ULAB
UNIVERSITY OF LIBERAL ARTS BANGLADESH

Open-Ended Lab Report

Course Code: CSE 2104

Course Title: Object Oriented Programming (CSE2104)

Submitted By: Md. Tazminur Rahman Tanim

ID: 242014124

Section: 4L

Submitted By: Task Management System (GUI Based)

Submitted To: Towsif Zahin Khan

Lecturer, Department Of Computer Science And Engineering

Date of Submission: Aug 24, 2025

Objectives of the Software :

The primary objectives of this Task Management System are:

- To design and implement a **GUI-based task management application** in Java.
- To demonstrate the **core principles of Object-Oriented Programming** (Encapsulation, Inheritance, Polymorphism, Abstraction).
- To integrate **file handling** for storing and retrieving task data persistently in `tasks.csv`.
- To allow users to perform **basic CRUD operations** (Add, Update, Delete, View Tasks).
- To provide a **user-friendly graphical interface** using Java Swing.
- To ensure the project aligns with the **open-ended lab requirements** of using modern tools, OOP, and GUI.

System functionalities and features:

The **Task Management System (GUI-based)** incorporates the following features:

- **Add Task**: Users can add tasks with *title, description, due date, and priority*.
- **Update Task**: Existing tasks can be selected from the table and updated.
- **Delete Task**: Users can delete any task after confirmation.
- **View Tasks**: A table displays all tasks in a structured format.
- **Persistence (File Handling)**: Tasks are stored in `tasks.csv` and automatically loaded at startup.
- **GUI (Java Swing)**: User-friendly interface with forms, buttons, and a table.
- **OOP Principles Implemented**:
    - Encapsulation (private fields with getters/setters in `Task`)
    - Inheritance (`ImportantTask` extends `Task`)
    - Polymorphism (overridden `toString()` method)
    - Abstraction (`TaskOperations` interface)

This ensures the software is both **functional** and **aligned with real-world task management needs**.

Design structured algorithms :

Algorithm for Adding a Task

1. Start
2. Accept input: Title, Description, Due Date, Priority
3. Create a new `Task` object with these details
4. Append the task into the `ArrayList` of tasks in `TaskManager`
5. Save tasks into `tasks.csv` using file handling
6. Refresh the GUI table to show the new task
7. End

Algorithm for Updating a Task

1. Start
2. User selects a task from the GUI table
3. Accept updated values from input fields
4. Replace the old `Task` object with a new one at the selected index
5. Save changes to `tasks.csv`
6. Refresh the GUI table
7. End

Algorithm for Deleting a Task

1. Start
2. User selects a task row in the GUI
3. Confirm deletion with a dialog box
4. If confirmed, remove the task from the `ArrayList`
5. Save changes to `tasks.csv`
6. Refresh the GUI table
7. End

Algorithm for Loading Tasks at Startup

1. Start
2. Check if `tasks.csv` exists
3. If exists, read each line and parse task details
4. Create `Task` objects and insert into `ArrayList`
5. Populate the GUI table with these tasks
6. End

Screenshots of the Code :

Implemented in **Java** with:

- `Task.java` → Encapsulation
- `ImportantTask.java` → Inheritance & Polymorphism
- `TaskOperations.java` → Abstraction (Interface)
- `TaskManager.java` → File Handling + Core CRUD logic
- `TaskManagementSystem.java` → GUI (Java Swing)

Task.java (Encapsulation)

```java

package com.mycompany.taskmanagementsystem;


// Encapsulation: private fields + getters/setters
public class Task {
    private String title;
    private String description;
    private String dueDate; // yyyy-MM-dd
    private String priority; // Low/Medium/High

    public Task(String title, String description, String dueDate, String priority) {
        this.title = title;
        this.description = description;
        this.dueDate = dueDate;
        this.priority = priority;
    }

    public String getTitle() { return title; }
    public String getDescription() { return description; }
    public String getDueDate() { return dueDate; }
    public String getPriority() { return priority; }

    public void setTitle(String title) { this.title = title; }
    public void setDescription(String description) { this.description = description; }
    public void setDueDate(String dueDate) { this.dueDate = dueDate; }
    public void setPriority(String priority) { this.priority = priority; }

    // Polymorphism: override toString for quick display/logging
    @Override
    public String toString() {
        return title + " | " + dueDate + " | " + priority;
    }
}

