ClickEase

System/Subsystem Design Description

Sara Nabulsi, Kyle Nehman, Marc Neisser, Parth Patel, Garrett Porter

Version 1.1

November 20, 2018

Version History

Date	Version	Description
11-20-18	1.0	Initial version from the template provided. Requirements Traceability will be added after the SRS rewrite.
11-20-18	1.1	Added additional tables and diagrams to in place of text heavy areas to give a better visual representation of design and improve organization.

Table of Contents

1. Scope.

1.1 Identification.

The ClickEase system is designed for operation within the University of Baltimore Maryland (UMBC) lecture system.

1.2 System overview.

The ClickEase system was designed to address a gap area in the current University of Maryland Baltimore County (UMBC) clicker system. The system is a free-to-use application designed for teachers and students for use in the classroom as a compliment to the lecture. It provides hub for teachers to create quizzes for students who can submit answers in real time through their devices during class. The system supplies teachers with data on student accuracy and attendance per quiz and holistically. The goal of the system is to provide a straightforward, free method of facilitating quizzes in class without the hassle of clicker subscriptions and fees.

The ClickEase system was loosely designed based on a suggestion from Michael Neary, and with implementation feedback from Dr. Russell Cain.

1.3 Document overview.

This ClickEase SSDD describes the system-wide design and architectural design of the ClickEase system hardware and software as of its current state. The SSDD will be updated when and as directed by the team to reflect changes in the design of the ClickEase system architecture for subsequent software releases and site implementations deployed in the future.

Section 1, Scope, identifies the ClickEase and provides a brief description of the system.

Section 2, Referenced Documents, identifies all documents referenced in or used in the creation of this document.

Section 3, System-wide Design Decisions, presents the decisions about the system's behavioral design and other decisions affecting the selection and design of the system components.

Section 4, System Architectural Design, describes the ClickEase system's architectural design and functional flow. The ClickEase is considered to be one Computer Software Configuration

Item (CSCI). It is also a collection of Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), computer resources, and manual operations.

Section 5, Requirements Traceability, presents the traceability from each system component identified in this SSDD to the system requirements allocated to it as well as traceability from each system requirement to the system components to which it is allocated.

Section 6, Notes, contains any additional information pertaining to the ClickEase that may not fit into one of the aforementioned sections.

2. Referenced documents.

DSP SSDD: https://faaco.faa.gov/index.cfm/attachment/download/63151

3. System-Wide Design Decisions

3.1 Student Interface Design Decisions

Student inputs and interaction will be quite limited in comparison to teach input and interaction with the interface. Students create an account with the application through an online form. The form will catch errors such as non-UMBC email addresses and incorrect input types (numerical when it shouldn't be or vice versa). They then have the option to add courses to their course list using a unique registration key that the teacher provides to them. They input the key, if it's incorrect, they are prompted to re-enter their key. After inputting a correct key, the course is added to their list. When a teacher begins a quiz session, students enter by selecting the course in their course list. They are presented with the answer options as clickable buttons and may choose one response, after which that response is highlighted to indicate it was chosen. Each response that is sent is accompanied by a timestamp that shows when the student selected the response. There will be a limited period of time in which after a question ends, packets will be accepted to account for slow network speeds in lecture halls. Students have a cooldown period after selecting a response. All the responses at this point are greyed out until the cooldown ends and they can send another response. This is to avoid overloading the system with a large amount of entries. After a teacher ends a question period, the student can view on their screen if the response they chose was correct or not. Additionally, Students can only be in one quiz instance at a time so if they attempt to open the application in another tab on their web browser, it would show the same UI instance.

After a quiz instance, students can view their past quiz metrics and see their accuracy and correct responses to questions. They will also receive graphical representation of their holistic quiz accuracy.

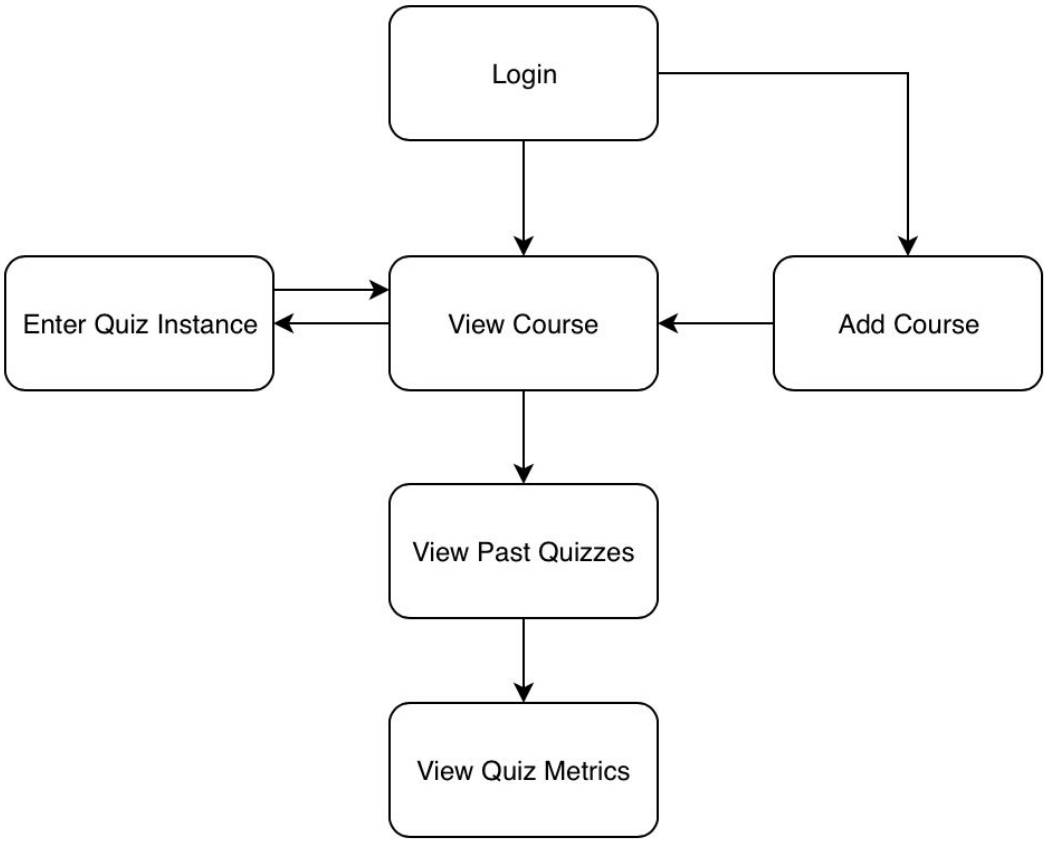


FIGURE 4-1 Student Flow Diagram

3.2 Teacher Interface Design Decisions

Teachers create an account with the application through an online form. They create new courses by entering a course name and they receive a registration key that they distribute to the class. Teachers can enter a time period after which the registration key will expire and no longer be able to be used to enter the course. Their courses are organized on a homepage so they can switch between different courses. In a course page, the teacher can add a quiz. They enter the name of the quiz and can begin adding questions to the quiz. Each question has answer options

that the teacher creates. The course page shows a list of all quizzes created for that course by name. The course page also shows a student roster, indicating which students have signed up for this course. Teachers select a quiz and then can click a button to start the quiz instance.

