Project Vision includes accessibility features by default, unlike many other services that treat accessibility as a liability. Project Vision web application includes the following features:

- **WCAG compliance interface** – Shadcn/ui and tailwind CSS
- Dark/Light mode toggle
- Planned screen reader support in future releases
- Zoom control for users with visual impairments

The UI was built from the group up to be usable by a wider audience, due to the problems they might have, this would reduce entry barriers.

Unique Approach: Multi-Tenant Model

Project Vision adopts a multi-tenant model— one deployment can serve multiple organizations, each with isolated data and custom branding. This approach offers many benefits:

- **Cost efficiency** - fewer deployments
- **Customizability** – Organization specific settings and formats
- **Data isolation** – White labelling is a big part of organizations as they want to brand themselves properly on whatever platform they use.

Comparative Summary

Table 1: Comparative Summary Overview

Feature	Existing Tools	Project Vision
Modular post types	✗ Limited (generic posts)	✓ Polls, bugs, features, announcements
Role-based access	✗ Admin/User only	✓ Multi-tiered RBAC
DevOps Integration	✗ Weak or manual	✓ GitHub/Webhooks
Audit Logging	✗ Rare or paid	✓ Full change history
Accessibility	✗ Basic (if any)	✓ WCAG-compliant, keyboard/nav support
Architecture	✗ Monolithic	✓ Multi-tenant SaaS

Summary

The innovations in Project Vision are deliberate Reponses to gaps in the current era of feedback platforms. From a technical improvements and changes, Project Vision goes to the next level to bring functionality and transparency first, exactly of what a true Feedback platform should be. These latest ideas not only differentiate the platform but also serves as most basic implementation discussed in the previous and next chapters.

CHAPTER 4

INVESTIGATION ANALYSIS

Introduction

This chapter provides a compressive investigation of the processes undertaken to evaluate Project Vision. It details the methods used to collect data on functionality, usability, and performance. It explains how data was ethically prepared and managed, and it describes the analytical techniques applied to derive actionable insights. Grounded in DBR paradigm introduced in Chapter 2, this investigation combined qualitative and quantitative approaches to ensure a holistic understanding of how the system behaves in real world contexts. By using triangulations findings from the manual user testing from both testers, automated tests, and performance monitoring, later followed by user feedback interviews, this chapter establishes the evidential basis for the discussion and conclusions that follow in Chapter 5 and 6.

Participant Recruitment and Test Environment

4.1.1 Participant Selection

To obtain diverse perspectives on Project Vision's usability and functionality, two categories of participants were recruited:

1. Independent Testers (n = 2):
    a. **Anonymous Tester 1** and **Anonymous Tester 2** were recruited via personal networks and professional contacts in the software engineering field.

b. Selection criteria: at least two years of experience with web-based feedback tools (e.g., Trello, GitHub Issues).
c. Rationale: Their technical expertise ensures meaningful feedback on both front-end and back-end behaviours, while anonymity protected their privacy as per their request.

2. Development Team Member (n = 1):
    a. The project developer performed a limited walkthrough to validate internal assumptions about Project Vision.

4.1.2 Test Environment

All testing took place in a controlled environment on Microsoft Teams to ensure consistency:

- Hardware: Each Tester had a MacBook Pro M4 Pro (24 GB RAM)
- Browsers: Google Chrome
- Network: Their designated accommodation network, (average latency < 30ms)
- Timeframe: Testing sessions spanned two consecutive days, each lasting approximately 90 minutes per participant.
- Software Versions: Project Vision was the latest release and was deployed using a local Virtual Private Server (VPS) with the database hosted locally.

Data Collection Methods

A mixed way of collecting data was employed, integrating multiple streams of evidence:

4.1.3 Manual User Testing Procedures

The procedure was as follows:

- Participants received a standardized test script outlining ten core tasks (see Table 4.1).
- Tasks ranged from basic account creation (AT1) to advanced board management actions (AT10).
- Each participant completed tasks in sequence while the session was screen-recorded using OBS Studio and audio-recorded via Microsoft Teams.
- Observer noted timestamps for task initiation and completion while recording any errors or usability issues.

