Online Interactive Quiz Implementation

An online interactive quiz with disconnection handling and feedback.

Alec Braynen
College of Engineering
The University of South Florida
Tampa, FL, 33647
alecbraynen@usf.edu

Sefat E Rahman
College of Engineering
The University of South Florida
Tampa, FL, 33647
sefaterahman@usf.edu

Kyle Walker
College of Engineering
The University of South Florida
Tampa, FL, 33647
kylewalker@usf.edu

Althaaf Shaik
College of Engineering
The University of South Florida
Tampa, FL, 33647
althaaf@usf.edu

Leela Sumanth Manduva
College of Engineering
The University of South Florida
Tampa, FL, 33647
leelasumanth@usf.edu

ABSTRACT

In this paper, we present an implementation of an online examination system with disconnection handling and feedback. As computing becomes more integral to the college education experience, students are increasingly participating in quizzes and exams online. This increasing use of online examination was further spurred by the COVID-19 pandemic, which forced institutions to adapt and adopt more wholistically, online education practices. However, current technologies fail to implement most modernly, the HCI design principle of feedback. Furthermore, disconnections and connection issues hurt the examination process and cause a decrease in the accuracy of the measurement of student performance due to the emotional fluctuations that uncertainty during exams can cause.

We present in the response, our Online Interactive Quiz System. Our examination system allows for the examination or quiz to be continued after disconnection and provides immediate interactive and informative feedback to the user if they are disconnected. The software and frameworks used to implement this system are Angular, Java, PostgreSQL, and Spring boot. For the user-facing side of the application, AngularJS is used to implement the software. For the backend, Java with Spring Boot is used to create the middle layer, and the backend database is implemented as a PostgreSQL.

CCS CONCEPTS

• HCI • Feedback • Examination Interface

KEYWORDS

HCI, Design Principles, Feedback

1 Introduction

Computers continue to ubiquitously integrate into more facets of our lives. As this ubiquity grows, the demand for computing solutions that fit in and seem natural to our lives grows along with it. Academia is no exception; nowadays, academic life is not possible without a computer of some sort. Assignments now require that they be typed or submitted in a digital format, registration for classes and campus events take place online, and class examinations along with standardized tests now occur digitally.

Additionally, the COVID-19 pandemic accelerated this adoption of computers in every modern industry, academia included. When society became cognizant and responded to the pandemic, academia responded as well, moving to virtual classes, remote learning, and online examinations. This transition to virtual and remote learning brought with it some convenience and other benefits, but it also brought challenges and difficulties since technology became more integral to the teaching and examination process.

One of the more contentious parts of the transition to remote/online academia was the examination process. Academia scrambled to create workable solutions for examinations that would not allow for cheating or the general denigration of the academic system. This rush for a solution to online examination led to examination interfaces that did follow HCI design principles. We aim to contribute a solution to this specific problem with our online examination system implementation.

Our online examination system is intended to be used by **college students taking quizzes and examinations online**. Our goal is to improve the remote/online examination experience by providing to our users: 1) an examination system that allows a quiz or exam to continue uninterrupted if the user experiences network issues, 2) informative feedback to the user about their connection state and their examination session state and, 3) the ability to submit a quiz despite the current connection status.

The current solutions do not provide informative feedback to users in a timely manner, do not allow examinations to continue despite the connection status, and do not allow users to submit an exam if they experience technical difficulties. **The lack of feedback and the dependence on technical reliability decreases the accurate measure of student ability** due to exams not being completed and students getting frustrated or worried during the use of the system.

Our implementation solves this problem by utilizing a back-end system that allows for the synchronization of the exam state so that if users are disconnected or experience technical problems when they reconnect the system updates and synchronizes the user's quiz states without interrupting their quiz. Additionally, our system provides fast and informative feedback about the connection status of the user's quiz session to reduce the user's gap of evaluation while completing exams.

In this paper, we describe in detail our examination system and the technologies utilized to create it. Additionally, we provide information about its level of completeness and discuss the features essential to the design.

