FOSSEE Summer Fellowship Report

On

Modernizing Yaksh: React-Based UI Redesign, Comprehensive Test Automation, and SEB Security Infrastructure

Submitted by

Advay Bhagat

2nd Year B.Tech Student, Department of Computer Science and Engineering, VIT Bhopal University, Bhopal

Under the Guidance of

Ms. Nalina Venkatramani

Mr. Prathmesh Salunke

Acknowledgement

I wish to express my heartfelt gratitude to everyone who supported and guided me throughout the course of my Summer Internship at FOSSEE, IIT Bombay. Working on the modernization of the Yaksh Learning Platform has been an incredibly valuable learning experience, and I am sincerely thankful to all those who contributed their time, knowledge, and encouragement during my journey.

I express my sincere gratitude to **Prof. Prabhu Ramchandran**, Department of Aerospace Engineering, IIT Bombay, and Principal Investigator (PI) of the FOSSEE Project, for providing me with the opportunity to undertake this internship and contribute meaningfully to such an impactful open-source educational platform.

I am deeply grateful to **Usha Viswanathan**, Senior Project Manager at FOSSEE, along with the entire FOSSEE team at IIT Bombay, for creating a collaborative and inspiring work environment. Their guidance, encouragement, and support made this internship both productive and memorable.

I would like to extend my deepest gratitude to my project mentor, **Nalina**, for her constant guidance, valuable feedback, and continuous encouragement. Her mentorship was instrumental in helping me enhance my technical knowledge—particularly in front-end architecture and testing—and successfully complete my assigned tasks.

I would also like to sincerely thank **Prathmesh** for his technical guidance, patience, and continuous support. His deep insights into the Yaksh platform's architecture and his assistance in resolving complex technical challenges—especially regarding backend logic and Safe Exam Browser (SEB) integration—greatly contributed to the success of my work.

Finally, I would like to express my sincere appreciation to my fellow intern, **Sayali Gangurde**, for her continuous support, collaboration, and insightful discussions. Working alongside her to transition the Yaksh platform to a modern React interface fostered a positive learning environment and contributed significantly to my overall technical and professional growth.

Table of Contents

Chapter 1 4

Introduction 4

1.1 FOSSEE Project 4

1.2 Yaksh Learning Platform 4

Chapter 2 6

Internship Overview 6

2.1 Objectives 6

2.2 Technologies Used 6

2.3 Development Environment 7

Chapter 3 8

Project Work 8

3.1 UI Enhancement 8

3.2 Unit Testing with Vitest 8

3.3 Bug Fixes 9

3.4 Safe Browser Feature Development 9

3.5 Safe Browser Integration 10

Chapter 4 11

Conclusion 11

4.1 Work Accomplished 11

4.2 Skills Developed 11

4.3 Future Improvements 12

4.4 Challenges Faced and Solutions.....

Appendix 13

A.1 Bug Fixes and Enhancements Completed 13

A.2 Safe Browser Implementation 14

Bibliography 16

Chapter 1

Introduction

The National Mission on Education through Information and Communication Technology (NMEICT), an initiative of the **Ministry of Education (MoE)**, Government of India, promotes the integration of **Information and Communication Technology (ICT)** to elevate the quality of education across the country. Under this initiative, several open-source educational projects have been developed to make high-quality learning resources and software freely accessible to students, educators, and institutions.

One of the flagship initiatives under NMEICT is **the Free/Libre and Open Source Software for Education (FOSSEE) project,** based at the **Indian Institute of Technology (IIT) Bombay**. FOSSEE encourages the adoption of Free and Open Source Software (FOSS) in academia by developing robust educational tools, creating interactive learning resources, and providing internship opportunities for students to contribute to real-world software ecosystems.