During a quiz instance, the question appears on the screen, followed by the answer options. The correct answer is not shown so that the teacher can project this image to their display for the entire class to see. The teacher can either pre-set how long each question will remain on the screen for on their quiz page, or they can move through the questions by clicking their mouse. After a question period, a chart will appear on the screen that indicated the percentage of students who selected each answer choice as well as indicates which answer choice was correct. Clicking again moves forward to the next question. After all the questions have been cycled through, the quiz instance will end itself. Teachers can only host one quiz instance at a time. Attempting to open another application tab in the web browser will show the same UI instance.

After a quiz, the teacher can view overall quiz metrics for accuracy per question and overall. Teachers can also view a graphical representation of quiz accuracy for the entire class so far or quiz accuracy per student for duration of the entire class so far.

Teachers additionally have the option to remove students from a course. They receive a notice asking for secondary authentication that they wish to have the student removed. If not, the student remains in the course and nothing is changed. If they do, the student data no longer appears in any of the class metrics.

It’s imperative to limit the possibility of database injection caused by questions submitted that use certain characters. There are workarounds to this issue, so it is something to consider when in development.

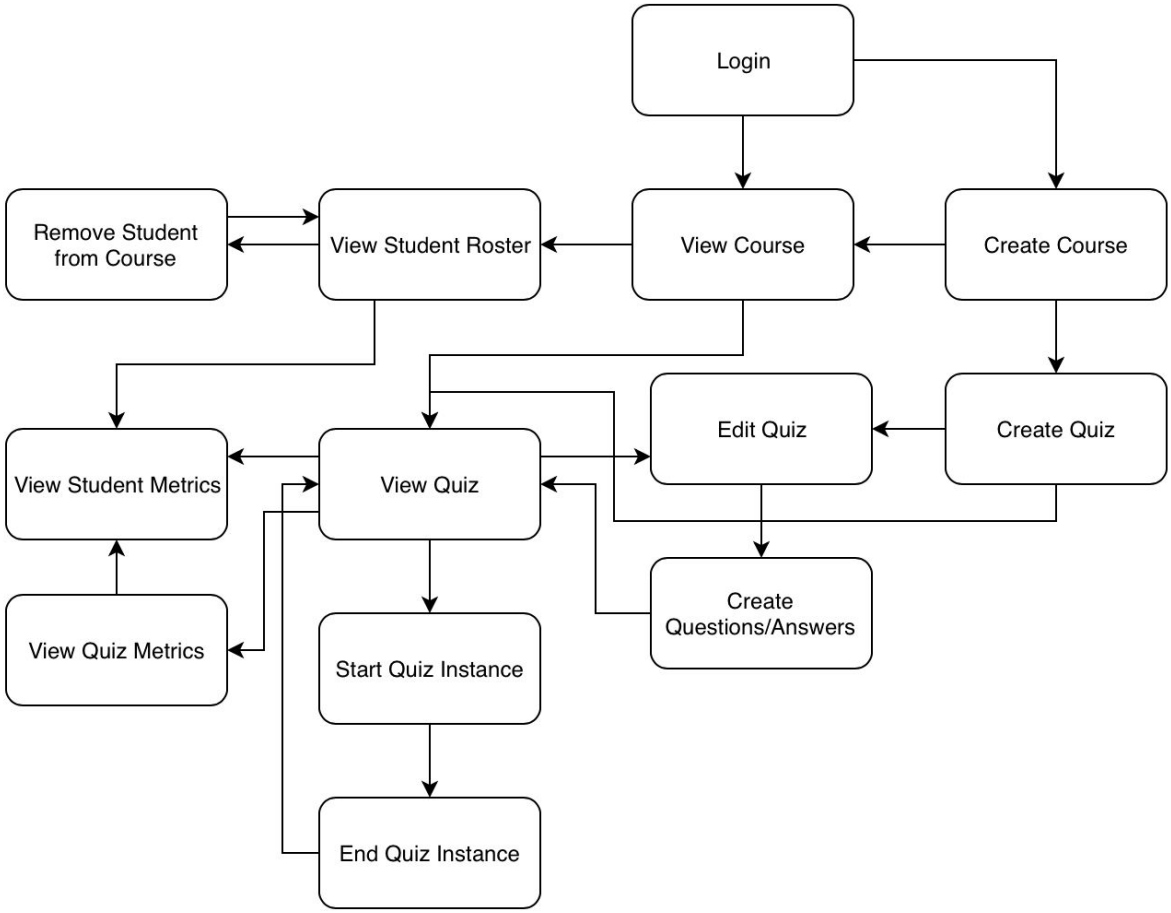


FIGURE 4-2 Teacher Flow Diagram

4. System Architectural Design.

This section shall be divided into the following paragraphs to describe the system architectural design. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one paragraph, it may be presented once and referenced from the other paragraphs. Design conventions needed to understand the design shall be presented or referenced.

4.1 System components

The following subsection addresses the computer resources and hardware and software components of the clicker WebApp. The clicker Web App uses minimal hardware resources. The server software is deployed on AWS for reliable scalability.

The Clicker WebApp system computer resources include:

- AWS Ubuntu Server
- User Machine (Smartphones or Computers)
- MERN (MongoDB, Express.Js, React, Node.Js) Stack

4.1.1 Software Units

The following subsections describe the software units of the Clicker WebApp system, including their descriptions, purpose, requirements, and development status. These components, and their unique identifiers that they will be referred to as, are listed in Table 4.1-1.

4.1.1.1 TEACHER_CLIENT

A TEACHER_CLIENT is an end user who is logged into the system and registered as a teacher, that is that they have at least one course which they run and can create quizzes for. A TEACHER_CLIENT controls a COURSE and may create QUESTIONs and QUIZZES for that course.

4.1.1.2 STUDENT_CLIENT

A STUDENT_CLIENT is an end user who is logged into the system and may only respond to QUESTIONS, not create them. The STUDENT_CLIENT is concerned with providing answers for their given COURSEs in order to receive credit during a QUIZ.

4.1.1.3 DATABASE

DATABASE holds information about all of the other software components for the Clicker WebApp. All of the Objects mentioned in section 4.1.1, excluding 4.1.1.3, are represented directly inside of NoSQL Collections.

The individual Objects reference each other via ObjectIDs, a form of UUID. A COURSE consists of STUDENT_CLIENTs, TEACHER_CLIENTs, QUIZZES, QUESTIONS. One COURSE Object therefore will contain multiple ObjectID fields to reference other Objects in other Collections.

MongoDB Database has been deployed to the WebApp server and has Collections for each Object. Database is ready for insertion of each software unit as they finish development.

4.1.1.4 QUIZ

QUIZZES are made up of groups of QUESTIONS and presented to STUDENT_CLIENTS in order for them to receive a grade in the class. QUIZZES are

created by TEACHER_CLIENTs and are a member of a specific COURSE.

4.1.1.5 QUESTION

QUESTIONS are prompts created by TEACHER_CLIENTs and are members of a QUIZ. STUDENT_CLIENTs will respond to individual QUESTIONS while taking a QUIZ.

4.1.1.6 COURSE

COURSES are created by TEACHER_CLIENTs and joined by STUDENT_CLIENTs. They contain QUIZZES and are used primarily as a way of grouping people together in order to quickly provide access for all relevant STUDENT_CLIENTs to the same QUIZ.

