

**DESIGN DOCUMENT**

**FOR**

**THE MATH QUEST CHALLENGE SYSTEM**

GitHub Link: <https://github.com/makmot256/G-32-MATH-QUEST-CHALLENGE-SYSTEM-.git>

**G-32**

**SUPERVISOR:** Mr. Brian Muchake

Group Members’ Details:

|  |  |  |
| --- | --- | --- |
| **Name** | **Registration number** | **Student number** |
| Johnson Makmot Kabira | 23/U/26794/EVE | 2300726794 |
| Mwesigwa Isaac | 23/U/12539/PS | 2300712539 |
| Bataringaya Bridget | 23/U/07471/EVE | 2300707471 |
| Jonathan Katongole | 23/U/27072/EVE | 2300727072 |
| Mariam Wambui | 23/U/18494/PS | 2300718494 |

**Date**: 18*/06/2024*

**TABLE OF CONTENTS**

[**LIST OF TABLES** iii](#_Toc169724459)

[**LIST OF FIGURES** iv](#_Toc169724460)

[**1. INTRODUCTION** 1](#_Toc169724461)

[1.1 Purpose 1](#_Toc169724462)

[1.2 Scope 1](#_Toc169724463)

[1.2.1 Goals and Objectives 1](#_Toc169724464)

[1.3 Document Overview 2](#_Toc169724465)

[1.4 Definitions and Acronyms 3](#_Toc169724466)

[1.4.1 Definitions 3](#_Toc169724467)

[1.4.2 Acronyms 4](#_Toc169724468)

[**2. SYSTEM OVERVIEW** 5](#_Toc169724469)

[**3. SYSTEM ARCHITECTURE** 8](#_Toc169724470)

[3.1 Architectural Design 8](#_Toc169724471)

[3.2 Decomposition Description 9](#_Toc169724472)

[3.3 Design Rationale 12](#_Toc169724473)

[**4. DATA DESIGN** 13](#_Toc169724474)

[4.1 Data Description 13](#_Toc169724475)

[4.2 Entity Relationship Diagram 15](#_Toc169724476)

[4.3 Data Dictionary 15](#_Toc169724477)

[**5. COMPONENT DESIGN** 17](#_Toc169724478)

[5.1 Admin Module 17](#_Toc169724479)

[5.1.1 uploadQuestions(String filePath) 17](#_Toc169724480)

[5.1.2 uploadAnswers(String filePath) 17](#_Toc169724481)

[5.1.3 setChallengeParameters(Date startDate, Date endDate, int duration, int numberOfQuestions) 18](#_Toc169724482)

[5.1.4 login(String username, String password) 18](#_Toc169724483)

[5.1.5 logout() 18](#_Toc169724484)

[5.2. Registration Module 19](#_Toc169724485)

[5.2.1 registerParticipant(String username, String firstname, String lastname, String email, Date dob, String schoolRegNum, String imageFile) 19](#_Toc169724486)

[5.2.2 confirmParticipant(String username, Boolean isConfirmed) 19](#_Toc169724487)

[5.2.3 viewApplicants() 19](#_Toc169724488)

[5.3. Challenge Module 20](#_Toc169724489)

[5.3.1 attemptChallenge(int participantId, int challengeId) 20](#_Toc169724490)

[5.3.2 viewChallenge(int challengeId) 20](#_Toc169724491)

[5.3.3 getRandomQuestions(int challengeId, int numberOfQuestions) 21](#_Toc169724492)

[5.3.4 calculateScore(int participantId, List<Answer> answers) 21](#_Toc169724493)

[5.4. Analytics Module 21](#_Toc169724494)

[5.4.1 generateReports() 21](#_Toc169724495)

[5.4.2 getMostCorrectlyAnsweredQuestions() 22](#_Toc169724496)

[5.4.3 getSchoolRankings() 22](#_Toc169724497)

[5.4.4 getParticipantsWithIncompleteChallenges() 22](#_Toc169724498)

[5.4.5 getStudentRanking() 22](#_Toc169724499)

[5.4.6 getPerformanceOverYears() 23](#_Toc169724500)

[5.4.7 getQuestionRepetitionPercentage(int participantId) 23](#_Toc169724501)

[**6.HUMAN INTERFACE DESIGN** 24](#_Toc169724502)

[6.1 Overview of User Interface 24](#_Toc169724503)

[6.2 Screen Images 24](#_Toc169724504)

[6.2.1 Administrator Interface 24](#_Toc169724505)

[6.2.2 School Representative Interface 27](#_Toc169724506)

[6.2.3 Participant Interface 29](#_Toc169724507)

[6.2.4 Additional User Interface Elements 30](#_Toc169724508)

[6.3 Screen Objects and Actions 31](#_Toc169724509)

# 

# 

# **LIST OF TABLES**

Table 1.1 Document Overview

Table 1.2 Data entities for The Math Quest Challenge System

Table 1.3 Data Dictionary for The Math Quest Challenge System

# 

# **LIST OF FIGURES**

Figure 3.1 System Architectural Design

Figure 3.2 Use Case Diagram of the system

Figure 3.3 Sequence Diagram of the system

Figure 3.4 Activity diagram showing participant registration

Figure 3.5 Activity diagram showing School Representative

Figure 3.6 Entity Relationship diagram of the Math Quest System

Figure 6.1 Administrator Login Interface for The Math Quest Challenge System

Figure 6.2 Uploading Schools Administrator Interface for The Math Quest Challenge System

Figure 6.3 Uploading Questions and Answers Interface for The Math Quest Challenge System

Figure 6.4 Setting Competition parameters Interface for The Math Quest Challenge System

Figure 6.5 School Representative login Interface for The Math Quest Challenge System

Figure 6.6 School Representative Applicants viewing Interface for The Math Quest Challenge System

Figure 6.7 School Representative Applicant Confirmation or Rejection Interface for The Math Quest Challenge System

Figure 6.8 Participant Registration Interface for The Math Quest Challenge System

Figure 6.9 Participant Challenge viewing Interface for The Math Quest Challenge System

Figure 6.10 Participant Challenge attempting Interface for The Math Quest Challenge System

Figure 6.11 Notifications for Rejected Registrations Interface for The Math Quest Challenge System

Figure 6.12 Error Handling Interface for The Math Quest Challenge System

# **1. INTRODUCTION**

## 1.1 Purpose

This document serves as a comprehensive guide to the design and functionality of the Math Quest Challenge system. This document shows the system's objectives, features, and requirements, providing stakeholders with a clear understanding of its scope and capabilities.

