# Front-End for a Semi-Automated Grading Tool

Harper Doak

***Abstract*—Motivated by the shift towards online teaching, an application which semi-automates the grading of long-form questions would not only save time but could be more accurate. The education sector frequently grapples with the challenge of efficiently grading a large volume of student submissions. Addressing this challenge, this project aims to develop the front-end interface of a semi-automated grading tool that promises to streamline the grading process, thereby enhancing efficiency and reducing potential biases. The foundation of this tool's evaluation mechanism is built upon the utilization of four distinct distance metrics, each meticulously selected to calculate the disparity between student submissions. Questions and submissions can be imported directly into the application, and once the grading has been completed, they can be downloaded and exported. To cater to a dynamic and interactive user experience, the project leverages the capabilities of Next.js, a robust React framework known for its performance and scalability. This framework was selected not only for its rich set of features but also for its server-rendering capabilities, which will facilitate the real-time processing of data, critical for instantaneous feedback during the grading process. This application provides an education tool that can be used by lecturers to help better utilise their time, spending less time marking and more time teaching.**

***Index Terms*—Article submission, IEEE, IEEEtran, journal, LaTeX, paper, template, typesetting.**

## I. Introduction

In the world of education, lecturers, tutors, and other academic professionals spend countless hours marking and grading exams and tests, especially those involving long-format questions. Marking the same questions over and over can be a very tedious task.

In the computer science field, exams usually consist of short written questions e.g. " Explain this...". For these questions there are usually one or two answers or ways to approach the question, resulting in many students having very similar answers. This leaves the door open for an automation tool that removes the monotonous task of marking the same answer repeatably. Automation of grading is relatively straight forward when it comes to multiple-choice questions, however, for long-format written questions automation is a lot more challenging.

The semi-automated grading application (SAGA) utilises distance metrics to group similar submissions together, thus allowing the marker to give the same score and feedback for all submissions which are similar (based on the distance metric selected).

This SAGA will empower markers by placing control at their fingertips. Through a dynamic slider bar, markers can fine-tune the chosen distance metric, allowing for precise customization of the grading process. The aim of this project is to develop the front-end of this application utilising Next.js, pairing this with Firebase. The result of this project will be a web application which offers a good user experience, semi-automation marking of long-format answers, and supplies an education tool which lecturers and teachers can use to better manage their responsibilities. Following the completion of this project, the SAGA will help in making markers across all types of disciplines better utilise their time [1], spending less time marking and more time teaching.

This project was supervised by Arman Khouzani (Primary).

### A. Problem Statement

The education landscape underwent a seismic shift during the COVID-19 pandemic, accelerating the transition to remote learning [2]. As a consequence, a growing number of examinations and tests have migrated from traditional in-person formats to online settings. However, this transformation has not been accompanied by a evolution in the way these online assessments are marked, particularly in fields like engineering, and computer science.

In these technical disciplines, assessments often consist of questions with well-defined answers, a characteristic that distinguishes them from many other academic fields where questions are inherently open-ended.

While the move to online assessments offers many advantages, it has also exacerbated a long-standing challenge: the repetitive and time-consuming nature of manual grading. Markers in engineering and computer science frequently find themselves reevaluating identical or nearly identical submissions repeatedly. This iterative grading process presents several significant challenges:

- Tedious and Repetitive Work: Markers face the arduous task of reviewing and grading a large volume of submissions, each requiring the same set of criteria to be applied repeatedly. The repetitive nature of this work can be mentally taxing and monotonous.
- Risk of Error: As markers sift through countless responses with uniform solutions, it is easy to succumb to lapses in concentration, leading to human errors in grading. These errors can have far-reaching consequences, affecting the fairness and accuracy of assessments.
- Lost of Engagement: The sheer volume of similar submissions can lead to marker fatigue, causing a decline in attentiveness and engagement. This reduced focus further compounds the risk of grading inaccuracies.
- Time-Intensive: The traditional manual grading process is time-intensive, consuming valuable resources that could be redirected to more impactful educational activities, such as providing timely feedback to students, or designing innovative instructional materials.
- Quality Assurance: Ensuring consistency and fairness in grading across the multitude of submissions is a persistent challenge. Variability in human judgment can lead to

discrepancies in grading, undermining the credibility of the assessment process.

This grading application seeks to tackle these issues head-on by introducing automation, efficiency, and accuracy into the grading process, providing educators in engineering and computer science with a more effective means of evaluating student work.

### B. Solution and Deliverables

The main solution for this problem is an application which semi-automates the grading of long-format questions. This will result in a tool which increases the efficiency of marking and produces potentially higher accuracy and consistency compared to marking in person.

It will also make the process of marking more engaging with things like visual feedback on the progress status, easier searching capabilities, seamless registration of the marks and feedback, and easy exporting. This application sets out to remove the tedious tasks involved with marking, as much as possible. It also provides an interface which facilitates better management of submissions, as well as ensuring that students receive their grades in a timely manner.

Some of the basic functionalities and deliverables of the application are listed below:

- Authentication and authorization, sign up functionalities, or login with Google.
- An admin interface, where admin users can control the users and questions on the application.
- An interface for importing the submission and exporting the results.
- An application designed with user experience in mind.
- Automating the opening of each submission, and the organisation of the presentation of the answers to each question.
- Showing progress status per question.
- Storing the feedback and score for the corresponding submission.

The next steps include adding features to group similar submissions, enabling the marker to assign the same score and feedback to all submissions thus bypassing the generic marking process and saving time. This feature is implemented by comparing the texts based on four different distance metrics. The features below represent this next step in the application:

- A distance metric selection tab, where the user can switch between the applied distance metric.
- A slider bar adjusts the aggressiveness of the selected distance metric, thus allowing a grader to see the effect dynamically.

### C. Distance Metrics

The four different distance metrics used for this application are Levenshtein, Bert+Cosine, TfIdf+Cosine, and Jaccard. Each one offers a slightly different way of sorting the submissions which have been uploaded to the database.

*1) Levenshtein:* Levenshtein distance, also known as Edit Distance, is a metric that measures the difference between two texts based on the minimum number of character edits required to transform one text into another [3]. Character edits include operations like additions, deletions, and substitutions of numbers.

For example, if we have two words, ”kitten” and ”sitting”, the Levenshtein distance between them is 3 because you would need to perform three character edits (substitute ’k’ with ’s’, insert ’i,’ and insert ’g’) to transform ”kitten” into ”sitting”. This metric quantifies the level of similarity and dissimilarity between two texts in terms of how different they are at the character level.