4.1.1.7 RECORD

RECORDs represent the individual STUDENT_CLIENT's responses and grade for an individual QUIZ. These are contained in the STUDENT_CLIENT Object but are immutable. Both the TEACHER_CLIENT and STUDENT_CLIENT may view the results of a RECORD, and a TEACHER_CLIENT may view multiple RECORDs at once for their COURSE in order to make comparisons and view statistics.

Table 4.1-1

Unit Unique Identifier	Description
TEACHER_CLIENT	An end user that is registered as a Teacher that runs a Course and can create Quizzes
STUDENT_CLIENT	An end user that is enrolled in a course and using the webapp to respond to Quizzes
CLIENT	May refer to either TEACHER_CLIENT or STUDENT_CLIENT
DATABASE	MongoDB NoSQL database to store all of the WebApp's data
QUIZ	A grouping of Questions assigned to a specific Course
QUESTION	Prompt with multiple choice answers to be responded to by a Student

COURSE	Grouping of Students taught by an Instructor
RECORD	Individual Student's results for a specific Quiz

4.1.2 Hardware Components

The only hardware components required by the Clicker WebApp, aside from the actual server itself, is that of the end user client devices to connect to the web app. A client device may consist of a computer, smartphone, tablet, or any internet capable device with a modern web browser and some form of user input.

The WebApp server is currently being hosted on an AWS free-tier t2.micro EC2 VM and has the capability of being scaled up to a larger server, or to be moved to on premises if desired.

The AWS free-tier EC2 Ubuntu server hardware configuration includes:

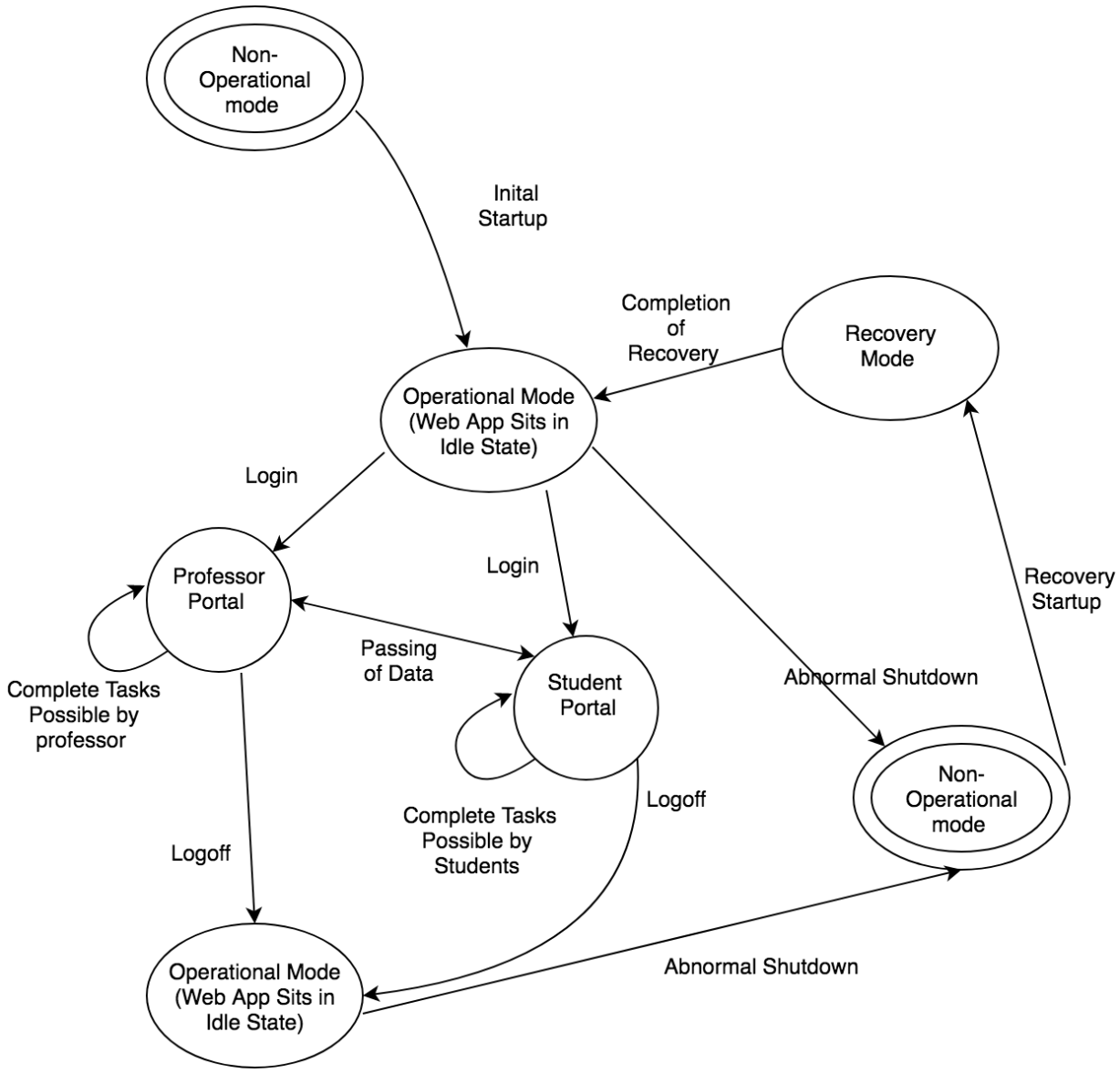
- 3.3 GHz Intel Xeon CPU
- 1 GiB Ram
- 8 GB HDD storage

4.2 Concept of execution

This subsection presents the Clicker WebApp system as a concept of execution by defining and discussing the states and modes that the web app will be in during its lifecycle.

- The initial startup of the Clicker web app
- The Webapp sitting in an idle state
- Runtime Operation
    - Runtime Operation of the Professor Side (creating classes
    - Runtime Operation of students joining classes

    - Runtime Operation of Professors creating questions (quizzes)
    - Runtime Operation of Students Answering Questions
    - Runtime Operation of Professors Closing Quizzes
    - Runtime Operation of Professors creating statistics
- Dynamic Relationship between Components
- System Monitoring
- The eventual downfall of the clicker web app (AWS shutdown/migration to UMBC DoIT)



4.2.1 Initial Startup of the Clicker WebApp

The startup of the clicker web app is controlled in a simple few steps:

1. The AWS server (Mentioned above in 4.1) is created and used to host the web app
2. The MERN stack is deployed upon the server and created in a way to allow the different customer portals to complete tasks.
3. A Second database is updated once every day in case there is a need of a recovery state
4. Finally-- The Server stays up, and functions in an idle state while Professors and Students complete their tasks and view their profiles.

4.2.1.1 Normal Startup

A normal startup is defined as the ‘initial startup’ (explained above in 4.2.1) completed after a planned update or shutdown for maintenance.
Normal Startup (Initial Startup) consists of the following steps

1. The AWS server has been taken offline and run for maintenance. This maintenance might be anywhere from changing updating the source code with patches and or updated features. As well as, migrated to a larger server to handle more and more requests.
2. The AWS server is than spun up again and the webapp is transitioned into a idle state once again.
3. Students and Professors can go about their way and access the new added features and patches as well as have the ability to complete tasks as previously stated.

