Muscle Mind

Software Design Document

Version <2.0>

10/30/2024

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 2.0 |
| Indicator of Last Page in Document: | 25 |
| Date of Last Review: | 10/30/2024 |
| Date of Next Review: | 11/4/2024 |
| Target Date for Next Update: |  |

Distribution List

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members: Veronica Rivas

Customer: CS 4374 Software Construction and Bhanukiran Gurijala

Software Team Members: Ruben Martinez, Alfonso Hernandez, David Duru, Logan Armendariz, Francisco Jimenez, Caleb Lopez, Jose Legarreta

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 10/6/24 | RM and AH | Initial establishment of design |
| 1.1 | 10/14/24 | RM and AH | Aggregate classes and collaboration graphs |
| 2.0 | 10/27/24 | RM | Perform Design Review, Refactor Subsystems |

Table of Contents

[Document Control ii](#_Toc322626963)

[Approval ii](#_Toc322626964)

[Document Change Control ii](#_Toc322626965)

[Distribution List ii](#_Toc322626966)

[Change Summary ii](#_Toc322626967)

[Introduction 1](#_Toc322626968)

[1.1 Purpose and Intended Audience 1](#_Toc322626969)

[1.2 Scope of Product 1](#_Toc322626970)

[1.3 References 1](#_Toc322626971)

[1.4 Definitions, Acronyms, and Abbreviations 1](#_Toc322626972)

[1.1.1 Definitions 1](#_Toc322626973)

[1.4.1 Acronyms and Abbreviations 1](#_Toc322626974)

[1.5 Overview 1](#_Toc322626975)

[2 Decomposition Description 2](#_Toc322626976)

[2.1 System Collaboration Diagram 2](#_Toc322626977)

[2.2 Subsystem Descriptions 2](#_Toc322626978)

[3 Detailed Description of Subsystem <name> 3](#_Toc322626979)

[3.1 Subsystem Description (subsystem cards) 3](#_Toc322626980)

[3.2 Class Description (CRC cards) <class 1 name> 3](#_Toc322626981)

[3.2.1 Contract <contract 1 name> 3](#_Toc322626982)

[3.2.2 Contract <contract 2 name> 3](#_Toc322626983)

[3.3 Class Description <class 2 name> 3](#_Toc322626984)

[3.3.1 Contract <contract 1 name> 3](#_Toc322626985)

[Appendix 4](#_Toc322626986)

# Introduction

## Purpose and Intended Audience

The **Muscle-Mind** project is designed to create an immersive fitness and learning experience by combining quiz-based gameplay with physical activity. The core idea of this project is to encourage users to learn about fitness, health, and nutrition in a fun, engaging, and interactive manner. Often, traditional learning tools, especially in the areas of physical education or health, can feel static or passive. This project seeks to disrupt that by adding a physical challenge to the mental effort. Each incorrect quiz answer results in a randomized exercise penalty, transforming a simple quiz format into an engaging real-world activity that promotes physical well-being.

The primary goal of the game is to foster both mental engagement and physical movement, making it stand out compared to conventional learning platforms. Users aren't just passively absorbing information; they are actively participating by moving, exercising, and engaging their bodies. The penalty feature, which includes exercises like push-ups, squats, or jumping jacks for incorrect answers, encourages users to stay active—especially relevant in today's increasingly sedentary society. The game offers a unique combination of mental and physical activities, helping users stay both intellectually and physically sharp.

The target audience for this platform is broad. On one hand, it caters to fitness enthusiasts who are eager to combine learning with an active workout routine. On the other hand, it appeals to anyone who enjoys quiz games or trivia-style challenges and is open to integrating physical activity into the experience. This audience could range from personal trainers, gym instructors, and educators to casual users at home looking for a fun and socially interactive way to stay active with friends or family. For instance, educators might use the platform to reinforce health-related lessons in a classroom setting, while fitness instructors could use it as an entertaining warm-up or cool-down tool during sessions.

This Software Design Document (SDD) is primarily intended for those who want an in-depth overview of the system design and implementation process. This includes not only the development team members but also the design manager Francisco Jimenz, the course instructor Bhanukiran Gurijala, and the instructional team member Veronica Rivas. The development team will rely on the SDD as a technical guide for building the system, ensuring that all components—from quiz creation to exercise penalty enforcement—function as described. For Francisco Jimenz, the design manager, this document will help oversee the technical alignment with project goals, while Bhanukiran Gurijala and Veronica Rivas will use the document to monitor project progress and ensure that the final product meets course standards. Additionally, the SDD serves as a roadmap for constructing the platform in a scalable and maintainable way, ensuring the project’s long-term success and usability.

## Scope of Product

The Muscle-Mind system is designed to create an interactive fitness quiz platform that integrates educational content with physical challenges. The software products to be produced include components responsible for managing user authentication, quiz creation, quiz retrieval, quiz submission, and enforcing exercise penalties for incorrect answers. These components will ensure that users can log in, create customized quizzes, and engage in fitness challenges as part of their learning experience. In the earlier needs analysis, it was identified that traditional fitness and educational tools lack engagement, which often results in lower motivation for users to stay active or continue learning. To address this problem, Muscle-Mind offers a solution that makes learning about health, fitness, and nutrition more dynamic and motivating by combining quizzes with physical exercise penalties. The system will allow users to take quizzes on health-related topics, and when incorrect answers are given, users will face physical challenges such as squats, push-ups, or jumping jacks. This approach not only enhances knowledge retention but also promotes physical well-being by ensuring that users stay active while learning. The system’s key objectives include offering an enjoyable and interactive learning experience, increasing user motivation for both learning and physical activity, and making fitness education more engaging. The main goal is to create a versatile platform that can be used for individual fitness routines, classroom settings, or group activities, offering both educational and physical benefits. Through the combination of mental and physical challenges, Muscle-Mind aims to bridge the gap between static learning tools and the need for more engaging, active experiences.