It does this through different ways such as describing the software’s architecture and components, specifying what the software should do (e.g., user registration, question selection), setting standards and constraints for developers to follow, explaining how data will be stored (e.g., questions & answers), showing mockups of screens and interactions, address authentication, authorization, and privacy.

This document also identifies external systems or APIs.

## 1.2 Scope

As a group we identified a problem with the current rigid system of having to write long essays on Mathematics questions and therefore we set out to make the process of examining students with Mathematics course units much more convenient for students and teachers.

### 1.2.1 Goals and Objectives

**Goals:**

i. Efficient Competition Management.

ii. Enhanced User Experience.

iii. Accurate and Fair Challenge Administration.

iv. Comprehensive Reporting and Analytics.

v. Secure and Scalable Architecture.

**Our objectives** are to design, implement and test the said system.

The Math Quest challenge is meant to cover aspects that will enable students in the various primary schools nationwide to participate in the competition smoothly and also ensure proper management and monitoring of the system.

Some of the key aspects the system is expected to cover are registration. (both the schools and the students intending to participate), Validation and confirmation of participants, Challenge creation with the respective answers, Question selection for individual participants, Participant management and score reporting.

It caters for three user groups which are: **administrators** responsible for system setup and management, **school representatives** responsible for ensuring registration of the schools hoping to engage in the math quest challenge, **participants** (primary school children) that are eager to engage in math quest challenge.

## 1.3 Document Overview

*Table 1.1 Document Overview*

|  |  |
| --- | --- |
| **Section** | **Description** |
| **Introduction** | Provides an overview of the Mathematics Quest Challenge System, including its purpose, scope, target audience, definitions and acronyms as well as the document overview. |
| **System Overview** | Provides a high-level description of system components and their interactions as well as details on registration, question management, challenge participation, and results processing (System description). It also includes software and hardware requirements needed to run the system efficiently. |
| **System Architecture** | Provides the technical blueprint of the system’s structure and design principles like diagrams and descriptions of system architecture and component relationships (architectural design and decomposition description). It also gives justification for architectural choices based on system requirements and constraints (Design rationale). |
| **Data Design** | Outlines the database schema, including tables such as Schools, Participants, Challenges, Questions, and Answers, along with their attributes and relationships. (ERD, data description and dictionary) |
| **Component Design** | Specifies individual system components and their functionalities including modules like Admin, Registration, Challenge, and Analytics. |
| **Human Interface Design** | Describes the various user interfaces in the system, including the Administrator Interface, Representative Interface, and Participant Interface, along with their functionalities. It includes screen images, objects and actions. |

## 

## 1.4 Definitions and Acronyms

### 1.4.1 Definitions

|  |  |
| --- | --- |
| i. Activity Diagram | Describes the flow of activities of the different actors in the system. |
| ii. Entity Relationship Diagram | Provides a visual starting point for the database design. |
| iii. Sequence Diagram | Shows the sequence of messages passed between objects of the system. |
| iv. Use case diagram | Summarizes the details of the system and the users within the system. |

### 1.4.2 Acronyms

|  |  |
| --- | --- |
| API | Application Programming Interface |
| CLI | Command Line Interface |
| DBMS | Database Management System |
| ERD | Entity Relationship Diagram |
| MQC | Math Quest Challenge |
| UI | User Interface |

# 

# **2. SYSTEM OVERVIEW**

The country at large is currently faced with a problem of organizing and managing competitions involving numerous schools and participants and also particularly, a rigid system of having to write long essays on mathematics questions which is quite complex and prone to errors since it is being handled manually.

Therefore, a web-based system is required to manage and conduct a mathematics competition for primary school children at a national level. The Math Quest Challenge system is designed to ensure efficiency, fairness, scalability and accuracy in organizing and managing the nationwide mathematics competition for primary school children. For smooth operation and comprehensive management, it encompasses key components and functionalities which include,

**i. Registration and Validation:**

Participants register using a java command-line interface (CLI) with a username, first name, last name, email address, date of birth, school registration number and an image file for identification.

**Functionality:**

The system validates school registration numbers against a list of registered schools and sends email notifications to school representatives for confirmation of participant registration.

School representatives are also normally validated before being registered into the system.

**ii. Administrator Functions:**

An administrator must be registered to use the system.

The administrator’s roles include,

1. Uploading school data including name, district, registration number, email of representative, and name of representative into the system.
2. Uploading questions (questions.xlsx) and answers (answers.xlsx) into the system and ensuring questions and answers are correctly paired and aligned for challenge creation.
3. Setting parameters for each challenge including opening and closing dates, duration of the challenge, number of questions to be presented per challenge attempt (that is to say, if a challenge is to consist of 10 questions, 100 questions should be uploaded such that for each attempt, the 10 questions are selected randomly from the 100 questions).

**iii. Participant Interaction:**

The prospective participant logs into the system and sees details of the challenges using view challenges command via the command line interface.

**View Challenges:**

1. Displays valid challenges that participants can choose to participate in.
2. Allows participants to select which challenge they wish to participate in and they can participate in more than one if they would like to.

**Challenge Participation:**

As long as the dates are valid, a participant can log in and attempt challenges using the attempt challenge command, which:

1. Randomly selects questions from the uploaded pool (questions.xlsx).
2. Limits participants to three attempts per challenge.
3. Deducts three marks for incorrect answers and awards zero marks for unsure responses (the participant can enter a negative or - for unsure answers).
4. Generates a score and detailed report upon completion of the challenge.

***Also***, the questions are presented one by one and each time a question is presented, the number of remaining questions and time are indicated above that question.

**iv. School Representative Interaction:**

School representatives log into the system and use the view applicants command via the command line interface.

**View Applicants:**

1. Displays prospective participants awaiting confirmation.
2. Allows school representatives to confirm or reject applicants (confirm yes/no username).
3. Sends notifications to applicants based on confirmation status (accepted or rejected).