The normal startup after a planned maintenance period is administered until the state of the web app is brought once again into the ability to allow the users to complete their planned tasks.

4.2.1.1 Recovery Startup

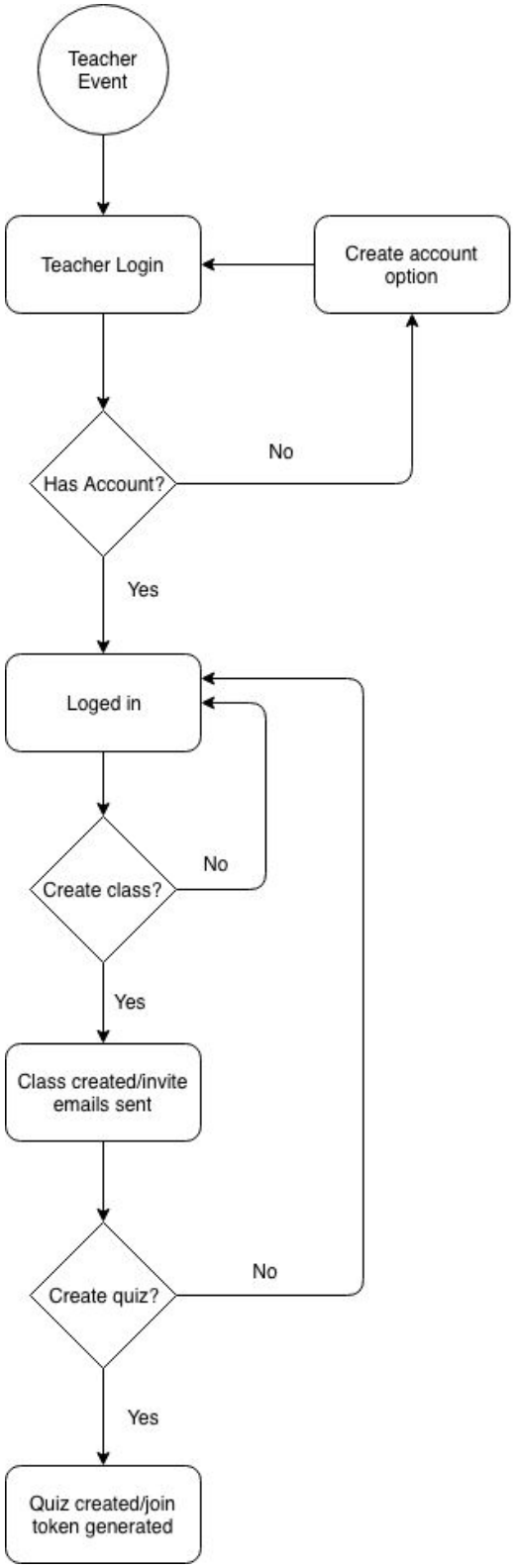
A recovery startup occurs after the web app was not functioning properly and had a unplanned shutdown. The clicker web app will have an adminstarter check to see if any data was lost by the shutdown.

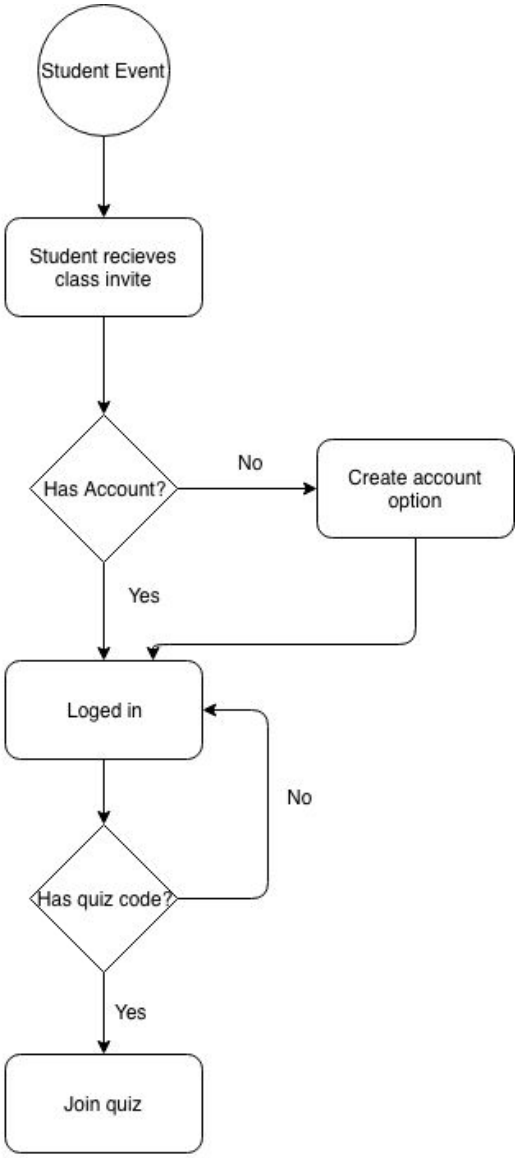
If no data was lost in between the last active(idle) state. The issue that caused the unplanned shut is investigated and possible fixed will be tried.

After fixing the issue, the webapp will be brought back into a idle state and work as planned. If no fix was found then the web app is brought back into an idle state after establishing the databases to the most current backup taken.

4.2.2 Runtime Operation

After getting the web app into an idle state, the software components are executed initially once per user. The MERN stack will be able to deploy an initial startup page that would allow the user to create actual profiles to allow the user to sign in and access their respected portals. This is done by clicking the ‘sign up’ button on the login page. The creation of profiles is initially done by the professor who can then go in and create a token to allow students to join their respected classes. The process of creating profiles and joining classes is displayed in the diagram below.


Teacher Event
Teacher Login
Create account option
Has Account?
No
Yes
Loged in
Create class?
No
Yes
Class created/invite emails sent
Create quiz?
No
Yes
Quiz created/join token generated


Student Event
Student recieves class invite
Has Account?
No
Create account option
Yes
Loged in
No
Has quiz code?
Yes
Join quiz

4.2.2 Runtime Operation

The WebApp software components are created to work asynchronously, with ability of a professor creating questions and students being able to answer questions within the given time limit. All the following subsections (4.2.2.x) all will be running asynchronously. To reiterate the following software components will be:

- Professor runtime operations
    - Create Classes
    - Create Questions
    - Close Questions
    - Look at statistics
- Student runtime Operations
    - Join Classes
    - Answer Questions
    - View Grades

4.2.2.1 RunTime Operations of Professors

The Run time Operations of Professors are all run asynchronously and allow to professor to do all the following. All the following will be drawn out in a diagram in the User interaction portion of the document. Section 4.2.3.2 further below.

4.2.2.1.1 Create Classes

Professor Users who are logged in have the ability to create classes. When creating classes the user can state the name of the class, the possible number of max students the class can support. The most important information that comes out of creating the class is the ability to assign (or randomly generate) a token that would allow the students to join the respected class.

4.2.2.1.2 Create Questions

After professors have successfully created classes, they have the ability to create questions. These questions can vary from Multiple Choice, to Short answer, to possible ‘Long-Short’ answer. The student interaction with the webapp does change slightly depending on the question type. They can set possible time limits on how long the database will continue to store student responses to a particular question. They can also choose to manually close the question from taking anymore responses.