## References

[Project Plan Sheet](https://minersutep-my.sharepoint.com/:w:/r/personal/rjmartinez12_miners_utep_edu/_layouts/15/Doc.aspx?sourcedoc=%7BA507532D-06CA-47CE-B503-89A5DB94CCD5%7D&file=scm-plan-template.doc&action=default&mobileredirect=true)

[Project Report](https://minersutep-my.sharepoint.com/:w:/r/personal/rjmartinez12_miners_utep_edu/_layouts/15/Doc.aspx?sourcedoc=%7B189B9891-0520-459D-A31E-4D2C6A17539B%7D&file=Team%203%20Software%20Construction%20Part%201%20Report.docx&action=default&mobileredirect=true)

[GitHub Repository](https://github.com/riddle-me-ruben/muscle-mind)

[Updated Design](https://minersutep-my.sharepoint.com/:b:/r/personal/rjmartinez12_miners_utep_edu/Documents/Team%203%20Software%20Construction/Project%20Part%202%20Documents/Other/Updated%20Design%20Part%202.pdf?csf=1&web=1&e=C9Jqjc)

## Definitions, Acronyms, and Abbreviations

### Definitions

* **Game Session** - An instance of a quiz game that tracks the player's progress, including questions answered, scores, and penalties.
* **Exercise Challenge** - A feature that assigns physical activities (e.g., push-ups, squats) to users as penalties when they answer quiz questions incorrectly, promoting physical engagement.
* **Performance Report** - A summary generated after a quiz game session that shows the user’s scores, completed exercises, and overall performance metrics.

### Acronyms and Abbreviations

* **MM** – Muscle Mind

## Overview

* **Introduction**: This section explains the purpose of the system and its intended audience. It sets the context for the document by describing the main goals of the project, including promoting learning through an interactive fitness quiz platform.
* **Definitions, Acronyms, and Abbreviations**: This section provides explanations for the terms, acronyms, and abbreviations used throughout the document to ensure clarity and consistency in understanding.
* **System Overview**: This high-level description of the system outlines the main goals and objectives. It offers a conceptual understanding of how the system integrates mental and physical challenges, combining quiz-based gameplay with exercise penalties.
* **Decomposition Description**: This section breaks the system down into individual subsystems and their interactions. It explains the collaboration between different components and subsystems, ensuring a modular design that is easier to maintain and scale.
* **System Collaboration Diagram**: The diagram visually represents how the various subsystems of **Muscle-Mind** interact with one another. This includes both the user management subsystem and the quiz management subsystem, showing how they collaborate to provide a seamless user experience.
* **Subsystem Descriptions**: Each subsystem is described in detail, including its purpose, supported functions, and any relevant contracts that define its interactions with other components of the system.
* **Detailed Description of Subsystems**: This section delves deeper into the design of each subsystem and their respective classes. It includes descriptions of the classes, their responsibilities, and collaborations with other parts of the system.
* **Class Responsibility Collaborator (CRC) Cards**: This section outlines the responsibilities and collaborations of each class within the system, making it clear how each component contributes to the overall functionality of the platform.
* **Contract Descriptions**: Each contract defines the protocols and expected behaviors of the classes, ensuring that their functions are well-documented and adhere to the system’s design requirements.
* **Appendix**: This section contains additional supporting materials such as high-level and subsystem collaboration diagrams, which further illustrate the interactions between different components of the system.