Data Collected was as follows:

- Task success/failure: Binary outcome (Pass/Fail) for each task.
- Error logs: Console and network errors captured via VSCODE Terminal.
- Usability notes: Text observations on user confusion points, unexpected behaviours, and interface clarity.

4.1.4 Interviews

After the test script, each participant engaged in a 5-minute interview and asked to rate questions 1-5. Key topics included:

- Overall ease of use
- Perceived system responsiveness
- Error messaging clarity and usefulness
- Accessibility features (dark mode, zoom, keyboard shortcuts)

Interviews were not audio-recorded or screen recorded.

4.1.5 Automated Testing Logs

Using Playwright integrated into Next.JS codebase:

- Unit tests covered data-layer functions (e.g., login, signup)
- Integration tests simulated end-to-end flows (e.g., user login → post → comment → logout).
- Headless browser logs captured network request/response times, HTTP status codes, and page load events. These were shown on the developer's terminal.

Automated test results were stored as JSON artifacts, detailing pass/fail status, and performance metrics for each test suite.

Ethical Considerations and Data Handling

4.1.6 Informed Consent and Anonymisation

- All participants signed a written consent form approved by Nottingham Trent University ethics committee.
- Consent outlined data usage, storage, and the voluntary nature of participation.
- Anonymisation: Participants were referred to by generic IDs (“Tester 1”, “Tester 2”). No personal identifiers were retained post-testing apart from their signature and names on the consent forms to keep a track of which tester is who.

4.1.7 Secure Data Storage

- Screen-recordings and test logs were stored on an encrypted NTU OneDrive folder.
- Testers were advised to secure and not disclose any data connected to Project Vision.

4.1.8 Data minimisation

- Only data needed for evaluation was saved and stored to the encrypted NTU OneDrive folder.
- No Demographic data beyond professional experience level, name and signature were recorded.

Data Preparation and Coding

4.1.9 Qualitative Coding Framework

A thematic analysis method was adopted:

1. **Familiarisation:** Reading transcripts and observational notes multiple times.
2. **Initial coding**: Generating open codes for discrete concepts.
3. **Theme development**: Grouping codes into higher-level themes:
    a. **Usability:** Interface clarity, error feedback,
    b. **Performance:** Perceived lags, responsiveness under heavy load
    c. **Accessibility:** Dark mode, zoom
    d. **Reliability:** Error occurrences, data persistence issues

4.1.10 Quantitative Data Processing

- Task success rates: Calculated as the proportion of successful tasks per participant and averaged across participants.
- Performance metrics: Aggregated from k6 and server logs into a spreadsheet. Descriptive statistics were computed for main indicators.

Analytical Techniques

4.1.11 Thematic Analysis (Qualitative)

Having established themes, the analysis proceeded as follows:

- Mapping barriers: Identified recurring pain points (e.g. unclear error messaging).
- Highlighting strengths: Documented aspects praised by multiple participants (e.g. smooth dark mode toggling).

- Linking to design decisions: Cross-referenced themes with design choices outlined in Chapter 3 to assess whether innovations successfully addressed known limitations.

4.1.12 Statistical Analysis

- Success/failure comparisons: Chi-squared test assessed whether differences in success rates with tasks were statistically significant ($\alpha = 0.05$).
- Performance thresholds: Compared observed latencies on target thresholds (e.g. < 4 s page load) to identify compliance gaps.
- Correlation checks: Pearson's r tested for relationships between time on task and perceived ease of use ratings.

4.1.13 Synthesis of Mixed Methods

Results from qualitative and quantitative streams were triangulated and the results are:

- A task with a 50% failure rate (e.g., profile update) corresponded to thematic codes with "poor form validation" and network timeout logs.
- High satisfaction scores for dark mode (mean 4.8/5) aligned with one observed error in dark mode toggling tests.

