Efficient task management using Java Swing and MySQL database

Mini Project Report
submitted by

Abhishek Johns (VML22CS009)
Adithya KP VML22CS015)
Adithya U (VML22CS017)
Aghosh Vijayan (VML22CS020)
Aida Rose Benny (VML22CS021)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERINGVIMAL JYOTHI ENGINEERING COLLEGE
CHEMPERI CHEMPERI P.O. - 670632, KANNUR, KERALA, INDIA
May 2024

Content

PAGENO.

Chapter 1

Introduction

Task management is a crucial aspect of productivity in both personal and professional spheres. The Task Manager application serves as a comprehensive tool tailored to streamline task management processes for users. Offering an array of functionalities, including task addition with due dates, task completion tracking, deletion capabilities, and real-time task list updates, this application is designed to enhance efficiency and organization. Leveraging Java Swing for its intuitive graphical user interface (GUI) and seamlessly integrating with a MySQL database, the Task Manager ensures reliable storage and retrieval of task information. This report explores the features, architecture, and usability of the Task Manager application, shedding light on its role in facilitating effective task management.

Objective

- Streamline Task Management: Develop a user-friendly graphical interface using Java Swing to facilitate efficient task organization and management for users.
- Seamless Task Manipulation: Enable users to add tasks with due dates, mark tasks as completed, delete tasks, and refresh the task list effortlessly through intuitive GUI interactions.
- Real-time Data Interaction: Establish connectivity between the Java Swing GUI and a MySQL database to ensure seamless storage, retrieval, and manipulation of task information.
- Secure Data Storage: Implement robust data handling mechanisms to securely store task-related data in the MySQL database, safeguarding against unauthorized access or data loss.

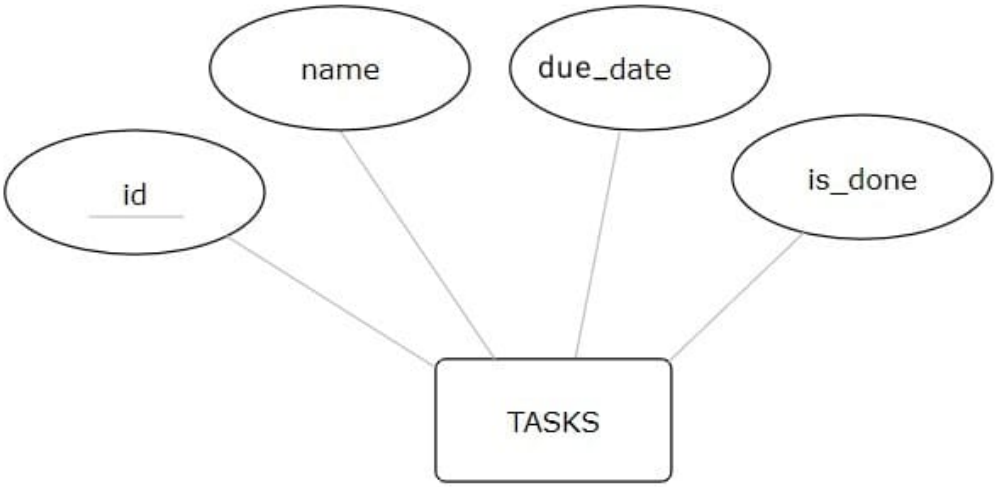
Technologies Used

- Java SWING
- MySQL

Chapter 2

Design Phase

Entity-Relationship Diagram (ERD)