*2) BERT+Cosine:* Bert+Cosine utilises the powerful BERT (Bidirectional Encoder Representations from Transformers) model, which is a state-of-the-art natural language processing (NLP) model, combined with the cosine similarity metric to determine text similarity [4]. BERT is a deep learning model that is pre-trained on a massive amount of text data. It can understand context and relationships between words in a text, making it highly effective for various NPL tasks, including similarity analysis. BERT+Cosine leverages BERT’s contextual understanding of language to create vector representations of text data and then applies cosine similarity to measure the similarity between these vectors. It is a powerful approach for assessing text similarity that considers semantic meaning and context.

*3) TfIdf+Cosine:* TfIdf+Cosine employs Term Frequency-Inverse Document Frequency (TfIdf) analysis and the cosine similarity metric to evaluate the similarity between texts [5]. TF-IDF quantifies the significance of terms within a document relative to a larger corpus, considering both term frequency and rarity. Cosine similarity measures the similarity between text vectors, calculated from their TF-IDF representations. Higher cosine similarity scores indicate greater textual similarity, while lower scores indicate dissimilarity. This approach is widely used in text analysis and document clustering, enabling the assessment text similarity while accounting for the importance of terms within documents and across a corpus.

*4) Jaccard:* Jaccard is a metric that evaluates text similarity based on the size of the intersection of terms in two texts divided by the size of the union of those terms [6]. This is clearly shown in the figure below. In other words, it quantifies how many terms two texts have in common relative to the total number of unique terms in both texts. Jaccard is a simple but effective metric for measuring text similarity, especially when you want to focus on the presence or absence of terms in texts.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Fig. 1. The Jaccard Formula

## II. Related Work

Before, undergoing the development of this application I conducted research into preexisting solutions and applications which are available. With the growth in the popularity of AI over the last few years, there has been an increase in the number of AI applications and tools used in the education sector. This section identifies the advantages and disadvantages of some of the existing similar tools and academic studies.

*1) Gradescope:* Gradescope is an online platform that offers a comprehensive set of tools for grading exams and assignments efficiently [7]. It combines the benefits of AI-based grading with instructor control and flexibility.

One main feature of Gradescope is the exam and assignment setup process. This allows lecturers and teachers to create and set up their own exams with various question types. They can define the point values, add rubrics, and specify any specific grading criteria needed. This platform also employs AI-based grading to assist with grading. It uses machine learning techniques to automatically match student answers with correct responses and assign scores accordingly; It can also identify patterns or any common mistakes. Another advantage of Gradescope is its feedback generation feature, which generates detailed student feedback based on the rubrics and grading criteria. The students can then access their graded work, view the feedback, gaining insight into their performance. Finally, one major advantage of this platform is its built-in integration with learning management systems like Blackboard and Canvas. This enables easy syncing of course rosters, grade transfers, and single sign-on for students and instructors.

This platform has a few disadvantages compared to other grading applications. One of those is subjectivity in grading. Gradescope provides exam and assignment software for all subjects and topics. This, therefore, results in AI grading not always being accurate, as it can't calculate the grade of assignments or exams which have subjective analysis. Also, assignments that involve creativity, critical thinking, or subjective interpretation may not be effectively assessed solely through AI algorithms, resulting in human graders having to step in. Another disadvantage is its limited applicability. The platform states that it can handle any subject, exam, or assignment. However, assignments which require physical or hands-on components, such as labs, or presentations will struggle to utilise this platform to the fullest.

Overall, Gradescope aims to streamline the grading process, save time for lecturers, provide consistent feedback, and enhance the overall grading experience. It combines AI-based grading with human control and flexibility to cater for a variety of exam and assignment types.

*2) Grademark:* Grademark is an online tool which is correlated with the popular plagiarism detector Turnitin [8]. It can facilitate paperless grading and provides feedback on student assignments. This tool doesn't utilise AI, however, it has features which are designed to make grading very efficient.

The platform enables assignment submission utilising Turnitin's plagiarism detection system, allowing lecturers to access and review the submissions online.

Grademark's interface provides a user-friendly environment for grading, annotating, and providing feedback directly on digital documents. Lecturers can highlight text, add comments, insert audio or video feedback, and assign grades.

This platform allows lecturers to create and use rubrics to evaluate assignments. These rubrics can be customized with specific criteria and point values, enabling consistent and transparent grading; like some of the other platforms I have described. Lecturers can then apply the rubrics to assess various aspects of the assignment and assign grades accordingly. Grademark utilises QuickMarks library which are pre-loaded comments and feedback templates. Lecturers can access a range of commonly used comments and easily apply them to assignments, saving time, and ensuring consistent feedback.

Finally, Grademark offers data and analytics on student performance and assignment outcomes. Lecturers can access reports that provide insights into class-wide performance, identify trends, and compare student progress over time.

While there are a lot of advantages to this system, there are quite a few disadvantages. First, is the reliance on text-based assignments and the system being designed for essays, reports, and research papers. This makes the tool more niche compared to other platforms which can facilitate both assignments and exams. Also, the tool has the potential for reliance on automated feedback by utilising the QuickMarks library of pre-loaded comments and templates. This is where other systems have an edge as their feedback and comments are written using AI algorithms and machine learning techniques. By using QuickMarks, it means that students might not receive feedback which is meaningful and doesn't provide insight into where they can improve.

Overall, GradeMark enhances the grading and feedback process by providing lecturers with a range of tools to evaluate student assignments digitally. It enables efficient grading and ensures that lecturers can better manage their time.

*3) Automatic short answer grading study using text mining methods:* A study in the UK by Neslihan Suzen focused on automation of the grading process utilising similar methods to what is proposed for this project [9]. Data mining techniques were used to measure the similarity of students' answers to the model answer. The submissions were then clustered together based on the level of similarity acquired and each cluster was then given the same grade as well as providing useful feedback to the answers. Their aim for the research was to design a model of automatic short answer marking and feedback. They conducted an analysis of how precise the text mining methods they used were when marking submissions. They concluded that the number of correctly used words in submission has more influence on marks than the particular order of the words. Their model to predict marks for submissions compared the distance between the model answer and the student's submission, clustering similar submissions together, and giving similar grades to submissions in the same cluster. Overall, their tool and model was developed to have more of a focus on assisting human marking, not fully replacing it with an automated system, similar to our project. This paper provided insight into a very similar project the SAGA, however, they focused on developing a model rather than an application.

| | Our Project | Grademark | Gradescope |
|---|---|---|---|
| Semi-Automation | ✓ | ✗ | ✓ |
| Designed for Long-format Questions | ✓ | ✗ | ✗ |
| Exam Setup functionality | ✗ | ✗ | ✓ |
| Exporting Submissions | ✓ | ✗ | ✗ |

Fig. 2. The above figure shows the comparisons of the other related tools and the application which I have built for this project.