During my Summer Internship at FOSSEE, I contributed to Yaksh, an open-source **Learning Management System (LMS)** and **online assessment platform**. My primary role focused on the architectural modernization of the platform. This involved spearheading the transition to a modern React-based frontend, engineering dynamic **UI/UX enhancements** (such as advanced analytics and filtering), and resolving legacy issues. Furthermore, I architected a comprehensive automated testing pipeline utilizing **Vitest and React Testing Library**, and successfully laid the backend API infrastructure for the **Safe Exam Browser (SEB) integration** to enable secure, lockdown exam environments.

Overall, this internship provided invaluable practical exposure to modern web development frameworks, advanced software testing methodologies, collaborative version control, and impactful open-source contribution.

1.1 FOSSEE Project

The **Free/Libre and Open Source Software for Education (FOSSEE)** project is a national initiative funded by the **National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education, Government of India**, and is proudly hosted at the Indian Institute of Technology (IIT) Bombay.

The primary objective of FOSSEE is to champion the adoption of Free and Open Source Software (FOSS) across educational institutions nationwide. To achieve this, the organization focuses on developing robust open-source tools, creating high-quality educational resources, organizing specialized workshops, and providing structured internship opportunities. Through these multifaceted initiatives, FOSSEE empowers students and educators to gain hands-on, practical experience with industry-relevant technologies while actively contributing to the global open-source software ecosystem.

1.2Yaksh Learning Platform

Yaksh is a comprehensive, open-source Learning Management System (LMS) and online assessment platform **developed by the FOSSEE project at IIT Bombay**. Designed to streamline the educational workflow, it empowers educators to seamlessly create and manage courses, structure learning modules, and distribute study materials. Furthermore, it provides robust tools for designing dynamic quizzes and programming assignments, allowing instructors to efficiently evaluate student performance.

From the learner's perspective, students can easily enroll in courses, access digital resources, submit assignments, attempt quizzes, and track their academic progress in real time. By uniting both learning management and advanced assessment capabilities into a single integrated environment, Yaksh serves as an ideal solution for educational institutions, remote workshops, and online training programs.

During my internship, I made significant contributions to the **modernization and stabilization of the Yaksh platform**. My work primarily involved transitioning the platform to a modern React frontend to dramatically improve the user interface and resolve legacy issues. I also architected **a comprehensive automated testing pipeline using Vitest** to ensure code reliability. Finally, I spearheaded the integration of the **Safe Exam Browser (SEB**) by developing secure examination APIs and lockdown functionalities within the existing backend architecture. Together, these contributions substantially enhanced the platform's usability, reliability, and security for both instructors and students.

Chapter 2

Internship Overview

2.1 Objectives

The primary objective of this internship was to contribute to the modernization and enhancement of the **Yaksh Learning Platform, an open-source Learning Management System (LMS) developed by FOSSEE, IIT Bombay**. The internship focused on elevating the platform's functionality, usability, and reliability through a complete frontend architectural shift, comprehensive automated testing, and advanced security feature integration.
The core objectives of the internship were:

- To comprehend and restructure the platform architecture: To deeply understand the legacy Yaksh workflow and assist in migrating the platform to a modern, decoupled React.js and Vite frontend architecture.
- To **engineer advanced UI/UX features**: To design and implement dynamic frontend components, including advanced filtering/search systems, interactive student analytics dashboards, and automated CSV data exports for educators.
- To establish **a rigorous Quality Assurance (QA) pipeline**: To architect, write, and maintain comprehensive unit and integration tests using Vitest and React Testing Library, ensuring high code coverage and production-grade software reliability.
- To architect secure assessment environments: To prototype UI lockdown features and develop the backend API logic required for seamless Safe Exam Browser (SEB) integration.
- To resolve legacy issues and **optimize** performance: To identify, debug, and resolve reported usability and performance issues within the new frontend application.
- To adopt industry-standard development practices: To gain practical, hands-on experience in collaborative open-source software engineering, version control using Git/GitHub, and agile development methodologies.