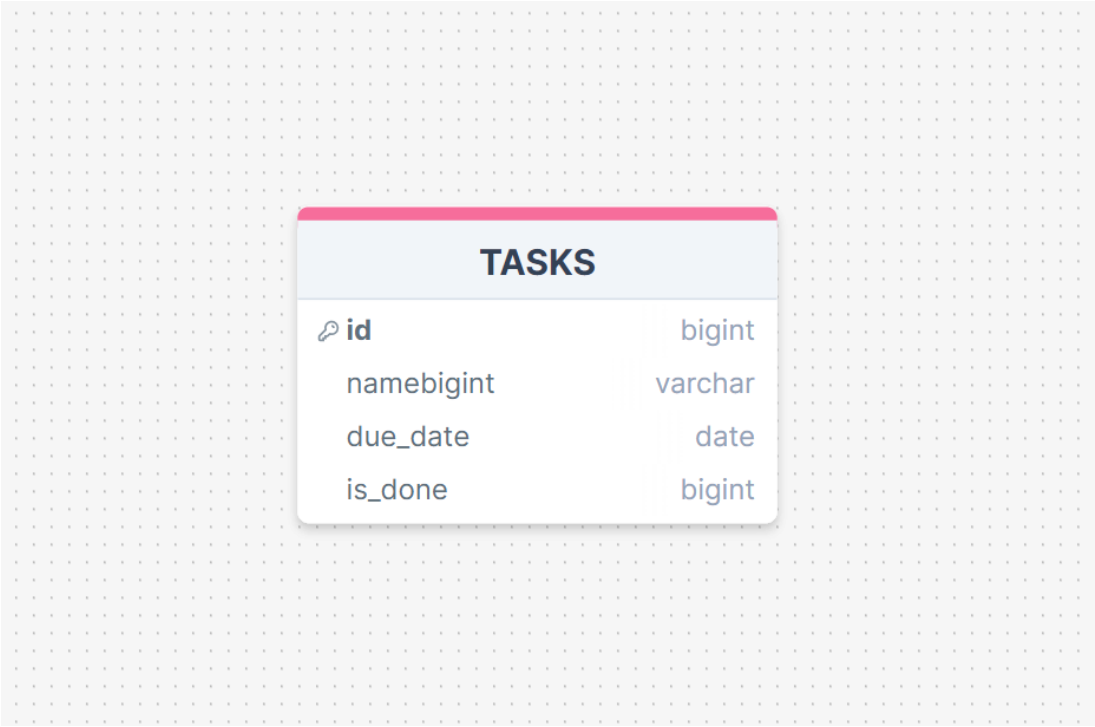
Entities and Attributes

1. **TASKS**: This is the main entity shown in the diagram. It represents a collection of tasks.
2. **Attributes**:
    - **id**: unique identifier for each task.
    - **name**: The title of the task.
    - **due_date**: deadline of the respective task.
    - **is_done**: A boolean attribute indicating whether the task has been completed.

Relationships

- The diagram shows that the "TASKS" entity has a direct relationship with each of the attributes (id, name, date, is_done). Each line connecting the TASKS entity to its attributes represents this relationship, indicating that each task is characterized by these four attributes.

Database Schema



Chapter 3

Implementation Phase

Database Setup

- Install and configure MySQL Database.
- Create the necessary database and user permissions.

Schema Implementation

- Translate the designed schema into SQL scripts.

```
CREATE TABLE tasks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    due_date DATE NOT NULL,
    is_done BOOLEAN NOT NULL DEFAULT FALSE
);
```

- Execute the scripts to create the database structure.

Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
public class TaskManager extends JFrame {
private ArrayList<Task> tasks = new ArrayList<>();
private JTextField taskNameField;
private JTextField dueDateField;
private JTextArea taskListArea;
private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/task_manager";
```

```
private static final String USERNAME = "adhi";
private static final String PASSWORD = "tellM3why";
private Connection connection;
public TaskManager() {
super("Task Manager");
// UI initialization
JLabel taskNameLabel = new JLabel("Task Name:");
taskNameLabel.setHorizontalAlignment(SwingConstants.CENTER);
taskNameField = new JTextField(20);
JLabel dueDateLabel = new JLabel("Due Date
(DD/MM/YYYY):");
dueDateLabel.setHorizontalAlignment(SwingConstants.CENTER);
dueDateField = new JTextField(20);
JButton addButton = new JButton("Add Task");
addButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
addTask();
}
});
JButton deleteButton = new JButton("Delete Task");
deleteButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
deleteTask();
}
});
deleteButton.setBackground(Color.RED); // Set
background color to red
JButton markDoneButton = new JButton("Mark Done");
markDoneButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
markTaskDone();
}
});
markDoneButton.setBackground(Color.GREEN); // Set
background color to green
JButton refreshButton = new JButton("Refresh Tasks");
refreshButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
refreshTasksFromDatabase();
}
});
taskListArea = new JTextArea(10, 30);
JScrollPane scrollPane = new JScrollPane(taskListArea);
// Layout
JPanel panel = new JPanel(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.gridx = 0;
```