***Also***, approved applicants are moved to a participant database and rejected ones are stored separately.

**v. Analytics:**

The system identifies;

1. Most correctly answered questions.
2. School rankings based on performance.
3. Participants with incomplete challenges.
4. Performance of schools and participants over the years and time.
5. Percentage repetition of questions for a given participant across attempts.
6. Student ranking (List of worst and best performing schools for all challenges).
7. Other reports which may add value to the project.

**vi. Communication:**

**Email Notifications:** Automated emails notify participants and school representatives at various stages (registration, challenge completion).

**Reports:** Participants receive reports detailing right answers for all attempted questions (PDF reports), scores, time taken for each attempted question and the total time they have taken to complete the challenge.

**Technology used:**

**Interface:** Java Command-line interface (CLI) for participant and school representative interactions.

**Data Management:** Excel files (questions.xlsx, answers.xlsx) for initial data upload.

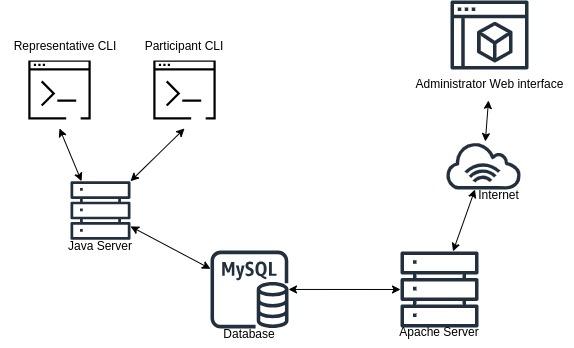
**Apache Web Server:** Periodically picks information from the java file, stores it in the system database and also posts onto the webpage that is accessed by the users.

**Backend:** Web-based system for administration, data storage (participant details, challenge data), and analytics generation.

The above system integrates administrative, participant, and analytical functionalities to efficiently manage the competition and also leverages automation to streamline processes, provide detailed performance analytics and ensure effective communication among stakeholders.

# **3. SYSTEM ARCHITECTURE**

## 3.1 Architectural Design



*Figure 3.1 System Architectural Design*

These are the components present in the architectural design of the Math Quest Challenge System;

**Command Line Interface**: The platform on which pupils register and School Representative login into the system.

**Web Interface**: The platform on which participants or school representatives are able to view the challenges, attempt challenges and also view applicants. It is also the platform on which the administrator views the summarized information about the performance of the participants.

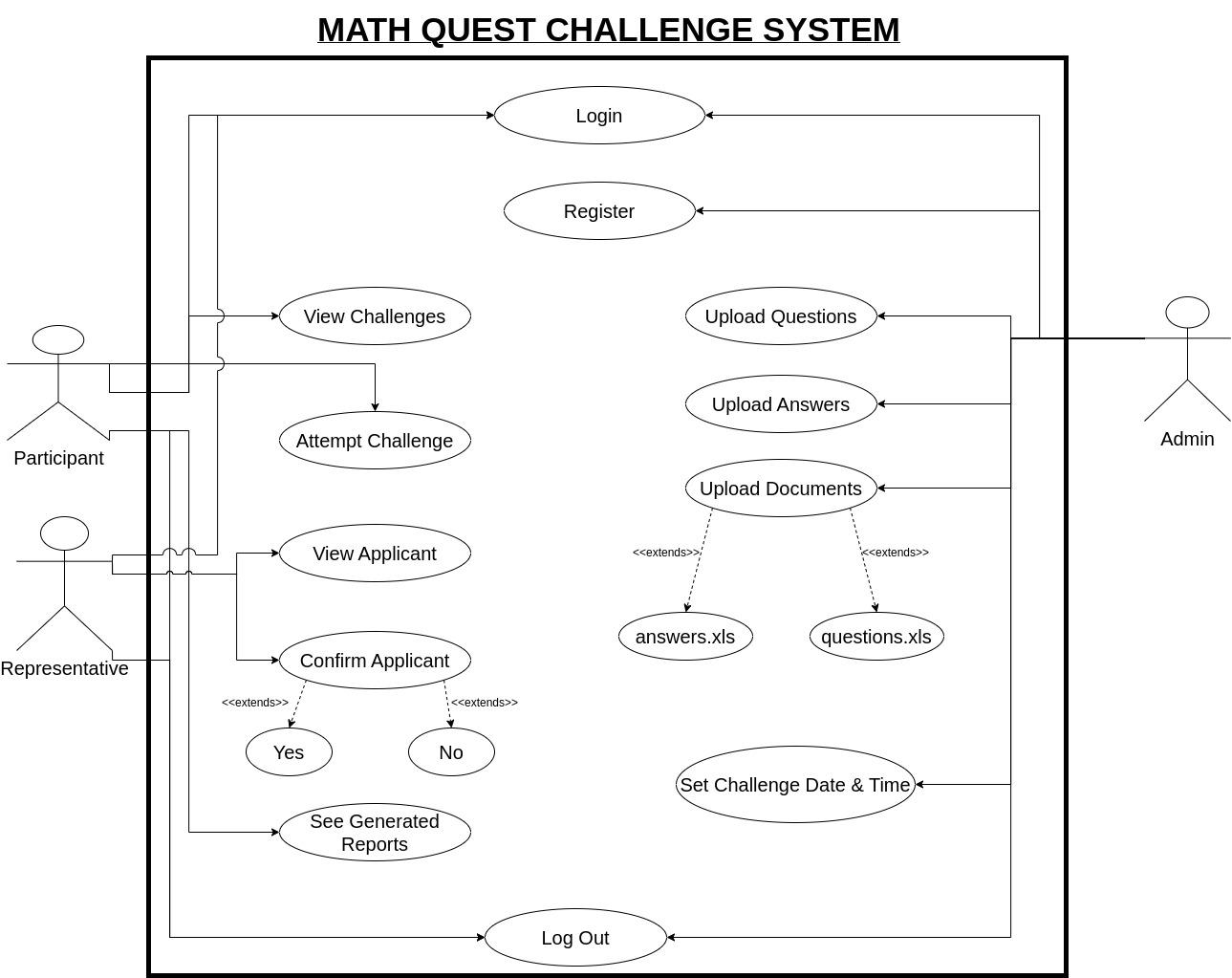
**Internet**: Provides a Network connection between the web interface and the Apache Web Server.