```

ImportantTask.java (Inheritance & Polymorphism)

```java
package com.mycompany.taskmanagementsystem;

// Inheritance: ImportantTask extends Task
public class ImportantTask extends Task {
    private String note; // extra field to show extension

    public ImportantTask(String title, String description, String dueDate, String priority, String note) {
        super(title, description, dueDate, priority);
        this.note = note;
    }

    public String getNote() { return note; }

    // Polymorphism: different toString
    @Override
    public String toString() {
        return "[IMPORTANT] " + super.toString() + " | Note: " + note;
    }
}
```

TaskOperations.java (Abstraction with Interface)

```java
package com.mycompany.taskmanagementsystem;

import java.util.List;

// Abstraction: interface defines behaviour
public interface TaskOperations {
    void addTask(Task task);
    void updateTask(int index, Task task);
    void deleteTask(int index);
    List<Task> getAllTasks();
    void save() throws Exception;
    void load() throws Exception;
}
```

TaskManager.java (File Handling & Data Storage)

```java
package com.mycompany.taskmanagementsystem;

import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;

// Concrete implementation + File Handling (CSV)
public class TaskManager implements TaskOperations {
    private final List<Task> tasks = new ArrayList<>();
    private final Path file = Paths.get("tasks.csv");

    @Override
    public void addTask(Task task) {
        tasks.add(task);
        try { save(); } catch (Exception ignored) {}
    }

    @Override
    public void updateTask(int index, Task task) {
        if (index >= 0 && index < tasks.size()) {
            tasks.set(index, task);
            try { save(); } catch (Exception ignored) {}
        }
    }

    @Override
    public void deleteTask(int index) {
        if (index >= 0 && index < tasks.size()) {
            tasks.remove(index);
            try { save(); } catch (Exception ignored) {}
        }
    }

    @Override
    public List<Task> getAllTasks() {
        return tasks;
    }

```

TaskManager.java (File Handling & Data Storage)

```java
@Override
public void save() throws Exception {
    // CSV Header + rows; quote fields safely
    try (BufferedWriter bw = Files.newBufferedWriter(file, StandardCharsets.UTF_8)) {
        bw.write("title,description,dueDate,priority\n");
        for (Task t : tasks) {
            bw.write(csv(t.getTitle())); bw.write(",");
            bw.write(csv(t.getDescription())); bw.write(",");
            bw.write(csv(t.getDueDate())); bw.write(",");
            bw.write(csv(t.getPriority()));
            bw.write("\n");
        }
    }
}

@Override
public void load() throws Exception {
    tasks.clear();
    if (!Files.exists(file)) return;
    try (BufferedReader br = Files.newBufferedReader(file, StandardCharsets.UTF_8)) {
        String line;
        boolean headerSkipped = false;
        while ((line = br.readLine()) != null) {
            if (!headerSkipped) { headerSkipped = true; continue; }
            List<String> cols = parseCsvLine(line);
            if (cols.size() >= 4) {
                String title = cols.get(0);
                String desc  = cols.get(1);
                String date  = cols.get(2);
                String prio  = cols.get(3);
                tasks.add(new Task(title, desc, date, prio));
            }
        }
    }
}

// --- CSV helpers ---
private String csv(String s) {
    if (s == null) s = "";
    String q = s.replace("\"", "\"\"");
    return "\"" + q + "\"";
}
```

```java
private List<String> parseCsvLine(String line) {
        List<String> out = new ArrayList<>();
        StringBuilder cur = new StringBuilder();
        boolean inQ = false;
        for (int i = 0; i < line.length(); i++) {
            char c = line.charAt(i);
            if (inQ) {
                if (c == '"') {
                    if (i + 1 < line.length() && line.charAt(i + 1) == '"') {
                        cur.append('"'); i++;
                    } else {
                        inQ = false;
                    }
                } else cur.append(c);
            } else {
                if (c == '"') inQ = true;
                else if (c == ',') { out.add(cur.toString()); cur.setLength(0); }
                else cur.append(c);
            }
        }
        out.add(cur.toString());
        return out;
    }
}

