Software Requirements Specification for

Version 1.0

02.15.2024

Team Members: Mofoluwake Adesanya, Ethan Dettmann, Rizwan Khan, Rohit Poolust, Brandon Swirsky

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

This specification describes the purpose, interfaces, software components, operating platforms, and functional and nonfunctional requirements of **Pollaris, V*ersion 1.0***.

1.2 Document Conventions

In this specification, texts are surrounded by double quotes (“”) to indicate text in a button of a web page. Any italicized or bolded and italicized text indicates an important word or phrase to draw the attention of the reader. The stated priorities of listed features are assumed to be inherited by detailed requirements listed under it. Priorities range from High, Medium and Low priority. In this specification, only high priority requirements are listed. High priority indicates that a feature is on the critical path of development.

1.3 Intended Audience and Reading Suggestions

This specification is intended for:

- Developers and testers seeking to understand Pollaris’s design and architecture,
- Project managers tracking progress of Pollaris’s development and versions,
- Marketing staff using this document to promote Pollaris to clients and organizations
- Users verifying new updates and features, and
- Documentation writers referencing the product’s use.

1.4 Project Scope

Pollaris is a live polling platform that facilitates seamless in-class interactions between Instructors and Students. In lecture halls with hundreds of Students in a single section, Instructors struggle to keep Students engaged and participating. A limited number of Students' opinions are heard and valued, therefore stifling learning and teaching efforts of both parties. As a solution, live polling and quizzes can be embedded in presentations to both check on attendance and test understanding of course materials. Pollaris provides live polling, Q&As, quizzes and more. It offers solutions to Instructors who want to manage classroom discussions effectively and to Students to engage with the course materials. Pollaris provides Instructors with options to create quizzes, edit upcoming quizzes, view past quizzes and generate live class reports. Students can enjoy an interactive class session and make the most of their class time with Pollaris. It is straightforward and accessible to use for both Students and Instructors.

1.5 References

The following interface and software components that Pollaris is built upon are referred here:

- Firebase
- WebSockets
- HTTP

2. Overall Description

2.1. Product Perspective

Pollaris is an online service that runs on web browsers with database support from Firebase. It allows teachers to create polls that Students can complete on their devices within a classroom setting.

2.2. Product Features

With Pollaris, Instructors can design interactive quizzes for Students. During a session, Students can join a quiz created by the Instructor from their device by entering their full name, the quiz's name, and a generated code for that quiz. Questions are answered live by each Student. Pollaris analyzes chosen answer choices for each question and displays a bar chart of the chosen answers anonymously. Instructors can generate reports later of Students' scores accordingly.

2.3. User Classes and Characteristics

The primary user classes are Instructors and Students. Instructors create and/or log into an account. They can manage a preexisting quiz or make a new quiz. They can obtain a unique code to their quiz for Students to participate in that quiz. Instructors can view the collective results of people who have taken the quiz. Instructors can generate reports for a classroom or a published quiz with each participating Students' full name and score. Students enter their full name and a code to participate in a quiz. Students choose answers to each question the Instructor renders. Upon completion, Students receive

individual reports with their names, overall score and a listing of questions answered incorrectly.

2.4. Operating Environment

Pollaris's database is supported by Google's Firebase. Pollaris runs on any computing device with a web browser including Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge, and with access to an Internet connection. Instructors and Students can use a system that supports the following operating systems: Apple MacOS, Apple iOS, Microsoft Windows, and more.

2.5. Design and Implementation Constraints

Constraints related to the development of Pollaris include knowledge with the required tools like Firebase, and front-end programming languages like Javascript, HTML and CSS. Pollaris is written and documented in the American English language so developers must have basic to fluent knowledge of the English language.

2.6. User Documentation

User documents are made available on version control systems like Github, and identify the components and functions of components within the system. Documentation is also available along with the web application and is accessible on the Work Page and Loading page of Instructors and Students respectively.