```
gbc.gridy = 0;
gbc.anchor = GridBagConstraints.CENTER;
gbc.insets = new Insets(5, 5, 5, 5);
panel.add(taskNameLabel, gbc);
gbc.gridy++;
panel.add(taskNameField, gbc);
gbc.gridy++;
panel.add(dueDateLabel, gbc);
gbc.gridy++;
panel.add(dueDateField, gbc);
gbc.gridy++;
panel.add(addButton, gbc);
gbc.gridy++;
panel.add(deleteButton, gbc);
gbc.gridy++;
panel.add(markDoneButton, gbc);
gbc.gridy++;
panel.add(refreshButton, gbc);
gbc.gridy++;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.BOTH;
panel.add(scrollPane, gbc);
add(panel);
setSize(400, 500);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
try {
connection = DriverManager.getConnection(JDBC_URL,
USERNAME, PASSWORD);
} catch (SQLException ex) {
ex.printStackTrace();
}
}
private void sortTasks() {
refreshTaskList();
}
private void addTask() {
String taskName = taskNameField.getText();
String dueDateString = dueDateField.getText();
SimpleDateFormat dateFormat = new
SimpleDateFormat("dd/MM/yyyy");
Date dueDate;
try {
dueDate = dateFormat.parse(dueDateString);
} catch (ParseException e) {
JOptionPane.showMessageDialog(this, "Invalid date
```

```
format. Please use DD/MM/YYYY.");
return;
}
String sql = "INSERT INTO tasks (name, due_date,
is_done) VALUES (?, ?, ?)";
try (PreparedStatement pstmt =
connection.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {
pstmt.setString(1, taskName);
pstmt.setDate(2, new
java.sql.Date(dueDate.getTime()));
pstmt.setBoolean(3, false); // Initially task is
not done
pstmt.executeUpdate();
try (ResultSet generatedKeys =
pstmt.getGeneratedKeys()) {
if (generatedKeys.next()) {
int taskId = generatedKeys.getInt(1);
taskListArea.append("New task added with
ID: " + taskId + "\n");
}
}
} catch (SQLException e) {
JOptionPane.showMessageDialog(this, "Failed to add
task.");
e.printStackTrace();
}
refreshTaskList();
clearFields();
}
private void refreshTasksFromDatabase() {
refreshTaskList();
}
private void deleteTask() {
if (tasks.isEmpty()) {
JOptionPane.showMessageDialog(this, "No tasks to
delete.");
return;
}
String taskToDelete = JOptionPane.showInputDialog(this,
"Enter task ID to delete:");
try {
int taskId = Integer.parseInt(taskToDelete);
// Delete task from the ArrayList
boolean found = false;
for (Task task : tasks) {
if (task.getId() == taskId) {
```

```
tasks.remove(task);
found = true;
break;
}
}
if (!found) {
JOptionPane.showMessageDialog(this, "Task with
ID " + taskId + " not found.");
return;
}
// Delete task from the SQL table
String sql = "DELETE FROM tasks WHERE id = ?";
try (PreparedStatement pstmt =
connection.prepareStatement(sql)) {
pstmt.setInt(1, taskId);
int rowsAffected = pstmt.executeUpdate();
if (rowsAffected > 0) {
JOptionPane.showMessageDialog(this, "Task
with ID " + taskId + " deleted successfully.");
} else {
JOptionPane.showMessageDialog(this, "Task
with ID " + taskId + " not found in the database.");
}
} catch (SQLException e) {
JOptionPane.showMessageDialog(this, "Failed to
delete task.");
e.printStackTrace();
}
refreshTaskList();
} catch (NumberFormatException e) {
JOptionPane.showMessageDialog(this, "Please enter a
valid number.");
}
}
private void markTaskDoneById(int taskId) {
// Find the task by ID
Task taskToUpdate = null;
for (Task task : tasks) {
if (task.getId() == taskId) {
taskToUpdate = task;
break;
}
}
if (taskToUpdate == null) {
JOptionPane.showMessageDialog(this, "Task with ID "
+ taskId + " not found.");
return;
```