```

TaskManagementSystem.java (Main GUI Implementation)

```java
package com.mycompany.taskmanagementsystem;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.List;

public class TaskManagementSystem extends JFrame {

    private final TaskManager manager = new TaskManager();
    private final DefaultTableModel model =
            new DefaultTableModel(new String[]{"Title","Description","Due Date","Priority"}, 0) {
                @Override public boolean isCellEditable(int r, int c) { return false; }
            };

    private JTable table;
    private JTextField tfTitle, tfDue;
    private JTextArea taDesc;
    private JComboBox<String> cbPriority;

    public TaskManagementSystem() {
        setTitle("Task Management System - GUI (OOP + File I/O)");
        setSize(900, 520);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout(10,10));

        // ---- Top form panel ----
        JPanel form = new JPanel(new GridBagLayout());
        GridBagConstraints gc = new GridBagConstraints();
        gc.insets = new Insets(6,6,6,6);
        gc.fill = GridBagConstraints.HORIZONTAL;

        tfTitle = new JTextField();
        tfDue   = new JTextField(); // format hint: yyyy-MM-dd
        taDesc  = new JTextArea(4, 20);
        taDesc.setLineWrap(true);
        taDesc.setWrapStyleWord(true);
        cbPriority = new JComboBox<>(new String[]{"Low","Medium","High"});

        int r=0;
        gc.gridy = r; gc.gridx=0; form.add(new JLabel("Title:"), gc);
        gc.gridx=1; form.add(tfTitle, gc);
```

TaskManagementSystem.java (Main GUI Implementation)

```
r++; gc.gridy = r; gc.gridx=0; form.add(new JLabel("Due Date (yyyy-MM-dd):"), gc);
gc.gridx=1; form.add(tfDue, gc);

r++; gc.gridy = r; gc.gridx=0; form.add(new JLabel("Priority:"), gc);
gc.gridx=1; form.add(cbPriority, gc);

r++; gc.gridy = r; gc.gridx=0; gc.anchor = GridBagConstraints.NORTHWEST;
form.add(new JLabel("Description:"), gc);
gc.gridx=1; gc.weightx=1; gc.weighty=1; gc.fill = GridBagConstraints.BOTH;
form.add(new JScrollPane(taDesc), gc);

add(form, BorderLayout.NORTH);

// ---- Table center ----
table = new JTable(model);
table.setRowHeight(22);
add(new JScrollPane(table), BorderLayout.CENTER);

// ---- Buttons bottom ----
JPanel actions = new JPanel(new FlowLayout(FlowLayout.RIGHT, 10, 8));
JButton btnAdd = new JButton("Add");
JButton btnUpdate = new JButton("Update");
JButton btnDelete = new JButton("Delete");
JButton btnClear = new JButton("Clear");
JButton btnReload = new JButton("Reload from File");

actions.add(btnAdd);
actions.add(btnUpdate);
actions.add(btnDelete);
actions.add(btnClear);
actions.add(btnReload);
add(actions, BorderLayout.SOUTH);

// ---- Events ----
btnAdd.addActionListener(e -> {
    Task t = readFormAsTask();
    if (t == null) return;
    manager.addTask(t);
    refreshTable();
    clearForm();
});
```

TaskManagementSystem.java (Main GUI Implementation)

```java
btnUpdate.addActionListener(e -> {
            int row = table.getSelectedRow();
            if (row < 0) { msg("Select a row to update."); return; }
            Task t = readFormAsTask();
            if (t == null) return;
            manager.updateTask(row, t);
            refreshTable();
        });

        btnDelete.addActionListener(e -> {
            int row = table.getSelectedRow();
            if (row < 0) { msg("Select a row to delete."); return; }
            int c = JOptionPane.showConfirmDialog(this, "Delete selected task?", "Confirm", JOptionPane.YES_NO_OPTION);
            if (c == JOptionPane.YES_OPTION) {
                manager.deleteTask(row);
                refreshTable();
                clearForm();
            }
        });

        btnClear.addActionListener(e -> clearForm());

        btnReload.addActionListener(e -> {
            try {
                manager.load();
                refreshTable();
                msg("Reloaded from file.");
            } catch (Exception ex) {
                msg("Failed to load: " + ex.getMessage());
            }
        });

        // Table selection -> fill form
        table.getSelectionModel().addListSelectionListener(e -> {
            int row = table.getSelectedRow();
            if (row >= 0) {
                tfTitle.setText(val(row,0));
                taDesc.setText(val(row,1));
                tfDue.setText(val(row,2));
                cbPriority.setSelectedItem(val(row,3));
            }
        });

        // Initial load
        try { manager.load(); } catch (Exception ignored) {}
        refreshTable();
    }

    private Task readFormAsTask() {
        String title = tfTitle.getText().trim();
        String due   = tfDue.getText().trim();
        String desc  = taDesc.getText().trim();
        String prio  = (String) cbPriority.getSelectedItem();

        if (title.isEmpty()) { msg("Title is required."); return null; }
        if (due.isEmpty())   { msg("Due date is required (yyyy-MM-dd)."); return null; }
        if (prio == null) prio = "Low";

        // Example: if title starts with "!" create ImportantTask (shows inheritance/polymorphism in action)
        if (title.startsWith("!")) {
            return new ImportantTask(title, desc, due, prio, "Flagged by user with '!'");
        }
        return new Task(title, desc, due, prio);
    }
```