2.7. Assumptions and Dependencies

The third-party component of Pollaris is its Firebase-powered database. Since the database is managed and maintained by a third-party, Pollaris can be subject to

downtimes and any constraints that Firebase applies to its system. This might obstruct task flow by user classes which users are notified of well before. Pollaris does not make any assumptions that can affect its functionality.

3. System Features

The major services provided by Pollaris are determined by the user class. Functional user classes are Instructors and Students. In this specification, each system feature specifies the user class it is provided to. Each feature can be understood by looking at Figures 1 and 2.

3.1 Role-based Access

3.1.1 Description and Priority

This feature is accessed from the landing page of Pollaris for both Instructors and Students. Role-based access is of *High priority* because it provides an interface for users to choose their user class that suits the tasks they wish to accomplish.

3.1.2 Stimulus/Response Sequences

Here is a list of user actions and system responses that define role-based access:

- User clicks the Instructor role and is taken to the Instructor interface.
- User clicks the Student role and is taken to the Student Interface.

3.1.3 Functional Requirements

R-1: A button with the text “Instructor” and another button with the text “Student”

3.2 User Authentication

3.2.1 Description and Priority

This feature differs between Instructor and Student. Instructors are taken to a page where signing in or logging in, if the instructor is not registered, is possible. HTTPS authentication is only available to Instructors because this user class can utilize a specific set of features needed to coordinate their sessions. Students are taken to a page where they can fill a form to enter a poll or quiz. This feature is of *High priority* for users to access their work spaces according to roles.

3.2.2 Stimulus/Response Sequences

Here is a list of user actions and system responses that define user authentication:

- If the Instructor is not registered, a login page is provided to create a new account.
- If the Instructor is registered, a form to input a username and password is presented.
- The Instructor fills in their credentials and is authenticated to access their personal work page.
- A Student fills in a form that requests a name and code to access a quiz or poll.

- A Student also fills in their full name to be identified by their Instructor.

3.2.3 Functional Requirements

R-2: Create Sign Up/ Login Page with user input form for an Instructor to fill

R-3: Create “Login” button on that page to submit the filled out form

R-4: Produce error messages for invalid credentials

R-5: Provide “Forgot Password?” hyperlink to allow Instructor to request a password reset

R-6: Create “Sign Up” button for Instructors who are not registered

R-7: Create Sign Up interface on the same page for Instructors who are not registered

R-8: Create a Student page with a user input form for Students to fill in their full name, quiz name, and code

R-9: Provide a “Join” button on Student page to enter an Instructor’s session given the code

3.3 Quiz Processing

3.3.1 Description and Priority

This feature is made available to only Instructors to create a quiz for an upcoming session. Quiz Processing includes creating, editing, and publishing a quiz. Pollaris allows Instructors to create multiple questions with a maximum of ten answer choices. Each question can be of two types:

- Multiple choice questions

- “Check All That Apply” questions

This is of *High priority* as this is the purpose of Pollaris.

3.3.2 Stimulus/Response Sequences

Some functionalities in quiz processing can be accessed on both the work page and the quiz page. Here is a list of user actions and system responses that define quiz processing:

- Creating A Quiz:
    - The Instructor clicks on “Create” on a blank quiz or poll on the work page to go to the page where a quiz is created.
    - On that page, a single question frame exists to start off the quiz or poll. The Instructor adds new questions by pressing the plus button.
    - For each question, the Instructor clicks an option plus sign to add answer choices.
- Editing A Quiz:
    - An unnamed quiz is automatically named “Sample Test” by default.
    - The Instructor clicks on “Edit” on an unpublished quiz or poll to edit and modify a quiz or poll.
    - The Instructor clicks on a question to edit it.
    - The Instructor clicks on the drop down menu of the question to change its type.