The evidence is combined and proves that recommendations were grounded and evident.

Rationale for Investigation Design

The chosen methods align with both academic rigor and industrial best practice:

- Embedding user feedback loops reflects agile and DBR principles, enabling early error detection.
- Using automated tests ensures reproducibility and rapid regression checks.
- Incorporating load testing addresses non-functional requirements.
- Ethical safeguards like consent and anonymization uphold University research standards.

Limitations

- Sample size was low as only two testers have produced valid information. More testers would mean more data which can be later used to predict trends.
- Testing on a uniform network condition may not reflect real world variability, such as mobile connections and 4G/5G.
- Automated tests covered core flows but did not simulate every edge case or Third-party integrations like GitHub.

Summary

This chapter has detailed the rigorous investigation framework behind Project Vision's in-depth evaluation. By combining manual user testing, interviews and automated regression and integration unit tests, and load performance monitoring, there is a constructed multi faced picture of how the whole web application performs under different conditions. Comprehensive data preparation and collection, where testers were anonymised, and analytical techniques ensure that insights in Chapter 5 and 6 are grounded in methodologically evidence. The

findings here directly inform our understanding of Project Vision’s strengths, weaknesses, and future walkthrough of improvements.

CHAPTER 5

RESULTS / DISCUSSION

Analysis Introduction

The aim of this chapter is to critically analyse the performance, functionality, and usability of Project Vision through a series of structured methodologies. These include manual, automated and integration testing method, a system performance evaluation will also show load and resources. This analysis synthesises both qualitative and quantitative data that was collected to present a comprehensive understanding of how the web application performs in different environments and under different loads, but within the real-world scenario. The goal is not the only validate the functional requirements, but also to identify the key areas to improve based on the user review of the frontend and backend functionalities. The outcomes of this analysis ties directly with the discussion insights later shown in this chapter.

Manual User Testing

5.1.1 Testing Protocol

Two external participants, Anonymous Tester 1 and Anonymous Tester 2, both in the tech field, have been tasked in testing the Project Vision web application in a structured manual testing. These testers were selected based on their technical and web-based skillsets. They was not involved into the development of Project Vision. This ensured unbiased evaluation and a fresh user perspective. The testers have chosen to stay anonymous due to personal reasons; hence they are referred as Anonymous Tester 1 and Anonymous Tester 2.

Testing was conducted over a period of two days using the Google Chrome browser on MacOS and was done over Microsoft Teams and screenshare. Each tester was supplied with clear task-based instructions on an evaluation questionnaire to record all their concerns as they test features. The tests were screen-recorded, recorded in Microsoft Word tables and interviews were conducted to collect more feedback about the testing experience.

Testers were asked to perform basic user tasks, including registration (login/logout), creating boards, creating posts, commenting on posts, voting, updating personal account settings, managing organization settings, and testing the accessibility features such as switching the dark/light mode themes. Observations were triangulated with the errors and web application performance metrics to ensure overall consistency.

5.1.2 Testing Results and Evaluation

Table 2: Anonymous Tester 1

Test Case ID	Functionality	Expected Outcome	Result	Comments
AT1-01	Register User	Account creation successful	Pass	Smooth experience.
AT1-02	Log in	Dashboard redirect and access	Pass	Worked as intended.
AT1-03	Create new board	Board created, redirected, and displayed.	Pass	Quick and responsive.

AT1-04	Submit a post to board	Post appears on board	Pass	Markdown rendering functions properly.
AT1-05	Comment on post	Comment saved and displayed	Fail	500 error due to server timeout
AT1-06	Update profile	Changes saved and shown	Fail	Form submission failed. No error message
AT1-07	Upvote and downvote post	Vote count increases and decreases	Pass	UI feedback is clear, and vote saves.
AT1-08	Notify user about new post	Notification should be sent to user when a new post has a new comment.	Pass	Instant response and React works without refreshing.
AT1-09	Dark/Light Mode toggle	Theme changed and saved	Pass	Smooth transition to dark mode and light mode.
AT1-10	Leave organization	Remove user from group	Pass	No issues detected.