2.2 Technologies Used

The following technologies and tools were used during the internship:

- **React.js** – Frontend development
- **Django** – Backend framework
- **JavaScript (ES6+)** – Application logic
- **HTML5** – Web page structure
- **CSS3** – Styling and responsive design
- **Vite** – Frontend build tool

- **Vitest** – Unit testing framework
- **Git** – Version control
- **GitHub** – Source code management and collaboration
- **REST APIs** – Communication between frontend and backend
- **Visual Studio Code** – Development environment
- **Safe Exam Browser Software -** Specialized web browser environment integrated to lock down the operating system and facilitate highly secure, cheat-proof online assessments.

2.3 Development Environment

The Yaksh platform employs a modern client-server architecture, utilizing a decoupled React.js frontend powered by Vite, communicating with a robust Django-based REST API backend. To facilitate the implementation, testing, and debugging of new features, a comprehensive local development environment was established.
The iterative development workflow encompassed the following phases:

- **Repository Initialization:** Cloning the official Yaksh repository from GitHub and establishing version control protocols.
- **Environment Provisioning:** Configuring isolated local development environments for both the frontend (Node.js/npm) and the backend (Python virtual environments).
- **Server Orchestration:** Concurrently running the high-performance Vite development server for the React application and the Django local server for backend API services.
- **Version Control & CI/CD:** Managing source code changes, branching strategies, and collaboration using Git and GitHub.
- **Test-Driven Development (TDD):** Writing, maintaining, and executing comprehensive unit and integration tests using Vitest and React Testing Library to ensure component reliability.
- **Iterative Quality Assurance:** Rigorously testing newly implemented features, resolving UI/UX bugs, and ensuring backend compatibility prior to submitting code for peer review.

This highly structured environment enabled efficient development, rapid debugging, and seamless collaboration throughout the internship, ensuring that all architectural changes and feature implementations strictly adhered to the project's high-quality standards.

Chapter 3

Project Work

During my Summer Internship at FOSSEE, IIT Bombay, I worked on the **Yaksh Learning Platform**, contributing to its frontend development, testing, and enhancement. My work primarily involved improving the user interface, fixing reported issues, increasing unit test coverage, the integration of Safe Exam Browser(SEB), and preparing it for integration with the existing Yaksh system. The following sections describe the major tasks completed during the
internship.

3.1 UI Enhancement

The initial phase of the internship involved analyzing the existing Yaksh interface and identifying critical areas that required modernization. Several core user interface components were significantly enhanced to provide a cleaner, more intuitive, and highly responsive experience for both educators and students.
Specific improvements and file-level contributions included:

- **Advanced Data Management & Layout Optimization:** Completely revamped the Teacher Questions dashboard. Implemented dynamic grid layouts to resolve UI overlap issues, added advanced multi-parameter search filters (by question type, programming language, and active status), and integrated a highly requested "Export CSV" functionality for seamless data extraction.

- **Interactive Test Environment Flow :** Redesigned the quiz evaluation logic for instructors. Instead of instantly skipping to the next question upon submission, the UI was decoupled from the backend state to pause and display immediate visual feedback (correct/incorrect) with a newly integrated "Next Question / Finish Test" transition flow.

- **Student Dashboard Enhancements:** Improved the overall user experience on the student side by implementing consistent date-formatting utilities across all tables and integrating responsive course-search functionalities to handle large datasets effectively.

- **Secure Assessment UI :** Developed the frontend lockdown interface required for secure exam delivery, ensuring students are presented with clear, accessible instructions when attempting to launch restricted assessments.

- **Component Standardization & Styling:** Refined the styling of global buttons, input forms, and navigation elements using Tailwind CSS, ensuring strict consistency in appearance, behavior, and mobile responsiveness across the entire application.