TaskManagementSystem.java (Main GUI Implementation)

```
private void refreshTable() {
        model.setRowCount(0);
        List<Task> list = manager.getAllTasks();
        for (Task t : list) {
            model.addRow(new Object[]{ t.getTitle(), t.getDescription(), t.getDueDate(), t.getPriority() });
        }
    }

    private void clearForm() {
        tfTitle.setText("");
        tfDue.setText("");
        taDesc.setText("");
        cbPriority.setSelectedIndex(0);
        table.clearSelection();
    }

    private String val(int row, int col) {
        Object v = model.getValueAt(row, col);
        return v == null ? "" : v.toString();
    }

    private void msg(String s) {
        JOptionPane.showMessageDialog(this, s);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new TaskManagementSystem().setVisible(true));
    }
}
```

Evidence of Core OOP Features Implemented:

Encapsulation

Implemented in `Task` class using **private fields** and **getters/setters**.

```
private String title;

private String description;

private String dueDate;

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }
```

Inheritance

`ImportantTask` class inherits from `Task`.

```
class ImportantTask extends Task {
    private String note;
    public ImportantTask(String t, String d, String date, String p, String note) {
        super(t, d, date, p);
        this.note = note;
    }
}
```

Polymorphism

Overriding `toString()` method in both `Task` and `ImportantTask`.

```
@Override
public String toString() {
    return "[IMPORTANT] " + super.toString() + " | Note: " + note;
}
```

Abstraction

`TaskOperations` interface defines the **contract** for all task management operations.

```
interface TaskOperations {
    void addTask(Task task);
    void updateTask(int index, Task task);
    void deleteTask(int index);
    List<Task> getAllTasks();
}
```

