

Virtual Classroom Manager Documentation

Table of Contents

Introduction

- Purpose
- Scope
- Architecture Overview

Setup Instructions

- Prerequisites
- Installation
- Configuration

Usage Guide

- Command-Line Interface (CLI)
- Example Usage

Class Diagram

Database Schema

- Entity-Relationship Diagram (ERD)

Database Structure

- Database
- Tables

Functionality

- Classroom Management
- Student Management
- Assignment Management

Code Quality and Design

- Design Patterns
- SOLID Principles
- Exception Handling
- Transient Error Handling

Testing

- Unit Testing
- Integration Testing

Introduction

➤ Purpose

The Virtual Classroom Manager is a terminal-based application designed to manage virtual classrooms, student enrolment, and assignment scheduling and submission for an EdTech platform.

➤ Scope

This documentation provides an overview of the project's architecture, setup instructions, usage guide, code structure, functionality, and other relevant information.

➤ Architecture Overview

The application follows the **Model-View-Controller (MVC)** architecture, where MySQL databases act as the model. Key components include:

- **Database: *MySQL database*** named "Virtual_Classrooms" with tables for STUDENTS (studentId, studentName, classroom_name), CLASSROOMS (className), ASSIGNMENTS (classroom_name, AssignmentDetails), SUBMITTED_ASSIGNMENTS (studentId, classroom_name, AssignmentDetails).
- **Command-Line Interface (CLI):** The terminal-based user interface for interacting with the application.
- **Java Codebase Connectivity:** The *JDBC* responsible for handling classroom, student, and assignment operations.

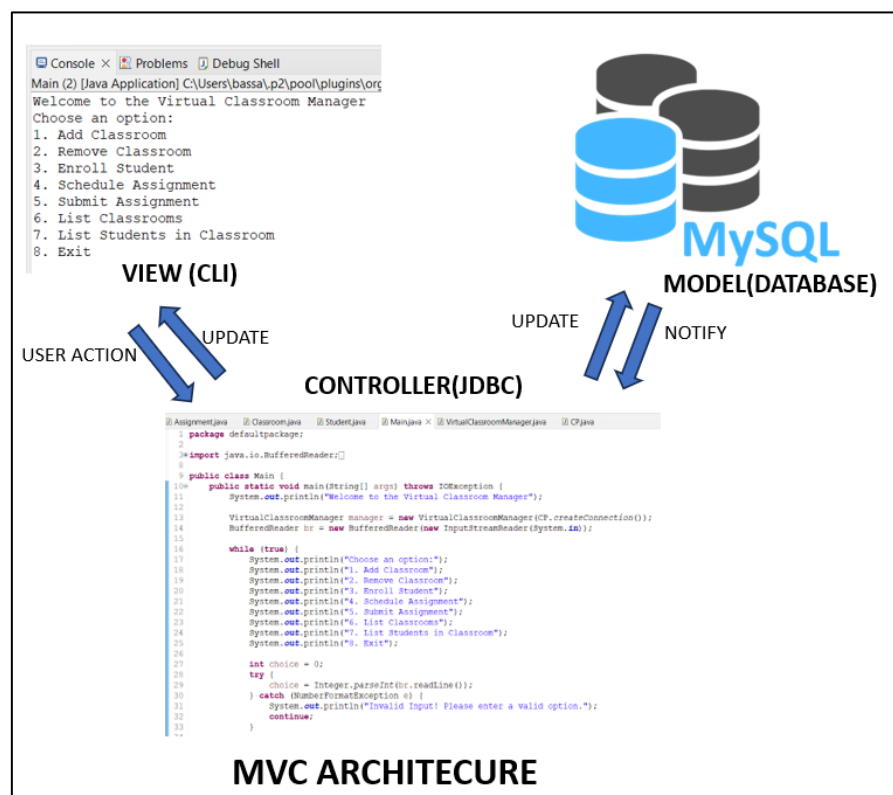


Fig1: Model-View-Controller (MVC) architecture

Setup Instructions

➤ **Prerequisites:**

Before running the Virtual Classroom Manager, make sure you have the following prerequisites installed on your system:

1. Java Runtime Environment (JRE): You can download and install the latest JRE from the official Oracle website: [Download JRE] (<https://www.oracle.com/java/technologies/javase-downloads.html>)

2. MySQL Database Server: Install MySQL Database Server if it's not already installed. You can download it from the official MySQL website: [MySQL Download] (<https://dev.mysql.com/downloads/mysql/>)

➤ **Installation Steps:**

Follow these steps to set up and run the Virtual Classroom Manager:

Step 1: Clone the Project Repository

Clone the project repository from the provided GitHub URL. You can use Git to clone the repository

Step 2: Create the Necessary Database and Tables

Execute the provided SQL script to create the required database and tables. You can use a MySQL client, such as MySQL Workbench or the command-line client, to run the script.

Step 3: Configure the Database Connection in the Application

In your Java application code, configure the database connection. You'll need the MySQL JDBC driver for this. Follow these steps:

- **Download the MySQL JDBC Driver:** Download the MySQL Connector/J JDBC driver from the MySQL website: [MySQL Connector/J Download] (<https://dev.mysql.com/downloads/connector/j/>)
- **Include the JDBC Driver:** Add the downloaded JDBC driver (a JAR file) to your project's class path. You can do this using your preferred Java development environment (IDE), build tool (e.g., Maven or Gradle), or by manually placing the JAR file in your project's library folder.

Step 4: Compile and Run the Java Application

Compile your Java application. You can typically do this from your IDE.

With these steps completed, your Virtual Classroom Manager should be ready to use. It will connect to the MySQL database and allow you to manage virtual classrooms, students, and assignments through the terminal-based interface.

➤ **Configuration**

Configure the database connection details in the application configuration file (config.properties) or through environment variables.

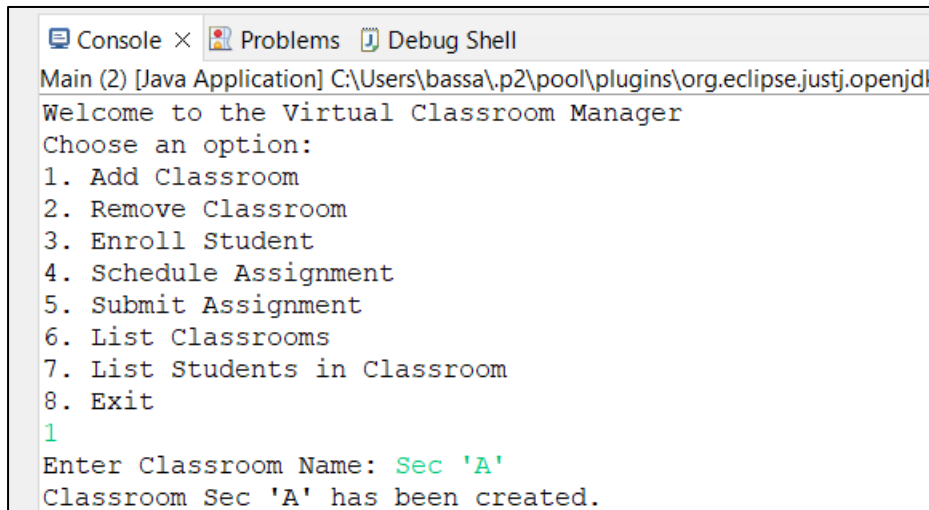
Usage Guide

Command-Line Interface (CLI)

The Virtual Classroom Manager can be used through a command-line interface (CLI). Users can perform various operations using commands like add classroom, remove classroom, add student, schedule assignment, and submit assignment, list students in each classroom, list the classroom.