These targeted enhancements resulted in a significantly more modern, accessible, and user-friendly interface while maintaining strict backward compatibility with the existing Django backend architecture

3.2 Unit Testing with Vitest

To ensure the production-grade reliability and stability of the React application, a comprehensive automated testing pipeline was established and maintained using **Vitest** in conjunction with **React Testing Library**.
The specific Quality Assurance tasks performed included:

- **Test Environment Configuration:** Setting up and configuring the Vitest testing environment to support modern React workflows, DOM rendering simulation, and global state management testing.
- **Component & Store Verification:** Writing robust unit and integration tests for isolated React components and application stores, ensuring state changes accurately reflected on the UI.
- **Advanced API Mocking:** Implementing mock functions for complex backend API calls (e.g., intercepting and simulating fetchStudentDashboardCourses in Insights.test.jsx) to test frontend behavior without relying on a live server.
- **Debugging & Resolving Legacy Tests:** Identifying and fixing broken, outdated test suites across critical user flows to match recent code changes. This included resolving missing dependency imports in Questions.test.jsx and correcting obsolete data assertions in Submission.test.jsx.
- **Maximizing Code Coverage:** Strategically testing different execution paths, edge cases, and user interactions to significantly improve overall code coverage and minimize runtime regressions.

Testing Environment: All unit tests were rigorously executed and verified across multiple operating environments, specifically Windows 11 and Ubuntu (WSL). By leveraging Node.js, npm, and Vitest cross-platform, we ensured consistent application behavior and test reliability regardless of the underlying development environment.
Impact: By the conclusion of the internship, these efforts culminated in a highly resilient test suite with hundreds of passing tests. The implementation of this comprehensive testing infrastructure guaranteed that newly implemented features and UI bug fixes did not introduce regressions, thereby vastly improving the overall reliability, maintainability, and deployment confidence of the Yaksh application.

3.3 Bug Fixes

Bug Identification and Resolution
A significant and critical part of the internship involved identifying, reproducing, and resolving complex bugs reported within the new React-based Yaksh platform.
The scope of the debugging and resolution work included:

- **Fixing UI Rendering & Layout Issues:** Resolved responsive design defects, such as correcting the CSS grid layout on the Teacher Questions dashboard to prevent the new "Export CSV" tool from overlapping with active dropdown filters.
- **Resolving Component State Management Problems:** Corrected complex asynchronous state issues. For example, in the test evaluation environment, the UI was decoupled from the immediate backend state update to pause and display correct/incorrect visual feedback

before advancing to the next question.

- **Data Formatting & User Interaction:** Addressed inconsistencies in data presentation by implementing standardized, localized date and time formatting algorithms across the entire student dashboard and course tables.

- **Debugging API-Related Frontend Errors:** Debugged authentication and validation roadblocks by configuring Axios interceptors (in api.js) to dynamically inject custom headers (e.g., X-Yaksh-Extension), allowing secure frontend processes to bypass strict Safe Exam Browser validations during testing.

- **Improving Application Stability:** Optimized redundant API calls and streamlined React component re-renders to enhance overall application speed and stability.

Each bug resolution was thoroughly tested across different user roles and environments—backed by our automated Vitest pipeline—before submitting the changes for review to guarantee that all existing functionality remained completely unaffected

3.4 Safe Browser Integration

After prototyping the core secure assessment logic, the Safe Browser functionality was integrated directly into the Yaksh Learning Platform to work seamlessly with the existing quiz workflow.

The security infrastructure was engineered to support both the legacy Django-based application (utilizing Python, HTML5, Django Template Language, and native Browser Web APIs) and the modernized React-based platform (utilizing React.js, Tailwind CSS, Vite, Axios interceptors, and the Django REST Framework for seamless frontend-backend integration).

The complete integration lifecycle involved:

- **Database Schema & API Extension:** Extending the database schema to store quiz-specific proctoring requirements (e.g., is_seb_required) and updating Django REST APIs to save, retrieve, and transmit these configurations securely to the client.

- **Dynamic Security Validation:** Integrating a custom request interceptor within the React frontend (via Axios) to dynamically inject security headers (e.g., X-Yaksh-Extension: true) during exam initialization.