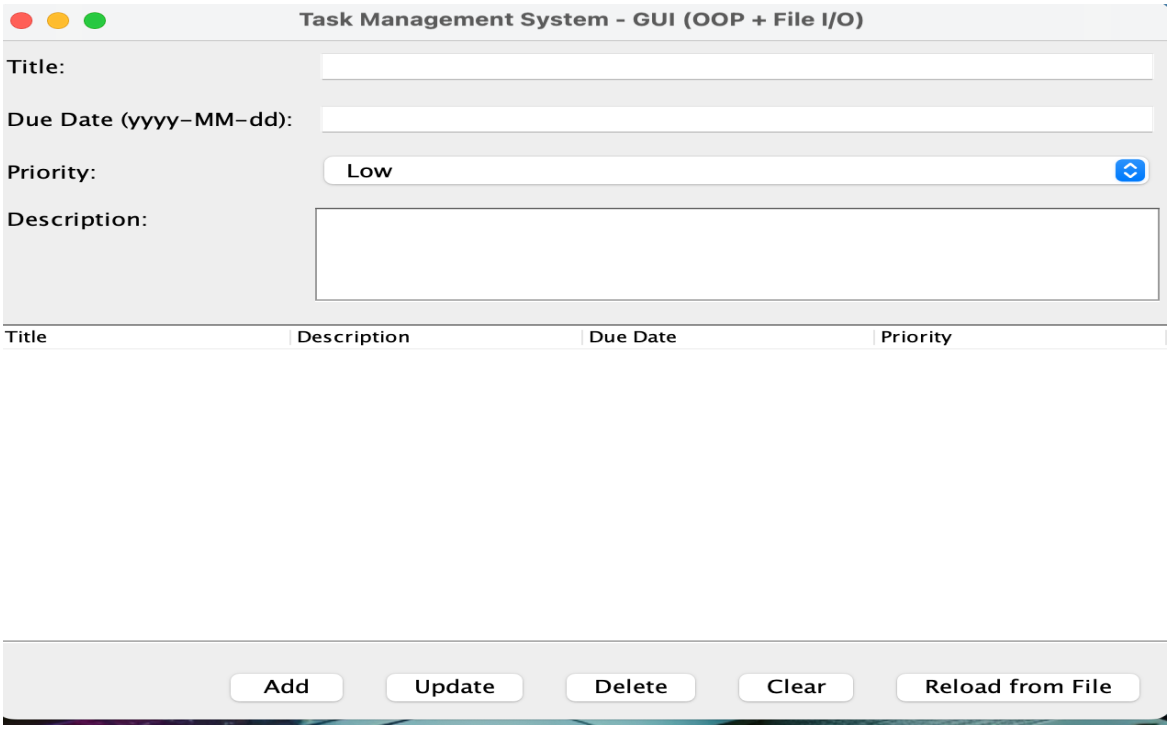
File Handling

Implemented in `TaskManager` class using **CSV-based save & load**.

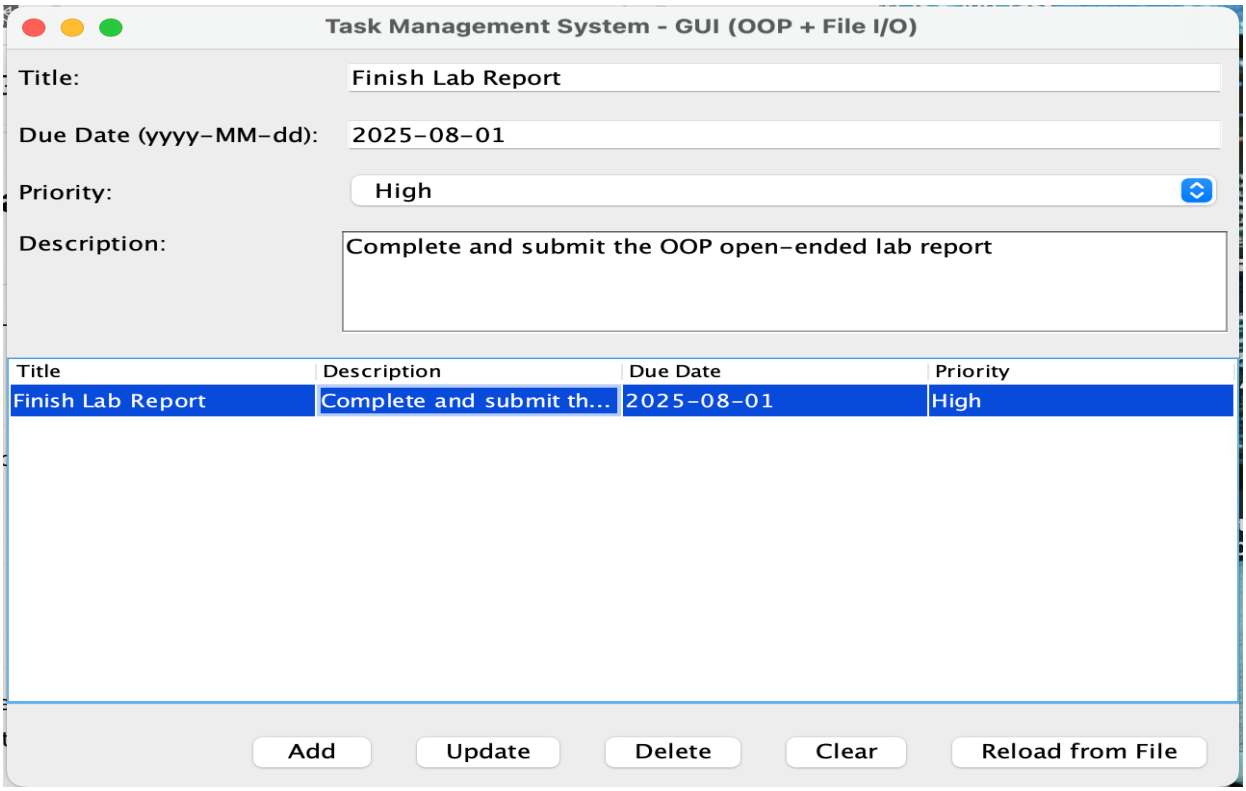
```
public void save() throws Exception {
    try (BufferedWriter bw = Files.newBufferedWriter(file, StandardCharsets.UTF_8)) {
        bw.write("title,description,dueDate,priority\n");
        for (Task t : tasks) {
            bw.write(csv(t.getTitle()) + "," + csv(t.getDescription()) + "," +
                    csv(t.getDueDate()) + "," + csv(t.getPriority()) + "\n");
        }
    }
}
```