## III. Requirements of the System

This section gives an overview of the requirements of the system. These requirements serve as a basis for designing, developing, and evaluating the application.

*1) Clustering and Pruning:* A pivotal functional requirement of this application involves the application of the four distance metrics in order to group submissions together. These submissions are grouped based on the distance values which is given to each submission based on its text. The grader is granted the capability to finely adjust how the values are interpreted, thus influencing the returned submissions. By incorporating this functionality, the application empowers semi-automation of the marking and grading processes, enhancing efficiency and precision.

*2) User Authentication and Authorisation:* User authentication and authorization represent essential functional requirements for this application, guaranteeing that users gain appropriate permissions upon login [10]. This is critical to enable graders to access pending submissions for grading. Robust access controls must be integrated into the system to enforce user privileges, limiting access to resources, data, and features according to their assigned roles. The system should offer user registration, implementing checks for data uniqueness to maintain data integrity. Furthermore, users should have the capability to reset their passwords in the event of forgetfulness, thus ensuring a seamless user experience. User roles, with distinct access rights, should be clearly defined to manage user permissions effectively. This comprehensive approach to user management enhances security and ensures a tailored user experience.

*3) Searching and Filtering:* Searching and filtering are important for the usability of this application, enabling the user to be able to search and filter through the submissions which they have to mark in any way they see fit. There will be functionality to filter the submissions by student id or question number, as well as the ability to search the id number to find submissions if grades need to be changed or the marker wants to review submissions.

*4) Data Import and Export:* Data import and export is another functional requirement which is key for the usability of this application. The system should be able to support importing data from various file formats including PDF and JSON. The user of the application should be able to specify the source of the imported data, whilst the system validates the data to ensure that it meets the required format. Once the data has been imported the system should have the capability to store it in the database.

*5) Capacity:* Ensuring that the system has the capacity to operate with high performance is crucial for an application of this sort. The system should be optimised to deliver fast response times and minimise latency. This will include optimising the clustering algorithm as well as the database queries for importing and exporting submissions. Performance testing the system with multiple graders actively working on the application will identify the system's maximum capacity and stress limits.

*6) Usability:* The system should have an intuitive user interface that is easy to navigate and understand. It should employ consistent design patterns, clear labelling, and logical organisation of information to enhance user experience. The system should also be responsive and adapt to different screen sizes and devices. It should ensure optimal display and usability on desktop computers and tablets. The system should provide meaningful error messages and feedback to users when they encounter errors or perform invalid actions.

## IV. Design

Design choices are of paramount importance in any project because they serve as the foundational blueprint upon which the entire endeavour is built. Effective design choices ensure that a project not only meets its technical requirements but also aligns seamlessly with its intended purpose and objectives. They determine the project's scalability, maintainability, security, and overall success. Moreover, well-considered design choices promote efficiency in development, reduce the risk of errors, and save valuable time and resources by preventing the need for major revisions down the road. In essence, the design choices made ensure that this project keeps on course, enabling to not only function but thrive in its intended context.

### A. System Architecture

Before commencing the implementation of the application, the system architecture of the application was meticulously designed to prioritise scalability, maintainability, security, and performance, while ensuring compliance with the requirements specifications which were discussed in the preliminary report of this project.

The architecture of this application is grounded in the client-server model, a widely recognised and efficient approach in web development [11]. In this model, Next.js assumes the role of the client, directly interfacing with the server, which, for this application, is facilitated by Firebase. Below I will break down both parts of this model:

*1) Client:* The client constitutes the user-facing part of this web application, residing and operating directly on the user's device. This component plays a pivotal role in delivering the user interface (UI) and facilitating user interactions. The key components of this model are:

- User Interface (UI): The client is responsible for rendering the UI, ensuring that it is visually appealing, responsive, and intuitive for users. It encompasses the design, layout, and presentation of all elements that users interact with.
- User Interactions: Beyond the visual aspect, the client also manages user interactions. It responds to user inputs, such as clicks, keyboard inputs, and gestures, ensuring that actions trigger appropriate responses within the application.
- Performance Optimisation: Performance is a paramount concern. The client optimizes resource utilisation to deliver swift response times and a seamless user experience, even under varying network conditions and device capabilities.
- Client-Side Validation: Certain validations and operations are performed on the client-side to enhance user experience and reduce the load on the server. However, critical validation and business logic are enforced server-side for security and consistency.

*2) Server:* The server is the backend component of the web application responsible for handling various tasks, such as processing requests from clients, managing data, and performing business logic. In this architecture, Firebase is used as the backend service to provide server-side functionality. Key aspects of the server component are:

- Request Processing: The server processes incoming requests from clients, translating them into meaningful actions. It handles various client requests, such as retrieving data, saving changes, or executing specific operations.
- Data Management: The server manages data storage, retrieval, and manipulation. In this architecture, Firebase serves as the backend service, offering robust capabilities for real-time data management.
- Business Logic: The server executes critical business logic, ensuring that application functionality aligns with defined requirements. It performs operations like calculating grades, generating reports, and enforcing security policies.

By adopting this client-server architecture, the SAGA capitalizes on the strengths of each component, delivering a user-friendly, secure, and high-performance experience. Firebase's robust backend capabilities complement Next.js on the client side, providing a robust and reliable foundation for the grading application.

### B. User Interface

The design of the user interface (UI) was an important aspect of this project. The UI plays a pivotal role in shaping the overall experience of the grading application. It is the bridge between the apps features and the users who use them such as lecturers, administrators, and markers. This subsection, looks into the crucial significance of UI design, emphasising its profound impact on user satisfaction, efficiency, and the success of the platform.

At the heart of the grading application's user interface (UI) design philosophy lies the concept of User-Centered Design (UCD). UCD is a fundamental approach that places the needs, preferences, and expectations of the users at the forefront of the design process [12].