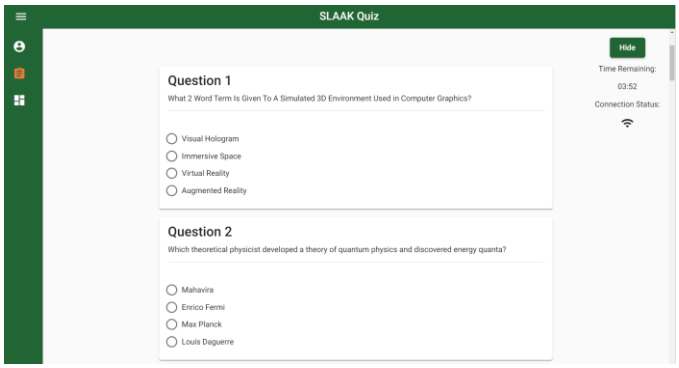

Figure 1: Quiz System Image

2 Initial Design

Our initial design seeks to give students helpful feedback during tests and enable them to finish a test even if there is a connectivity problem or service interruption. User interviews, user-centered design principles, and inspiration from existing online learning systems are all factors that influenced our design decisions. Giving students a hassle- and stress-free atmosphere to write exams is the main goal of our design.

We intended the quiz user interface to have two features: an offline quiz mode and connection status notifications. If the internet is interrupted, the offline quiz mode will assist students in finishing their exams. By doing this, students can write exams without worrying about any network disconnections that might hinder their mental state at the time. Students will be kept up to date on the connectivity status due to the notification status function. In the event of a disconnect, students can still complete their exam, and once it is finished, they will wait for the connection to come back before submitting.

Most of our design's user-facing elements, including the color palette, affordances, button placement, design, and so forth, remained relatively the same throughout the design and implementation phases. Using this initial design as our inspiration, we conducted a few surveys asking students what features they are looking for in online quiz systems. It was discovered students wanted the option to hide the running timer because doing so made them anxious and had a negative impact on their performance on the exam.

After considering all the survey comments, we adjusted our design to incorporate a hide/show timer option. Taking this concept further, we created a high-fidelity prototype that accurately represents the final product we planned to develop and presented to the class. Most students thought it was great that the issue they were facing was addressed, and they gave mostly favorable feedback. We decided to continue with our current design because we received no feedback that would have inspired us to add or remove any element of our design.

We finalized our design to include an offline quiz mode, connection status notifications, and a hide/show timer option. This coincides with one of our design goals, which is to protect the user's mental state during the exam. We want our examination system to capture our user's capabilities more accurately, and this requires that we keep them calm, in control, and stress-free during their exam.

The offline quiz mode is designed to continue the test while the timer is running even if there is an internet connection problem. The user is informed of the connection status using a pop-up and the connection icon on the exam page. The user will be informed if the connection resumes. If the connection is lost and the user finishes the exam, they can click the submit button and keep the page open until the connection is restored, at which point the quiz will be submitted automatically.

The hide/show timer option works with the click of a button at the top right corner of the page where the user hides it or pops it back up if he needs to. The user interface we created for all our features is clean and user-friendly, setting out each component so that the user can easily access it.

3 System

The quiz application allows users to take a quiz regardless of internet connectivity after a quiz has been started. The interface implementation closely matches our high-fidelity prototype, this is due to the positive feedback received. The menu on the left allows users to access the profile, quiz, and submission pages. Users have the choice to begin the exam on the quiz page. The application displays all the questions on one page, saving the user from

having to go back and forth between pages to move through questions. The backend is designed in such a way that it can take multiple sets of questions. Based on the user's ID number, a set is chosen by integrating with a third-party trivial question API. The questions are retrieved at random from a variety of different categories the API provides.

The timer and connection status is displayed on the right side of the test page [Fig 2]. The user has the option to hide the timer and connection status. The connection status icon changes depending on the user's network. In addition to the icon, a notification popup appears at the bottom of the page to inform the user of the network status [Fig 3] [Fig 4]. This enables a distraction-free experience for the user. The system alerts the users if they attempt to submit the quiz without answering all the questions [Fig 5]. The test is automatically submitted once the timer expires.

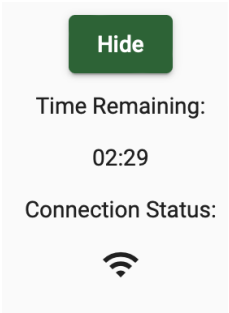

Figure 2 - Time Remaining and Connection Status.

The page verifies the status of the connection once the quiz has been submitted, either by the user or the system. If the connection is available, the user is welcomed by the report page, which displays the overall score and test duration time [Fig 6]. If the network status is offline when the quiz is submitted, the system leads the user to a loading screen where it waits for the network to be established before automatically submitting the quiz. [Fig 7] shows the loading page when a quiz is submitted without a proper connection.