4.2.2.1.3 Close Question

Professor can manually close a particular question from taking any more responses from students. Used in place of timed limit response.

4.2.2.1.4 View Statistics

Professor's also have the ability to view statistics on their classes. This can range from viewing the percentage of students who answered correctly. Or a break down of how students did in relation to the possible answers. They can also compare the results of different classes they manage.

4.2.2.1 RunTime Operations of Students

The Run time Operations of Students also run asynchronously and allow to professor to do all the following. All the following will be drawn out in a diagram in the User interaction portion of the document. See section 4.2.3.2 further below.

4.2.2.1.1 Join Classes

After receiving the token for a particular class either by email or in class from the professor, the dashboard of the student portal would allow the student to enroll in the class they want. They will require the token to enroll in the class, and must link their profiles to a student id so professors can gather relevant grades.

4.2.2.1.2 Answer Questions

After professors have created a question, the student can respond to an active question. Based upon the type of question (MC, short answer, 'long-short'), the student will be prompted to answer either using buttons relating to the possible MC option or text boxes of different sizes when the question requires a text based answer.

4.2.2.1.3 View Grades

Throughout the semester, a student user has the ability to check how they are doing in a class so far. This can be in the form of seeing past questions they have answers, and whether they have answered it correctly. As well what their overall grade based on the quizzes taken so far.

4.2.3 Dynamical Relationship Between Components

The components that are explained above can be split into three different relationships

1. User Interaction with their specific User Portals (Student vs Professor)
2. Database Interaction with the responses.
3. Updating the database with the final response to the questions and updating the backup database.

4.2.3.1 User Interaction

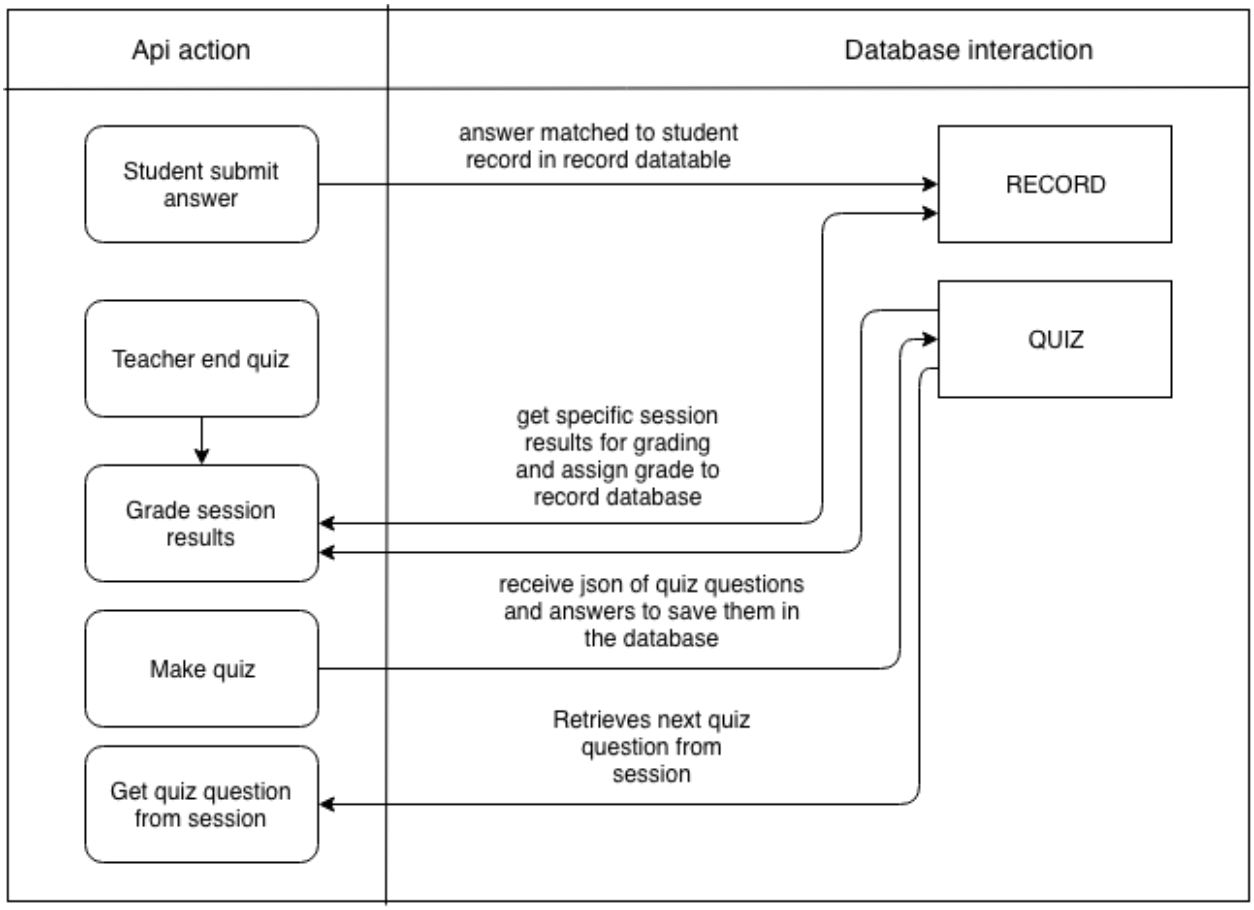
The two user groups (Professor and Student) dictate a majority of the way that the system software components react to one another.
The figure below shows the user flowchart of possible interactions.

The Figure is a repeat from the section above, however, it also displays the interaction between student users and professor users. Look to section three for diagram of user interaction with the Web App.

4.2.3.2 Database interaction

Section 4.2.3.1 discusses how the users dictate the way the components react to one another and the dynamic relationship that they share. This section will explain how the user actions are interrelated with the underlying API and Database.

The API will handle passing of data to the client as well as the setting of client data in the database. Additionally, API functionality will implement a number of checks to ensure the integrity of the data.



4.2.3.2 Database Interaction

4.2.3.3 Processing of the API information and Database

The API will be handled by a REST endpoint accepting HTTP POSTs and GETs which will be passed to the back-end to be processed. All API transmissions will contain JWT authentication tokens and be SSL encrypted for security purposes. Once a API command has been validated the corresponding data will be written to / read from the Database and a result will be sent in the form of either the requested data, or a boolean success flag.

API POSTs will additionally contain a timestamp, to be compared with the TCP timestamp of the actual HTTP transmission, in order to determine if the POST should be processed by the server and added to the database. The timestamps will have brief limits on them to avoid abuse, but are designed to allow student submissions shortly after a quiz has officially ended - during brief local (client) network downtimes or periods of slow transmission speeds.

4.2.4 System Monitoring

Throughout the life cycle there will be patches that are integrated into the system as features are added/removed/changed. AWS has monitoring tools that will be used to check throughout to see how the server is handling the constant pushes and requests.