Example Usage

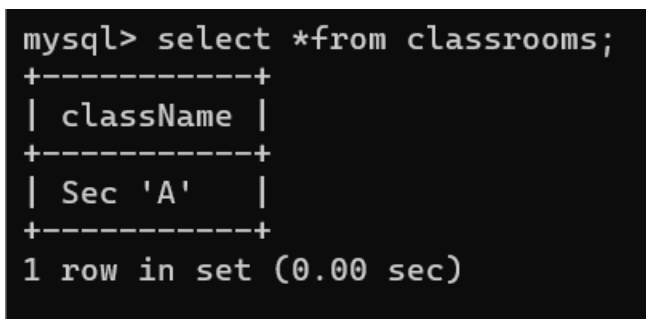
Add classroom:



```

Main (2) [Java Application] C:\Users\bassa\p2\pool\plugins\org.eclipse.justj.openjdk
Welcome to the Virtual Classroom Manager
Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
1
Enter Classroom Name: Sec 'A'
Classroom Sec 'A' has been created.

```



```

mysql> select *from classrooms;
+-----+
| className |
+-----+
| Sec 'A'   |
+-----+
1 row in set (0.00 sec)

```

Fig 2: Command-Line Interface and table classroom for Add classroom

Remove Classroom:

```

Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
2
Enter Classroom Name to Remove: Sec 'B'
Classroom Sec 'B' has been removed.

```

```
mysql> select *from classrooms;
+-----+
| className |
+-----+
| Sec 'A' |
| Sec 'B' |
+-----+
2 rows in set (0.00 sec)
```

Before

```
mysql> select *from classrooms;
+-----+
| className |
+-----+
| Sec 'A' |
+-----+
1 row in set (0.00 sec)
```

After

Fig 3: Command-Line Interface and table classroom for Remove classroom

Enrol Student:

```
Console x Problems Debug Shell
Main (2) [Java Application] C:\Users\bassa\p2\pool\plugins\org.eclipse.justj.openjdk.f
Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
3
Enter Student ID: 1
Enter Student Name: PREETAMGOUDA
Enter Classroom Name: Sec 'A'
Student PREETAMGOUDA has been enrolled in Sec 'A'.
```

```
mysql> select *from students;
+-----+-----+-----+
| studentId | studentName | classroom_name |
+-----+-----+-----+
| 1 | PREETAMGOUDA | Sec 'A' |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Fig 4: Command-Line Interface and table students for Enrol Student

Schedule Assignment:

```
Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
4
Enter Classroom Name: Sec 'C'
Enter Assignment Details: write a full essay on the importance of education
Assignment for Sec 'C' has been scheduled.
```

```
mysql> select *from assignments;
+-----+-----+
| assignmentdetails | classroom_name |
+-----+-----+
| write a full essay on the importance of education | Sec 'C' |
+-----+-----+
1 row in set (0.00 sec)
```

Fig 5: Command-Line Interface and table assignments for Schedule Assignment

Submit Assignment:

```
Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
5
Enter Student ID: 3
Enter Classroom Name: Sec 'C'
Enter Assignment Details: write a full essay on the importance of education
Assignment submitted by Student 3 in Sec 'C'.

mysql> select *from submitted_assignments;
+-----+-----+-----+
| student_id | classroom_name | assignment_details |
+-----+-----+-----+
| 3 | Sec 'C' | write a full essay on the importance of education |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Fig 6: Command-Line Interface and table submit_assignment for Submit Assignment

List Classrooms:

```
Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
6
List of Classrooms:
List of Classrooms:
Sec 'A'
Sec 'B'
Sec 'C'
```