**Apache Server**: Controls access to the database, web interface and text file.

**Database**: Location where the files are stored i.e. questions and answers excel sheets.

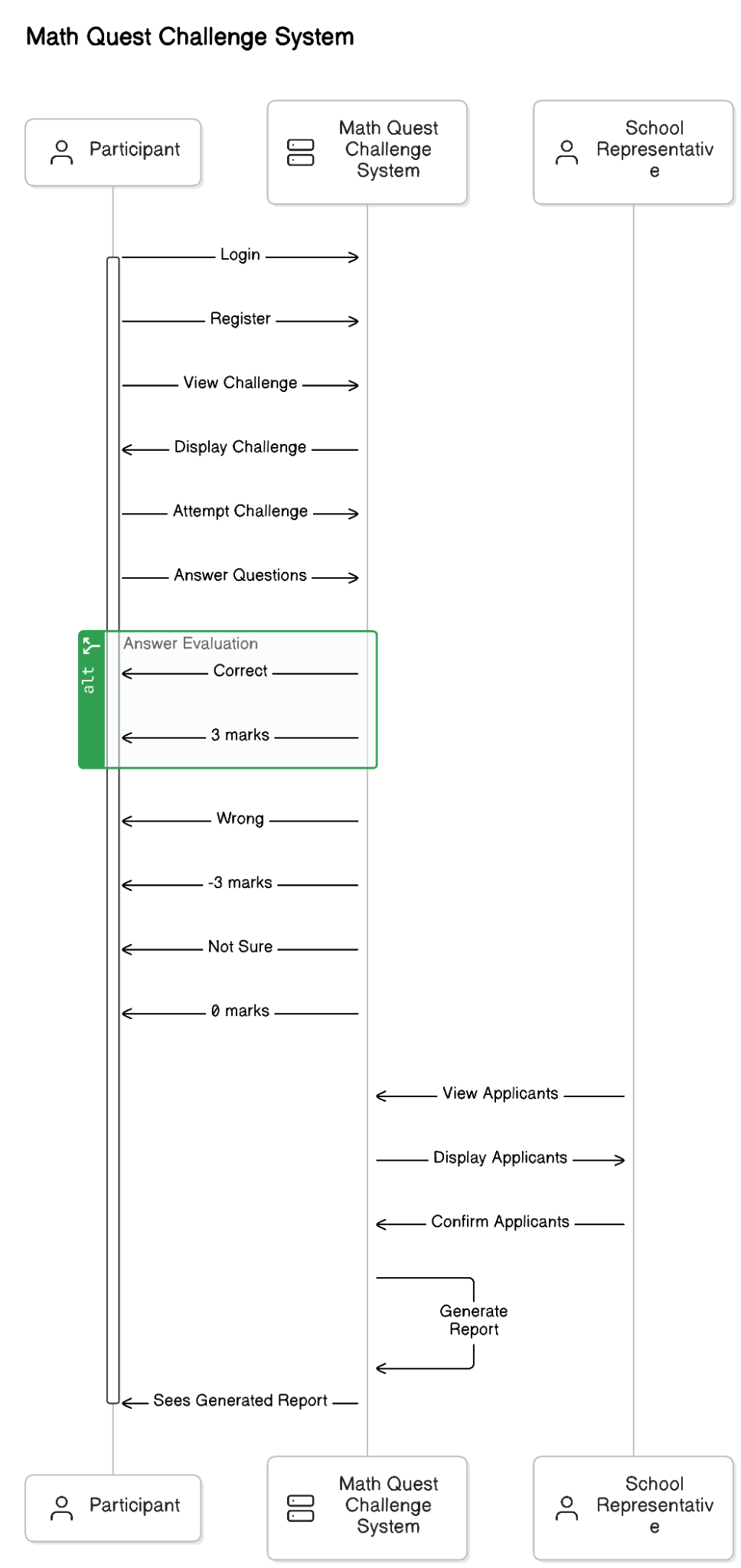
**Java server**: Implements the backend logic, data management and security features of the system.

## 3.2 Decomposition Description



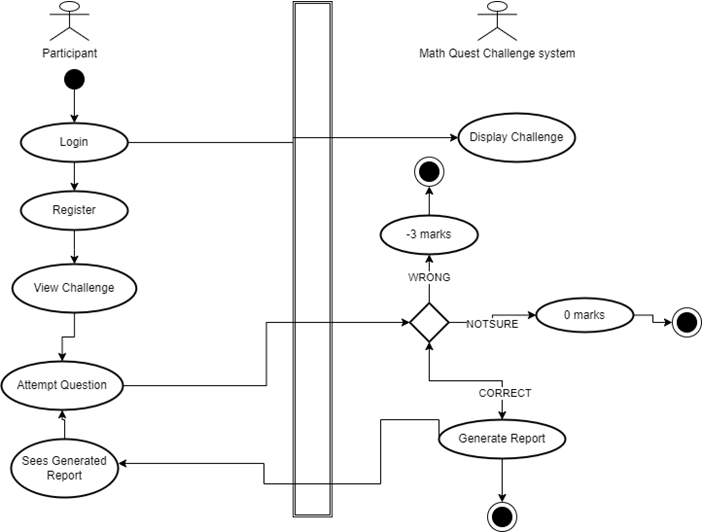
*Figure 3.2 Use Case Diagram of the system*

Figure 3.2 shows a Use Case Diagram of the system. This describes the different actors as they interact with the system and what they do.