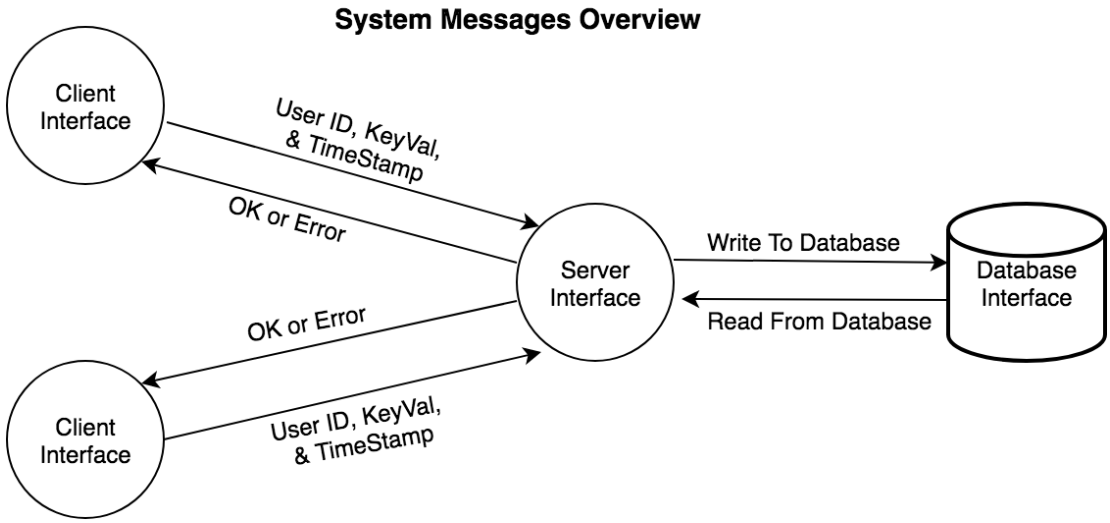
4.2.5 Shutdown of Clicker Web App.

The shutdown of the clicker app is when the AWS server is terminated and or migrated to the official UMBC servers.

4.3 Interface design

4.3.1 Interface identification and diagrams.

- Client Interface (Subsection 4.3.1.1)
- Server Interface (Subsection 4.3.1.2)
- Database Interface (Subsection 4.3.1.3)

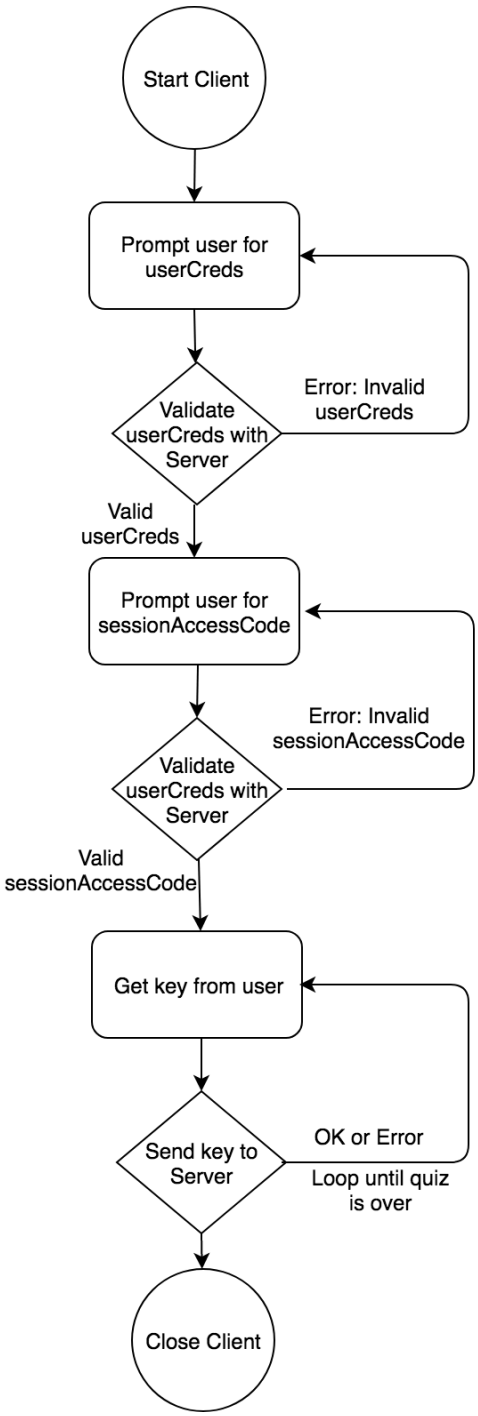


Data is transferred between Client Interface, Server Interface, and Database Interface using HTTP, JSON, and REST.

Variables Used

Variable	Description
userCreds	Contains the type of the user (teacher or student), username, and password. This is sent from the client to the server as a means of authentication and associating the client with a database user.
key	The quiz answer that the user is sending to the Server
userAccessToken	This given to each client to associate it with a specific user and instance.
error	A nil error acts as an OK, meaning there were no problems. If error is not nil then this signifies that the operation could not be completed as expected. An error type is returned each time there is communication between Client, Server, or Database.
session	Each operation between Client and Server is related to a particular session. The session specifies which class and quiz are being referred to, and who the user is.
timestamp	The time that the answer was selected on the client. Used to correlate a key with a specific question.
questionInfo	Contains the session, the question, the key(s) chosen.Server sends this to Database so Database can store the key(s) in the appropriate place(s). May contain a single student's response or multiple students' responses.
stats	This data structure contains overall statistics for the session, for each individual student, and for each question. This is sent from the Database to the Server then to the teacher's Client.
sessionAccessCode	This is the password used to join a specific quiz instance.