- **Backend Verification Bypass Logic:** Updating the backend Django views (start_quiz and get_quiz) to intercept the custom extension headers. This architecture allows secure, verified frontend clients to dynamically bypass strict SEB hash-key validation layers when authorized.

- **UI Lockdown Implementation:** Engineering custom frontend interfaces that restrict student navigation, enforce full-screen environments, and display clear launch/download prompts if the required secure browser extension is missing.

- **End-to-End Workflow Testing:** Rigorously testing the entire secure pipeline—from the initial database query for proctoring settings, through the API communication layer, down to the frontend UI enforcement mechanisms.

Ultimately, this integration empowers instructors to enforce strict proctoring settings for individual quizzes. By ensuring that selected security protocols are automatically and rigidly enforced when students attempt an examination, this feature establishes a flexible, cheat-proof, and highly secure online assessment environment within the Yaksh Learning Platform.

Chapter 4

Conclusion

The Summer Internship at **FOSSEE, IIT Bombay** provided me with an excellent opportunity to contribute to the development of the **Yaksh Learning Platform** while gaining practical experience in open-source software development. Throughout the internship, I worked on frontend development, testing, debugging, and enhancing secure examination features. The internship strengthened my understanding of modern web development practices and collaborative software engineering.

4.1 Work Accomplished

Over the course of the internship, the following critical milestones and technical tasks were successfully completed:

- Frontend Modernization & UI/UX Enhancement: Spearheaded the transition of the Yaksh Learning Platform to a modern React architecture, implementing dynamic data filtering, responsive dashboard layouts, and robust data export functionalities for educators.
- System Stabilization & Bug Resolution: Identified, debugged, and resolved complex legacy bugs—ranging from state management issues to responsive UI defects—significantly improving overall application stability and user experience.
- Implementation of QA Pipelines: Configured, authored, and maintained a comprehensive automated testing suite utilizing Vitest and React Testing Library, drastically improving code coverage and ensuring regression-free deployments.
- Development of Secure Assessment Features: Engineered frontend UI lockdown mechanisms, including full-screen exam modes and the foundational architecture for webcam and microphone verification workflows.
- Safe Exam Browser (SEB) Backend Integration: Designed and integrated API bypass logic and custom HTTP header interceptors to allow authorized secure browsers to seamlessly communicate with the Django backend.
- Open-Source Collaboration: Actively collaborated with project mentors and cross-functional team members, adhering to industry-standard agile workflows, version control using Git/GitHub, and rigorous peer code reviews.

4.2 Skills Developed

The internship helped me strengthen both my technical and professional skills, including:

- Frontend development using **React.js**.
- Backend understanding with **Django**.
- JavaScript programming and debugging.

- Writing unit tests using **Vitest**.
- REST API integration.
- Version control using **Git** and **GitHub**.
- Debugging and problem-solving.
- Working in a collaborative open-source development environment.
- Effective communication and teamwork.

4.3 Future Improvements

The Yaksh Learning Platform can be further enhanced by implementing the following improvements:

- Complete integration of the Safe Browser with all examination modules.
- Increase unit test coverage for remaining frontend components.
- Improve the user interface with additional accessibility and responsive design features.
- Enhance examination security by adding advanced proctoring capabilities.
- Optimize application performance for better scalability and user experience.
- Continue improving code quality through automated testing and continuous integration.