Fig 7: Command-Line Interface List Classrooms

List Students in Classroom:

```

Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
7
Enter Classroom Name: Sec 'A'
List of Students in Sec 'A':
PREETAMGOUDA

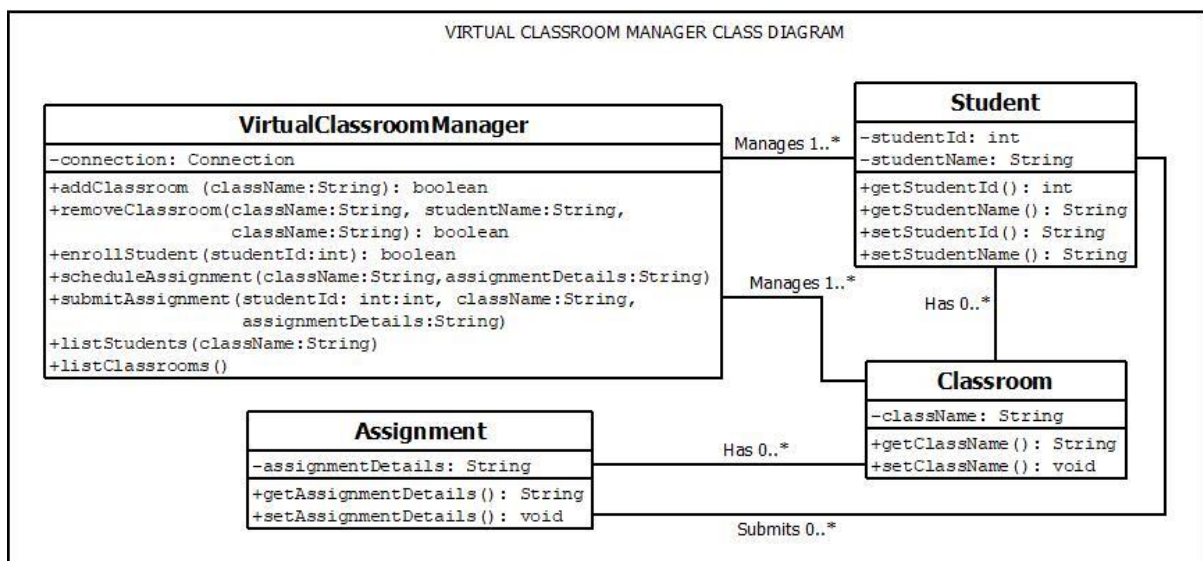
```

Fig 8: Command-Line Interface List Students in Classroom**Exit:**

```

Choose an option:
1. Add Classroom
2. Remove Classroom
3. Enroll Student
4. Schedule Assignment
5. Submit Assignment
6. List Classrooms
7. List Students in Classroom
8. Exit
8
Exiting the Virtual Classroom Manager.

```

Fig 9: Command-Line Interface exiting the virtual classroom manager**Class Diagram****➤ Class Diagram****Fig10: Class diagram for virtual classroom manager**

Database Schema

➤ Entity-Relationship Diagram (ERD)

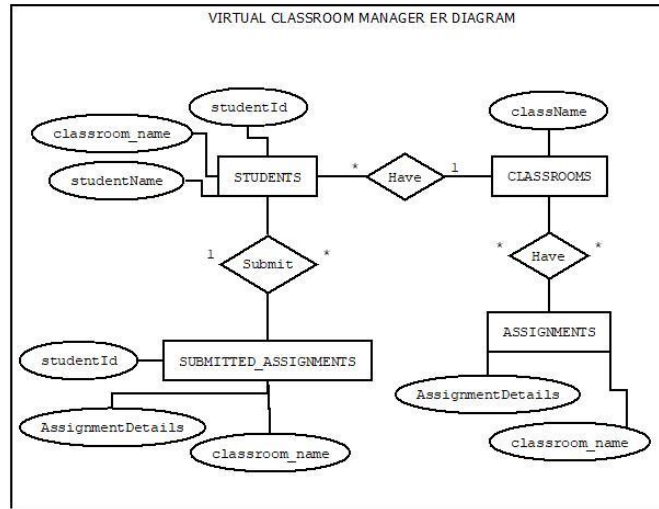


Fig11: ER diagram for virtual classroom manager

Database Structure

- Database

Creating database named as Virtual_Classroom.

```
mysql> create database Virtual_Classroom;
Query OK, 1 row affected (0.06 sec)

mysql> USE Virtual_Classroom;
Database changed
```

Fig12: creating database and using it for further operations.