```
}
taskToUpdate.setDone(true); // Mark task as done in the
Swing application
// Update is_done field in the SQL table
String sql = "UPDATE tasks SET is_done = ? WHERE id =
?";
try (PreparedStatement pstmt =
connection.prepareStatement(sql)) {
pstmt.setBoolean(1, true);
pstmt.setInt(2, taskId);
int rowsAffected = pstmt.executeUpdate();
if (rowsAffected > 0) {
JOptionPane.showMessageDialog(this, "Task
marked as done successfully.");
} else {
JOptionPane.showMessageDialog(this, "Task with
ID " + taskId + " not found in the database.");
}
} catch (SQLException e) {
JOptionPane.showMessageDialog(this, "Failed to mark
task as done.");
e.printStackTrace();
}
refreshTaskList(); // Refresh the task list in the
Swing application
}
private void markTaskDone() {
if (tasks.isEmpty()) {
JOptionPane.showMessageDialog(this, "No tasks
available.");
return;
}
String taskToMark = JOptionPane.showInputDialog(this,
"Enter task ID to mark as done:");
try {
int taskId = Integer.parseInt(taskToMark);
markTaskDoneById(taskId);
} catch (NumberFormatException e) {
JOptionPane.showMessageDialog(this, "Please enter a
valid number.");
}
}
private void refreshTaskList() {
taskListArea.setText("");
// Fetch tasks from the database and populate the
ArrayList
tasks.clear();
```

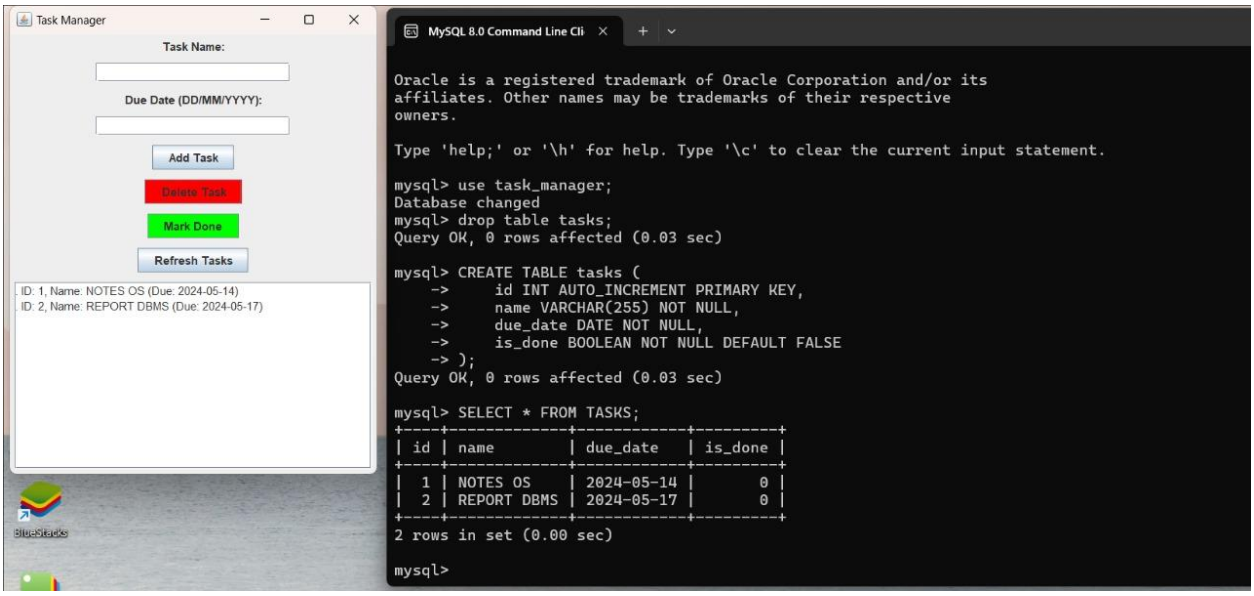
```
try (Statement statement =
connection.createStatement();
ResultSet resultSet =
statement.executeQuery("SELECT * FROM tasks")) {
while (resultSet.next()) {
int id = resultSet.getInt("id");
String name = resultSet.getString("name");
Date dueDate = resultSet.getDate("due_date");
boolean isDone =
resultSet.getBoolean("is_done");
tasks.add(new Task(id, name, dueDate, isDone));
}
} catch (SQLException e) {
e.printStackTrace();
}
for (int i = 0; i < tasks.size(); i++) {
Task task = tasks.get(i);
String status = task.isDone() ? " (Done)" : "";
taskListArea.append(". ID: " + task.getId() + ",
Name: " + task.getName() + " (Due: " + task.getDueDate() + ")"
+ status + "\n");
} //(i + 1) +
}
private void clearFields() {
taskNameField.setText("");
dueDateField.setText("");
}
public static void main(String[] args) {
SwingUtilities.invokeLater(new Runnable() {
public void run() {
new TaskManager();
}
});
}
}
class Task {
private int id;
private String name;
private Date dueDate;
private boolean done;
public Task(int id, String name, Date dueDate, boolean
done) {
this.id = id;
this.name = name;
this.dueDate = dueDate;
this.done = done;
}
```