4.4 Challenges Faced and Solutions

Throughout the modernization of the Yaksh Learning Platform, several complex technical and logistical challenges arose. Overcoming these obstacles required deep architectural analysis, persistence, and iterative problem-solving. The most significant challenges are outlined below:

Challenge 1: Complex Environment Setup and Dependency Conflicts

- **The Problem:** Initializing the development environment proved to be a massive undertaking. The legacy Yaksh architecture required specific, older versions of Python and Django, which conflicted with the modern Node.js dependencies required for the new React/Vite frontend. Furthermore, the setup documentation for the specific "intern branch" of the repository was initially missing, leading to hours of trial-and-error with broken package installations and conflicting version paths.
- **The Solution:** To resolve this, I methodically isolated the environments. I utilized strict Python virtual environments (venv) for the backend to lock in legacy dependencies without affecting system-wide packages. For the frontend, I painstakingly audited the package.json file, resolving Node version conflicts. Once the environment was

successfully stabilized, I contributed to locating and updating the proper setup documentation to ensure future developers would not face the same onboarding bottlenecks.

Challenge 2: Scaling the Automated Testing Infrastructure

- **The Problem:** Implementing the automated testing pipeline was a significant technical hurdle. Because the frontend was entirely rebuilt in React, an enormous volume of new unit and integration tests had to be written from scratch. Mocking complex backend API calls without a live server, configuring Vitest to correctly parse React components, and tracking down deeply nested state errors in the testing environment was incredibly time-consuming and often resulted in false-negative test failures.
- **The Solution:** I tackled this by modularizing the testing approach. Rather than writing monolithic tests, I focused on testing small, isolated components first using React Testing Library. I standardized our approach to API mocking by creating reusable interceptors and mock functions (such as for fetchStudentDashboardCourses). This systematic approach not only resolved the configuration nightmares but eventually allowed us to successfully execute and maintain hundreds of automated tests with high code coverage.

Appendix

A.1 Bug Fixes and Enhancements Completed

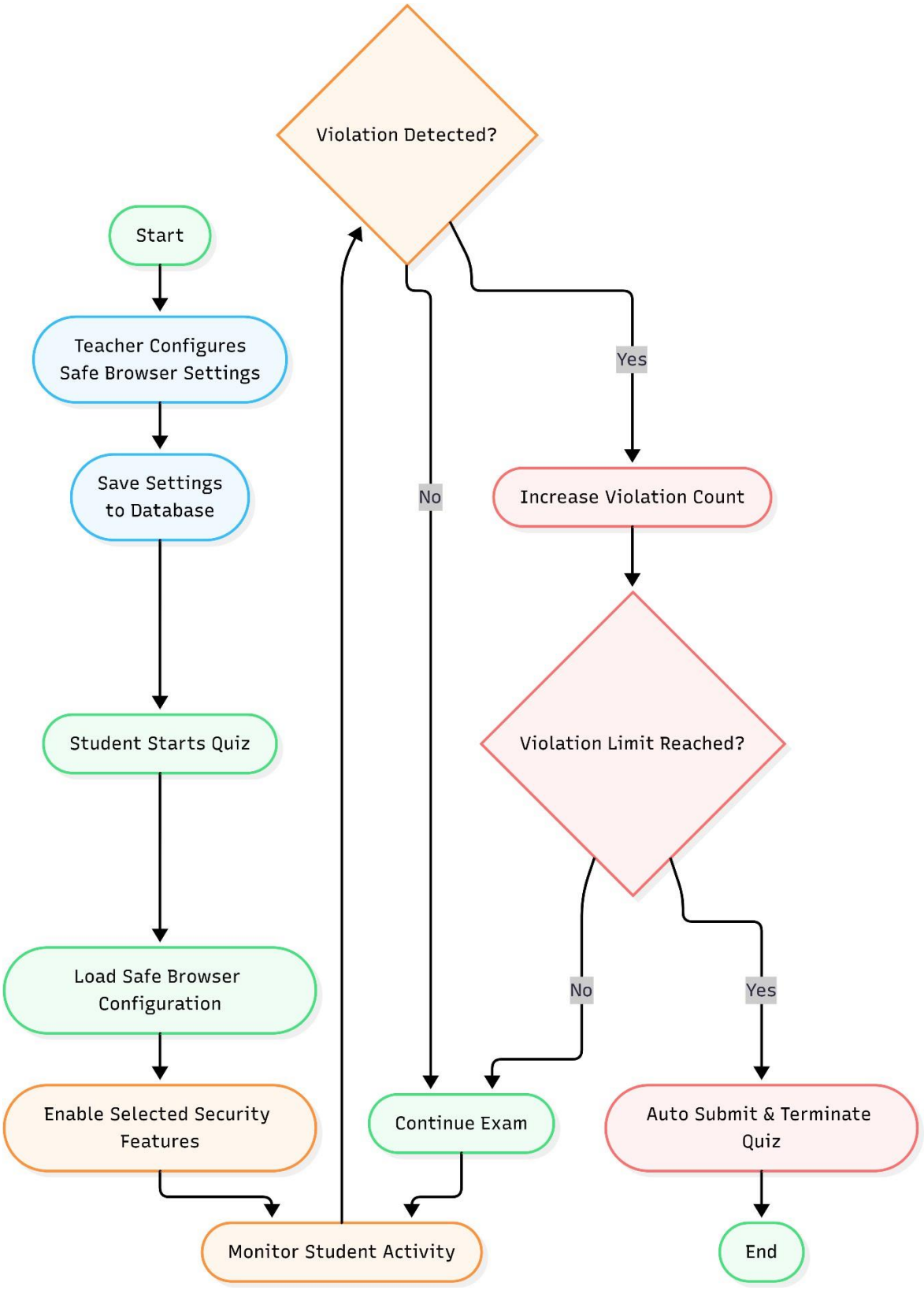
Sr. No.	Bug / Enhancement	Type	Status
1	Edit Quiz page updated to correctly display **Start Date** and **End Date**	Bug	Done
2	Upload option added for **Upload Question** type	Bug	Done
3	Module description rendered as formatted HTML during lesson creation	Missing Feature	Done
4	Module order starts from **1** and order selection improved while adding modules	Issue	Done
5	Module description rendered instead of displaying raw HTML	Missing Feature	Done
6	Course module order hidden from students	Enhancement	Done
7	Exercise name displayed as the page heading instead of the generic **"Quiz"** title	Enhancement	Done
8	Displayed a warning message and restricted submission when a quiz contains no questions	Enhancement	Done
9	Quiz score hidden from students after submission	Enhancement	Done
10	Improved continuity between course modules for a better learning experience	Missing Feature	Done