- Tables

Creating tables named as classrooms students, assignments, submitted_assignments.

```
mysql> CREATE TABLE classrooms (
    ->     className VARCHAR(255) PRIMARY KEY
    -> );
Query OK, 0 rows affected (0.09 sec)
```

Fig13: creating table classrooms

```
mysql> CREATE TABLE students (
    -> studentId INT PRIMARY KEY,
    -> studentName VARCHAR(255) NOT NULL,
    -> classroom_name VARCHAR(255) NOT NULL,
    -> FOREIGN KEY (classroom_name) REFERENCES classrooms(className)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Fig14: creating table students


```
mysql> CREATE TABLE submitted_assignments (
  -> student_Id INT NOT NULL,
  -> classroom_name VARCHAR(255) NOT NULL,
  -> assignment_details TEXT,
  -> FOREIGN KEY (student_Id) REFERENCES students(studentId),
  -> FOREIGN KEY (classroom_name) REFERENCES classrooms(ClassName),
  -> PRIMARY KEY (student_id, classroom_name)
  -> );
Query OK, 0 rows affected (0.05 sec)
```

Fig15: creating table assignments

```
mysql> CREATE TABLE assignments (
  -> assignmentDetails TEXT,
  -> classroom_name VARCHAR(255) NOT NULL,
  -> FOREIGN KEY (classroom_name) REFERENCES classrooms(ClassName)
  -> );
Query OK, 0 rows affected (0.05 sec)
```

Fig16: creating table submitted_assignments

```
mysql> desc classrooms;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| className | varchar(255) | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> desc students;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| studentId     | int           | NO   | PRI | NULL    |       |
| studentName   | varchar(255) | NO   |     | NULL    |       |
| classroom_name | varchar(255) | NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc assignments;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| assignmentDetails | text         | YES  |     | NULL    |       |
| classroom_name   | varchar(255) | NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> desc submitted_assignments;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| student_Id     | int           | NO   | PRI | NULL    |       |
| classroom_name | varchar(255) | NO   | PRI | NULL    |       |
| assignment_details | text         | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Fig17: initial tables before doing any update operations.

Functionality

Explanation of the logic for all the functional requirements in the Virtual Classroom Manager:

➤ Classroom Management:

a. Add Classroom:

Logic: This functionality allows adding a new classroom to the system.

Explanation:

- The addClassroom method constructs an SQL query to insert the provided className into the "classrooms" table.
- It sets the className parameter in the prepared statement and executes the query.
- If successful, it prints a message indicating the classroom creation and returns true.
- If an error occurs during database operations, it catches SQL Exception, prints the error, and returns false.

```
public boolean addClassroom(String className) throws ClassroomCreationException {
    try {
        String sql = "INSERT INTO classrooms (className) VALUES (?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, className);
            stmt.executeUpdate();
            return true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new ClassroomCreationException("Failed to create classroom: " + e.getMessage());
    }
}
```

Fig18: Code Snippet for the Add Classroom method

b. Remove Classroom:

Logic: This functionality allows removing an existing classroom from the system.

Explanation:

- The removeClassroom method constructs an SQL query to delete the specified classroom from the "classrooms" table.
- It checks the number of affected rows after executing the query. If greater than 0, it prints a success message; otherwise, it indicates that the classroom was not found.
- It handles database errors with SQL Exception.

```
public class ClassroomRemovalException extends Exception {
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public ClassroomRemovalException(String message) {
        super(message);
    }
}
```

Fig19: Code Snippet for the Remove Classroom method

c. List Classrooms:

Logic: This functionality lists all classrooms in the system.