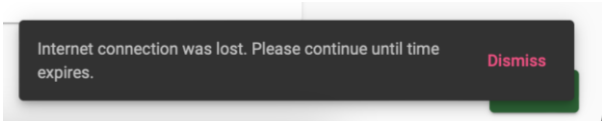

Figure 3 - Notification showing that the internet connection is lost.

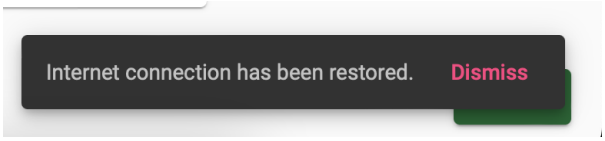

Figure 4 - Notification showing that the internet connection is restored.

3.1 Front-end:
The front-end interface is built using the Angular framework and the Angular Material Component UI Library for a consistent page design with a minimalistic look. The web application's pages are all uniformly styled in a two-tone color scheme.

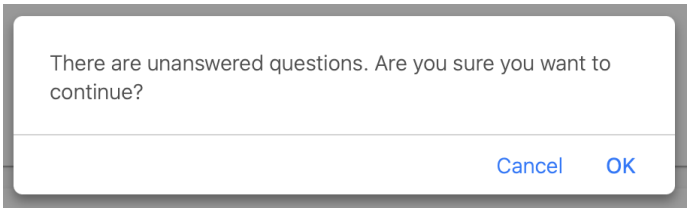

Figure 5 - Conformation message shown to the user if try to submit the quiz without answering all the questions.

Notifications are an important aspect of this system regarding the HCI questions we are trying to answer. While the user is interacting with the quiz web application there are various situations that warrant a notification so that the user can clearly understand what is happening while conducting their tasks. When the application is offline, and the user tries to start a quiz a notification will inform the user and ask them to try again later. Once the user is online and clicks the start quiz button they will be prompted to acknowledge if they want to start the quiz and will have to confirm before the quiz can start.

When the quiz is in progress the user can encounter two additional notifications. The first is a notification informing that the user is offline and assuring them to continue until finished. This feature is the main difference between what the current system does. The current system will pause the quiz until all issues are resolved, resulting in the user losing time on the quiz. The user is also notified the moment the internet connection has been restored.

3.2 Backend:
The backend is implemented as a Java Spring Boot microservice, which is used to serve requests from the UI. Postgres, a relational database was used for storing the backend data for the project.

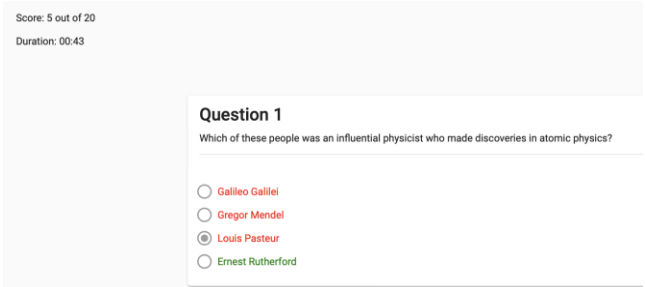

Figure 6 – Quiz submission report page

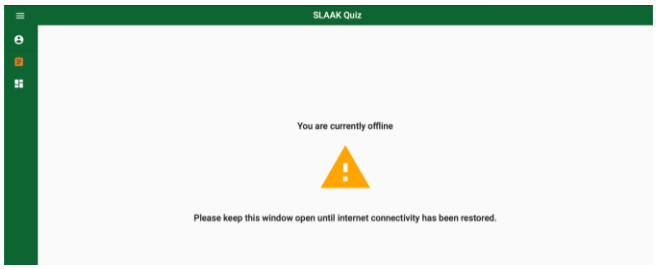

Figure 7 - Quiz application waiting for the network connection to submit the quiz.

To develop different functionality, a wide variety of APIs [Fig 5] were employed in this project.

- UpdateConnectionStatus: This is a put API that is used to update a user's connection status on the server. Primarily to simulate a network disconnection.
- PutAnswers: This is a put API that is used to send the options selected by the user to the server for report creation. Questions are updated with the user-selected options. The results are then able to be displayed when calling the QuizResult API.
- PostQuestions: This API is used to initiate a new quiz. When called quiz questions are retrieved from a third-party API that supplies trivial questions. Those questions are saved to the Quiz, Questions, and Options tables. The questions are then returned in the response that will be displayed to the user. This interaction happens when the Start Quiz button is clicked on the UI.
- QuizResult: This is a get API. It retrieves the quiz results containing how long the user took, the number of correct answers and all the questions from the latest quiz taken for the user.
- GetConnectionStatus: This is a get API that fetches the connection status from the server.