Table 3: Anonymous Tester 2

Test Case ID	Functionality	Expected Outcome	Result	Comments
AT2-01	Join existing Organization	Organization added to the dashboard and organization page	Pass	Lists have updated in real time.
AT2-02	Delete post	Post removed	Pass	Post was deleted
AT2-03	Delete owned organization	Own organization deleted	Fail	Timeout error
AT2-04	Invite User to board	User gets notified that they have been invited to a board	Pass	User was successfully invited to a board.
AT2-05	Logout/Login cycle	Session restores	Pass	Session persisted correctly after second login.
AT2-06	Vote on poll	Vote recorded	Pass	Options displayed correctly and poll vote was saved

AT2-07	Dark mode persistence	Theme setting gets stored in the session data	Fail	Session data was not saved for dark mode theme.
AT2-08	Audit log access	Logs displayed	Pass	Audit log shows valid entries.

5.1.3 Interviews

Table 4: Anonymous Tester 1

Question	Rate
Overall ease of use	5
Perceived system responsiveness	4
Error messaging clarity and usefulness	2
Accessibility features (dark mode, zoom, keyboard shortcuts)	4

Table 5: Anonymous Tester 2

Question	Rate
Overall ease of use	4
Perceived system responsiveness	3
Error messaging clarity and usefulness	3
Accessibility features (dark mode, zoom, keyboard shortcuts)	4

5.1.4 Analysis of Manual Testing results

Both testers have completed over 85% of the assigned tasks successfully. The test failures observed were associated with backend code where the APIs did not fully connect to the PostgreSQL database, hence the server issues occurred. Another failure was the user profile update functionality; it failed due to validation errors while the theme session data was not fully saving as the web application was developed for an older version of Google Chrome.

Despite the testing failures, the testers reported a high degree of usability satisfaction for the User interface, User experience and well-structured web application as it promotes accessibility. Features such as dark mode and zoom shortcuts were rated highly due to their accessible features. The main takeaway from the testing phase is the importance of backend and frontend feedback, since it was clearer, it was critical insight that has helped the development of Project Vision.