Explanation:

- The listClassrooms method constructs an SQL query to retrieve all classroom names from the "classrooms" table.
- It iterates through the result set, printing each classroom name to the console.
- It handles database errors with SQL Exception.

```
public List<String> listClassrooms() {
    List<String> classrooms = new ArrayList<>();
    try {
        String sql = "SELECT className FROM classrooms";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            try (ResultSet resultSet = stmt.executeQuery()) {
                System.out.println("List of Classrooms:");
                while (resultSet.next()) {
                    String className = resultSet.getString("className");
                    classrooms.add(className);
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return classrooms;
}
```

Fig20: Code Snippet for the List Classroom method

➤ **Student Management:**

a. Enrol Student:

Logic: This functionality allows enrolling a student into a classroom.

Explanation:

- The enrollStudent method constructs an SQL query to insert the student's information (ID, name, classroom name) into the "students" table.
- It sets the parameters in the prepared statement and executes the query.
- If successful, it prints a message indicating the student's enrolment and returns true.
- It handles database errors with SQL Exception.

```
public boolean enrollStudent(int studentId, String studentName, String className) throws StudentEnrollmentException {
    try {
        String sql = "INSERT INTO students (studentId, studentName, classroom_name) VALUES (?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, studentId);
            stmt.setString(2, studentName);
            stmt.setString(3, className);
            stmt.executeUpdate();
            return true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new StudentEnrollmentException("Failed to enroll student: " + e.getMessage());
    }
}
```

Fig21: Code Snippet for the Enrol Student method

b. List Students:

Logic: This functionality lists all students in a specified classroom.

Explanation:

- The listStudents method constructs an SQL query to retrieve student names in the specified classroom from the "students" table.
- It iterates through the result set, printing each student's name to the console.
- It handles database errors with SQL Exception.

```
public List<String> listStudents(String className) {
    List<String> students = new ArrayList<>();
    try {
        String sql = "SELECT studentName FROM students WHERE classroom_name = ?";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, className);
            try (ResultSet resultSet = stmt.executeQuery()) {
                while (resultSet.next()) {
                    String studentName = resultSet.getString("studentName");
                    students.add(studentName);
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return students;
}
```

Fig22: Code Snippet for the List Student method

➤ Assignment Management:

a. Schedule Assignment:

Logic: This functionality allows scheduling an assignment for a classroom.

Explanation:

- The scheduleAssignment method constructs an SQL query to insert the assignment details (classroom name and assignment details) into the "assignments" table.
- It sets the parameters in the prepared statement and executes the query.
- It prints a message indicating the assignment scheduling.
- It handles database errors with SQL Exception.

```
public boolean scheduleAssignment(String className, String assignmentDetails) throws AssignmentSchedulingException {
    try {
        String sql = "INSERT INTO assignments (classroom_name, assignmentDetails) VALUES (?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setString(1, className);
            stmt.setString(2, assignmentDetails);
            stmt.executeUpdate();
            System.out.println("Assignment for " + className + " has been scheduled.");
            return true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new AssignmentSchedulingException("Failed to schedule assignment: " + e.getMessage());
    }
}
```

Fig23: Code Snippet for the Schedule Assignment method

b. Submit Assignment:

Logic: This functionality allows a student to submit an assignment for a classroom.

Explanation:

- The submitAssignment method constructs an SQL query to insert the submitted assignment details (student ID, classroom name, and assignment details) into the "submitted_assignments" table.
- It sets the parameters in the prepared statement and executes the query.
- It prints a message indicating the assignment submission.
- It handles database errors with SQL Exception.

```
public boolean submitAssignment(int studentId, String className, String assignmentDetails) throws AssignmentSubmissionException {
    try {
        String sql = "INSERT INTO submitted_assignments (student_id, classroom_name, assignment_details) VALUES (?, ?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(sql)) {
            stmt.setInt(1, studentId);
            stmt.setString(2, className);
            stmt.setString(3, assignmentDetails);
            stmt.executeUpdate();
            System.out.println("Assignment submitted by Student " + studentId + " in " + className + ".");
            return true;
        }
    } catch (SQLException e) {
        e.printStackTrace();
        throw new AssignmentSubmissionException("Failed to submit assignment: " + e.getMessage());
    }
}
```