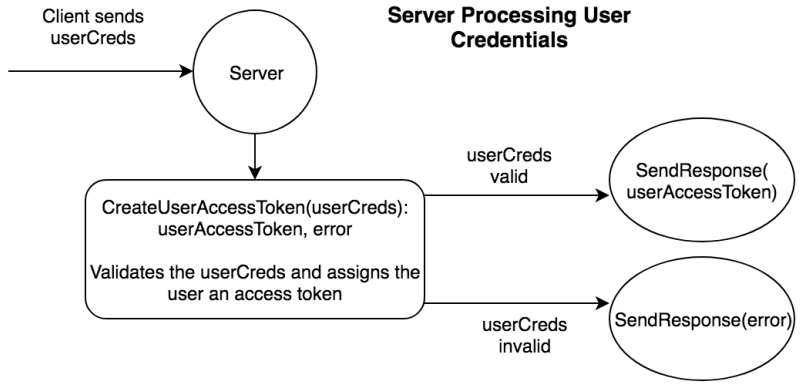
4.3.1.1 Client Interface

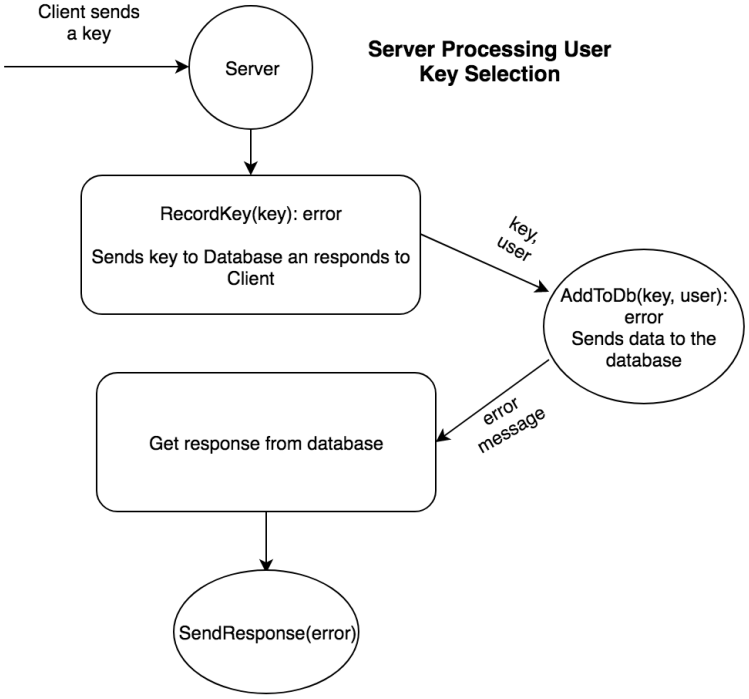


Client Interface Functions

Function	Description
GetLoginToken(userCreds): userAccessToken, error	Sends user credentials to the Server. If the credentials are valid then the user is given an access token and error is nil.
SetupSession(userAccessToken, sessionAccessCode): session, error	Sends the access code for the session and the user's token to the Server. The server returns a session which is used to identify the user and the quiz instance.
SendKey(key, timestamp): error	Send the user's key selection to the server.

4.3.1.2 Server Interface



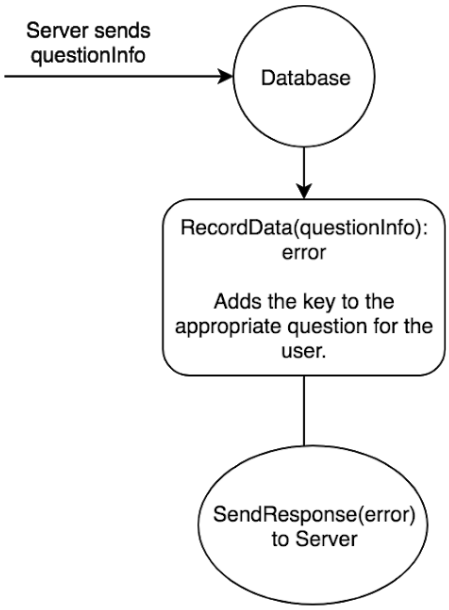


Server Interface Functions

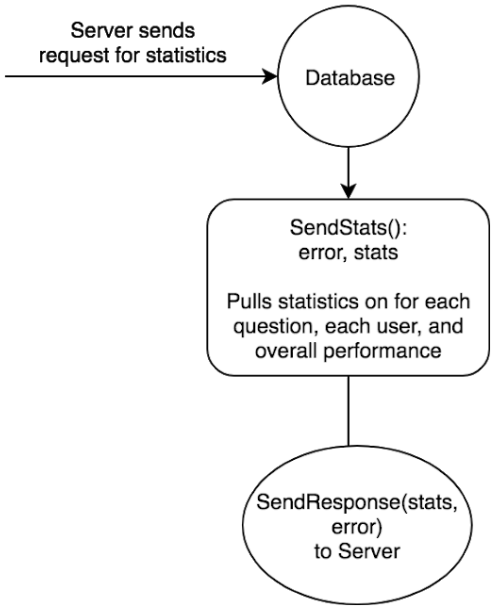
Function	Description
SendResponse(error)	A nil error tells the Client that the key was successfully added to the database. An non nil error states why the key could not be added to the database.
RecordKey(key): error	Sends the user's key choice to the Database if it is a valid selection.
GetStats(): stats	Sends a request to the database for statistics

4.3.1.3 Database Interface

Write to Database



Read From Database



Database Interface Functions

Function	Description
AddToDb(key, user): error	Records the user's selection for the specific question.
SendResponse(stats, error)	Sends session, user, and class statistics to the server.
SendResponse(error)	Used to tell the server whether the data was written successfully. Nil error means the operation was successful.

4.3.2 Interface Details

The following subsections provide details on the interfaces listed and described in Subsection 4.3.1 and Table 4.1-1. Interface details are presented in the same order as listed in Subsection 4.3.1.

- CLIENT Interface (Subsection 4.3.2.1)
- SERVER Interface (Subsection 4.3.2.2)
- DATABASE Interface (Subsection 4.3.2.3)

4.3.2.1 CLIENT Interface

4.3.2.1.1 Not Applicable.

4.3.2.1.2 This interface is used for real time data transfer.

4.3.2.1.3 The CLIENT Interface is for sending data.

4.3.2.1.3.1 Names/identifiers

4.3.2.1.3.1.1 Not Applicable.

4.3.2.1.3.1.2 Not Applicable.

4.3.2.1.3.1.3 Not Applicable.

4.3.2.1.3.1.4 The function mainly used to communicate with the server is called SendKey in the code.

4.3.2.1.3.1.5 Not Applicable.

4.3.2.1.3.2 The data type is alphanumeric used by these functions is alphanumeric.

4.3.2.1.3.3 Client requests are in JSON format and use HTTP. The requests are very small because each key is represented by a single character.

4.3.2.1.3.4 Time is measured in milliseconds.

4.3.2.1.3.5 Possible values default to 1-5, but may be configured for each quiz.

4.3.2.1.3.6 Not Applicable.

4.3.2.1.3.7 The client is limited to sending a key to the server ever 100ms to avoid DOS attacks

4.3.2.1.3.8 Username, password, and token must be encrypted before being sent to the server.

4.3.2.1.3.9 The client sends user input to the receiving server.

4.3.2.1.4 The client will send data structures containing the user's session information, and data associated with the specific key entry such as the key value and the time of the keypress.

4.3.2.1.4.1 userCreds and session are the data structures sent from the client for authentication.

4.3.2.1.4.1.1 Not Applicable.

4.3.2.1.4.1.2 Not Applicable.

4.3.2.1.4.1.3 Not Applicable.

4.3.2.1.4.1.4 Not Applicable.

4.3.2.1.4.2 Not Applicable.

4.3.2.1.4.3 Not Applicable.

4.3.2.1.4.4 Not Applicable.

4.3.2.1.4.5 Not Applicable.

4.3.2.1.4.6 The key may be updated every 100ms while the question is still active.

4.3.2.1.4.7 userCreds and session shall be encrypted in transit to prevent a man-in-the-middle from intercepting sensitive information.

4.3.2.1.4.8 The client sends user input to the receiving server.

4.3.2.1.5 Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:

4.3.2.1.5.1 Not Applicable.

4.3.2.1.5.2 Not Applicable.

4.3.2.1.5.3 Messages are in JSON format.

4.3.2.1.5.4 Flow control is handled by TCP and HTTP.

4.3.2.1.5.5 The transfer is limited to one request every 100ms.

4.3.2.1.5.6 Not Applicable.

4.3.2.1.5.7 Requests will be handled in FIFO order.

4.3.2.1.5.8 HTTPS shall be used to encrypt sensitive information, and username, password, session token, and user tokens are all used to assure that user is who they claim to be.