        - The Instructor clicks on an answer choice after clicking on a question to edit it.
    - Publishing A Quiz:
        - The Instructor clicks the “Publish” button to publish a quiz.
        - A pop-up menu comes up to allow the Instructor to *rename* the quiz.
        - The “Publish” button on this pop-up menu makes a quiz available to be accessed by Students (*where a code is made public*).

3.3.3 Functional Requirements

R-10: Create a Work page for Instructors with a list of quiz icons with a “Sample Test” auxiliary quiz for reference

R-11: Implement a “Create” button on the work page to create a new quiz

R-12: On Quiz Page, create an auxiliary question for Instructors to create questions

R-13: Create a drop down menu, available for each question, for the type of question to be created

R-14: Create a “+” button at the bottom left of each question to add a new question

R-15: Create a “+” button at the bottom right ***in*** each question to add a new option

R-16: Create a “Delete” button at the bottom of the Quiz page to delete a quiz

R-17: On the work page, create an “Edit” button on the icon of an existing quiz that is not published

R-18: Create a “Publish” button at the top right corner of the Quiz Page that prompts a Call To Action.

R-19: Create a pop-up menu that allows Instructors to rename the quiz from “Sample Test” and finally make it public.

3.4 Code Generator

3.4.1 Description and Priority

The code generator allows an Instructor to copy a pre-generated code that makes a published quiz public. The pre-generated code is a 5-digit numeric code generated randomly by the system. This code is made public to Students so that they can access the published quiz.

3.4.2 Stimulus/Response Sequences

Here is a list of user actions and system responses that define code generator:

- The Instructor clicks “Publish” after creating a quiz and is presented with a pop-up menu.
- The Instructor clicks on “Copy Code” from the pop-up menu and receives the code in their device’s clipboard.

- The Instructor can access the code on the “Copy Code” button of each quiz on the work page.

3.4.3 Functional Requirements

R-20: On the pop-up menu produced by clicking on the “Publish” button, create a “Copy Code” button to copy the pre-generated code for the created quiz

R-21: On the work page, create a “Copy Code” button on each quiz that is accessed by hovering over the quiz icon.

3.5 Quiz Engagement

3.5.1 Description and Priority

This feature is provided to Students and Instructors. This allows Students to view and answer quiz questions. Assuming Students have received the access code for a quiz, this feature allows Students to interact in a session via a quiz on the interface. This feature can also be harnessed to track attendance of Students.

3.5.2 Stimulus/Response Sequences

Here is a list of user actions and system responses that define quiz engagement:

- A Student clicks “Join” on logging in and can enter a published quiz.
- The Student sees each question as a slide. The Student clicks on an option and the question closes.

- Other parts of engagement like displaying a poll are facilitated by the Instructor through their interface.

3.5.3 Functional Requirements

R-22: Create an interface for Students to interact with the quiz

R-23: Create color-coded buttons representing each answer choice

3.6 Poll Generator

3.6.1 Description and Priority

This feature allows both Students and Instructors to view a poll based on an answered question. This is of *High priority*.

3.6.2 Stimulus/Response Sequences

Here is a list of user actions and system responses that define poll generator:

- A Student clicks an option which is saved by the system.
- The Instructor prompts a poll to be displayed ending the timer of the current question.
- The poll is automatically displayed on the screens on both Instructor and Students.
- The Instructor initiates the next question on their interface by clicking “Next” underneath the poll on their screen.
- The “Next” button is not made available on the Student interface.

3.6.3 Functional Requirements

R-24: Provide a timer for the Instructor to view the overall time being spent on a question

R-25: Create a "Next button for the Instructor to end the time and prompt the poll display

R-26: Gather answer choices chosen by Student users

R-26: Utilize Google Analytics on Firebase to create a poll

R-27: Display poll on the screen on all users and give Instructor control to change screen by changing to the next question or finish the quiz.

4. External Interface Requirements

4.1 User Interfaces

Pollaris has the following user interfaces:

- Home Page:
    - Pollaris logo is in the middle of the screen,
    - Student and Instructor rectangular access buttons are beneath the logo, and each access button connects the user to the respective interface depending on their roles.
    - User roles include Instructor and Student (Fig. 1).
- Instructor
    - Login Page:
        - A login form with username and password input boxes labeled appropriately.