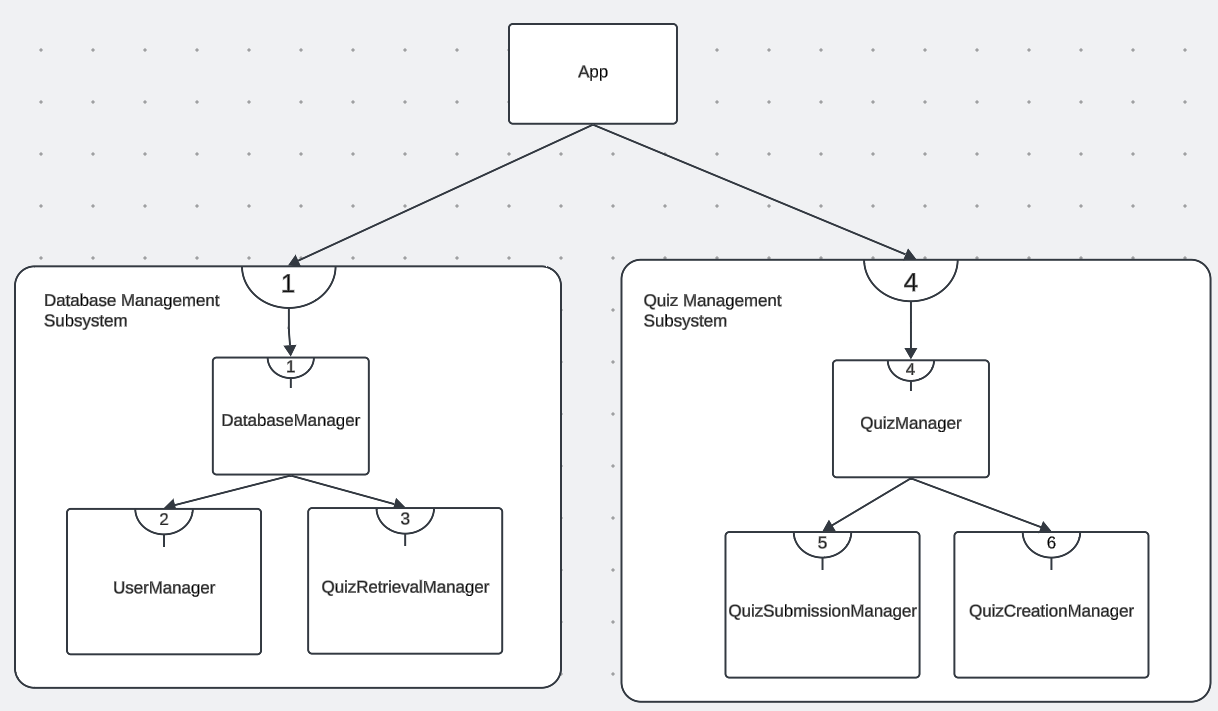
1. Decomposition Description

Component descriptions in the MuscleMind system serve as a critical reference for designers and maintainers to understand the system's structure and identify the major design entities. Each class within the system is defined with a clear set of responsibilities, and by outlining these responsibilities, designers can quickly trace which entity is responsible for specific functions. For example, the DatabaseManager class is responsible for executing database queries and managing data storage, making it the primary entity for any data-related operations. Similarly, the UserManager and QuizRetrievalManager classes handle user authentication and quiz retrieval respectively, clearly demarcating their roles in the system. This structured approach not only aids in the initial development but also simplifies future maintenance by making it easy to locate and modify specific functionality when updates or bug fixes are required.

Additionally, these component descriptions provide a clear mapping between system requirements and the corresponding design entities. For instance, if a requirement specifies that user data must be securely stored and retrieved, the designer can trace this requirement directly to the UserManager and DatabaseManager classes within the DatabaseManagementSubsystem. This traceability ensures that each requirement is accounted for within the system design and is correctly implemented by the appropriate component. By maintaining this clear link between requirements and design entities, the development process remains organized and manageable, and future modifications to the system can be implemented with confidence, knowing exactly where changes need to be made.

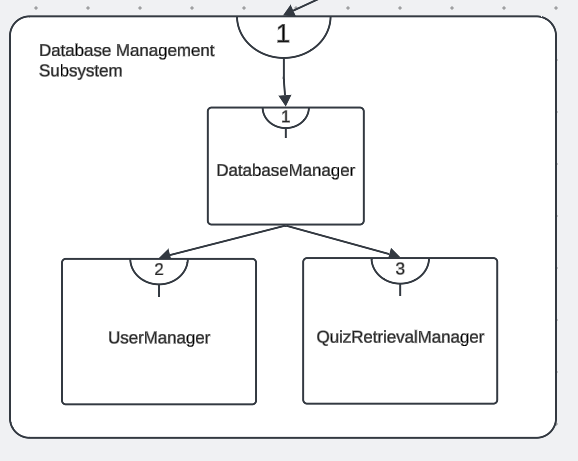
## System Collaboration Diagram

**2.1.1 High Level Overview of MuscleMind System**



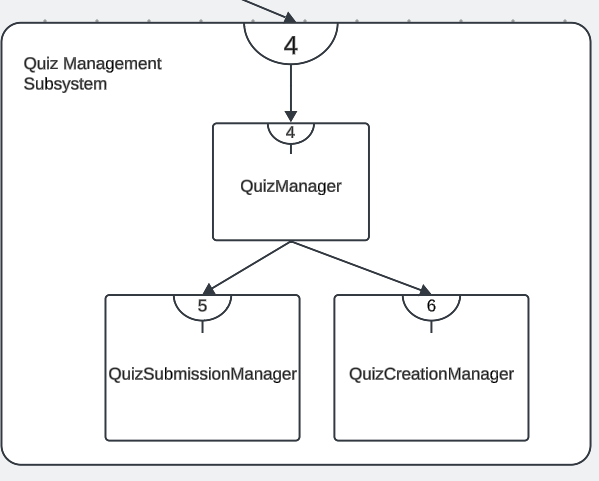
In MuscleMind, the system is structured around two key subsystems, each composed of several critical classes, which collaborate to create an interactive and educational experience. These subsystems are the Database Management Subsystem and the Quiz Management Subsystem. Each subsystem focuses on a specific area of functionality, making the system modular, scalable, and easier to manage. This modular design allows for clear separation of concerns, helping developers and maintainers trace requirements to specific design entities and identify which components are responsible for certain functions. The App class is the central component that ties both subsystems together, utilizing the services provided by the Database Management Subsystem to handle user data and quiz retrieval and the services from the Quiz Management Subsystem to enable users to create and interact with quizzes. This architecture ensures a clear distribution of responsibilities between data management and quiz functionality, making the system more maintainable and adaptable to future changes. By structuring the system this way, designers and maintainers can easily trace functional requirements—such as user login, quiz creation, or quiz submission—to specific classes within their respective subsystems, ensuring that responsibilities are clearly defined, and modifications can be localized within individual components.

**2.1.2 Database Management Subsystem**



The Database Management Subsystem includes three primary classes: DatabaseManager, UserManager, and QuizRetrievalManager. This subsystem is responsible for managing all data interactions, such as retrieving user and quiz data from the database, storing new data, and ensuring secure connections. The App class interacts with this subsystem to handle the retrieval of user profiles, quiz questions, and results, as well as to ensure that user authentication is properly managed. For example, the DatabaseManager class manages the low-level database interactions, while the UserManager class focuses on user authentication processes like login and account creation. The QuizRetrievalManager handles fetching quizzes by ID and managing quiz-related queries, allowing the app to access the necessary information when users participate in or create quizzes.

**2.1.2 Quiz Management Subsystem**



The Quiz Management Subsystem is composed of the QuizManager, QuizSubmissionManager, and QuizCreationManager classes. This subsystem is responsible for handling all quiz-related actions, including creating, submitting, and taking quizzes. The App class interacts with this subsystem to manage the quiz lifecycle—from creation to submission and grading. The QuizManager oversees the high-level management of quizzes, including fetching user-created quizzes and managing penalties for incorrect answers. The QuizCreationManager focuses on handling the creation process, ensuring that quizzes are built correctly with valid questions and stored appropriately. The QuizSubmissionManager manages the process of taking quizzes and calculating the score, including applying exercise penalties when users answer questions incorrectly.