UCD begins with gaining an understanding of the user base and understanding their workflows and goals. There are some important design goals which are relevant for an application of this sort:

*1) Intuitive Navigation:* One of the key principles of UCD is the creation of an intuitive and user-friendly navigation system. One of the aims of this application is to minimize the learning curve, ensuring that users can effortlessly find the features they need. Information architecture is meticulously planned to match the mental model of our users, making the application's structure feel natural and logical. Below are some important aspects of having an Intuitive Navigation:

- Logical Structure: The application's structure is designed to feel natural and logical to users, organising content and features in a way that mirrors how educators, administrators, and markers think and work.
- Well-organised Navigation Bar: At the core of the navigation system is a well-organised navigation bar. This element serves as a central hub for accessing various functions. It is designed to categorise functions logically, ensuring that users can quickly locate the tools they require.
- Clear Labels: Clear and descriptive labels accompany each navigation item. Clarity in labelling is crucial for users to understand the purpose of each feature without ambiguity.
- Efficient Task Flow: The navigation flow optimises common tasks. Users can seamlessly move between related functions, reducing the need to backtrack or navigate through unnecessary steps. This is especially important for an application of this nature, as the design of the task flow has huge impact on the time it will take for marking submissions.

*2) Modern Theme:* A modern theme incorporates contemporary design principles, such as clean and minimalistic aesthetics, the use of white space, and a visually appealing color palette. Ensuring that this application has a modern theme helps in making the application appear fresh and appealing to users.

By having a well-designed UI it can enhance usability and efficiency in the grading process

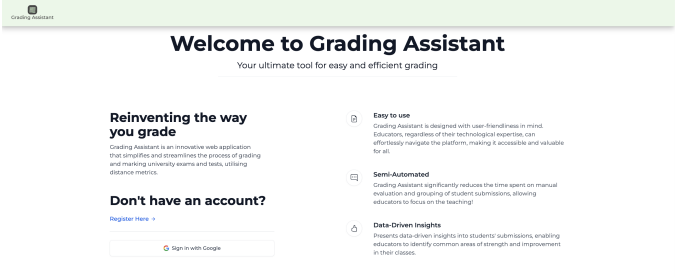


Fig. 3. The landing page of the web application (the first page that users see on the application)

## V. Implementation

This section shows the transition from design theory to practical realisation. It serves as a comprehensive account of the actual development and construction of the Grading Assistant system, detailing its technologies, tools, and methodologies employed to transform the design into a functional artifact. The following section will shed light on how the vision of this project was transformed into a tangible reality while adhering to the project's defined requirements and constraints.

### A. Technologies and Tools

*1) Next.js:* The frontend of the Grading Assistant web application was developed using Next.js, a popular React framework for its server-side rendering (SSR) capabilities and excellent developer experience [13]. Next.js provided several advantages in realising the project goals:

- Server-Side rendering (SSR): I leveraged Next.js to implement SSR, which enhances page load performance and SEO optimisation, rendering pages on the server and delivering pre-rendered content to users.
- React Component Structure: Next.js's integration with React allowed the creation of a modular and component-based front-end architecture, ensuring code reusability and maintainability.
- Routing: The built-in routing system in Next.js simplified navigation between different parts of the application, enhancing user experience.
- Development Environment: Next.js offers hot module replacement and a development server, streamlining the development process and facilitating rapid prototyping.

*2) Tailwind CSS:* The Grading Assistant web application's front-end was styled and designed using Tailwind CSS, a utility-first CSS framework known for its streamlined development process and highly customisable styling capabilities [14]. Tailwind CSS brought several benefits to the project:

- Utility-First Approach: Tailwind CSS's utility-first approach offers a unique and highly efficient way to style elements. It provides a comprehensive set of utility classes that can be applied directly to HTML elements, making styling and design quick and accessible. This approach eliminates the need for writing custom CSS for most styling tasks, saving time and reducing the size of the stylesheet.
- Customisation: Tailwind CSS is exceptionally flexible, allowing for easy customisation to match the specific design and branding requirements of the application. You can create and define custom themes and utility classes, giving you full control over the visual appearance of the application.
- Community and Ecosystem: Tailwind CSS boasts a thriving community and ecosystem with numerous plugins and extensions. This community support provides access to additional functionality and design elements that can be seamlessly integrated into the project.
- Responsive Design: Tailwind CSS includes built-in responsive design classes, simplifying the creation of responsive layouts. Enabling adaption of the UI to different screen sizes and devices.

*3) Firestore:* For the backend and data storage of the Grading Assistant application, I employed Firestore, Google's NoSQL cloud database [15]. Firestore was chosen for its scalability, real-time data synchronisation, and ease of integration.

- NoSQL Database: Firestore's NoSQL data model provided flexibility in handling unstructured data, which is well-suited for managing student submissions and grading data.
- Real-time Updates: Firestore's real-time data synchronisation capabilities allowed graders to receive instant updates when new submissions were made or grading adjustments were applied, enhancing collaboration and efficiency.
- Scalability: Firestore seamlessly scales with the growth of data and user activity, ensuring that the application remains responsive even during peak usage periods.
- Authentication Integration: Firestore integrates seamlessly with Firebase Authentication, providing user authentication and authorization features, aligning with our functional requirements for user roles and access controls.
- Data Import and Export: Firestore's capability with various data formats, including JSON, facilitated data import and export functionalities. This feature was essential for enabling users to import grading data from different file formats.

Firebase's Firestore database is designed around a NoSQL data model that organizes data into collections, providing a flexible and scalable structure for storing and managing information. Within Firestore, each collection serves as a container for related documents, and these documents are where the actual data resides. Firestore documents are essentially JSON-like objects that can contain multiple fields. These fields can store various types of data, such as strings, numbers, booleans, arrays, or nested objects. Initialising Firebase with the Next.js code base is very simple and done through a 'Firebase.js' file. This file contains the Firebase configuration, which includes the projects API key, ID, and AuthDomain [16]. The app is then initialised using the firebaseConfig file. Once this was completed data could be pushed and pulled to and from the database using some of Firebase's helper methods which were utilised across the implementation of this application [17].

Incorporating Next.js and Firestore into our implementation stack allowed me to efficiently build a responsive and high-performance web application that meets the project's functional and non-functional requirements. These technologies played a pivotal role in shaping the Grading Assistant's user interface, backend functionality, and data management processes.

### B. Code Structure

Next.js leverages a file system-based routing approach, whereby the organization of the project closely mirrors the architecture of the application itself. Notably, each folder housed within the 'pages' directory corresponds directly to a distinct page within the application [18]. This organizational strategy is

highly intuitive, affording the ability to systematically arrange code and effortlessly comprehend the application's flow.