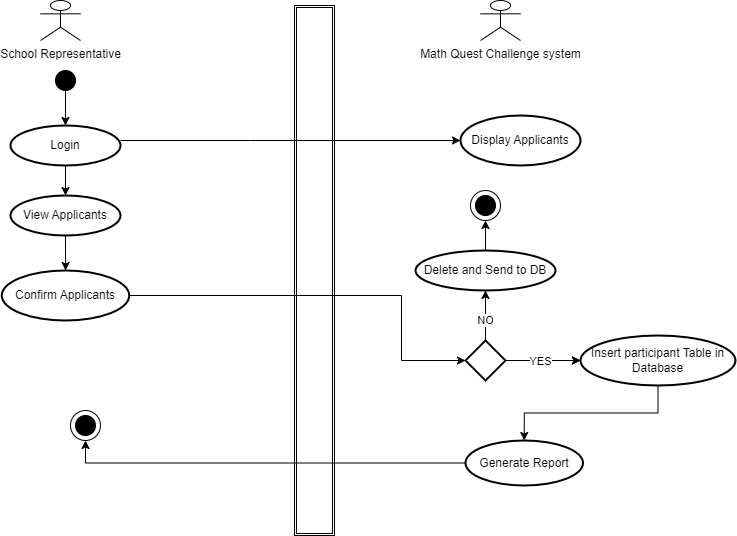
*Figure 3.3 Sequence Diagram of the system*

Figure 3.3 shows a sequence diagram displaying the sequence of message passing between objects of the system.



*Figure 3.4 Activity diagram showing participant registration*

Figure 3.4 shows an activity diagram displaying the flow of the participant’s interaction with the system.



*Figure 3.5. Activity diagram showing School Representative*

Figure 3.5 shows the flow of the School Representative interaction with the system.

## 3.3 Design Rationale

The architectural design in Figure 3.1 was considered because it fully expresses the requirements of the key players. The whole system revolves around two key players being the participant and the school representatives. The participant/pupil interacts with the command line interface on a desktop computer and all data is saved in a text file. The need to have a central repository of data that allows shared access attracts the deployment of a client-tier system; with a relational database management system that controls the movement of information to and from the text file on the participant’s side and the web interface on the customer’s side.

# **4. DATA DESIGN**

In today's data-driven world, unlocking the power of information starts with a strong foundation. This section defines how data will be organized, structured, and accessed, ensuring it's clear, reliable, and efficient to use. This meticulous planning empowers us to extract valuable insights, generate accurate reports, and ultimately, make data-driven decisions that propel the Math Quest system forward.

## 4.1 Data Description

Data is primarily stored in a relational database and accessed by both Java and PHP components. Key data entities include participants, challenges, questions, and results.

The data for Math Quest is stored in a relational database, accessed by Java and PHP components. Key data entities include:

**School**: Represents a school participating in the competition.

**Participant**: Represents students registered to participate in the competition.

**Question**: Contains questions for the challenges.

**Challenge**: Defines the challenges participants must complete.

**Result**: Stores the outcomes of the challenges.

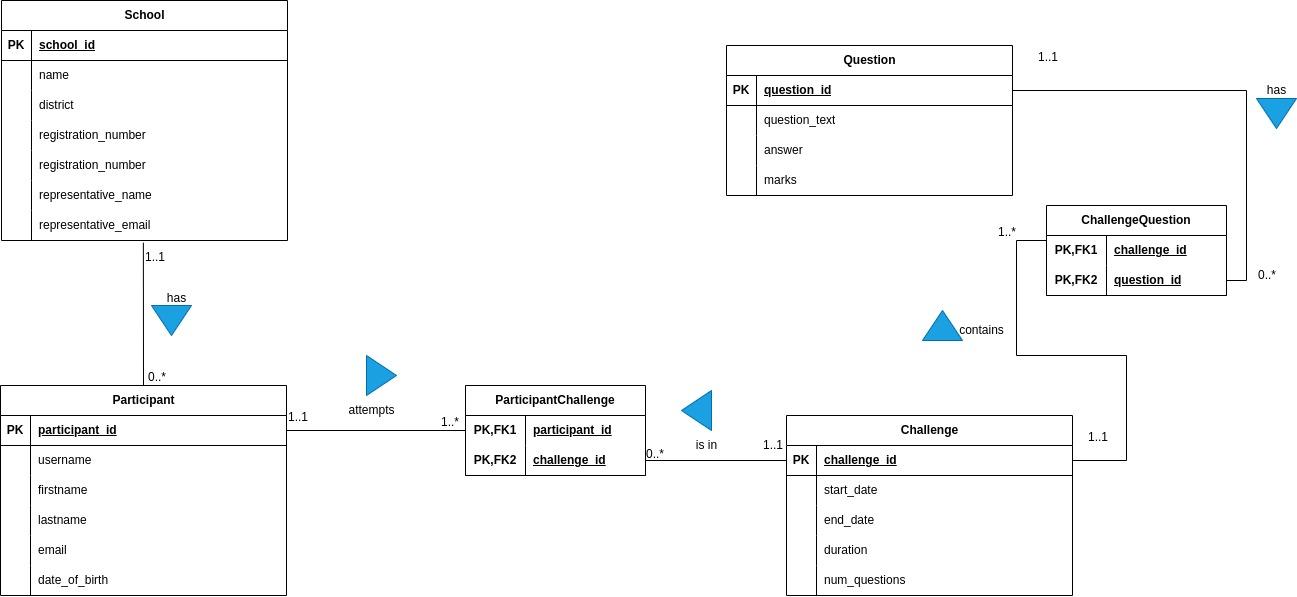
Data entities:

*Table 1.2 Data entities for The Math Quest Challenge System*

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Description |
| School | school\_id | Id of school in the database |
| name | Name of the school |
| district | District in the country where the school belongs |
| registration\_number | Registration number assigned to each school |
| representative\_number | Number assigned to every school representative |
| representative\_name | Name of school representative |
| representative\_email | Email of school representative |
| Question | question\_id | Id of each question |
| question\_text | Content contained in each question |
| answer | Answer to each question |
| marks | Marks assigned to each question |
| Participant | paticipant\_id | Id of each participant |
| username | Display name shown on the interface |
| firstname | Participant’s legal first name |
| lastname | Participant’s legal last name |
| email | Participant’s working email |
| date\_of\_birth | Participant’s legal date of birth |
| Challenge | challenge\_id | Id of challenge |
| start\_date | Date of starting challenge |
| end\_date | Date of ending challenge |
| duration | Duration of each challenge |
| num\_questions | Number of questions contained in each challenge |

## 4.2 Entity Relationship Diagram

This is an entity relationship diagram illustrating how the various entities will interact with each other within the Math Quest Challenge system.



