**Software Design Document**

**Team Name**

**Team Alpha**

**Project Title**

**Mathematics Challenge System**

**Name(s)**

**John Doe, Jane Smith, Alice Johnson**

**Lab Section**

**Section A**

**Workstation**

**WS-101**

**Date**

**06/17/2024**

**1. Introduction**

**1.1 Purpose**

This Software Design Document describes the architecture and system design of the Mathematics Challenge System. It is intended for developers, testers, and stakeholders involved in the development and maintenance of the system.

**1.2 Scope**

The Mathematics Challenge System is a web-based platform designed to host mathematics competitions for primary school children. The system allows administrators to manage schools, questions, and challenges, while participants can register, view, and attempt challenges. The project aims to provide a fair and engaging competition environment with automated scoring and reporting.

**1.3 Overview**

This document outlines the system architecture, data design, component design, and user interface design. It provides a detailed description of the system's structure and functionality, guiding the development process.

**1.4 Reference Material**

* IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions
* Project Assignment Document

**1.5 Definitions and Acronyms**

* SDD: Software Design Document
* SRS: Software Requirements Specification
* DFD: Data Flow Diagram
* PDL: Procedural Description Language
* OO: Object-Oriented

**2. System Overview**

The Mathematics Challenge System facilitates nationwide mathematics competitions for primary school pupils. Administrators manage schools, upload questions and answers, and set challenge parameters. Pupils register, view, and participate in challenges through a command-line interface (CLI). The system tracks attempts, scores participants, and provides detailed reports.

**3. System Architecture**

**3.1 Architectural Design**

The system is divided into three main components:

* **Client (Java)**: Provides menu items and reports through a CLI.
* **Server (Java)**: Handles file and database manipulations, processes CLI commands.
* **Web Interface (PHP, Laravel)**: Manages the administrative interface, participant registration, and report generation.

**3.2 Decomposition Description**

* **Client Component**:
  + CLI Menu
  + Registration Handler
  + Challenge Viewer
  + Challenge Attempter
* **Server Component**:
  + User Management
  + School Management
  + Challenge Management
  + Reporting
* **Web Interface**:
  + Admin Dashboard
  + Question Management
  + Result Analytics

**3.3 Design Rationale**

The architecture is designed to separate concerns between user interaction, data processing, and administrative functions. Java is chosen for its robustness in handling backend processes, while PHP with Laravel is selected for its simplicity and efficiency in creating web interfaces. This separation ensures scalability and maintainability.

**4. Data Design**

**4.1 Data Description**

The system uses a relational database to store data related to schools, participants, questions, challenges, and results. Data is imported from Excel files and processed accordingly.

**4.2 Data Dictionary**

* **School**: {school\_id, name, district, registration\_number, representative\_name, representative\_email}
* **Participant**: {participant\_id, username, firstname, lastname, email, date\_of\_birth, school\_registration\_number}
* **Question**: {question\_id, question\_text, correct\_answer, marks}
* **Challenge**: {challenge\_id, start\_date, end\_date, duration, question\_count}
* **Attempt**: {attempt\_id, participant\_id, challenge\_id, score, time\_taken, status}

**5. Component Design**

**Client Component**

* **Registration Handler**:
  + Input: User details
  + Process: Validate input, register user
  + Output: Confirmation message
* **Challenge Viewer**:
  + Input: User request
  + Process: Fetch valid challenges
  + Output: List of challenges
* **Challenge Attempter**:
  + Input: Challenge selection
  + Process: Fetch questions, track attempts
  + Output: Score and report

**Server Component**

* **User Management**:
  + Process: Handle user registration, validation, and login
* **School Management**:
  + Process: Manage school details, representative verification
* **Challenge Management**:
  + Process: Handle challenge creation, question assignment
* **Reporting**:
  + Process: Generate and send reports, analytics

**Web Interface**

* **Admin Dashboard**:
  + Functionality: Manage schools, users, challenges
* **Question Management**:
  + Functionality: Upload and manage questions and answers
* **Result Analytics**:
  + Functionality: Display performance statistics, rankings

**6. Human Interface Design**

**6.1 Overview of User Interface**

The CLI provides a text-based interface for participants to register, view challenges, and attempt challenges. The web interface offers a graphical interface for administrators to manage the system and view reports.

**6.2 Screen Images**

**CLI Menu:**

markdown

Copy code

Welcome to Mathematics Challenge

1. Register

2. View Challenges

3. Attempt Challenge

4. Exit

**Admin Dashboard (Web Interface):**

**6.3 Screen Objects and Actions**

* **CLI Menu**:
  + Options: Register, View Challenges, Attempt Challenge
  + Actions: Navigate through menu, enter commands
* **Admin Dashboard**:
  + Objects: School List, Participant List, Challenge List
  + Actions: Add, Edit, Delete, View Details

**7. Requirements Matrix**

| **Requirement ID** | **Description** | **Component** |
| --- | --- | --- |
| R1 | Register participant | Client, Server |
| R2 | Validate school registration | Server |
| R3 | Upload questions and answers | Web Interface |
| R4 | Create and manage challenges | Server, Web Interface |
| R5 | Attempt challenge | Client, Server |
| R6 | Generate reports | Server, Web Interface |
| R7 | Display analytics | Web Interface |

**8. Appendices**

* **Appendix A**: Sample Data
* **Appendix B**: Detailed Algorithm for Challenge Attempt