Furthermore, Next.js provides a means to establish a uniform application layout through the strategic use of components. Typically, this layout configuration is centralized within a file such as 'Layout.js,' located within the 'Components' directory. This central layout component ensures that the visual design and user experience remain consistent throughout the application, regardless of the chosen route.

One remarkable facet of this architectural approach is its adaptability in addressing user roles and permissions. Tailoring the accessibility of links and features based on a user's role becomes a straightforward task. Integration of role-based access control within 'Layout.js' or other pertinent components permits dynamic customization of the user interface. This empowers the application to provide a personalized and secure experience for each user, thereby elevating the overall quality of the user experience. Additionally, this approach streamlines the development process, as role-specific access logic can be managed from a centralized location, ensuring consistency and cohesion across the entire application.

### C. Core Features and Functions

In this section the core features and functionalities which have been implemented into the application will be discussed.

*1) Home / Landing Page:* When user's first navigate the the 'Grading Assistant' web page they will be on the landing page. This page provides general information about what the application is setting out to do, as well as explanations about how it works, and the four different distance metrics used. It also provides an interface where user's can sign up or sign back in.

*2) Grading:* The 'Grading' page serves as the hub for the core features and functionalities of our web application. This page is only accessible if the user is signed-in. It plays a central role in managing questions, submissions, and the application's core logic. During the development of this page, I frequently leveraged React's 'useEffect' hook to define post-render actions and responses to changing dependencies [19]. This proved invaluable when handling extensive data fetching operations, a common task on this page [20].

The paramount functions of this page involve fetching both questions and texts, followed by sorting the texts based on their relevant distance metric values. To maintain code clarity, maintainability, and readability, complex functionalities have been modularized into separate components, such as 'ImportData' and 'ExportData,' which are invoked as needed [21].

There are multiple different render methods in the file, which all eventually get called in 'RenderContent.js'. By breaking down the rendering logic into separate methods, you create modular code. Each rendering method is dedicated to specific part of UI. For example, the 'RenderTabs' method renders the question selection tabs on the left of the page, as well as 'RenderSlider' which renders the slider bar. This simplifies the implementation process, as you can focus on one piece of functionality at a time. By better organizing the rendering logic, it becomes easier to reuse specific methods in different parts of the application. I also found that by utilising this approach to my implementation, it dramatically helped with debugging code because it was easier to pinpoint the specific problem, as it was generally isolated inside of a separate function.

Once the user has logged in they will be automatically brought to the Grading page. With the knowledge that this page is the main page which graders will interact with on the application, it has been implemented it in a way which is easy to follow and understand. Tool-tips and modals have been implemented to offer information if some user's are struggling to understand how to utilise the page to its fullest. An example of this is the info icon next to the slider bar. When a user clicks this they greeted with a popup explaining exactly how the distance metric values are used with the slider. This ensures that first time users don't have to go back and forth between the grading page and home page, in order to get information about how the system works [22].

There is consistent error handling throughout the implementation of this page. During every key user interaction error handling has been implemented, e.g, for importing data, exporting data, and submitting grades and feedback. Due to the high level of user interaction built into the implementation of the application, we ensured that if the user imports incorrect data it doesn't break the application, as well as that the user is aware of why it isn't working [23]. This ensures a good user experience by directly offering feedback.

This page exhibits multiple states contingent on the presence of data in the database and user access permissions. If no questions have been loaded, users are prompted with an 'Import Questions' button. To ensure a smooth user experience, five questions are pre-loaded when users first access the web application for testing purposes.

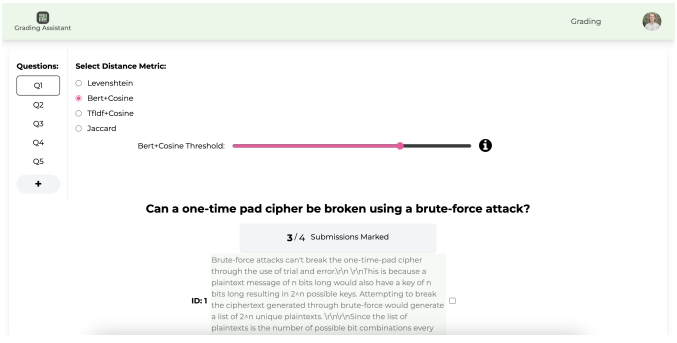


Fig. 4. The above figure shows the Grading Page

*3) Admin:* The 'Admin' page serves as an exclusive interface accessible only to administrators. It encompasses two primary functions: question management and user management, seamlessly organized within separate tabs [24].

- **Question Management Tab:** Within this tab, administrators can efficiently oversee and manipulate the question database. They are presented with a comprehensive table listing all existing questions, accompanied by a count of the submissions associated with each question. Additionally, administrators have the authority to remove ques-

tions from the database, ensuring streamlined database maintenance.

- **User Management Tab:** Similarly, the user management tab offers administrators control over user accounts. It provides a comprehensive list of all registered users, enabling administrators to modify user roles or initiate account removal as necessary. This level of control empowers administrators to uphold user management with ease.

This interface was implemented by using Firebase's 'getDoc', 'deleteDoc', and 'collection' functionalities.

In summary, the 'Admin' page offers administrators an organized and secure platform to manage questions and user accounts. It grants them the ability to oversee and manipulate database entries efficiently while maintaining the integrity of the application and ensuring that users' roles and accounts are appropriately administered.

Only users with admin privileges can access this page. Users with regular permissions will not find a link to this page in the navigation bar. If a non-admin user attempts to access the page by manually typing '/Admin' in the URL, they will be automatically redirected to the application's home page. This setup ensures that access is restricted to authorized personnel, maintaining the security and integrity of the application.

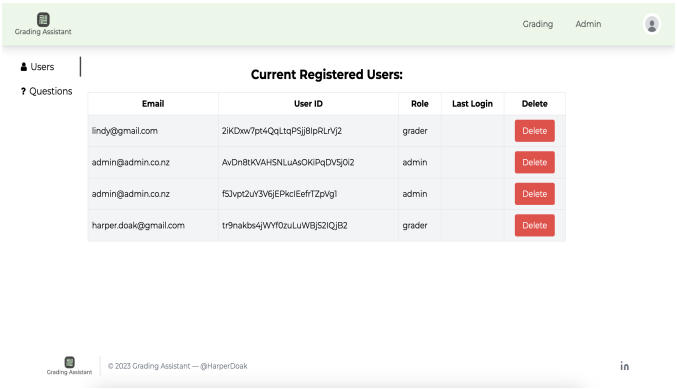


Fig. 5. The above figure shows the Admin Page, showing the user management

*4) Authentication and Authorisation:* Authentication and authorization are fundamental aspects of any web application, ensuring that users have secure access to specific features and data. In this project, Firebase, a robust and versatile cloud-based platform, was chosen to implement these critical functionalities. Firebase offers a comprehensive set of tools for implementing user authentication and authorization, simplifying the process of integrating user management features into the application [25]. The primary components used for authentication and authorization are:

- **Firebase Authentication:** This service provides a secure and straightforward way to allow users to sign up, sign in, and manage their accounts. Firebase supports various authentication methods, including email/password, social media logins (Google, Facebook, Twitter, etc.), and single sign-on (SSO) options.
- **User Management:** Through Firebase Authentication, user accounts and profiles can be managed efficiently. This includes features like password resets, profile updates, and user role management.