## Subsystem Descriptions

1. **Database Management Subsystem**
   1. **Purpose:** The Database Management Subsystem is responsible for handling all interactions between the system and the database. Its primary function is to securely store, retrieve, and manage user and quiz data, ensuring data integrity and consistency across the platform. This subsystem acts as the backend data handler, supporting user authentication, session management, and the retrieval of quizzes for gameplay.
   2. **Contracts:**
2. **ManageDatabaseOperations (DatabaseManager class)**:

This contract is responsible for handling all interactions with the database, including establishing and closing connections, executing SQL queries, and committing changes to the MySQL database. The DatabaseManager ensures that user and quiz data are handled efficiently and securely, supporting other components by providing a reliable interface for database operations.

1. **ManageUserSessions (UserManager class)**:

This contract manages user-related session data and authentication operations. It ensures that users can log in, log out, register, and maintain session states within the system. The UserManager works in collaboration with the DatabaseManager to store and retrieve user-specific information, such as login credentials and session data, enabling smooth user interactions and authentication processes.

1. **ManageQuizRetrieval (QuizRetrievalManager class)**:

This contract handles retrieving quiz data, including questions and options, from the database. It ensures that the quizzes are properly structured and made available for users to take. The QuizRetrievalManager is responsible for executing SQL queries to retrieve quiz content, ensuring that the data is accurately displayed to the user during quiz-taking or quiz review sessions.

* 1. **Location in Document:** Section 3.1 Page 13

1. **Quiz Management Subsystem**
   1. **Purpose:** The Database Management Subsystem is responsible for handling all interactions between the system and the database. Its primary function is to securely store, retrieve, and manage user and quiz data, ensuring data integrity and consistency across the platform. This subsystem acts as the backend data handler, supporting user authentication, session management, and the retrieval of quizzes for gameplay.
   2. **Contracts:**
2. **ManageQuizControl (QuizManager class)**:

This contract is responsible for overseeing the entire quiz lifecycle, including the creation, retrieval, and submission of quizzes. The QuizManager coordinates between the QuizCreationManager, QuizRetrievalManager, and QuizSubmissionManager to ensure that all quiz-related activities are managed efficiently and seamlessly for the user.

1. **ManageQuizSubmission (QuizSubmissionManager class)**:

This contract handles all aspects of quiz submissions, including validating answers and calculating scores. It ensures that user responses are checked against correct answers and that the final scores are calculated accurately. Additionally, it applies penalties, such as exercise tasks, for incorrect answers during the quiz. The QuizSubmissionManager is responsible for managing these interactions and ensuring the results are stored properly.

1. **ManageQuizCreation (QuizCreationManager class)**:

This contract oversees the process of creating and validating quizzes. It ensures that users can input quiz questions, multiple-choice options, and other related data. The QuizCreationManager ensures that the quiz creation form is properly structured, that quizzes meet system requirements, and that they are stored correctly in the database.

* 1. **Location in Document:** Section 3.5 Page 19

### Subsystem Summaries

**Database Management Subsystem**:

The Database Management Subsystem is the foundation of the system’s backend data operations. It handles all communication between the system and the database, ensuring that data is stored and retrieved securely and efficiently. The subsystem consists of the DatabaseManager, which executes queries and manages connections; the UserManager, which deals with user-related data like login credentials and profiles; and the QuizRetrievalManager, which is responsible for pulling quiz data from the database. Together, these entities ensure that the system can reliably access, store, and manage critical data, such as user accounts and quizzes.

**Quiz Management Subsystem**:

The Quiz Management Subsystem is designed to handle all quiz-related functionality, from creation to submission. It consists of the QuizManager, which coordinates quiz-related processes; the QuizCreationManager, which allows users to design and submit new quizzes; and the QuizSubmissionManager, which processes user responses and calculates results. This subsystem ensures that quizzes can be created, taken, and scored efficiently, providing users with immediate feedback and maintaining the integrity of the quiz-taking experience. The Quiz Management Subsystem works closely with the Database Management Subsystem to store quiz results and access user data, allowing for a fully integrated user experience.

### Final Summary:

In designing the Muscle-Mind system, we used Design by Contract (DbC) to clearly define the expected interactions between our classes and subsystems, making sure each component fulfills its role consistently. DbC allows us to outline requirements (pre-conditions) and guarantees (post-conditions) for each function, helping our team to know exactly what each part of the system expects and provides. Instead of handling raw data in isolation, our system relies on Flask’s state management, so we needed flexible contracts that account for session data and dynamic user interactions. With DbC, we can ensure each subsystem—whether handling user sessions, quiz management, or database interactions—behaves predictably and fulfills its intended purpose, contributing to a seamless experience for the user.

Since Muscle-Mind isn’t dealing purely with static data but rather interactive components, we kept our DbC approach informal yet precise. For each function, we wrote “requires” clauses that describe the needed state or inputs, like ensuring a valid user session exists before attempting to retrieve quiz data. This way, the system only moves forward with operations when the necessary conditions are met, preventing unexpected errors. Similarly, our “ensures” clauses describe expected outcomes, such as guaranteeing that quiz data is formatted correctly after retrieval or that user authentication has succeeded before granting access. These informal descriptions allow us to remain agile within Flask’s stateful environment while maintaining clear expectations.

This design method also helps us manage dependencies between components effectively. Each class or function only takes on tasks that meet its pre-conditions and only hands off data when post-conditions are guaranteed, making troubleshooting easier. By working with informal contracts, we focus on the core goals—like ensuring quiz data integrity, reliable user authentication, and secure database connections—without being bogged down by rigid specifications. This approach enables our system to be robust and adaptable, particularly in a development environment that involves continuous user interactions, session handling, and dynamic content. The result is a structured yet flexible system where each part reliably performs its role, enhancing both usability and maintainability.