*Figure 4.1. Entity Relationship diagram of the Math Quest System*

## 4.3 Data Dictionary

*Primary keys and data types listed in detail here as we store them in MySQL:*

*Table 1.3 Data Dictionary for The Math Quest Challenge System*

|  |  |  |  |
| --- | --- | --- | --- |
| TABLE | ATTRIBUTES | DATA TYPE | SIZE |
| schools | school\_id (Primary Key)  name  district  registration\_number  representative\_name  representative\_email | INT  VARCHAR  VARCHAR  VARCHAR  VARCHAR  VARCHAR | 10  255  40  10  255  255 |
| questions | question\_id (Primary Key)  question\_text  answer  marks | INT  VARCHAR  VARCHAR  INT | 10  255  255  10 |
| participants | participant\_id (Primary Key)  username  firstname  lastname  email  date\_of\_birth | INT  VARCHAR  VARCHAR  VARCHAR  VARCHAR  DATE | 10  255  255  255  255  12 |
| challenges | challenge\_id (Primary Key)  start\_date  end\_date  duration  num\_questions | INT  DATE  DATE  INT  INT | 10  12  12  10  10 |

# **5. COMPONENT DESIGN**

This section provides detailed algorithms for the functionalities implemented within the ‘Admin Module’, ‘Registration Module’, ‘Challenge Module’, and ‘Analytics Module’ classes. Each algorithm delineates the systematic process for the corresponding functionalities.

## 5.1 Admin Module

### 5.1.1 uploadQuestions(String filePath)

This method is responsible for uploading questions from an Excel file located at the specified ‘filePath’.

*Algorithm:*

1. Input: Path to the Excel file (‘filePath’).
2. Initialize: Open the Excel file located at ‘filePath’.
3. Read: For each row in the Excel file:
4. Extract the question text.
5. Extract any other relevant data (e.g., question type, difficulty level).
6. Store: Save each extracted question into the database or internal storage. Close: Close the Excel file.
7. Output: Confirmation message that questions have been uploaded successfully.

### 5.1.2 uploadAnswers(String filePath)

This method uploads answers from an Excel file located at the given ‘filePath’.

*Algorithm:*

1. Input: Path to the Excel file (‘filePath’).
2. Initialize: Open the Excel file located at ‘filePath’.
3. Read: For each row in the Excel file:
   1. Extract the answer text corresponding to each question.
   2. Extract any other relevant data (e.g., marks for the answer).
4. Store: Save each extracted answer into the database or internal storage.
5. Close: Close the Excel file.
6. Output: Confirmation message that answers have been uploaded successfully.

### 5.1.3 setChallengeParameters(Date startDate, Date endDate, int duration, int numberOfQuestions)

This method sets the parameters for a challenge, including the start and end dates, duration, and number of questions.

*Algorithm:*

1. Input: startDate, endDate, duration, numberOfQuestions.
2. Validate: Ensure the dates are valid and the number of questions is within a permissible range.
3. Store: Save the challenge parameters into the database or configuration storage.
4. Output: Confirmation message that the challenge parameters have been set successfully.

### 5.1.4 login(String username, String password)

This method allows a user to log into the system using their credentials.

*Algorithm:*

1. Input: Username and password.
2. Validate:Check if the provided credentials match any user record in the database.
3. Authorize:
   1. If valid, create a session for the user.
   2. If invalid, return an error message.
4. Output: Confirmation message that the user has logged in successfully or an error message.

### 5.1.5 logout()

This method logs the current user out of the system.

*Algorithm:*

1. Input: None.
2. Terminate Session: End the user’s session.
3. Output: Confirmation message that the user has logged out successfully.

## 5.2. Registration Module

### 5.2.1 registerParticipant(String username, String firstname, String lastname, String email, Date dob, String schoolRegNum, String imageFile)

This method registers a new participant with their details and stores their information in a file or database.

*Algorithm:*

1. Input: username, firstname, last name, email, dob, schoolRegNum, imageFile.
2. Validate: Check if the school registration number (schoolRegNum) exists. If not, display an error message and exit.
3. Store: Save the participant’s details into a file or database.
4. Notify: Send a notification to the school representative for confirmation.
5. Output: Confirmation message that the participant has been registered (pending confirmation).

### 5.2.2 confirmParticipant(String username, Boolean isConfirmed)

This method confirms or rejects a participant based on the ‘isConfirmed’ flag.

*Algorithm:*

1. Input: username, isConfirmed.
2. Fetch: Retrieve the participant’s pending registration details.
3. Decision:
   1. If ‘isConfirmed’ is true, move the participant from the pending list to the accepted list (database).
   2. If false, move the participant to the rejected list (database).
4. Notify: Send a notification email to the participant about their status.
5. Output: Confirmation message that the participant has been confirmed or rejected.

### 5.2.3 viewApplicants()

This method is responsible for allowing the school representatives to view all applicants who have registered for the competition but are pending confirmation.

*Algorithm*:

1. Input: None.
2. Fetch: Retrieve the list of applicants from the pending registration file or database.
3. Display: Show the list of applicants with details such as username, full name, email, date of birth, and school registration number.
4. Output: A list of applicants pending confirmation.

## 5.3. Challenge Module

### 5.3.1 attemptChallenge(int participantId, int challengeId)

This method initiates a challenge for a participant by presenting random questions.

*Algorithm:*

1. Input: participantId, challengeId.
2. Fetch: Retrieve the challenge details and participant’s information.
3. Select: Use ‘getRandomQuestions’ to pick random questions for the challenge.
4. Present: Display questions one by one to the participant.
5. Track: Monitor the time and remaining questions.
6. Calculate: At the end of the challenge, use ‘calculateScore’ to compute the score.
7. Store: Save the results of the challenge in the database.
8. Output: Display or send the score and report to the participant.

### 5.3.2 viewChallenge(int challengeId)

This method allows users to view details of all challenges available in the system.

*Algorithm:*

1. Input**:** None.
2. Fetch**:** Retrieve the list of all challenges from the database.
3. Display**:** Show the list of challenges with details such as challenge ID, start date, end date, duration, and number of questions.
4. Output**:** A list of all available challenges.