Screenshots of the GUI Implementation:

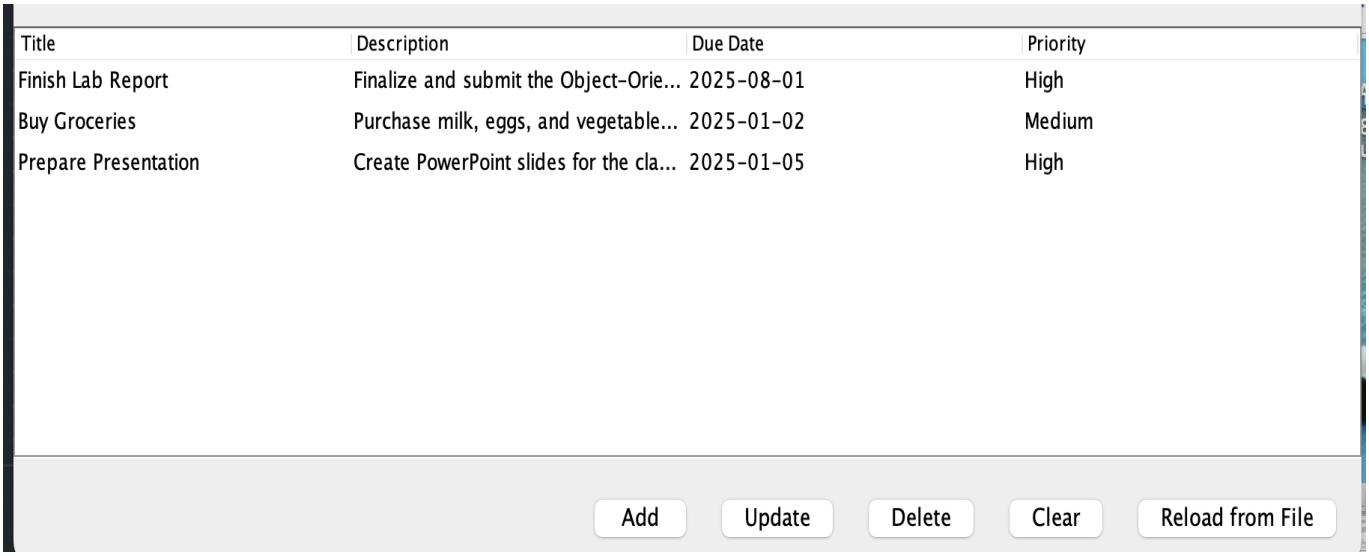
Insert screenshots taken while running the program:



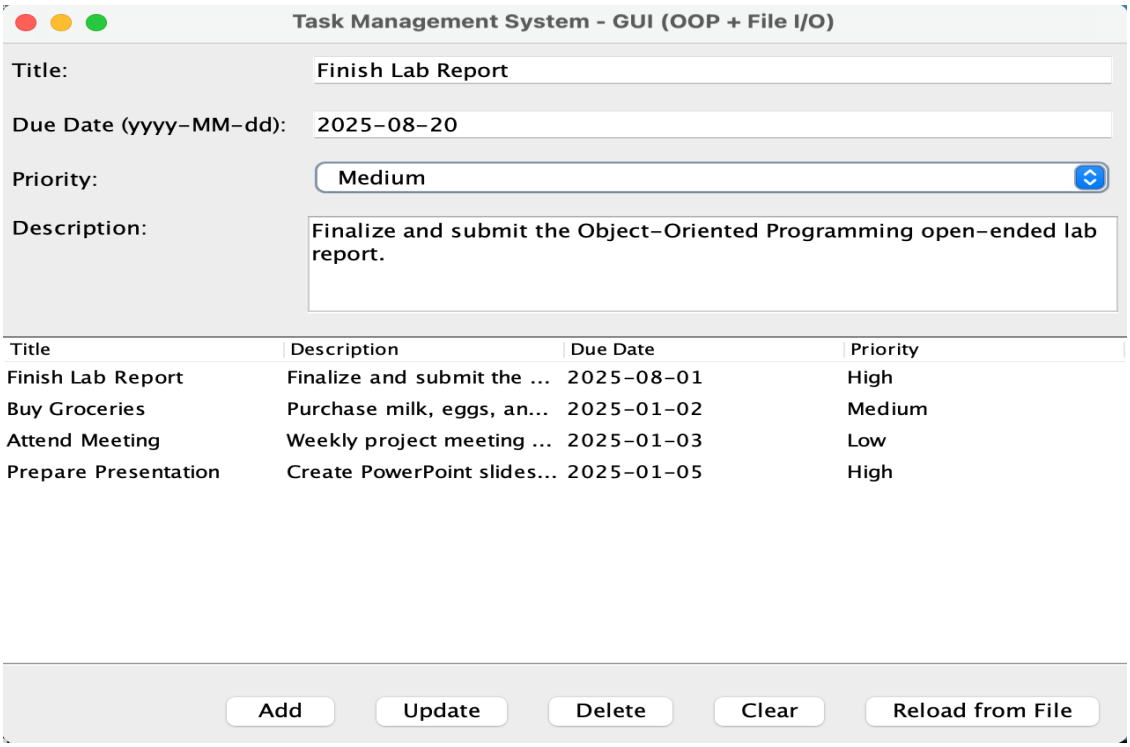
GUI Main Window (Empty Task List)