The MuscleMind system is a well-structured platform that leverages both the Database Management Subsystem and the Quiz Management Subsystem to create a seamless, interactive quiz experience for users. By dividing the system into these two core subsystems, MuscleMind achieves a modular architecture where each subsystem is responsible for a specific set of tasks. The Database Management Subsystem focuses on data handling, ensuring the secure storage and retrieval of user and quiz data. On the other hand, the Quiz Management Subsystem takes care of all quiz-related functionalities, from quiz creation to submission and scoring, ensuring that users can interact with the platform in an engaging and dynamic way. Both subsystems collaborate with each other through well-defined contracts, enabling the system to operate efficiently, meet its requirements, and remain scalable for future enhancements. This design allows for easy maintenance, updates, and potential expansions, ensuring the long-term success of the MuscleMind platform.

1. Detailed Description of Subsystems

## Subsystem Descriptions (subsystem cards)

|  |
| --- |
| **Subsystem Name:** Database Management Subsystem |
| **Classes:** DatabaseManager, UserManager, QuizRetrievalManager |
| **Collaboration Graph:** |
| **Description:** The Database Management Subsystem is a critical component in the MuscleMind system that handles all interactions between the system and its database. It ensures that user and quiz data are securely stored, retrieved, and managed throughout the system. This subsystem is composed of three key classes, each with its own responsibilities and contract numbers that define their functionalities and interactions with other components. |
| **Contracts:**   1. **ManageDatabaseOperations (DatabaseManager class)**:   This contract is responsible for handling all interactions with the database, including establishing and closing connections, executing SQL queries, and committing changes to the MySQL database. The DatabaseManager ensures that user and quiz data are handled efficiently and securely, supporting other components by providing a reliable interface for database operations.   1. **ManageUserSessions (UserManager class)**:   This contract manages user-related session data and authentication operations. It ensures that users can log in, log out, register, and maintain session states within the system. The UserManager works in collaboration with the DatabaseManager to store and retrieve user-specific information, such as login credentials and session data, enabling smooth user interactions and authentication processes.   1. **ManageQuizRetrieval (QuizRetrievalManager class)**:   This contract handles retrieving quiz data, including questions and options, from the database. It ensures that the quizzes are properly structured and made available for users to take. The QuizRetrievalManager is responsible for executing SQL queries to retrieve quiz content, ensuring that the data is accurately displayed to the user during quiz-taking or quiz review sessions. |

## Class Description (CRC cards) DatabaseManager

|  |  |
| --- | --- |
| **Class Name:** DatabaseManager | |
| **Description:** The DatabaseManager class is responsible for managing connections to the MySQL database, executing queries, and committing changes to ensure smooth interaction between the application and the database. | |
| **Super classes:** None | |
| **Contracts:**  **Contract 1:** ManageDatabaseOperations  **Description:** Defines the responsibilities related to managing the database connection, executing queries, and committing changes to the MySQL database.  - Establishes and closes connections with the database  - Executes the SQL queries and returns the results  - Commits changes to the database securely    **Private Responsibilities:**  - Manage connection lifecycle  - Safely execute queries and manage transaction commits | **Collaborations:**  UserManager(2)  QuizRetrievalManager(3) |
| **Comments:**  The DatabaseManager class provides a unified interface for all database operations, handling connections, query execution, and ensuring transaction security. | |

### Contract #1: ManageDatabaseOperations

* **Establishes and closes connections with the database**
  + **Function**: connect(), close()
  + **Pre-condition**:
    - connect(): Requires that the Flask application’s database configuration is set correctly (e.g., host, user, password).
    - close(): Requires an active connection to the database.
  + **Post-condition**:
    - connect(): Ensures that a connection to the MySQL database is established.
    - close(): Ensures that the active database connection is properly closed.
  + **Collaborations**:
    - The DatabaseManager(1) class collaborates with UserManager(2) and QuizRetrievalManager(3), providing them with the service of establishing and closing database connections to perform their respective operations.
* **Executes the SQL queries and returns the results**
  + **Function**: execute\_query(query: str, params: tuple = ())
  + **Pre-condition**: Requires a valid SQL query and parameters, and an active database connection.
  + **Post-condition**: Ensures that the SQL query is executed, and the results are returned as a tuple of rows.
  + **Collaborations**:
    - This function is primarily used by QuizRetrievalManager(3) and UserManager(2) for retrieving data from the database.
* **Commits changes to the database securely**
  + **Function**: execute\_commit(query: str, params: tuple = ())
  + **Pre-condition**: Requires a valid SQL query and parameters, and an active database connection.
  + **Post-condition**: Ensures that the SQL query is executed, and the changes are committed to the database.
  + **Collaborations**:
    - The UserManager(2) class relies on this method to commit user registration data, and other database-modifying actions.

## Class Description (CRC Cards) UserManager

|  |  |
| --- | --- |
| **Class Name:** UserManager | |
| **Description:** The UserManager class is responsible for handling all user-related operations, such as managing user sessions, logging in, signing out, and registering new users. | |
| **Super classes:** None | |
| **Contracts:**  **Contract 2:** ManageUserSessions  **Description:** This contract handles the responsibilities for managing user data and session activities, such as checking if users are signed in, registering new users, and validating login credentials.  - Checks if a user is signed in and maintains session states  - Registers new users and stores user data in the database  - Validates user login credentials    **Private Responsibilities:**  - Manage user sessions (login, logout)  - Register and validate users. | **Collaborations:**  DatabaseManager(1) |
| **Comments:**  The UserManager class ensures that user-related functions, such as registration, login, and session management, are handled securely and efficiently, utilizing the services of the DatabaseManager to store and retrieve user data. | |

### Contract #2: ManageUserSessions

* **Checks if a user is signed in and maintains session states**
  + **Function**: is\_signed\_in()
  + **Pre-condition**: Requires a session object that tracks whether the user is signed in.
  + **Post-condition**: Ensures that the method returns True if the user is signed in, False otherwise.
  + **Collaborations**:
    - This function collaborates with session management and checks user authentication in the UserManager(2) class.
* **Registers new users and stores user data in the database**
  + **Function**: add\_user()
  + **Pre-condition**: Requires that valid email and password inputs are submitted by the user.
  + **Post-condition**: Ensures that the new user’s data is saved to the database, provided the email doesn’t already exist.
  + **Collaborations**:
    - This function relies on DatabaseManager's(1) execute\_commit() to store user registration data.
* **Validates user login credentials**
  + **Function**: login()
  + **Pre-condition**: Requires a valid email and password entered by the user.
  + **Post-condition**: Ensures that the credentials are validated and that the user is signed in if the credentials are correct.
  + **Collaborations**:
    - This function interacts with DatabaseManager(1) through execute\_query() to check user login credentials.