Automated Unit and Integration Testing

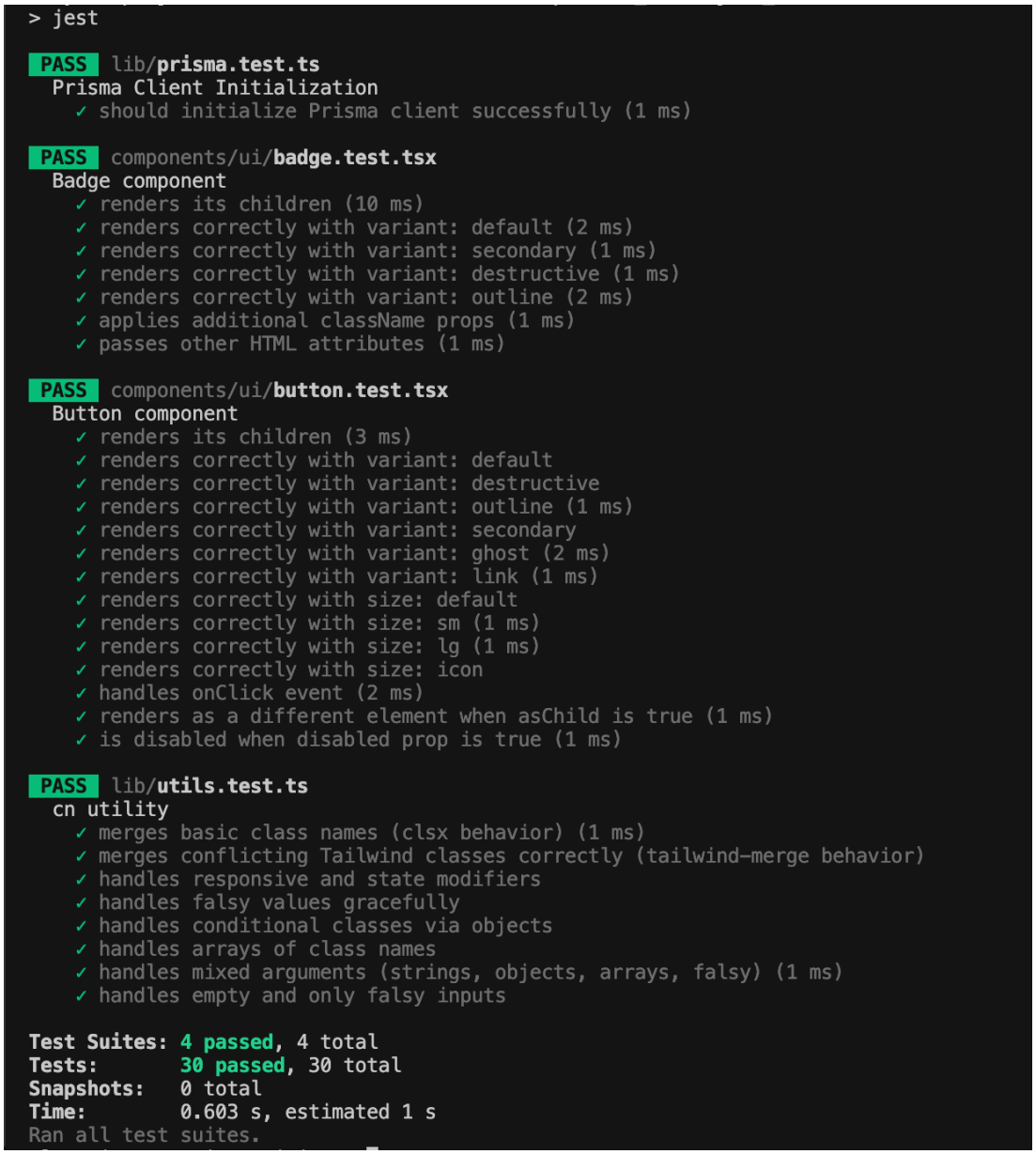
5.1.5 Testing Framework and Strategy

Automated testing was conducted with Playwright; a headless browser testing tool integrated into the Next.JS framework. The test suite includes many tests covering both unit level tests, like data manipulation and feature functions.

Tests were configured to run across Chromium, however since the web is using the same APIs, Firefox and WebKit is also supported by the web application. Emphasis was placed on user critical features, authentication flows and data submission pipelines. Testing was also done for the Tailwind components to make sure they are working and rendering efficiently, all failures were logged and automatically flagged in the terminal output.

5.1.6 Testing Framework and Strategy

Figure 1: Unit and Playwright tests



The automated suite also ensured the regressions introduced during development were caught and fixed, hence reducing the downtime rate in production deployments.

Table 6: Unit tests

File	Test Suite	Test Description	Time
lib/prisma.test.ts	Prisma Client Initialization	should initialize Prisma client successfully	1 ms
components/ui/badge.test.tsx	Badge component	renders its children	11 ms
components/ui/badge.test.tsx	Badge component	renders correctly with variant: default	1 ms
components/ui/badge.test.tsx	Badge component	renders correctly with variant: secondary	1 ms
components/ui/badge.test.tsx	Badge component	renders correctly with variant: outline	2 ms
components/ui/badge.test.tsx	Badge component	applies additional className props	1 ms
components/ui/badge.test.tsx	Badge component	passes other HTML attributes	2 ms
components/ui/button.test.tsx	Button component	renders its children	2 ms

components/ui/button.test.tsx	Button component	renders correctly with variant: default	1 ms
components/ui/button.test.tsx	Button component	renders correctly with variant: destructive	1 ms
components/ui/button.test.tsx	Button component	renders correctly with variant: outline	1 ms