When I first implemented these functionalities, I wrote authorization rules for importing data allowing only admin users to be able to do this. However, this was later changed to allow all users so that the user's who were testing my application could test this functionality.

### *D. Database*

In this subsection, I will describe I implemented the database on Firebase's Firestore NoSQL database [15], and more importantly how the data is both stored and used on the application. I have 3 different collections of data, one for questions, grades, and users, respectively.

*1) Questions:* Within the 'Questions' collection, each individual question is represented by a dedicated document. These question documents encompass three pivotal fields: 'name,' 'number,' and 'q-text.' Notably, the user enjoys complete autonomy in defining the content of these fields during the data import process. In the event that a user attempts to import a question with a name or number matching an existing question, an alert is promptly triggered, signifying an unsuccessful import due to the question's preexistence. The 'q-text' field, the core repository for the actual question that students respond to, encapsulates the query itself.

Within each question document, you'll find a nested subcollection known as 'submissions,' housing all the submission data specific to that particular question. Every question document boasts its dedicated 'submissions' sub-collection. Delving into the 'submissions' collection, you'll discover a document for each student who has submitted an answer to the given question. Within these documents, you'll encounter nine essential fields: 'id,' 'question-id,' 'text,' 'value1,' 'value2,' 'value3,' 'value4,' 'marked,' and 'active.' The 'id' field signifies the student's unique ID for their submitted text, while 'question-id' corresponds to the ID of the question they've addressed. The 'text' field serves as the repository for their response. The four 'value' fields pertain to the four distance metrics employed in this application. Both 'marked' and 'active' fields are Boolean, initially set to 'false' upon data import, and serve as markers to identify submissions yet to be assessed.

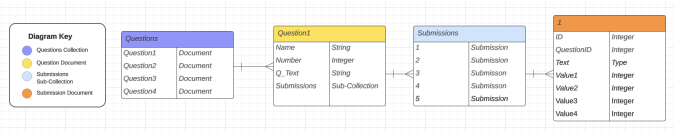


Fig. 6. The above figure shows a diagram of how the questions and submissions are structured

*2) Grades:* Within this 'Grades' collection, we meticulously store all the crucial grade and feedback data assigned by markers to individual submissions. When a user initiates the submission process by clicking the submit button, the system efficiently imports and records both the grade and the corresponding feedback in this collection. In addition to the grade and feedback information, we also capture and store

an array of 'active-Ids' in the database. This array serves as a key element in distinguishing each submission from the rest. It includes both the 'question-ID' and the 'student-ID' of every selected submission. This feature is particularly essential, as it allows graders to simultaneously evaluate multiple submissions while maintaining a clear and accurate distinction between them.

*3) Users:* Upon a user's initial registration with the application or their first sign-in via Google, their particulars are recorded and seamlessly integrated into the Users collection within Firestore. This process initiates the automatic generation of a document in this collection, featuring three essential fields: 'email,' 'role,' and 'uid. The 'role' field is set to grader by default when creating an account. If at some point a user wanted to gain admin authorization their account role would have to be manually changed in order for that user to be able to utilise the admin interface on the application. When an account is first made a random 'uid' is generated. This id is used for authentication and data association, ensuring that each user's data is appropriately associated with their profile.

The firestore database which we have utilised for this application employs real-time updates. What this means if that if the database is updated whilst a user is on actively on the application marking, during the next refresh the content on the page will reflect the most up to date state of the data. This feature of the database ensures data consistency across all clients. When one client makes a change, the database ensures that all other connected clients reflects the updated information without any conflicts.

### *E. Calculating the Distance Values*

In order to calculate the distance values for all four distance metrics, a python script for each metric was developed. Each python script pulled the question submissions from a JSON file, as well as a reference answer in order to calculate a value for comparing each submission text. These metrics provide insights into the degree of similarity or dissimilarity between a reference text and multiple text submissions.

*1) Jaccard:* The Jaccard similarity is calculated by dividing the size of the intersection of the sets of words by the size of their union. The resulting value ranges from 0 to 1, with 0 indicating no similarity and 1 indicating perfect similarity.

```
# Calculate Jaccard similarity for each text
for i, text in enumerate(texts):
    text_tokens = set(text.lower().split())
    intersection = len(reference_tokens.intersection(text_tokens))
    union = len(reference_tokens.union(text_tokens))
    jaccard_similarity = intersection / union
    question_similarity_scores[f"Text {i+1}"] = jaccard_similarity
```

Fig. 7. The above figure shows the code to calculate the Jaccard value.)

*2) Levenshtein:* Levenshtein Distance, also known as Edit Distance, quantifies the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one text into another. The script for calculating Levenshtein Distance is as follows: The levenshtein distance function calculates the distance between two strings s1 and s2. This distance represents the minimum number of character operations

```
def levenshtein_distance(s1, s2):
    if len(s1) < len(s2):
        return levenshtein_distance(s2, s1)

    if len(s2) == 0:
        return len(s1)

    previous_row = range(len(s2) + 1)
    for i, c1 in enumerate(s1):
        current_row = [i + 1]
        for j, c2 in enumerate(s2):
            insertions = previous_row[j + 1] + 1
            deletions = current_row[j] + 1
            substitutions = previous_row[j] + (c1 != c2)
            current_row.append(min(insertions, deletions, substitutions))
        previous_row = current_row

    return previous_row[-1]
```

Fig. 8. The above figure shows the code to calculate the Levenshtein.)

(additions, deletions, substitutions) required to transform s1 into s2.

The make the similarity scores more intuitive and comparable, we normalize the Levenshtein distance. Normalization scales the distance values to a range between 0 and 1, where 0 indicates no similarity and 1 indicates perfect similarity. This is achieved by dividing the Levenshtein distance by the maximum length of the two strings being compared and subtracting the result from 1.