## Class Description (CRC cards) QuizRetrievalManager

|  |  |
| --- | --- |
| **Class Name:** QuizRetrievalManager | |
| **Description:** The QuizRetrievalManager is responsible for fetching quiz data from the database and managing how quizzes are structured for use in the frontend. | |
| **Super classes:** None | |
| **Contracts:**  **Contract 3:** ManageQuizRetrieval  **Description:** This contract handles retrieving quizzes and their related data (e.g., questions, options) from the database, ensuring that quizzes are properly structured and ready to be rendered.  - Retrieves quiz data, including titles, quesiotns, and optons, based on quiz IDs  - Builds and formats quiz detials for display to the user  **Private Responsibilities:**  - Build and execute queries to retreive quizzes  - Format quiz data for frontend presentation | **Collaborations:**  DatabaseManager(1) |
| **Comments:**  The QuizRetrievalManager class ensures that quiz-related data is retrieved and formatted properly, ready to be displayed to users in the frontend. | |

### Contract #3: ManageQuizRetrieval

* **Retrieves quiz data, including titles, questions, and options, based on quiz IDs**
  + **Function**: get\_quiz\_by\_id(quiz\_id: int), fetch\_quiz(quiz\_id: int)
  + **Pre-condition**:
    - get\_quiz\_by\_id(): Requires a valid quiz ID and an active database connection.
    - fetch\_quiz(): Requires a valid quiz ID and an active database connection.
  + **Post-condition**:
    - get\_quiz\_by\_id(): Ensures that the quiz data, including questions and options, are returned.
    - fetch\_quiz(): Ensures that the raw quiz data is retrieved from the database.
  + **Collaborations**:
    - This function collaborates with DatabaseManager(1) to retrieve quiz data and with QuizManager to process quiz display.
* **Builds and formats quiz details for display to the user**
  + **Function**: build\_questions(quiz: tuple)
  + **Pre-condition**: Requires valid quiz data retrieved from the database.
  + **Post-condition**: Ensures that the questions and their corresponding options are formatted into a user-readable structure.
  + **Collaborations**:
    - This function collaborates with QuizManager(2) to render quiz details on the user interface.

## Subsystem Descriptions (subsystem cards)

|  |
| --- |
| **Subsystem Name:** Quiz Management Subsystem |
| **Classes:** QuizManager, QuizSubmissionManager, QuizCreationManager |
| **Collaboration Graph:** |
| **Description:** The Quiz Management Subsystem is a vital part of the MuscleMind system that is responsible for handling all operations related to quiz creation, submission, and management. It ensures that users can create custom quizzes, take quizzes, submit answers, and receive scores. The subsystem is designed to manage the flow of quiz-related data between the user interface and the database, ensuring a seamless experience for users. This subsystem consists of three main classes, each with distinct roles and responsibilities, and associated contract numbers that define their specific functions and interactions with other parts of the system. |
| **Contracts:**   1. **ManageQuizControl (QuizManager class)**:   This contract is responsible for overseeing the entire quiz lifecycle, including the creation, retrieval, and submission of quizzes. The QuizManager coordinates between the QuizCreationManager, QuizRetrievalManager, and QuizSubmissionManager to ensure that all quiz-related activities are managed efficiently and seamlessly for the user.   1. **ManageQuizSubmission (QuizSubmissionManager class)**:   This contract handles all aspects of quiz submissions, including validating answers and calculating scores. It ensures that user responses are checked against correct answers and that the final scores are calculated accurately. Additionally, it applies penalties, such as exercise tasks, for incorrect answers during the quiz. The QuizSubmissionManager is responsible for managing these interactions and ensuring the results are stored properly.   1. **ManageQuizCreation (QuizCreationManager class)**:   This contract oversees the process of creating and validating quizzes. It ensures that users can input quiz questions, multiple-choice options, and other related data. The QuizCreationManager ensures that the quiz creation form is properly structured, that quizzes meet system requirements, and that they are stored correctly in the database. |

## Class Description (CRC cards) QuizManager

|  |  |
| --- | --- |
| **Class Name:** QuizManager | |
| **Description:** The QuizManager class serves as the main controller for managing quizzes within the system. It coordinates the creation, retrieval, and submission of quizzes, ensuring that the interactions between the user and the quiz-related processes are smooth and efficient. | |
| **Super classes:** None | |
| **Contracts:**  **Contract 4:** ManageQuizControl  **Description:** This contract defines the core responsibilities for managing the quiz lifecycle, from creation to submission and scoring.  - Handles quiz creation and storage  - Manages the submission and validatino of quiz answers  - Coordinates the retrieval of quiz data for display to users  **Private Responsibilities:**  - Ensure proper quiz creation and submission  - Validate quiz answers and provide scores. | **Collaborations:**  QuizSubmissionManager(5)  QuizCreationManager(6) |
| **Comments:**  The QuizManager acts as a key coordinator for all quiz-related activities. This class not only manages the lifecycles of quizzes but also ensures the data flows correctly between different subsystems and classes, such as retrieving quiz data from the database or managing user submissions. | |

### Contract #4: ManageQuizControl

**Description**: This contract defines the responsibilities for managing the overall quiz lifecycle, from quiz creation to submission and scoring. It serves as a controller by delegating tasks to specific managers and ensuring the flow of the quiz-related processes.