Fig24: Code Snippet for the Submit Assignment method

Code Quality and Design

Code quality and design aspects implemented in the Virtual Classroom Manager codebase:

Design Patterns:

Singleton Pattern for Database Connection (CP class): The CP (Connection Pool) class is designed as a Singleton, ensuring that only one instance of the database connection is created and shared across the application. This promotes efficient resource management.

DAO (Data Access Object) Pattern: The codebase follows a DAO pattern, where database-related operations are encapsulated in separate methods within the VirtualClassroomManager class. This separation of concerns enhances code modularity and maintainability.

SOLID Principles:

Single Responsibility Principle (SRP): Each method in the VirtualClassroomManager class has a single responsibility related to classroom management, student management, or assignment management. This promotes code clarity and separation of concerns.

Open/Closed Principle (OCP): The codebase is designed to be open for extension (e.g., adding new functionality) but closed for modification of existing code. New features can be added by creating new methods or classes without altering existing ones.

Liskov Substitution Principle (LSP): The codebase doesn't explicitly involve inheritance or subclassing, so the LSP is not a prominent factor. However, it adheres to the principles of composition and interface-based design.

Interface Segregation Principle (ISP): The codebase doesn't have explicit interfaces, but it follows a clean separation of methods within classes, ensuring that each class has a specific set of methods related to its responsibilities.

Dependency Inversion Principle (DIP): The codebase follows the DIP indirectly through the use of dependency injection. The VirtualClassroomManager class depends on the database connection provided via the constructor, allowing for flexibility and testability.

Exception Handling:

Exception Handling in Database Operations: In methods that involve database operations (e.g., addClassroom, removeClassroom), exceptions related to SQL queries and database connections are caught and logged. This prevents unexpected errors from crashing the application and provides feedback to users.

Logging:

Console-Based Logging: The codebase uses `System.out.println` statements to provide feedback to users about the success or failure of various operations. While this is rudimentary, it serves the purpose of basic logging for user interaction.

Exception Logging: Exceptions that occur during database operations are caught and printed using `e.printStackTrace()`. This helps developers diagnose issues during development.

Code Quality:

Readability and Maintainability: The code follows Java coding conventions and is well-structured, making it readable and maintainable.

Comments and Documentation: The codebase includes comments and documentation for class methods, explaining their purpose and usage.

Code Reusability: Methods are designed for reusability, allowing them to be easily called from the main application logic.

Separation of Concerns: Different methods in the `VirtualClassroomManager` class handle specific concerns related to classroom, student, and assignment management, promoting code organization.

Overall, the codebase demonstrates good practices in terms of design patterns, adherence to SOLID principles, basic exception handling, and rudimentary logging for user feedback. Further improvements could involve adopting a more sophisticated logging framework for comprehensive log management and implementing advanced exception handling strategies.

Testing

Testing Approach

The testing approach for the Virtual Classroom Manager application encompasses both unit testing and integration testing to ensure its reliability and functionality.

Unit Testing:

Unit testing focuses on testing individual components or units of code in isolation to verify that they perform as expected. In the context of this application, unit tests are created to validate the behaviour of specific methods and classes.

Test Data: Unit tests employ various test data scenarios to validate different execution paths, including valid and invalid inputs, boundary cases, and edge cases.

Integration Testing:

Integration testing verifies the **interaction** and **compatibility between different components or modules of the application**. It ensures that these components collaborate correctly when integrated into the system.

End-to-End Testing: End-to-end testing evaluates the entire system's functionality, from user inputs through the user interface to database interactions and back. It tests scenarios involving multiple components working together to accomplish specific tasks, such as adding a classroom, enrolling a student, and scheduling an assignment.