### 5.3.3 getRandomQuestions(int challengeId, int numberOfQuestions)

This method selects a specified number of random questions for a given challenge.

*Algorithm:*

1. Input: challengeId, numberOfQuestions.
2. Fetch: Retrieve all available questions for the challenge.
3. Select: Randomly choose ‘numberOfQuestions’ from the available questions.
4. Return: Return the list of selected questions.

### 5.3.4 calculateScore(int participantId, List<Answer> answers)

This method calculates the score for a participant based on their answers.

*Algorithm:*

1. Input: participantId, answers.
2. Initialize: Set the initial score to 0.
3. Evaluate: For each answer in answers:
   1. Compare the given answer with the correct answer.
   2. Apply scoring rules (e.g., add marks, subtract for wrong answers, zero for unsure).
4. Store: Save the calculated score to the database.
5. Return: Return the calculated score.

## 5.4. Analytics Module

### 5.4.1 generateReports()

This method generates performance reports and analytics for the system.

*Algorithm:*

1. Fetch: Retrieve all relevant data (e.g., challenges, scores, participants).
2. Process: Analyze data to generate reports (e.g., performance summaries, rankings).
3. Format: Format the analyzed data into readable reports.
4. Store: Save the generated reports to a specified location (e.g., database, file).
5. Output: Provide access or notify stakeholders about the reports.

### 5.4.2 getMostCorrectlyAnsweredQuestions()

This method retrieves the most correctly answered questions.

*Algorithm:*

1. Fetch: Retrieve the answer data for all questions.
2. Analyze: Count the number of correct answers for each question.
3. Sort: Sort questions by the number of correct answers in descending order.
4. Return: Return the list of most correctly answered questions.

### 5.4.3 getSchoolRankings()

This method retrieves the rankings of schools based on their participants’ performances.

*Algorithm:*

1. Fetch: Retrieve performance data for all participants.
2. Aggregate: Group and aggregate data by schools (e.g., average scores).
3. Rank: Rank schools based on aggregated performance metrics.
4. Return: Return the list of school rankings.

### 5.4.4 getParticipantsWithIncompleteChallenges()

This method retrieves the list of participants who have incomplete challenges.

*Algorithm:*

1. Fetch: Retrieve data on all challenges attempted by participants.
2. Filter: Identify participants who have not completed their challenges.
3. Return: Return the list of participants with incomplete challenges.

### 5.4.5 getStudentRanking()

This method retrieves the ranking of students based on their performance.

*Algorithm:*

1. Input: None.
2. Fetch: Retrieve performance data for all participants.
3. Rank: Rank participants based on their scores.
4. Return: Return the list of participants ranked by their performance.

### 5.4.6 getPerformanceOverYears()

This method retrieves the performance of schools and participants over the years and time. *Algorithm:*

1. Input: None.
2. Fetch: Retrieve historical performance data from the database.
3. Analyze: Aggregate and analyze the data over different time periods (e.g., yearly, monthly).
4. Visualize: Create graphs or charts to represent the performance trends.
5. Return: Return the analysis results and visualizations.

### 5.4.7 getQuestionRepetitionPercentage(int participantId)

This method calculates the percentage of repeated questions for a given participant across attempts.

*Algorithm:*

1. Input: participantId.
2. Fetch: Retrieve all questions attempted by the participant across different challenges.
3. Analyze: Identify and count the repeated questions.
4. Calculate: Compute the percentage of repeated questions.
5. Return: Return the percentage of repeated questions for the participant.

# **6.HUMAN INTERFACE DESIGN**

## 6.1 Overview of User Interface

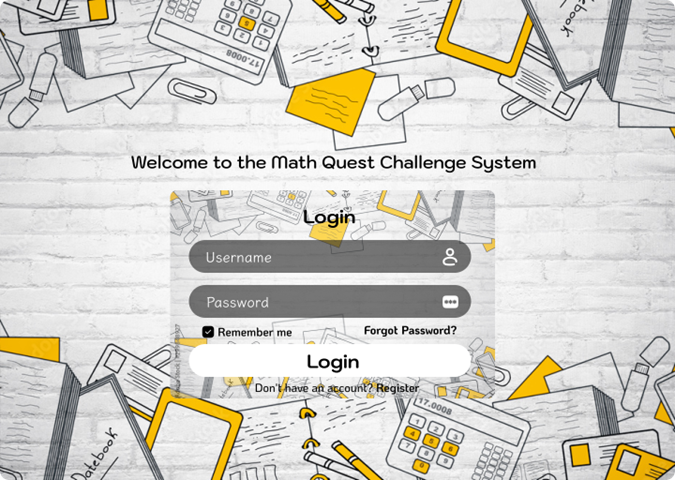
The user interface for the National Primary School Mathematics Competition system is primarily a command-line interface (CLI). This section provides detailed descriptions of the user interactions and command sequences for different user roles: administrators, school representatives, and participants.

## 6.2 Screen Images

Since the interface is text-based, the "screen images" will be represented as text screenshots showing typical interactions.