* **Handles quiz creation and storage**
  + **Function**: create\_quiz()
  + **Pre-condition**: Requires QuizCreationManager to be initialized and active.
  + **Post-condition**: Ensures that the quiz creation process is delegated to QuizCreationManager, and that the quiz is created and stored properly.
  + **Collaborations**: This function collaborates with QuizCreationManager to handle the input and storage of new quizzes.
* **Manages the submission and validation of quiz answers**
  + **Function**: submit\_quiz\_answer(quiz\_id: int, question\_num: int)
  + **Pre-condition**: Requires a valid quiz\_id and question\_num, along with an active QuizSubmissionManager.
  + **Post-condition**: Ensures that the answer is submitted for validation, and the quiz process continues to the next question or the final score.
  + **Collaborations**: This function collaborates with QuizSubmissionManager to handle answer submissions and validations.
* **Coordinates the retrieval of quiz data for display to users**
  + **Function**: get\_user\_quizzes(user\_email: str), quiz\_detail(quiz\_id: int)
  + **Pre-condition**: Requires a valid user\_email or quiz\_id, and the QuizRetrievalManager must be initialized.
  + **Post-condition**: Ensures that the correct quiz data is retrieved and presented to the user, either in a list format or detailed view.
  + **Collaborations**: These functions collaborate with QuizRetrievalManager to retrieve quizzes and display quiz details to the user.

## Class Description (CRC cards) QuizSubmissionManager

|  |  |
| --- | --- |
| **Class Name:** QuizSubmissionManager | |
| **Description:** The QuizSubmissionManager handles the process of submitting answers during a quiz and providing feedback based on the user's performance. It manages answer validation, score calculation, and the application of exercise penalties when necessary. | |
| **Super classes:** None | |
| **Contracts:**  **Contract 5:** ManageQuizSubmission  **Description:** This contract defines the process of managing the submission of quizzes, validating answers, and calculating scores.  - Validates user responses against correct answers  - Calculates and stores quiz scores  - Manages exercise penalties for incorrect answers.  **Private Responsibilities:**  - Validate answers and calculate scores  - Handle the application of exercise penalties for wrong answers | **Collaborations:**  QuizManager(4) |
| **Comments:**  The QuizSubmissionManager ensures that the quiz submission process is handled efficiently and that results are processed accurately. It’s also responsible for making the MuscleMind system interactive by incorporating exercise penalties for wrong answers, adding a unique dimension to the quiz-taking experience. | |

### Contract #5: ManageQuizSubmission

**Description**: This contract handles the submission of quizzes, validation of answers, and the calculation of scores. It also manages penalties for incorrect answers, ensuring that users are scored correctly and penalized when applicable.

* **Validates user responses against correct answers**
  + **Function**: submit\_quiz\_answer(quiz\_id: int, question\_num: int)
  + **Pre-condition**: Requires a valid quiz\_id and question\_num, as well as an answer from the user.
  + **Post-condition**: Ensures that the answer is validated and the quiz proceeds to the next question or displays the final score.
  + **Collaborations**: This function collaborates with QuizRetrievalManager to verify correct answers and with QuizManager to continue quiz flow.
* **Calculates and stores quiz scores**
  + **Function**: score(quiz\_id: int, score: int, total: int)
  + **Pre-condition**: Requires a valid quiz\_id, a final score, and the total number of quiz questions.
  + **Post-condition**: Ensures that the final quiz score is calculated and displayed to the user, and that the score is stored.
  + **Collaborations**: This function collaborates with DatabaseManager for storing quiz scores and with QuizManager for displaying the results.
* **Manages exercise penalties for incorrect answers**
  + **Function**: penalty(quiz\_id: int, question\_num: int)
  + **Pre-condition**: Requires a valid quiz\_id and question\_num, and an incorrect answer from the user.
  + **Post-condition**: Ensures that the penalty is displayed to the user, prompting them to complete an exercise challenge.
  + **Collaborations**: This function collaborates with QuizRetrievalManager to determine which question was answered incorrectly and applies the appropriate penalty.

## Class Description (CRC cards) QuizCreationManager

|  |  |
| --- | --- |
| **Class Name:** QuizCreationManager | |
| **Description:** The QuizCreationManager class is responsible for allowing users to create quizzes by inputting questions, multiple-choice options, and other quiz-related details. It ensures that the input data adheres to the correct format and that the quiz is stored in the database. | |
| **Super classes:** None | |
| **Contracts:**  **Contract 6:** ManageQuizCreation  **Description:** This contract defines the process of creating and validating quizzes, ensuring that they are formatted properly and stored in the database.  - Collects user input for quiz creation  - Validates quiz details and ensures proper storage in the database  - Manages the form for quiz creation and ensures it meets system requirements    **Private Responsibilities:**  - Handle user input for quiz creation  - Validate the format and structure of the quiz data | **Collaborations:**  QuizManager(4) |
| **Comments:**  The QuizCreationManager simplifies the process of quiz creation by guiding the user through the necessary steps and ensuring that quizzes adhere to the system’s specifications. Its primary focus is ensuring that data is entered correctly and efficiently stored for future use. | |