4.3.2.1.6 Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:

4.3.2.1.6.1 Not Applicable.

4.3.2.1.6.2 Not Applicable.

4.3.2.1.6.3 Not Applicable.

4.3.2.1.6.4 Each public function returns an error that will be handled at the top of the chain

4.3.2.1.6.5 The server shall send a message to disconnect to all connected clients when the quiz is over.

4.3.2.1.6.6 Auditing shall be implemented to record user connections.

4.3.2.1.7 Not Applicable.

4.3.2.2 SERVER Interface

4.3.2.2.1 Not Applicable.

4.3.2.2.2 The interface is used for real time data transfer and for storage and retrieval. Data is transferred between the server and client as well as the server and database. The server indirectly stores and retrieves data by sending instructions to the database.

4.3.2.2.3 The interface provides the ability to send data to the database, access data from the database, and receive key selections from clients. It acts as the logic between clients and the database

4.3.2.2.3.1 Names/identifiers

4.3.2.2.3.1.1 The function names in this interface are SendResponse, RecordKey, and GetStats.

4.3.2.2.3.1.2 Not Applicable.

4.3.2.2.3.1.3 Not Applicable.

4.3.2.2.3.1.4 Not Applicable.

4.3.2.2.3.1.5 Not Applicable.

4.3.2.2.3.2 A response is of type string, a key of of type integer, and stats is a data structure containing several double type objects,

4.3.2.2.3.3 A key is a single digit integer, a response can be between size 0 and an arbitrarily large integer, and stats the size of stats scales to the amount of data collected for that class.

4.3.2.2.3.4 Time is measured in milliseconds.

4.3.2.2.3.5 Keys shall be between 1 and 5 by default, and can be adjusted by the teacher.

4.3.2.2.3.6 Percentages shall be accurate up to the hundredths place, such as 99.99%.

4.3.2.2.3.7 Priority is FIFO order.

4.3.2.2.3.8 Data sent shall always be encrypted. Answers should never be sent to a client.

4.3.2.2.3.9 The server sends receives and sends data to both the database and clients.

4.3.2.2.4 The server will receive data structures containing the user's session information, and data associated with the specific key entry such as the key value and the time of the keypress. It sends the user's information and the key information to the database. The server may also grab statistics from the database

4.3.2.2.4.1 userCreds and session are the data structures received from the client for authentication, then send to the database along with the key. Object stats is received from the database.

4.3.2.2.4.1.1 Not Applicable.

4.3.2.2.4.1.2 Not Applicable.

4.3.2.2.4.1.3 Not Applicable.

4.3.2.2.4.1.4 Not Applicable.

4.3.2.2.4.2 Not Applicable.

4.3.2.2.4.3 Not Applicable.

4.3.2.2.4.4 Not Applicable.

4.3.2.2.4.5 Not Applicable.

4.3.2.2.4.6 Not Applicable.

4.3.2.2.4.7 Data in transit shall be encrypted to protect sensitive information from man in the middle attacks. User credentials shall be stored on the server as hash values.

4.3.2.2.4.8 The server both receives and sends data to the clients and the database.

4.3.2.2.5 Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:

4.3.2.2.5.1 Not Applicable.

4.3.2.2.5.2 Not Applicable.

4.3.2.2.5.3 JSON formatted HTTP messages are used.

4.3.2.2.5.4 Not Applicable.

4.3.2.2.5.5 Not Applicable.

4.3.2.2.5.6 Not Applicable.

4.3.2.2.5.7 Not Applicable.

4.3.2.2.5.8 Data in transit shall be encrypted to protect sensitive information from man in the middle attacks. User credentials shall be stored on the server as hash values.

4.3.2.2.6 Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:

4.3.2.2.6.1 Not Applicable.

4.3.2.2.6.2 Not Applicable.

4.3.2.2.6.3 Not Applicable.

4.3.2.2.6.4 Each public function returns an error that will be handled at the top of the chain

4.3.2.2.6.5 The server shall send a message to all connected clients when it is about to close the quiz instance sockets.

4.3.2.2.6.6 Certificates will be used to verify the identity of the server.

4.3.2.2.7 Server will be compatible with the client on modern web browsers.

4.3.2.3 DATABASE Interface

4.3.2.3.1 Not Applicable.

4.3.2.3.2 The interface is used for storage and retrieval. The database stores data, which is retrieved by the server.

4.3.2.3.3 The interface provides the ability to read and write data. It can only be called by the server.

4.3.2.3.3.1 Names/identifiers

4.3.2.3.3.1.1 The function names used are AddToDb and GetResponse.

4.3.2.3.3.1.2 Not Applicable.

4.3.2.3.3.1.3 Not Applicable.

4.3.2.3.3.1.4 Not Applicable.

4.3.2.3.3.1.5 Not Applicable.

4.3.2.3.3.2 A response is a data structure called stats. It contains several other data structures that are integer and double based. Stats can be thought of as several tables.

4.3.2.3.3.3 The size of stats scales to the amount of data collected for that class.

4.3.2.3.3.4 Time is measured in milliseconds.

4.3.2.3.3.5 Session, question, key, and username must all map to valid ones in the database.

4.3.2.3.3.6 Percentages shall be accurate up to the hundredths place, such as 99.99%.

4.3.2.3.3.7 Priority is FIFO order.
4.3.2.3.3.8 Data sent shall always be encrypted. Data should never be sent to a client or received from a client directly.
4.3.2.3.3.9 The database shall only communicate with the server,
4.3.2.3.4 The database will receive data structures containing the user's session information, the key, and the question.
4.3.2.3.4.1 Key, userInfo, and session are the data structures received from the server
4.3.2.3.4.1.1 Not Applicable.
4.3.2.3.4.1.2 Not Applicable.
4.3.2.3.4.1.3 Not Applicable.
4.3.2.3.4.1.4 Not Applicable.
4.3.2.3.4.2 Not Applicable.
4.3.2.3.4.3 Not Applicable.
4.3.2.3.4.4 Not Applicable.
4.3.2.3.4.5 Not Applicable.
4.3.2.3.4.6 Not Applicable.
4.3.2.3.4.7 Data in transit shall be encrypted to protect sensitive information from man in the middle attacks. Data at rest shall also be encrypted.
4.3.2.3.4.8 The database both receives and sends data to the server.
4.3.2.3.5 Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
4.3.2.3.5.1 Not Applicable.
4.3.2.3.5.2 Not Applicable.
4.3.2.3.5.3 JSON formatted HTTP messages are used. Data is stored as SQL.
4.3.2.3.5.4 Not Applicable.
4.3.2.3.5.5 Not Applicable.
4.3.2.3.5.6 Not Applicable.
4.3.2.3.5.7 Not Applicable.
4.3.2.3.5.8 Data in transit shall be encrypted to protect sensitive information from man in the middle attacks. Data at rest shall also be encrypted. User credentials shall be stored on the server as hash values.
4.3.2.3.6 Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:
4.3.2.3.6.1 Not Applicable.
4.3.2.3.6.2 Not Applicable.
4.3.2.3.6.3 Not Applicable.
4.3.2.3.6.4 Each public function returns an error that will be handled by the server.
4.3.2.3.6.5 The server will periodically check the status of the database. The server will also alert the database in the event that it is going offline.
4.3.2.3.6.6 Certificates will be used to verify the identity of the server with the database. This is to prevent unauthorized access.
4.3.2.3.7 The database only needs to be compatible with the server's infrastructure.

5. Requirements traceability.

Requirements are being reworked in the SRS document to achieve adequate requirements traceability.

6. Notes.

Not Applicable

A. Appendixes.

Not Applicable