- Beneath the form is the Login button which takes the instructor to their Work Page.

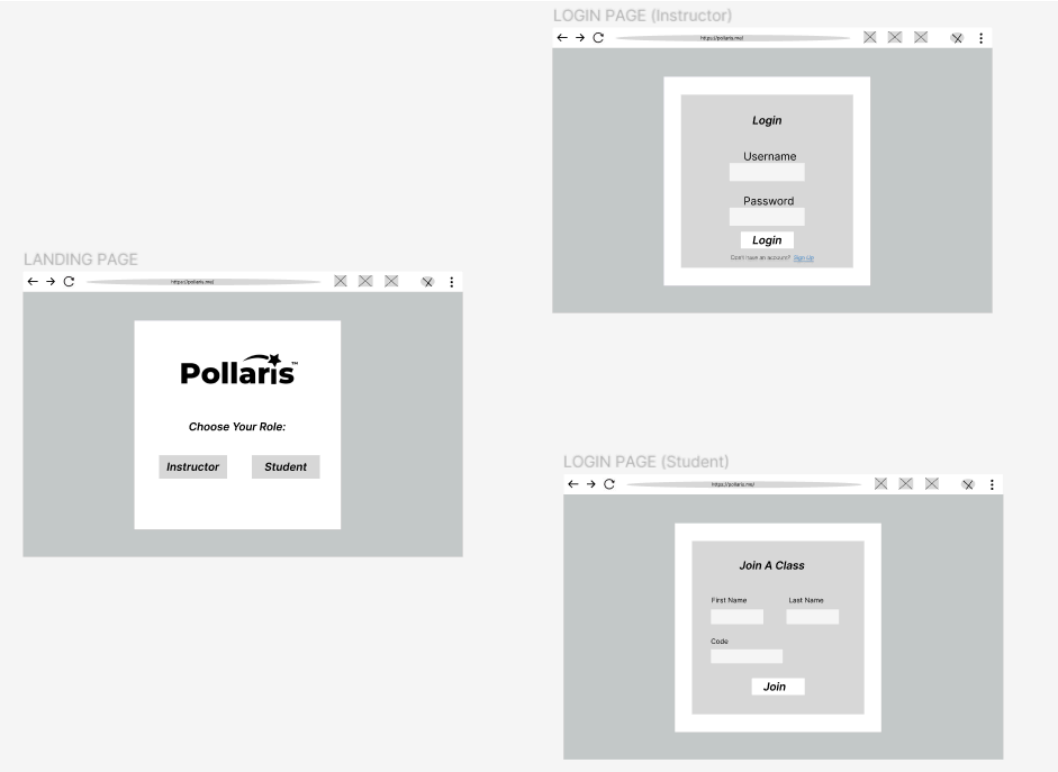

Figure 1 : Home Page, and Sign-In pages for Instructor and Student

- Work Page
  - Quizzes are displayed on this page and represented as squares.
  - Existing quizzes are listed first followed by an empty square with the text “Create Quiz” allows an instructor to go to the Question Creator.
  - Existing quizzes that are not published have an edit button that says “Edit” and is shown when the mouse hovers over it. There is a preset Sample Test to demo how a quiz is set up.