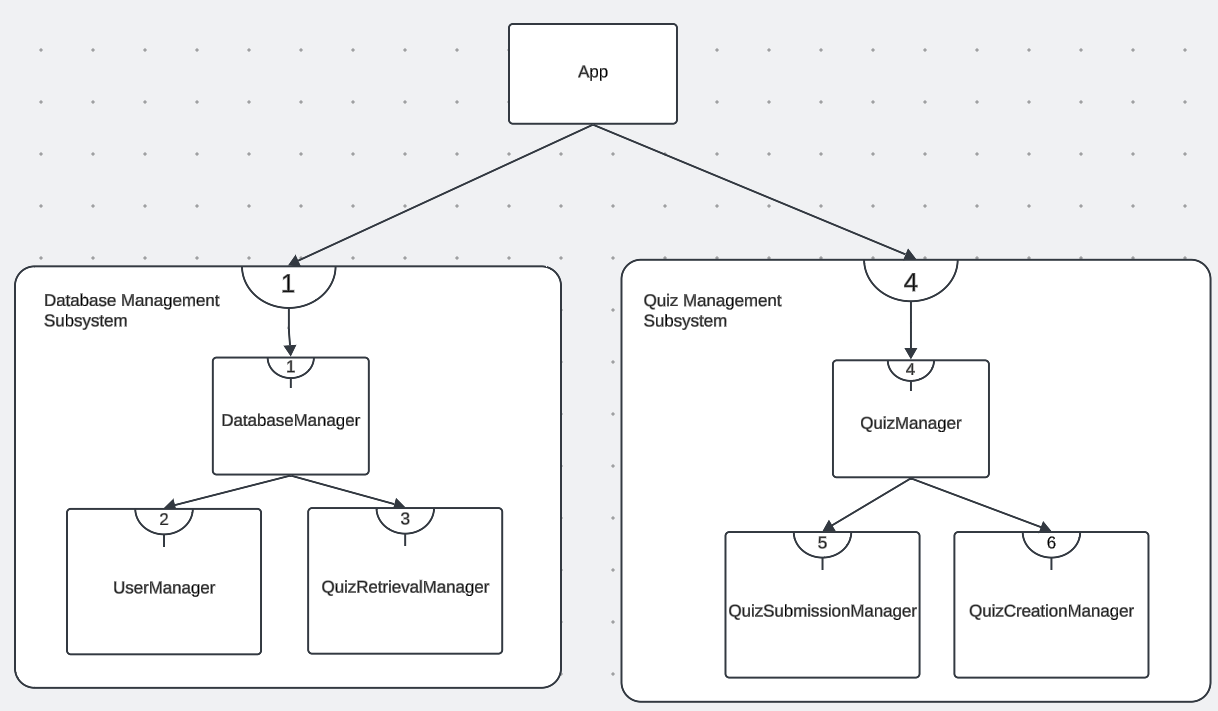
### Contract #6: ManageQuizCreation

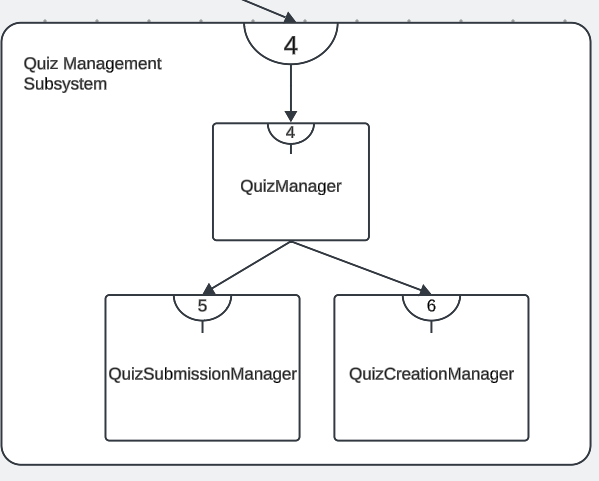
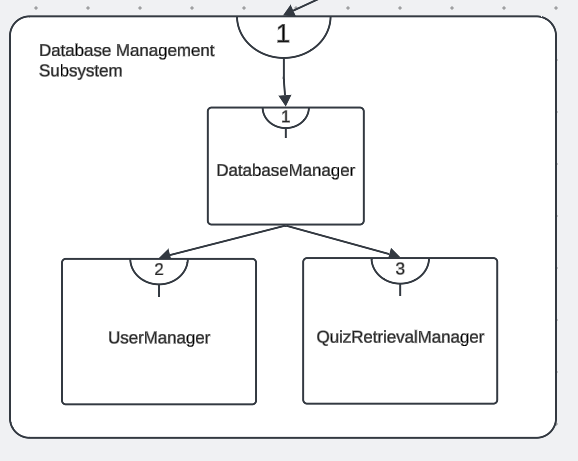
**Description**: This contract manages the creation of quizzes by collecting user input, validating the data, and storing it in the database. It ensures that the quiz creation process adheres to system requirements and that quizzes are stored securely.

* **Collects user input for quiz creation**
  + **Function**: create\_quiz()
  + **Pre-condition**: Requires valid user input for the quiz, including the title and number of questions.
  + **Post-condition**: Ensures that the quiz creation form is rendered or submitted for processing.
  + **Collaborations**: This function collaborates with QuizManager to handle the creation of quizzes and with DatabaseManager for storing the data.
* **Validates quiz details and ensures proper storage in the database**
  + **Function**: submit\_quiz()
  + **Pre-condition**: Requires valid quiz data, including questions, options, and answers, from the form submission.
  + **Post-condition**: Ensures that the quiz is properly formatted and stored in the database.
  + **Collaborations**: This function collaborates with DatabaseManager to store quiz data securely and with QuizManager to complete the creation process.
* **Manages the form for quiz creation and ensures it meets system requirements**
  + **Function**: render\_quiz\_creation\_form(num\_questions: int, title: str)
  + **Pre-condition**: Requires valid user input for the number of questions and quiz title.
  + **Post-condition**: Ensures that the form is dynamically generated based on the user's input and displayed for further quiz creation steps.
  + **Collaborations**: This function collaborates with QuizManager to facilitate the quiz creation process and ensure the user is able to create a valid quiz.

1. Appendix

High Level Overview of Muscle Mind System Collaboration Graph:





Subsystem #1: Database Management Subsystem (left)

Subsystem #2: Quiz Management Subsystem (right)