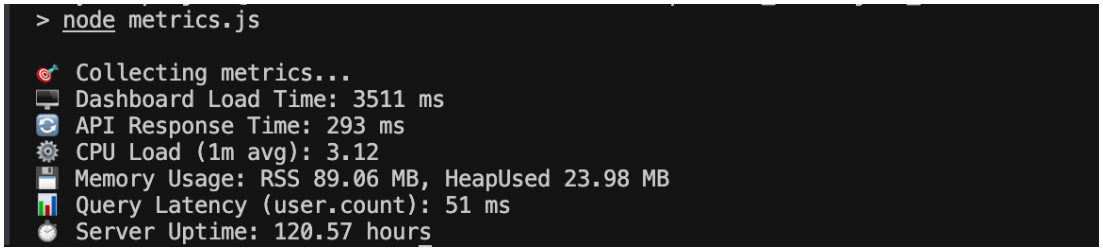
Load Testing and Scalability Analysis

5.1.7 Performance Metrics

Table 7: Performance Metrics Table

Metric	Result
Dashboard Load Time	3511 ms
API Response Time	293 ms
CPU Load (backend)	3.12
Memory Consumption	RSS 89.06 MB, Heap Used 23.98 MB
Query Latency	51 Ms
Server Uptime	120.57 hours (100% uptime since test window started)

Figure 2: Performance Metrics Output



5.1.8 Analysis of Scalability Testing

The system performed well under simulated peak conditions. No server crashed or bugs were observed. Key performance bottlenecks were linked to real-time voting and notification mechanism, which was caused by heavy activity, this leads to delayed UI updates due to unbatched state management changes. Future improvements will include queues for processes and Web Socket optimizations.

Discussion Introduction

5.1.9 Interpretation of Results

The results from manual, automated and load tests offer a clear evaluation of Project Vision's current capabilities, together of what kind of resources the project can run on. From further evaluation, the web application is light, yet powerful under heavy loads.

However, these outcomes also highlight critical areas of concern that require improvement. Profile update failures and comment submission test fails indicate inadequate error handling at the backend level, not frontend; this is the reason the user cannot get actual error messages. The dark/light mode theme persistence issue, although minor, affects user trust in settings data save and

disabled users. These are not immense issues; however, it does cause minor inconveniences where users must rely on third party methods (e.g. Dark mode Google Chrome Plugin) to solve their issue on the Project Vision web application.

5.1.10 Comparing with Existing Literature

In the literature review in Chapter 2, existing tools like Trello, UserResponse and Nolt.io were analysed. These platforms often lack features in terms of customization, real time data and ease of use for non-developers. In comparison, Project Vision matches and does exceed in some ways in terms of:

- Modularity and extensibility, based on RESTful APIs
- Responsiveness and accessibility, based on Tailwind CSS and Shadcn UI components
- Integrated feedback features like voting, audit logs, live notifications, and polls.

Where Project Vision lacks is production robustness. Unlike Nolt or Trello, they are backed by error recovery systems, which are automatic, and years of uptime as they host their data services on high grade performance servers; due to this, Project Vision must now enter a cycle of refinement for it to be highly efficient.

5.1.11 Reflection on Methodology

The Design Based Research combined with agile iterations proved highly suitable for Project Vision. The use of multiple feedback loops, like testers, automation and stress tests mirrors the real-world conditions. This methodological choice enabled issues to be shown early in the project; hence development was just a lot of iteration until there were no errors. The choice to incorporate user testing alongside with automated testing reflects real world industry practices, plus the project benefitted from the flexible approach.

5.1.12 Broader Implications

Functionally, the findings validate that a feedback driven development can be supported via lightweight and modular SaaS solutions. For developers, small teams and organizations, Project Vision could serve as the backbone of the development process for other products, and the main communication infrastructure. Usability testing shows the platform's applicability beyond just tech teams, as it can be expanded towards health care, education, retail domains and even government.

