```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
public class EnhancedTaskManager {
private static final String DB_URL = "jdbc:mysql://localhost:3306/Tasks";
private static final String DB_USER = "root";
private static final String DB_PASS = "Sanjay1234%";
private Connection connection;
private Frame mainFrame;
private TextField taskField;
private TextArea descriptionField;
private Choice priorityChoice;
private List taskList;
private Button submitButton, removeButton, markCompleteButton, clearAllButton,
editButton, sortButton;
private Label notificationLabel;
private ExecutorService executorService;
public EnhancedTaskManager() {
try {
connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
executorService = Executors.newFixedThreadPool(2);
prepareGUI();
fetchTasksFromDatabase(); // Load tasks on startup
} catch (SQLException e) {
e.printStackTrace();
notifyUser("Database connection failed!");
}
}
private void prepareGUI() {
```

```
mainFrame = new Frame("Enhanced Task Manager");
mainFrame.setSize(500, 600);
mainFrame.setLayout(new FlowLayout(FlowLayout.LEFT));
mainFrame.addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent windowEvent) {
executorService.shutdown();
try {
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
System.exit(0);
}
});
taskField = new TextField(35);
descriptionField = new TextArea(3, 35);
priorityChoice = new Choice();
priorityChoice.add("HIGH");
priorityChoice.add("MEDIUM");
priorityChoice.add("LOW");
taskList = new List();
submitButton = new Button("Add Task");
submitButton.addActionListener(e -> executorService.execute(this::addOrEditTask));
removeButton = new Button("Remove Task");
removeButton.addActionListener(e -> executorService.execute(this::removeTask));
markCompleteButton = new Button("Mark Complete");
markCompleteButton.addActionListener(e ->
executorService.execute(this::markTaskComplete));
clearAllButton = new Button("Clear All Tasks");
clearAllButton.addActionListener(e -> executorService.execute(this::clearAllTasks));
notificationLabel = new Label("Welcome to the Task Manager!");
```

```java
mainFrame.add(new Label("Task:"));
mainFrame.add(taskField);
mainFrame.add(new Label("Description:"));
mainFrame.add(descriptionField);
mainFrame.add(new Label("Priority:"));
mainFrame.add(priorityChoice);
mainFrame.add(submitButton);
mainFrame.add(removeButton);
mainFrame.add(markCompleteButton);
mainFrame.add(clearAllButton);
mainFrame.add(taskList);
mainFrame.add(notificationLabel);
mainFrame.setVisible(true);
}
private void addOrEditTask() {
String description = taskField.getText().trim();
String details = descriptionField.getText().trim();
String priority = priorityChoice.getSelectedItem();
if (description.isEmpty()) {
notifyUser("Task cannot be empty.");
return;
}
String query = "INSERT INTO tasks (description, priority) VALUES (?, ?)";
try (PreparedStatement stmt = connection.prepareStatement(query)) {
stmt.setString(1, description + ": " + details);
stmt.setString(2, priority);
stmt.executeUpdate();
notifyUser("Task added successfully.");
fetchTasksFromDatabase();
} catch (SQLException e) {
e.printStackTrace();
```

```
notifyUser("Failed to add task.");
}
}
private void removeTask() {
int selectedIndex = taskList.getSelectedIndex();
if (selectedIndex < 0) {
notifyUser("No task selected to remove.");
return;
}
String selectedTask = taskList.getItem(selectedIndex);
String query = "DELETE FROM tasks WHERE description = ?";
try (PreparedStatement stmt = connection.prepareStatement(query)) {
stmt.setString(1, selectedTask.split(" \\(")[0]);
stmt.executeUpdate();
notifyUser("Task removed successfully.");
fetchTasksFromDatabase();
} catch (SQLException e) {
e.printStackTrace();
notifyUser("Failed to remove task.");
}
}
private void markTaskComplete() {
int selectedIndex = taskList.getSelectedIndex();
if (selectedIndex < 0) {
notifyUser("No task selected to mark complete.");
return;
}
String selectedTask = taskList.getItem(selectedIndex);
String query = "UPDATE tasks SET completed = TRUE WHERE description = ?";
try (PreparedStatement stmt = connection.prepareStatement(query)) {
stmt.setString(1, selectedTask.split(" \\(")[0]);
stmt.executeUpdate();
notifyUser("Task marked as complete.");
fetchTasksFromDatabase();
} catch (SQLException e) {
e.printStackTrace();
notifyUser("Failed to mark task complete.");
}
}
private void clearAllTasks() {
String query = "DELETE FROM tasks";
try (PreparedStatement stmt = connection.prepareStatement(query)) {
stmt.executeUpdate();
notifyUser("All tasks cleared.");
fetchTasksFromDatabase();
} catch (SQLException e) {
e.printStackTrace();
notifyUser("Failed to clear tasks.");
}
}
private void fetchTasksFromDatabase() {
String query = "SELECT description, priority, completed FROM tasks";
try (Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery(query)) {
EventQueue.invokeLater(() -> {
taskList.removeAll();
try {
while (rs.next()) {
String taskText = (rs.getBoolean("completed") ? "[Completed] " : "")
+ rs.getString("description") + " (" + rs.getString("priority") + ")";
taskList.add(taskText);
}
} catch (SQLException e) {
e.printStackTrace();
}
});
} catch (SQLException e) {
e.printStackTrace();
notifyUser("Failed to fetch tasks.");
}
}
private void notifyUser(String message) {
EventQueue.invokeLater(() -> notificationLabel.setText(message));
}
public static void main(String[] args) {
new EnhancedTaskManager();
}
}
```