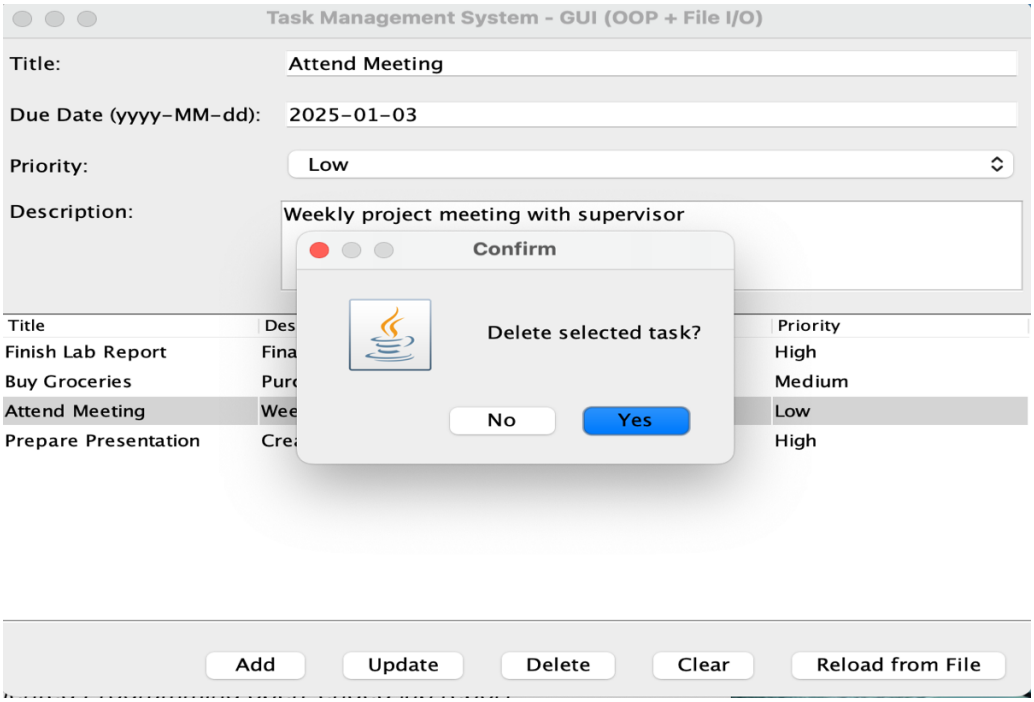
Adding a Task (Input Fields + Add Button)



Viewing Tasks in the Table



Updating a Task



Deleting a Task (Confirmation Dialog)

Conclusion:

This open-ended lab demonstrates a fully functional **Task Management System** built with **Java Swing, File Handling, and OOP principles**.
The project successfully covers:

- **Encapsulation, Inheritance, Polymorphism, and Abstraction**
- **CRUD operations with persistent storage**
- **Modern GUI-based interface**

Thus, the software meets both the **given problem requirements** and the **open-ended features** (modern tools, GUI, OOP).