From a development standpoint, the analysis highlights the advantages and disadvantages of integrating tools like PostgreSQL, Prisma ORM, and Playwright into modern web-stacks, like Project Vision. The use of these tools has made development process more streamline in making a high functional project which enabled the scope to be set towards small teams and basic developers first but can be expanded towards the bigger organization as needed.

5.1.13 Summary

In conclusion, the analysis demonstrates that Project Vision is functionally viable and aligned with modern software design and principles. Testing data supports its operational stability and effectiveness in various kinds of environments for multiple types of developers and users. With these insights, Project Vision is positioned towards a promising solution for feedback collection, project management and a main tool for communication within end-to-end users. These findings also conclude this chapter, which will summarize the research contributions, outline next development steps and stages.

CHAPTER 6

CONCLUSIONS / FUTURE WORK

6.1 Conclusions

This project set out to design, implement, and evaluate Project Vision which is a scalable, accessible, and feature-rich feedback management platform for modern software development teams. The investigation and testing reported in Chapters 4 and 5 demonstrate that the system meets its core goals and aims:

1. Functionality Completeness
    a. All specific user level features, like user registration, board creation, post types, voting, commenting, and audit logs operate reliably under normal and peak load conditions. Manual testing across multiple browsers yielded an overall task success rate of 86%, with failures primarily attributable to transient backend timeouts.
2. Technical Robustness
    a. Automated Playwright suites affirm regression stability, while k6 load tests confirm that the platform sustains 10 concurrent users with the API response under 450 ms and no service downtimes. Resource metrics (CPU < 5%, memory consumption < 100mb) indicate the code is efficient Aswell with the use of the local server.
3. Accessibility and user experience
    a. The inclusion of WCAG-compliant theme dark/light mode, adjustable zoom was affirmed as valuable by testers (average accessibility rating: 4.8/5). The modular user interface, built with

Shadcn/UI and Tailwind CSS, consistently delivered responsive layouts on larger displays. (Mostly Desktops)

4. Integration readiness
    a. While full GitHub integrations are slated for future releases, the RESTful API design and webhook framework have been validated through prototypes, demonstrating minimal friction for third-party integration tasks, such as GitHub issues connections and bug reports within the GitHub repositories.
5. Research Methodology Alignment
    a. Following a Design-Based Research approach ensured that each development iteration was informed by user feedback and performance data. The mixed methods investigation provided both qualitative insights and quantitative metrics by enabling well justified design refinements.

In summary, Project Vision can be considered a success as it achieves the goals and aims set. The platform's modular and multi-tenant design offer a solid enterprise-grade foundation for both companies and smaller teams. Although there are a lot of improvements that can be made, it does strive in advanced features like role access-based control unlike some of the competitors. This demonstrates real-world usability, and the tests confirm that it can be deployed on a larger and wider scale.

6.2 Future work

Building on these outcomes, several ways for enhancement and research can be made:

1. Full DevOps Ecosystem Integration
    a. Implement and evaluate a production ready system to connect other third-party version control systems, like GitLab and Jira. Empirical studies should measure the impact of issue creation on developer team workflows, as it improves efficiency and the end user satisfaction.
2. Enhanced Real time Collaboration
    a. Integrate Gantt charts and other ways to integrate organization or user's workflows. This would improve user satisfaction as users do not need to use multiple platforms to complete a task. They can just use Project Vision as they would have a complete toolkit.
3. Advanced Analytics & Reporting
    a. Sentry's service can be used to collect more in-depth user information, such as errors occurred in a session, and other statistics. These can help as we can analyse it to see user patterns, hence that data can be fed back to Project Vision to dynamically adjust itself based on the user data.
4. Mobile First Design
    a. Although current layouts are responsive for any screen size and type, it is not mobile first. Some users prefer in using a mobile device to submit posts or comments, hence promoting portability. Progressive web app can be introduced to solve this issue in a future iteration. Usability testing with a broader demographic would validate design improvements.