```
public int getId() {
return id;
}
public String getName() {
return name;
}
public Date getDueDate() {
return dueDate;
}
public boolean isDone() {
return done;
}
public void setDone(boolean done){
this.done = done;
}
}
```

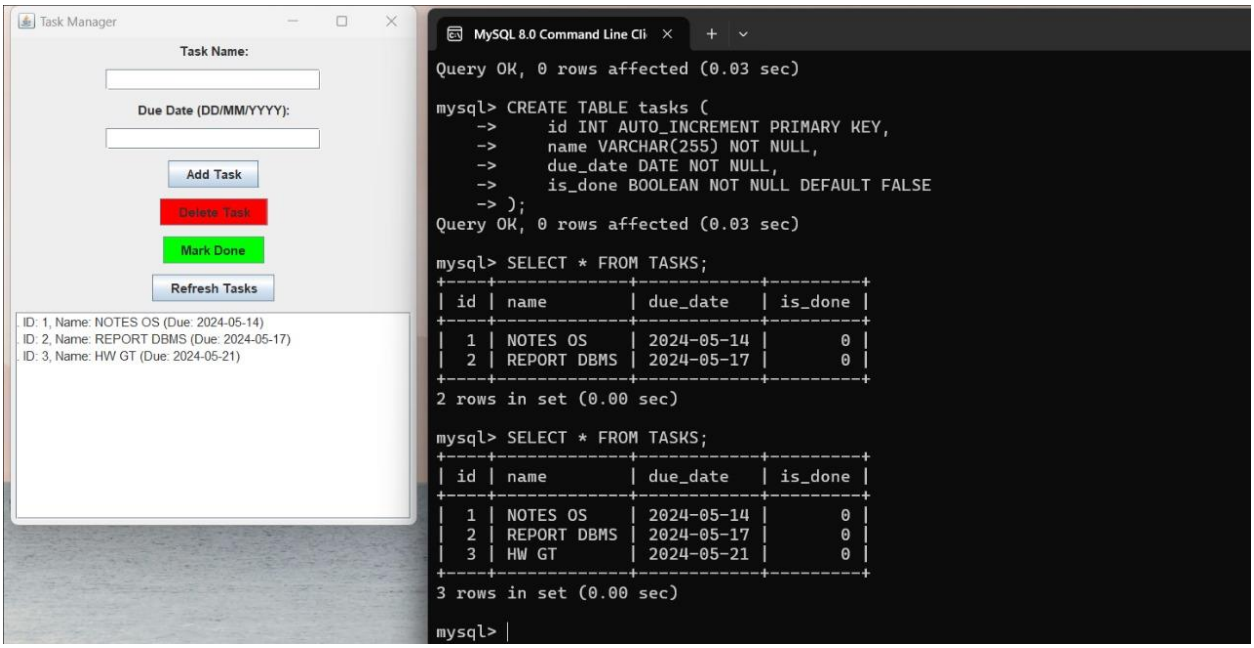
Chapter 4

Testing Phase