        - After a quiz is created, it is displayed on the Work Page.
        - Quizzes can be ordered according to *Date Created, Alphabetical,* or *Last Edited*, and filtered according to *Published* and *Not Published*.
        - A “Host Quiz” button takes you to the Host Quiz page.
    - Question Creator
        - A “Publish” button at the top of the screen allows the instructor to publish the created quiz. Clicking this button prompts you to name the quiz, saves the questions you created, and takes you back to the home screen.
        - A blank question is presented with a text box to type in a question, an option to choose the type of question being created, a list of answer choices that can be filled in, and a checkbox next to each answer choice indicating whether or not it is the correct answer.
        - The types of questions that can be created are multiple choice questions (one correct answer) and “check all that apply” questions. A question can have multiple or no correct answers. A dropdown menu allows the user to select if the question is multiple choice (picking any of the correct answers counts as correct), or a “select all that apply” question (must pick all correct answers to count as correct)

        - A question by default must have at least two possible answers. A plus button underneath the lowest answer choice allows the user to add up to two more answers
        - A button to the bottom right of the question that says “Add Question” creates another blank question template beneath the existing question
    - Quiz Page
        - Displays the title of the quiz and a five digit code that Students are able to input on their sign in page to join
        - A start/progress button that allows the user to start the quiz and continue on to the next question
        - When the quiz starts, the screen displays a timed question and once the timer completes, a poll is displayed according to chosen answer choices
- Student:
    - Login Page:
        - Four textboxes are presented, two for entering the Student’s first and last name, one for the poll name, and one for the poll ID.
        - A “Join” button is displayed beneath. If the code is correct, the student will begin taking the poll. Otherwise, there is a popup error message that asks the user to enter a valid code

        - A back button is in the top left corner to take the user back to the Initial Page
    - Quiz Pages
        - The Instructor displays a single question and it is extended to the Students' screen with answer choices in a similar question format on the Question Creator screen (Fig. 2).

Below are wireframes of the pages on Pollaris's critical path:

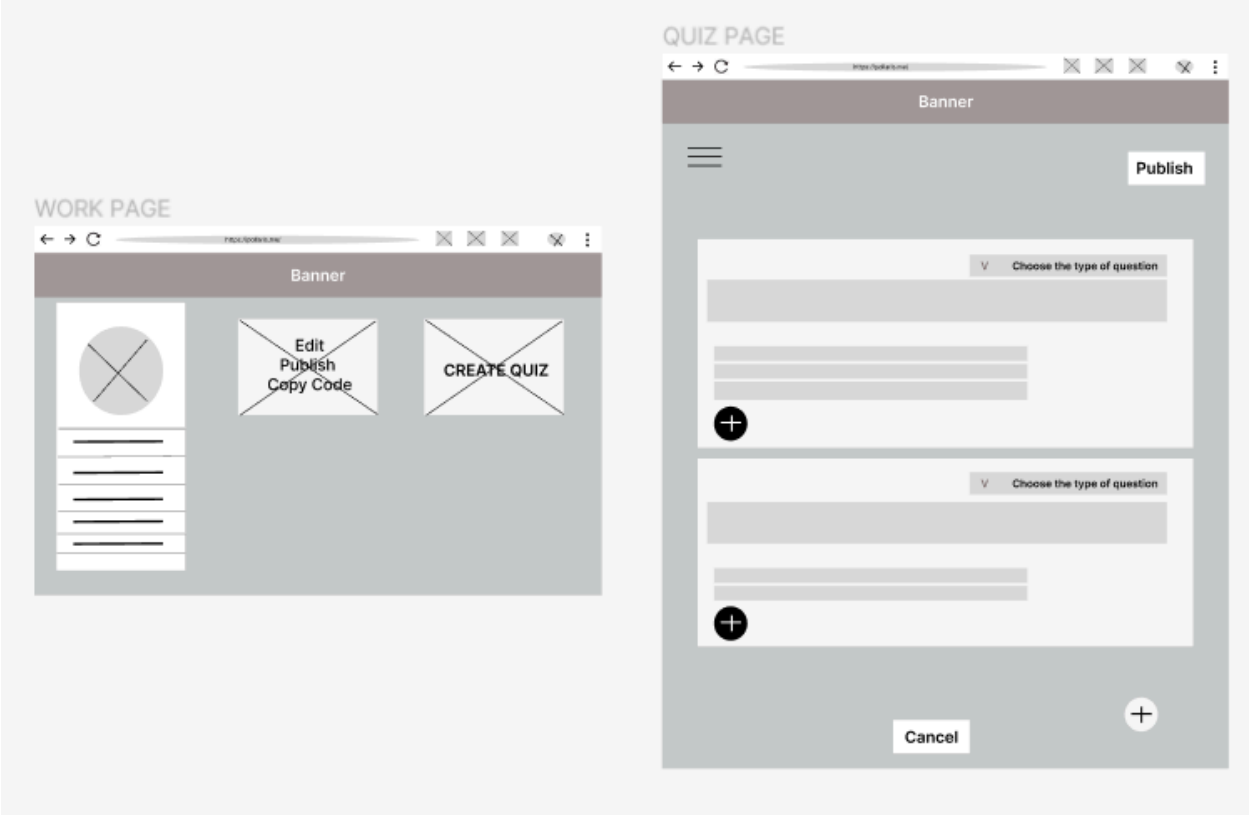

Figure 2 : Interface Flow for Instructor showing Work Page, and Question Creator

4.2 Hardware Interfaces

Pollaris has no hardware components. Its supported device types include computing devices with an access to Internet connection and a web browser. This includes laptops, desktops, mobile phones, tablets, notepads and more that have access to Internet connection and a web browser like Mozilla Firefox, Microsoft Edge, Safari and more.

4.3 Software Interfaces

Pollaris uses Firebase Realtime Database:

- Operating System: Windows and mobile-compatible
- Tools: Firebase SDK for JavaScript

- Connections and Communications:
  - Data Items Coming Into the System:
    - User authentication requests (Login: Username and password).
    - Quiz creation/editing requests.
    - Student participation requests (e.g. joining quizzes, submitting answer choice).
    - Real-time updates on quiz questions and answers during live quizzes.
    - Student participation status (success/failure).
- Services Needed:
  - Authentication Service: This is a part of Firebase's framework.

    - Real-time Database Service: Stores quiz data and student responses, and facilitates real-time updates.
    - Cloud Functions Service: Executes backend logic in response to events triggered by Firebase features.
    - Internet Domain: Required to host the website and be a gateway to the server that hosts all our files. The domain Pollaris has been acquired to host the web application. This domain is valid for one year with an expiry time of February 2025.
- Nature of Communications between database and server:
    - Client-Server Communication: Frontend interfaces communicate with Firebase services (authentication, real-time database) using HTTP requests and responses.
    - Real-time Data Synchronization: Firebase Realtime Database ensures synchronized data updates across clients in real-time.
    - Backend Logic Execution: Firebase Cloud Functions execute backend logic in response to specific events triggered by Firebase services.
- Data Sharing Mechanism:
    - Real-time Database Integration: Data shared across software components includes user authentication details, quiz questions, student responses, etc.
        - Data Type being used:
            - Username
            - Password

        - Random Game Pin generation
        - Full Name
        - Questions
        - Answer choices
        - Responses for each individual student
        - Unique ID (Created by framework)
        - Quiz Report
- Implementation Constraints:
    - Firebase SDK Integration: The system must utilize the Firebase SDK for JavaScript to interact with Firebase services. This ensures compatibility and seamless integration with Firebase features and services.
    - Use of Javascript for integration offers an advanced front end design for Pollaris

4.4 Communication Interfaces

The communication interfaces include communication functions between client and server. This include web sockets that facilitate communication as described below:

- Full-duplex communication channels over a single TCP connection are provided by WebSockets; this allows for real-time, bidirectional dialogue between clients and servers.
- WebSockets, unlike HTTP that adheres to a request-response model, permit initiation of communication at any time by both clients and servers.

- Real-time chat applications, live updates, online gaming and collaborative editing tools commonly utilize WebSockets.