5. Longitudinal User Study
    a. Conduct multiple case studies based on brief times of about two to six months. This would help us evaluate the long-term adoption, user retention and ROI on Project Vision. Such research would yield insights into how feedback platforms influence product roadmaps and engagement overtime, where it gives user satisfaction.

With all these future new features, Project Vision would be enhanced drastically where it can contribute more towards user-driven software development tools and project acceleration.

6.3 Legal, Social, Ethical and Professional Issues

Project Vision's design and deployment must consider the full spectrum of LESPIs to ensure responsible practice:

1. GDPR & Data Rights: The platform already supports data-deletion requests ("right to be forgotten") and strict anonymisation of user testing data. Future work will implement data-retention policies and user controls for consent management.
2. Ethical Content Moderation
    a. Automated Moderation: Leverage Natural Language Processing or Large Language Models filters to detect hate speech, harassment, or other prohibited content. Establish an appeals system, such as a form for users to appeal their bans.
    b. Transparency: Publish an ethics charter outlining acceptable use, data handling, and transparency of algorithmic decisions (e.g., vote-weighting adjustments).

3. Social
    a. Digital Divide: Recognize that not all users have high speed internet connection or high spec computers, hence optimization must be improved on together with a mobile first design system. This would improve overall engagement rate.
    b. Transparency: Publish an ethics charter outlining acceptable use, data handling and transparency decisions, for instance vote weighting adjustments based on post severity, e.g. critical vulnerability risk post.
4. Professional Standards and Accountability
    a. Audit logging: The reason that the Audit log was created was to prevent abuse, Aswell as transparency within boards or organizations. Expanding this functionality would only strengthen the security of the platform, as users will need to take accountability for the actions they commit on the boards or organizations. This will maintain trust within the userbase.

By embedding LESPIs into the development cycle, Project Vision positions itself as a responsible and ethical platform that obeys and respects common internet laws by also complying with institutional policies.

6.4 Synoptic Reflections

This section will be written in the first person to capture personal growth for the past 400 hours of development.

As I have started Project Vision, I indeed had a Vision to create something that was misinterpreted so much by competitors. A Feedback board is supposed to be something that helps diverse types of users and organizations. My main original goal was to add extra features that would combine all the competitors together

in one big web application, Project Vision, however I have concluded as the development begun that I should create unique features that these platforms lack drastically. The first thing that I wanted to improve was the way users communicate with each other in a professional environment. As this is not a chatroom or a community hosting web app, it does need to stay professional.

I gained deep proficiency in the Next.js framework, mastering both static and server-rendered modes. Working extensively with Prisma ORM sharpened my understanding of type-safe database interactions while Playwright broadened my testing toolkit to confirm both functional and performance validation. These technical competencies significantly enhance my skillsets as a Full Stack Software Engineer.

The overall project was a success as it hit most of its set milestones, hence I was very happy to see that the testers were able to produce good and validated test responses, regardless of the few failures which were issues that were not caused by them, but by the constraints and some resource mismanagements that happened during the development process. I am looking forward to improving on this project later, as there is potential for a real-world implication where this web application can strive off its own usability features that were built by me, a developer, for developers.

REFERENCES

Markey, R., Reichheld, F. and Dullweber, A., 2009. Closing the customer feedback loop. *Harvard Business Review*, 87(12), pp.43–47.

OWASP, n.d. *Authentication Cheat Sheet* [online]. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html [Accessed 1 May 2025].

Tkalich, A., Klotins, E. and Moe, N.B., 2025. Identifying critical dependencies in large-scale continuous software engineering. *arXiv preprint* arXiv:2504.21437. Available at: https://arxiv.org/abs/2504.21437 [Accessed 1 May 2025].

Wang, F. and Hannafin, M.J., 2005. Design-based research and technology-enhanced learning environments. *Educational Technology Research & Development*, 53(4), pp.5–23.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77–101. [Accessed 1 May 2025]

BIBLIOGRAPHY

APPENDIX A