*3) BERT+Cosine:* BERT + Cosine first tokenizes and encodes the reference text from the JSON file, then calculates BERT embeddings for each of them. The cosine similarity between the reference text and each text in the set is determined, providing a quantitative measure of text similarity. Below is script used in my implementation to generate these values for the submissions.

```
# Calculate BERT embeddings and Cosine similarity for each text
for i, text in enumerate(texts):
    text_tokens = tokenizer(text, return_tensors='pt', padding=True, truncation=True)
    text_embedding = model(**text_tokens).last_hidden_state.mean(dim=1).squeeze()
    cosine_sim = cosine_similarity(reference_embedding.detach().numpy().reshape(1, -1), text_embedding.detach().numpy().reshape(1, -1))[0][0]
    question_similarity_scores[f"Text {i+1}"] = cosine_sim
```

Fig. 9. The above figure shows the code to calculate the BERT+Cosine.)

*4) TfIdf+Cosine:* The process for calculating the TfIdf+Cosine values begins by vectorizing both the reference text and the collection of texts using the TF-IDF method, which captures the importance of individual words within the texts relative to their frequency across the entire dataset. Cosine similarity is then computed between these TF-IDF vectors, yielding a numerical measure of similarity.

```
# Vectorize the texts using TF-IDF
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([reference_text] + texts)
# Calculate cosine similarity between the reference text and each text
cosine_similarities = cosine_similarity(vectors)
# Initialize an empty dictionary to store similarity scores for this question
question_similarity_scores = {}
# Calculate cosine similarity scores
for i, text in enumerate(texts):
    similarity = cosine_similarities[0][i + 1]
    question_similarity_scores[f"Text {i+1}"] = similarity
```

Fig. 10. The above figure shows the code to calculate the TfIDF+Cosine.)

## VI. Evaluation

The evaluation of this project was broken up into two different parts. Firstly, was performance testing of the application, followed by user testing.

### A. Component Testing

Several critical components and functionalities of the application underwent rigorous performance testing to ensure optimal operation. These assessments encompassed authentication procedures, including login and registration, as well as the retrieval of data from Firebase.

In this evaluation, I employed Cypress, an open-source end-to-end testing framework specifically engineered for assessing web applications. End-to-end (E2E) testing constitutes a software testing methodology that concentrates on validating the seamless functioning of an application from start to finish, simulating real user interactions to verify its overall functionality. Cypress proves to be an ideal choice for Next.js applications, given that its test scripts are composed in JavaScript or TypeScript.

The initial test focused on the Grading page, which serves as the application's main interface. Among the pivotal aspects examined was the performance and speed of data retrieval from the Firestore database. Upon the page's load, all questions and submissions stored within the database are loaded into the web application. Impressively, it took a mere 2.8 seconds to fully load the questions database for the initial access. It is worth noting that the database size is relatively modest, comprising only five questions and five submissions for each question.

Moreover, the process of submitting grades and feedback to the Grades database occurred almost instantaneously, demonstrating the efficiency and robustness of the application's data manipulation capabilities.

### B. User Testing

The main goal of user testing for this application was to test the usability and user interface of the application.

A questionnaire was developed which covered multiple different aspects of both the usability and functionality of the web application. There were 8 different sections included which were general feedback, user registration, question importing and exporting, grading, user interface and design, performance, and any additional comments or observations. Ensuring that the questionnaire had clear objectives and goals, so that results would focus directly on the usability and UI of the application. Six different people filled out the questionnaire after spending time using different functionalities of the application. These people all had different disciplines, ranging from software engineering, to non-technical backgrounds. By including individuals from different backgrounds it provides me with unique perspectives of the user interface and how the application functions overall.

Six individuals spent 10 minutes on the application, testing all of the different functionalities which were developed. After this 10 min period, the questionnaire was given to the individuals to fill out. The results which received were varied, which was expected due to the difference in knowledge of UX best practices, and web development in general. However, there were also similarities with some of the responses which were received. Here are the results:

*1) Main Results:* Feedback collected from the six individuals who participated in testing the application revealed several recurring themes and opportunities for enhancement. Many users encountered challenges during the data importation process, struggling to ensure that their data was in the correct format. Even when they believed they had followed the right format, the import frequently failed. Users strongly suggested that the application provide more informative feedback on data import issues, proposing the implementation of a more robust validation system to assist users in identifying and resolving specific data format errors.

Users also found the process of calculating distances using the four Python scripts and then uploading the results into the database to be time-consuming and complex. All users recommended integrating the distance calculation into the importation process, transforming the application into an end-to-end solution for added convenience and efficiency.

On a positive note, all users praised the application's user interface design, noting its modern and clean aesthetics. They specifically appreciated the application's judicious use of white space, which contributed to an elegantly designed layout that was both visually pleasing and well-organized.

In summary, these valuable insights from the testers underscore the need to improve the data importation process, streamline distance calculations, and maintain the positive aspects of the user interface design. Implementing these suggestions will undoubtedly result in a more user-friendly and efficient application.

## VII. Future Work

This section outlines the vision for further elevating the capabilities and user experience the SAGA.

*1) Further Distance Metric Implementation:* The first part of potential future work is to change the way that both the distance metric is calculated and utilised on the application. Currently, the distance values are calculated by running the submissions through the 4 scripts (1 for each distance metric). This clearly isn't a very optimal way in order to collect these values, as it means that if a user wants to import more questions to the system, they have to also have the 4 python scripts on their computer. This additional requirement introduces complexity and an extra step in the importation process, potentially hindering the seamless integration of new questions or submissions. Following the user testing of the application, this was one of the main points which users mentioned could be improved. Recognizing the need for efficiency and user-friendliness, I envision a more integrated and streamlined solution by changing it to be fully included within the application. When the import button is selected and submissions have been uploaded to the database, the calculation of the distance values could be included in this process. There are many different benefits to this integration, firstly is a simplified user experience. By eliminating the

need for external Python scripts simplifies the user experience. Users can import questions and submissions without worrying about additional software requirements, reducing potential barriers to entry. As the grading application continues to grow, this integrated approach ensure scalability. Users can confidently import larger datasets and perform calculations without concerns about any external dependencies.

*2) Approved Data type:* In its current state, the application is designed to accept JSON files exclusively for data importation. While JSON is a versatile and widely used data format, I recognize that diversifying data type support can further enhance the application's flexibility and usability.