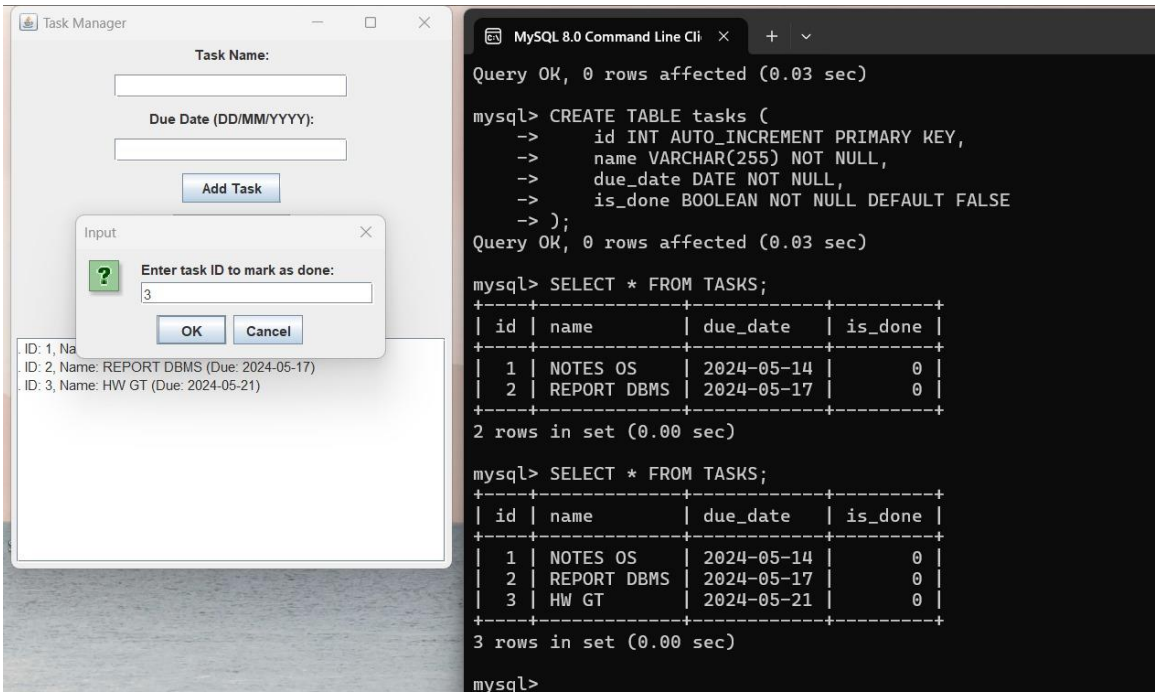
1. BEFORE INSERTION



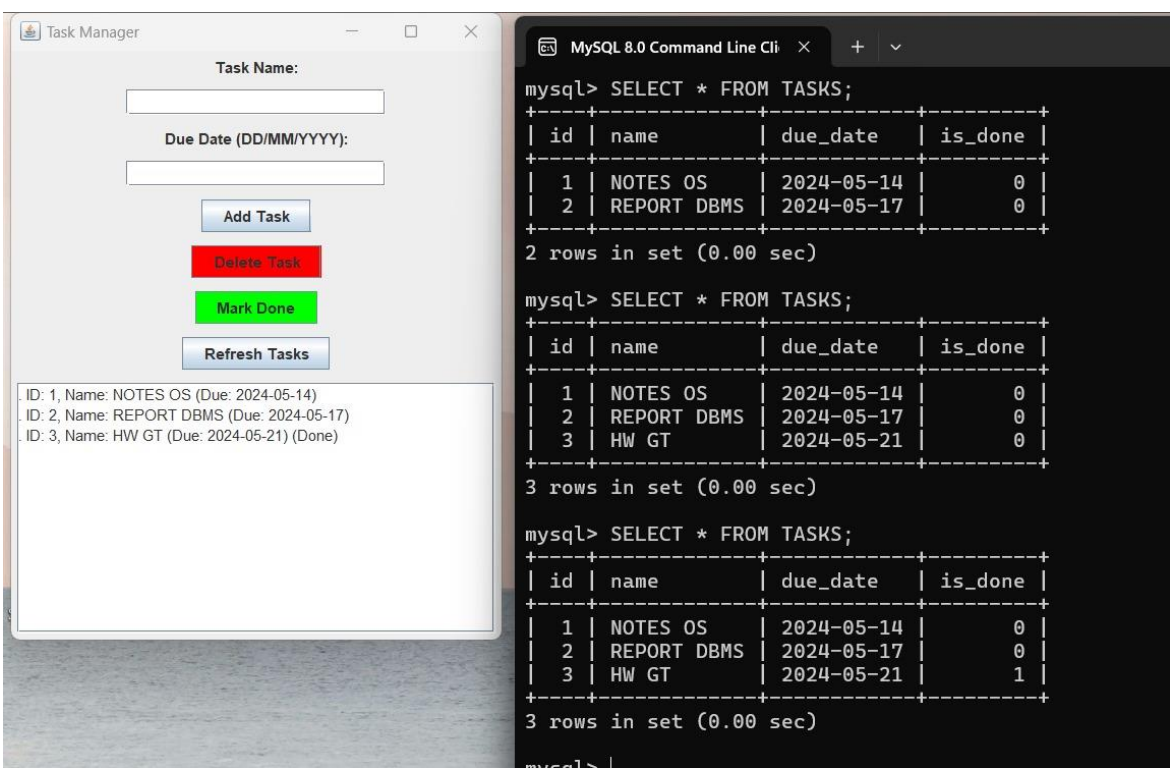
2. AFTER INSERTION



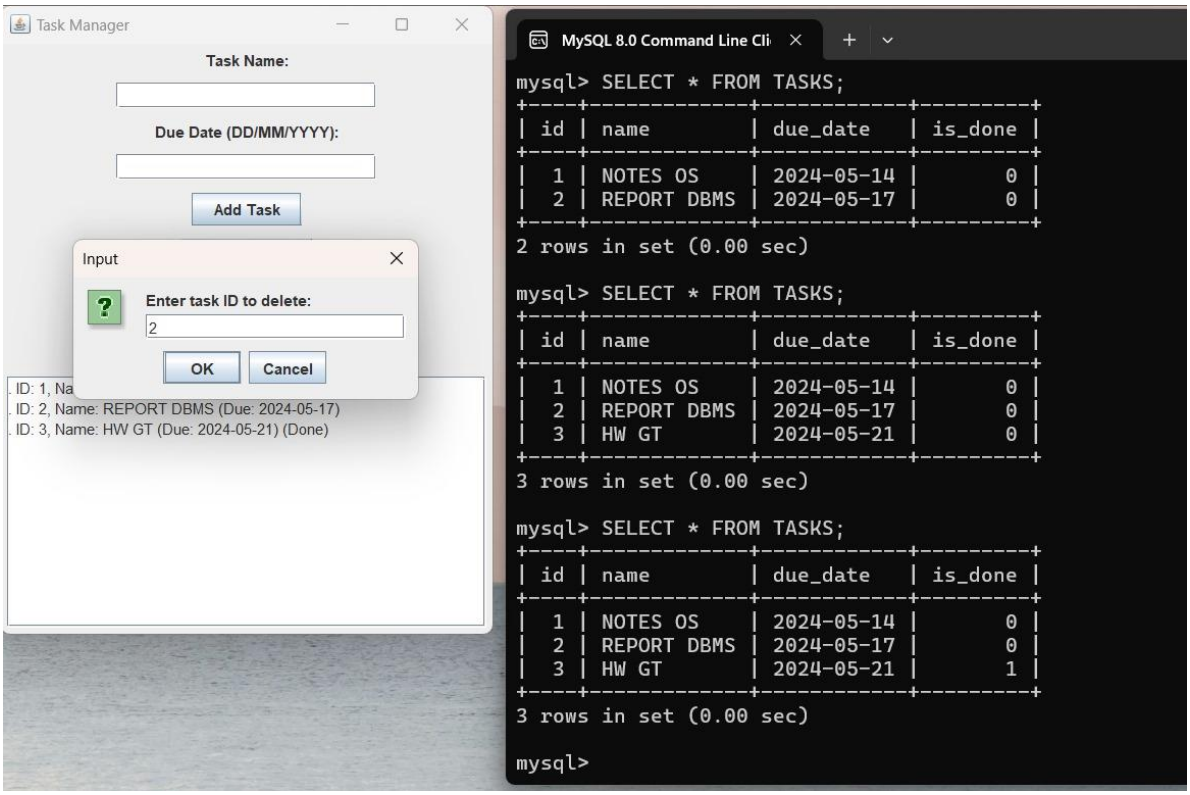
3. BEFORE UPDATION



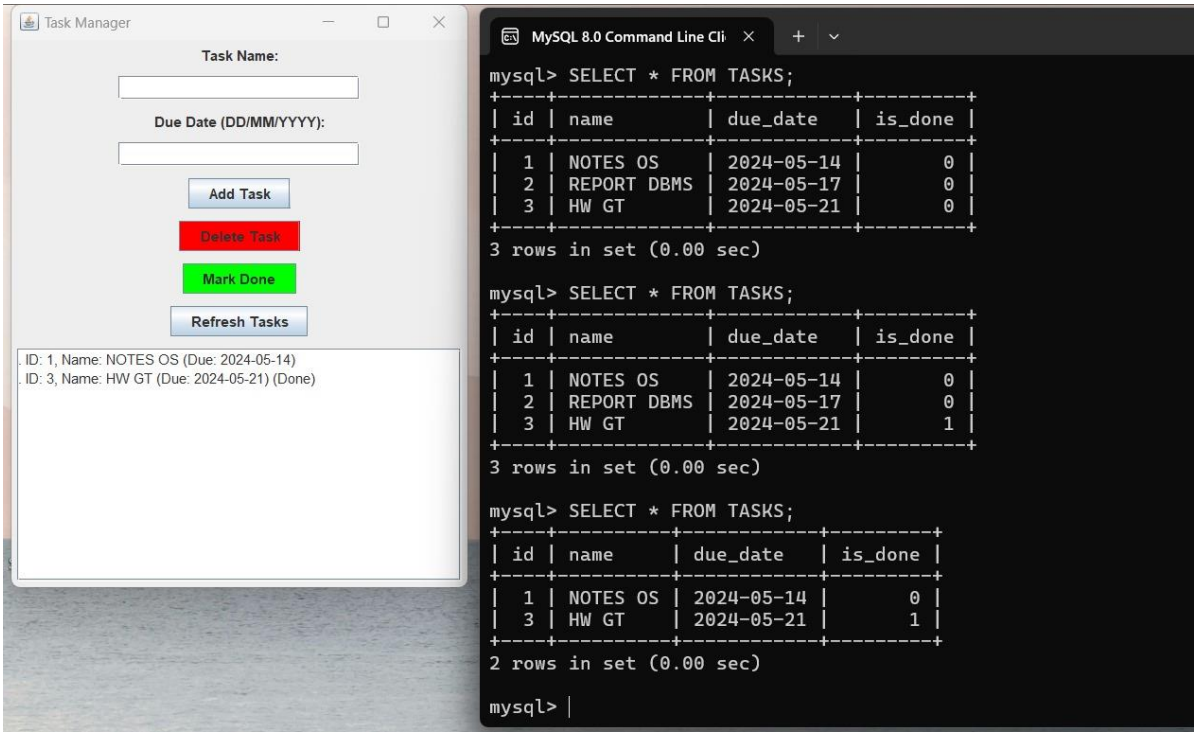
4. AFTER UPDATION



5. BEFORE DELETION



6. AFTER DELETION



Chapter 5

Conclusion

The Task Manager application offers a user •friendly interface for managing tasks efficiently.
It seamlessly integrates with a MySQL database for persistent storage and provides
essential task management features. With error handling mechanisms in place, the
application ensures a smooth user experience.
This report summarizes the functionality and implementation details of the Task Manager
application, fulfilling the requirements for the assignment submission.

References

- https://www.youtube.com/watch?v=E90z9Qw8cHQ
- https://docs.oracle.com/javase/tutorial/jdbc/index.html
- https://www.geeksforgeeks.org/sql-concepts-and-queries/