- Its Advantages:
    - Real-time: WebSockets enable low-latency, real-time communication between clients and servers. This is crucial for the quick transmission of polling/quiz data.
    - Unlike HTTP, WebSockets eradicate the necessity of recurring HTTP header overhead for each message; this creates more efficiency.
- Its Limitations:
    - Configure server to support WebSockets and manage WebSocket connections as well as message processing on the server side.
    - Modern web browsers widely support WebSockets, but full compatibility may not be ensured in older browsers and certain network proxies.

5. Other Nonfunctional Requirements

5.1 Security Requirements

Designing a system that has multiple communication interfaces—WebSockets, HTTP, and Firebase—necessitates some crucial security requirements to help safeguard the application. These requirements are highlighted below:

- WebSockets:

    - Implement Transport Layer Security (TLS) to encrypt data transmitted over WebSocket connections; this action prevents eavesdropping and tampering.
    - To ensure only authorized users establish connections, we authenticate clients connecting to the firebase database and server.
    - Enforce access control policies: This action restricts WebSocket connections and actions and is based specifically on the roles and permissions of users.
    - Validation of Input: Actively validate and sanitize the data received from WebSocket clients, effectively preventing injection attacks along with other potential security vulnerabilities.
    - To prevent abuse and mitigate denial-of-service (DoS) attacks targeting WebSocket endpoints, we can implement rate limiting mechanisms.
- HTTP (Client-to-Server):
    - HTTPS encrypts data transmitted over HTTP connections, safeguarding sensitive information from interception and manipulation; it employs TLS/SSL for this purpose.
    - Implement user authentication mechanisms, such as session cookies or JWT tokens; these verify the identity of clients who make HTTP requests.
    - Enforce access control policies at the application level: this approach restricts access to sensitive resources and endpoints--it's a method based on users' permissions.

    - Validation of Input: To circumvent prevalent security threats such as injection attacks, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF); adopt a rigorous process that validates and sanitizes user input.
    - Implement Content Security Policy (CSP) headers to diminish the risks linked with injecting and executing malicious content in the browser.
- Firebase:
    - Leverage Firebase Authentication to securely authenticate users using a variety of authentication providers: email/password, Google, Facebook--and others.
    - Define security rules for the Firebase Realtime Database: These restrictions limit access to stored data, and they are established based on users' authentication status and roles.
    - Similarly, configure security rules for Cloud Firestore to exercise control over document and collection access--a process used for adjusting read-write privileges in traditional database systems.
    - Implement server-side logic and enforce supplementary security measures--including data validation and access control--by utilizing Firebase Cloud Functions.

5.2 Software Quality Requirements

Pollaris has features to ensure it is efficient, accessible and understandable for all users and developers:

- Ease Of Use and Learning:
    - A tutorial page that explains how to use the website's features, including creating quizzes, questions, and joining a session.
    - A default Sample Test automatically part of the Work Page, which functions as a demo for new users.
- Portability:
    - It is available on every computing platform and device with access to the Internet and a web browser.
- Reliability:
    - A clear user interface that doesn't mislead its users.
    - All answer choices made by Instructors are visible to Students.
    - Answer choices are accurately recorded.
- Usability and Accessibility:
    - Buttons with text that accurately and simply describes its function.
    - A maximum of 10 answer choices can be created for each quiz question.
    - Alternative text and icon descriptions are used to make Pollaris accessible.

Appendix A: Glossary

- HTML: Hypertext Markup Language is a markup language used by programmers to place elements that can be displayed in a web browser.
- CSS: Cascading Styles Sheets is a styling language used to apply details to elements from an HTML file.
- HTTP: Hypertext Transfer Protocol. See link in References.
- HTTPS: The secure version of HTTP.

Appendix B: Work Split

Name	Contributions
Brandon	User Interfaces, Software Quality Requirements
Rizwan	Communication Interfaces, Security Requirements
Foluke	Cover Page, Table of Contents, System Features
Ethan	Overall Description
Rohit	Software Interfaces

Table 1 : Table shows the work contributions of each member of the team