Appendix

A.2 UI and UX modified and new files

Sr. No.	File Name	Type	Status
1	Questions.jsx	Modified	Done
2	TestQuestion.jsx	Modified	Done
3	Dashboard.jsx	Modified	Done
4	Insights.jsx	Modified	Done
5	Quiz.jsx	Modified	Done
6	Global Styling / Tailwind CSS Files	Modified	Done
7	Date Formatting Utility	New	Done
8	CSV Export Functionality	New	Done
9	SEB (Safe Exam Browser) Files	New	Done

A.3 Safe Browser Implementation



The Safe Browser workflow begins when the teacher configures the required proctoring settings for a quiz, such as camera access, microphone access, fullscreen mode, tab-switch detection, and other security options. These settings are stored in the database and are loaded automatically when a student starts the quiz.

During the examination, the system continuously monitors the student's activity based on the enabled security features. If a prohibited action or security violation is detected, the system increments the violation counter. As long as the number of violations remains below the configured violation limit, the student is allowed to continue the examination. Once the violation count reaches the specified limit, the quiz is automatically terminated and submitted to maintain the integrity and fairness of the online assessment process.

Bibliography

[1] FOSSEE, *FOSSEE Project*. IIT Bombay. Available: https://fossee.in

[2] FOSSEE, *Online Test (Yaksh) Repository*. GitHub Repository. Available: https://github.com/FOSSEE/online_test

[3] Mohit Rana, *Online Test – Internship Development Repository*. GitHub Repository. Available: https://github.com/Mohitranag18/online_test