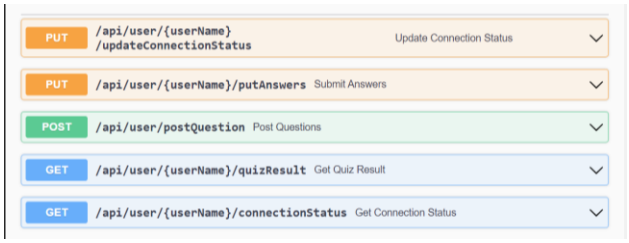

Figure 8 - All five back-end APIs

3.3 Database Schema

The below lists the five tables [Fig 9] that are implemented in the database schema for the interactive quiz examination.

Users

This allows the server to recognize each user separately, enabling it to choose a unique set of questions for each user.

Quiz

This schema contains information about the quiz, such as the total number of questions, the total number of questions that were answered correctly, and the overall time to complete the quiz.

Questions

This schema contains all the quiz questions. Each question has a question number attribute, and each question is linked to a quiz id.

Options

This serves as the repository for all options-related data. This schema stores all possible options for the question, the option chosen by the user, and the correct option for a specific question.

Connection Status

This schema contains the connection information. It is useful for determining the user's network status.

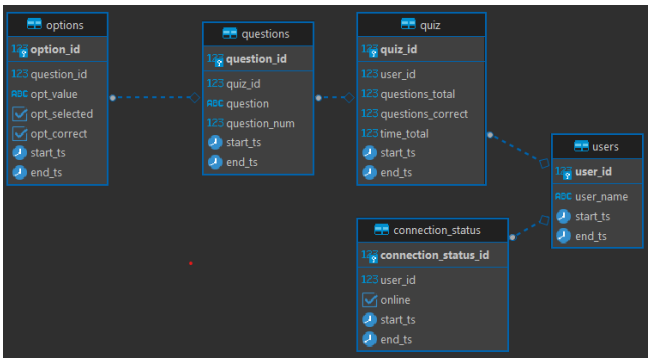

Figure 9 - PostgreSQL database schemas

3.4 Tools and Libraries for the project extension or replication:

3.4.1 Front-end:

1. Install NodeJs
2. Install IDE (we used Visual Studio Code)
3. Install Angular CLI packages via the command line
4. Deployment solution for Front-end:
    a. Build the project and deploy compiled files to AWS using S3 and configure it as a statically hosted site.
    b. http://slaak-hci-quiz-app.s3-website-us-east-1.amazonaws.com

3.4.2 Back-end:

1. Install OpenJDK 18
2. Install Java IDE (we used IntelliJ IDEA)
3. Create a Spring Boot base project (https://start.spring.io/)
4. Install Docker
5. Create a Postgres docker container
6. Deployment solution for Back-end:
    a. Build the project and deploy the JAR file to AWS using Elastic Beanstalk as a Web Service
    b. http://slaakhciquizapi-env.eba-n2tmhbnq.us-east-1.elasticbeanstalk.com/swagger-ui.html

4 Implementation Status

4.1 User Interface.

The user interface is successfully implemented. The theme matches and builds upon our high-fidelity prototype. Additionally, the user-interaction flows for submitting the quiz when online and when disconnected from the quiz system are functional.

4.1.1 Feedback System

The feedback system for connection status during the quiz is successfully implemented. The system notifies the user quickly of disconnection and reconnection events.

4.1.2 Toggleable Timer

The clock or timer shown during the quiz is toggleable as requested by the user survey.

4.2 Middleware and Backend

The middleware has been successfully implemented, allowing for communication between the front-end user interface and the back-end database. The back-end database has been implemented as well.

4.2.1 Local-Server Synchronization

The mechanisms and systems (middleware, backend) are in place to allow for the quiz to continue despite disconnection; in the best-case scenario, the user can continue the quiz and when they reconnect, the system synchronizes the changes that occurred, and the user can complete the test without any issues. In the worst-case scenario, where the user completes the test in a disconnected state, the system alerts the user to not close the window and that the test will be submitted as soon as the system reconnects. When the user reconnects, the quiz is submitted.

4.3 Summary

In our implementation, we have successfully implemented all the proposed core features. Our interface notifies the user of connection issues, allows for offline quiz submission, and synchronizes changes if the user disconnects and reconnects.