**Experiment 15**

**1. Problem Statement and Introduction**

Problem Statement:

Managing student marksheets manually in a school environment can be a time-consuming and error-prone process. It involves keeping track of each student's details and their respective marks, which can become increasingly difficult as the number of students increases. There is also a risk of data loss or errors in data entry. Therefore, there is a need for an automated system to manage student marksheets.

Introduction of the Java Project:

The "School Management System" is a Java-based application designed to address the above problem. It automates the process of managing student marksheets, thereby saving time and reducing the risk of errors.

The application provides a user-friendly interface for performing various operations related to marksheets. These operations include viewing available marksheets, selecting a specific marksheet, creating a new marksheet, and managing the selected marksheet (adding student details, displaying the marksheet, editing student details, and deleting student details).

The system is built using Java, ensuring platform independence and robustness. It interacts with a MySQL database for efficient data storage and retrieval, ensuring reliable performance and scalability.

In summary, the School Management System is a comprehensive solution that simplifies and enhances the process of managing marksheets in a school environment.

**2. System Architecture**

The system is built using Java and interacts with a MySQL database for data storage. The architecture can be divided into three main components:

- User Interface: This is where the user interacts with the system. It includes options to view, create, and manage marksheets.

- Business Logic: This is the core of the system, where the logic for managing marksheets is implemented. It includes functions for creating a new marksheet, displaying existing marksheets, and more.

- Database: This is where the data is stored. The system interacts with a MySQL database to store and retrieve marksheet data.

(Here, you can include a diagram showing these components and how they interact)

**3. Modules**

The system can be divided into the following modules:

- Database Connection: This module is responsible for establishing a connection with the MySQL database.

- Marksheet Management: This module includes functions for creating a new marksheet (`createNewClass()`), displaying existing marksheets (`showClasses()`), and managing a specific marksheet (`dashboard()`).

Each of these modules is implemented as a function in the `Main` class.

Sure, here's a more detailed and professional description of the modules:

1. Database Connection Module

This module is responsible for establishing and managing the connection to the MySQL database. It uses the `java.sql.Connection` class to create a connection object, which is then used throughout the application to interact with the database. The connection is established in the `connectDb()` function, which is called at the start of the `main()` function.

2. Marksheet Selection Module

This module provides the user with the ability to select an existing marksheet or create a new one. It uses a `Scanner` object to take user input and then processes this input to determine the next course of action. If the user chooses to work with an existing marksheet, they are asked to enter the name of the marksheet, and the `dashboard()` function is called with the connection object and the marksheet name as arguments. If the user chooses to create a new marksheet, the `createNewClass()` function is called with the connection object as an argument.

3. Marksheet Management Module

This module includes functions for creating a new marksheet (`createNewClass()`), displaying existing marksheets (`showClasses()`), and managing a specific marksheet (`dashboard()`). The `createNewClass()` function prompts the user for the necessary information to create a new marksheet and then inserts this information into the database. The `showClasses()` function retrieves the list of existing marksheets from the database and displays them to the user. The `dashboard()` function provides a user interface for managing a specific marksheet, including options to add student details, display the marksheet, edit student details, and delete student details.

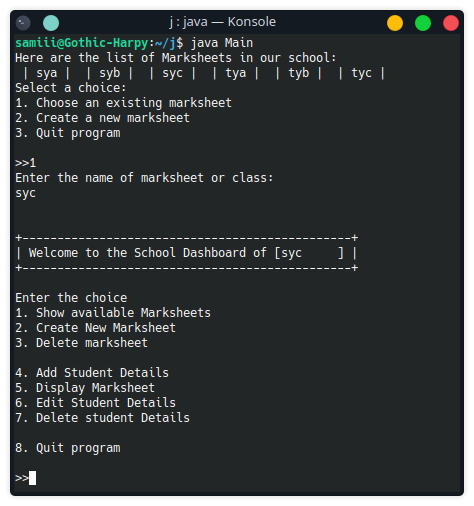
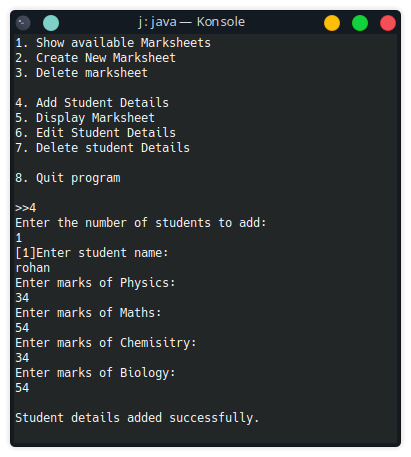
4. User Interface Module

This module is responsible for all user interactions. It uses the `System.out.println()` function to display information to the user and the `Scanner` class to take user input. It provides a clear and intuitive interface for the user to interact with the system, including a main menu with options to choose an existing marksheet, create a new marksheet, or quit the program, and a dashboard with options to manage a specific marksheet.

Each of these modules plays a crucial role in the functionality of the system and has been implemented with careful consideration of best practices for clarity, efficiency, and maintainability.

**4. Screenshots**

Initial Dashboard. Inserting new record.

Displaying all marksheet data.