The decision to accept only JSON files for importation has been driven by its popularity and ease of use. However, this does create some challenges:

- User Constraints: Users are required to convert their data into JSON format before importing it into the application. This adds an extra step and be cumbersome, particularly for those who are accustomed to using different data formats.
- Scalability: As the application evolves and accommodates various educational scenarios, it may encounter diverse data sources that benefit from different data formats.
- Data Compatibility: Some educational institutions or data sources may store information in formats other than JSON. This limitation restricts the application's compatibility with a broader range of data sources.

To address these challenges and provide a more versatile experience, the future work agenda includes expanding the range of approved data types that the application can accept for importation. By accommodating various data formats, we reduce the need for users to perform data format conversions before importing. This streamlines the data importation process and enhances user convenience. Also by expanding the approved data type support aligns with our commitment to user-centric design. Users can import data in a format that suits their needs, contributing to an overall more user-friendly experience.

## VIII. Conclusion

In summary, the development of this innovative application represents a significant step forward in enhancing the teaching experience for educators and academic professionals, allowing them to focus more on teaching and less on grading. The application's thoughtful design, tailored to the workflow of its users, showcases a contemporary user interface characterized by clean aesthetics, well-considered spacing, and a consistent color scheme, all of which contribute to an outstanding user experience. Our application follows a client-server architecture, with Next.js as the client and Firebase serving as the hosting platform for the server, ensuring a robust and efficient structure. We have meticulously implemented the core functionalities in strict adherence to predefined functional requirements, guaranteeing that the application effectively fulfills its intended purpose. Furthermore, we have conducted thorough user testing to gather valuable insights from our users, allowing us to refine and enhance the application further. This iterative process has provided valuable feedback for both the strengths and potential areas of improvement. Looking ahead, this project opens the door to exciting opportunities for future development and expansion, with the goal of continually enhancing the educational experience and streamlining the tasks of educators and academic professionals.

## References


[1] B. Spencer, "The Importance of Timely and Effective Feedback," blog.teamsatchel.com, Mar. 09, 2017. https://blog.teamsatchel.com/the-importance-of-timely-and-effective-feedback

[2] C. Li and F. Lalani, "The COVID-19 Pandemic Has Changed Education Forever," World Economic Forum, Apr. 29, 2020. https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/

[3] S. Grashchenko, "Levenshtein Distance Computation — Baeldung on Computer Science," www.baeldung.com, Aug. 03, 2020. https://www.baeldung.com/cs/levenshtein-distance-computation

[4] J. Briggs, "BERT For Measuring Text Similarity," Medium, Sep. 02, 2021. https://towardsdatascience.com/bert-for-measuring-text-similarity-eec91c6bf9e1

[5] A. Zhu, "Understanding TF-IDF and Cosine Similarity for Recommendation Engine," Geek Culture, Apr. 04, 2023. https://medium.com/geekculture/understanding-tf-idf-and-cosine-similarity-for-recommendation-engine-64d8b51aa9f9 (accessed Oct. 12, 2023).

[6] "Jaccard Similarity," www.learndatasci.com. https://www.learndatasci.com/glossary/jaccard-similarity/

[7] "Gradescope — Save time grading," @gradescope, 2000. https://www.gradescope.com/

[8] "GradeMark — Blackboard Basic," help.turnitin.com. https://help.turnitin.com/feedback-studio/blackboard/basic/instructor/grading/grademark.l (accessed May 29, 2023).

[9] N. Suzen, A. N. Gorban, J. Levesley, and E. Mirkes. "Automatic short answer grading and feedback using text mining methods." https://www.sciencedirect.com/science/article/pii/S1877050920302945 (accessed Mar. 31, 2023)

[10] "Authentication and Authorization," CyberArk. https://www.cyberark.com/what-is/authentication-authorization/

[11] GeeksforGeeks, "Client-Server Model," GeeksforGeeks, Oct. 23, 2019. https://www.geeksforgeeks.org/client-server-model/

[12] Interaction Design Foundation, "What is User Centered Design?," The Interaction Design Foundation, 2019. https://www.interaction-design.org/literature/topics/user-centered-design

[13] Vercel, "Next.js by Vercel - The React Framework," nextjs.org. https://nextjs.org/

[14] tailwindcss, "Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.," tailwindcss.com, 2023. https://tailwindcss.com/

[15] Google, "Firebase," Firebase, 2023. https://firebase.google.com/

[16] "Setting up Firebase," Firebase, 2015. https://firebase.google.com/docs/web/setup

[17] "Export and import data," Firebase. https://firebase.google.com/docs/firestore/manage-data/export-import

[18] "Building Your Application: Routing — Next.js," nextjs.org. https://nextjs.org/docs/pages/building-your-application/routing (accessed Oct. 16, 2023).

[19] "Using the Effect Hook – React," reactjs.org. https://legacy.reactjs.org/docs/hooks-effect.html

[20] H. Doak, "Fetch Data Now works," Gitlab. https://gitlab.ecs.vuw.ac.nz/course-work/project489/2023/doakharp/semi-automated-grading-project/-/commit/8b03dfcb2158183ec5f9a8505e6be91b2d58a166

[21] H. Doak, "Made some changes to Importing Data" GitLab. https://gitlab.ecs.vuw.ac.nz/course-work/project489/2023/doakharp/semi-automated-grading-project/-/commit/b2af993a16b5d46f14ec4f9fb2277595892b731d (accessed Oct. 15, 2023).

[22] "Grading Page Code" GitLab. https://gitlab.ecs.vuw.ac.nz/course-work/project489/2023/doakharp/semi-automated-grading-project/-/blob/Development-Part1/gradeassistantnext/src/app/Grading/page.jsx (accessed Oct. 15, 2023).

[23] “Added Error Handling to Button” GitLab. https://gitlab.ecs.vuw.ac.nz/course-work/project489/2023/doakharp/semi-automated-grading-project/-/commit/ddb781a369dbd875c222965fd7fe91e087120219 (accessed Oct. 15, 2023).
[24] “Admin Page” GitLab. https://gitlab.ecs.vuw.ac.nz/course-work/project489/2023/doakharp/semi-automated-grading-project/-/blob/Development-Part1/grade-assistant-next/src/app/Admin/page.jsx?ref-type=heads (accessed Oct. 16, 2023).
[25] [1]“Register Page” GitLab. https://gitlab.ecs.vuw.ac.nz/course-work/project489/2023/doakharp/semi-automated-grading-project/-/blob/Development-Part1/grade-assistant-next/src/app/Register/page.jsx (accessed Oct. 16, 2023).