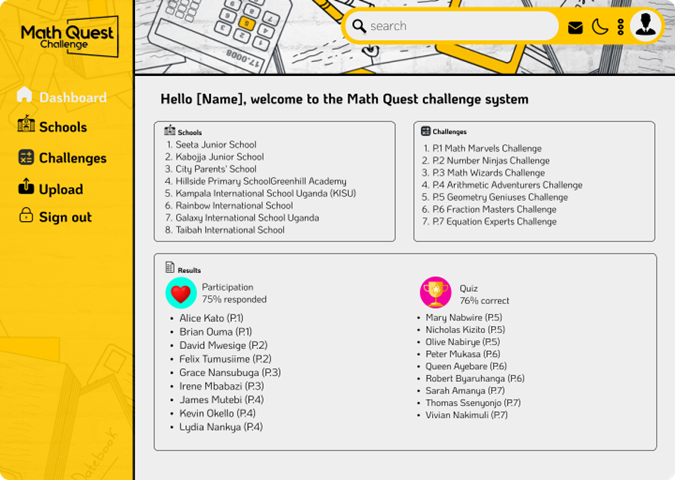
### 6.2.1 Administrator Interface

**Login Screen**

****

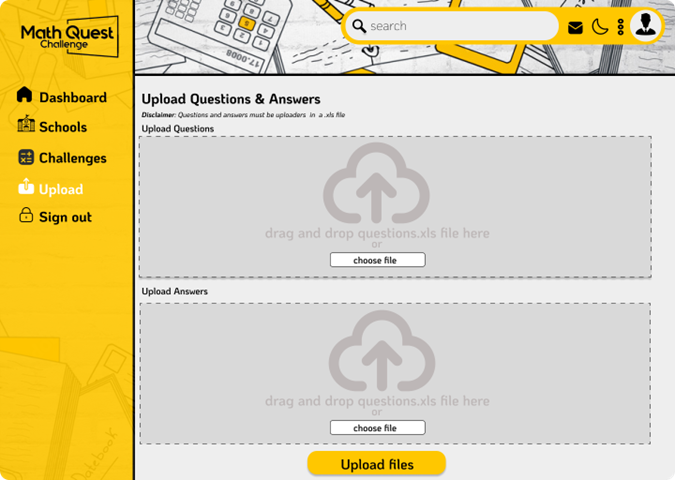
*Figure 6.1 Administrator Login Interface for The Math Quest Challenge System*

**Upload Schools**

****

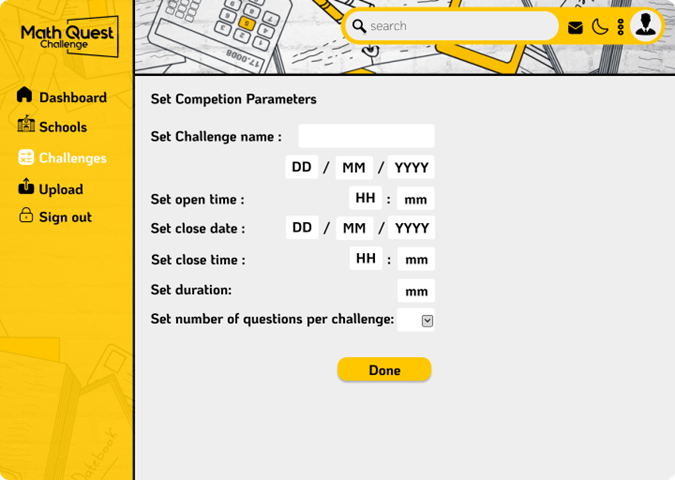
*Figure 6.2 Uploading Schools Administrator Interface for the Math Quest Challenge System*

**Upload Questions and Answers**



*Figure 6.3 Uploading Questions and Answers Interface for The Math Quest Challenge System*

**Set Competition Parameters**



*Figure 6.4 Setting Competition parameters Interface for The Math Quest Challenge System*

### 6.2.2 School Representative Interface

**Login Screen**

Welcome to the National Mathematics Competition System

Please enter your school representative credentials:

Username: rep\_smith

Password: \*\*\*\*\*\*

Login successful.

*Figure 6.5 School Representative login Interface for The Math Quest Challenge System*

**View Applicants**

rep\_smith@system: ~$ viewApplicants

Pending Applicants:

1. Username: john\_doe, Registration Number: SCH1234

2. Username: jane\_doe, Registration Number: SCH1234

Total: 2

*Figure 6.6 School Representative Applicants viewing Interface for The Math Quest Challenge System*

**Confirm or Reject Applicants**

rep\_smith@system: ~$ confirm yes john\_doe

Applicant john\_doe confirmed successfully.

rep\_smith@system: ~$ confirm no jane\_doe

Applicant jane\_doe rejected successfully.

*Figure 6.7 School Representative Applicant Confirmation or Rejection Interface for The Math Quest Challenge System*

### 6.2.3 Participant Interface

**Registration**

Welcome to the National Mathematics Competition System

Please register to participate:

Register username firstname lastname email date\_of\_birth school\_registration\_number image\_file.png

Example:

register john\_doe John Doe john.doe@example.com 2010-05-14 SCH1234 photo.png

Input:

register alice\_smith Alice Smith alice.smith@example.com 2011-03-22 SCH5678 alice.png

Output:

Registration successful. Awaiting confirmation from your school representative.

*Figure 6.8 Participant Registration Interface for The Math Quest Challenge System*

**View Challenges**

alice\_smith@system: ~$ viewChallenges

Available Challenges:

1. Challenge ID: 1, Open Date: 2024-07-01, Close Date: 2024-07-15, Duration: 60 minutes

Total: 1

*Figure 6.9 Participant Challenge viewing Interface for The Math Quest Challenge System*

**Attempt Challenge**

alice\_smith@system: ~$ attemptChallenge 1

Starting Challenge 1...

Question 1/10: What is 5 + 7?

Remaining Time: 59:50

Enter your answer: 12

Question 2/10: What is the capital of France?

Remaining Time: 59:30

Enter your answer: Paris

[...]

Challenge completed.

Your score: 75/100

A detailed report has been emailed to you.

*Figure 6.10 Participant Challenge attempting Interface for The Math Quest Challenge System*

### 6.2.4 Additional User Interface Elements

**Notifications for Rejected Registrations**

john\_doe@system: ~$ register john\_doe John Doe john.doe@example.com 2010-05-14 SCH1234 photo.png

Registration failed. You have been rejected by the school representative.

*Figure 6.11 Notifications for Rejected Registrations Interface for The Math Quest Challenge System*

**Error Handling**

alice\_smith@system: ~$ attemptChallenge 2

Error: Challenge ID 2 not found or not available.

alice\_smith@system: ~$ register alice\_smith Alice Smith alice.smith@example.com 2011-03-22 INVALID\_SCH photo.png

Error: School registration number INVALID\_SCH not found. Please check and try again.

*Figure 6.12 Error Handling Interface for The Math Quest Challenge System*

## 6.3 Screen Objects and Actions

It allows users to submit their credentials (usually a username/email and password) to gain access to the application.

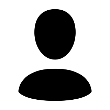
**Login**

It allows new users to create an account by submitting their personal information.

**Register**

Used to submit the form data to a server.

**Done**

 Typically represents a user's account or personal profile within an application.

 It allows users to express approval, enjoyment, or support for challenges.

 It symbolizes achievement, success, or recognition.