**REPORT**

**Title: To-Do List Application**

We are creating this to-do list application to help people manage their time and tasks more efficiently. In today’s busy world, it’s easy to get overwhelmed and forget important things. This app is our way of bringing a simple, effective solution that helps users stay organized, reduce stress, and achieve their daily goals with clarity and ease.

**Concept (Problem Statement):**

In our fast-paced daily lives, people often struggle to keep track of their tasks and responsibilities, leading to missed deadlines, forgotten activities, and unnecessary stress. Without a simple and reliable system to organize their work, many individuals find it difficult to stay productive and focused. This highlights the need for a user-friendly to-do list application that helps users manage their time and tasks effectively.

**To-Do List Application – DSA LAB Project**

**Created By:**  
Javeria Baqar (67152)

# 1. **Background / Theory**

We developed a To-Do List Application to help users manage their daily tasks effectively. In our fast-paced lives, it's common to forget tasks or miss deadlines. This application provides a simple, interactive way to track and organize tasks, set reminders, and mark tasks as complete. Currently, the application supports core functionalities such as task addition, deletion, reminders, and completion.

# 2. **Procedure / Methodology**

The project was developed in Java using object-oriented principles. The application comprises two classes: Task and ToDoListApplication. The Task class defines the structure of each task, while the main class handles user input and task management via a command-line interface. Java Collections (ArrayList) were used to store tasks. User inputs are managed using the Scanner class. Each task includes a name, due date, reminder time, and completion status.

# 3. **Data Collection**

No external data collection was required. All task data is user-provided during runtime.

# 4. **Analysis**

The current implementation uses ArrayList to manage tasks, which provides O(1) access and O(n) removal complexity. This structure is suitable for small to medium-sized task lists and allows for flexible task management via indexing and iteration.

# 5. **Discussion on Results**

The application fulfills its basic goals of task management. The structure is modular and allows for future enhancements. One limitation is the absence of a graphical user interface, which could make the application more user-friendly.

# 6. **Concluding Remarks**

This project demonstrates how core data structures and Java can be used to create useful real-world applications. The app currently handles basic task management well. Future improvements may include GUI development and persistent storage.

# 7. **References**

1. YouTube tutorials on Java OOP  
2 GitHub  
3. GeeksforGeeks

# **8.** **Implementation Of Code:**

package todolist;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class ToDoListApplication {

private List<Task> tasks;

public ToDoListApplication() {

tasks = new ArrayList<>();

}

public void addTask(String name, LocalDateTime dueDate, LocalDateTime reminder) {

Task task = new Task(name, dueDate, reminder);

tasks.add(task);

System.out.println("Task added: " + task);

}

public void displayTasks() {

if (tasks.isEmpty()) {

System.out.println("No tasks available.");

return;

}

System.out.println("Tasks:");

for (int i = 0; i < tasks.size(); i++) {

System.out.println((i + 1) + ". " + tasks.get(i));

}

}

public void markTaskDone(int index) {

if (index < 1 || index > tasks.size()) {

System.out.println("Invalid task number.");

return;

}

Task task = tasks.get(index - 1);

if (task.isDone()) {

System.out.println("Task is already marked as done.");

} else {

task.markDone();

System.out.println("Task marked as done.");

}

}

public void deleteTask(int index) {

if (index < 1 || index > tasks.size()) {

System.out.println("Invalid task number.");

return;

}

tasks.remove(index - 1);

System.out.println("Task deleted.");

}

public void checkReminders() {

LocalDateTime now = LocalDateTime.now();

boolean found = false;

for (int i = 0; i < tasks.size(); i++) {

Task t = tasks.get(i);

if (!t.isDone() && t.getReminder() != null && !t.getReminder().isAfter(now)) {

if (!found) {

System.out.println("=== Reminders Due ===");

found = true;

}

System.out.printf("%d. %s%n", i + 1, t);

}

}

if (!found) {

System.out.println("No reminders due right now.");

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

ToDoListApplication app = new ToDoListApplication();

Scanner scanner = new Scanner(System.in);

DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");

while (true) {

System.out.println("\n1. Add Task");

System.out.println("2. Display Tasks");

System.out.println("3. Tasks Done");

System.out.println("4. Remind Tasks");

System.out.println("5. Delete Tasks");

System.out.println("6. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter task name: ");

String name = scanner.nextLine();

System.out.print("Enter due date and time (yyyy-MM-dd HH:mm): ");

String dueDateStr = scanner.nextLine();

System.out.print("Enter reminder date and time (yyyy-MM-dd HH:mm): ");

String reminderDateStr = scanner.nextLine();

// Convert String to Date

try {

LocalDateTime dueDate = LocalDateTime.parse(dueDateStr, dateFormat);

LocalDateTime reminderDate = LocalDateTime.parse(reminderDateStr, dateFormat);

app.addTask(name, dueDate, reminderDate);

} catch (Exception e) {

System.out.println("Invalid date format. Please use yyyy-mm-dd HH:mm.");

}

break;

case 2:

app.displayTasks();

break;

case 3:

System.out.print("Enter task number to mark as done: ");

int doneTaskNum = scanner.nextInt();

scanner.nextLine(); // consume newline

app.markTaskDone(doneTaskNum);

break;

case 4:

app.checkReminders();

break;

case 5:

System.out.print("Enter task number to delete: ");

int taskNum = scanner.nextInt();

scanner.nextLine();

app.deleteTask(taskNum);

break;

case 6:

System.out.println("Exiting...");

scanner.close();

return;

default:

System.out.println("Invalid option. Please try again.");

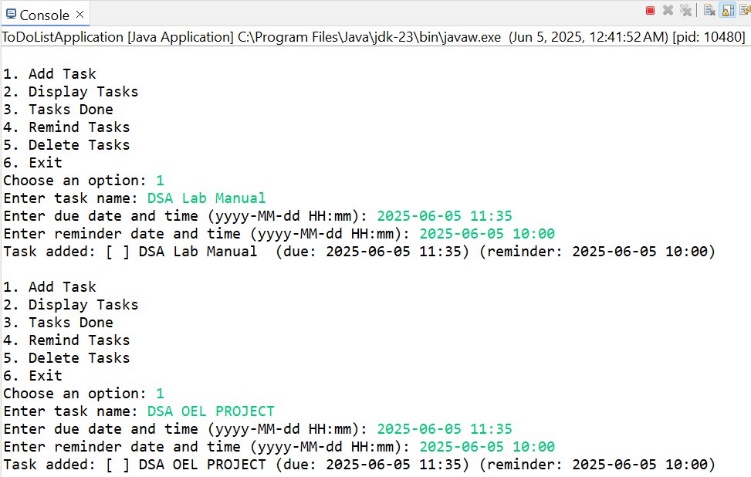
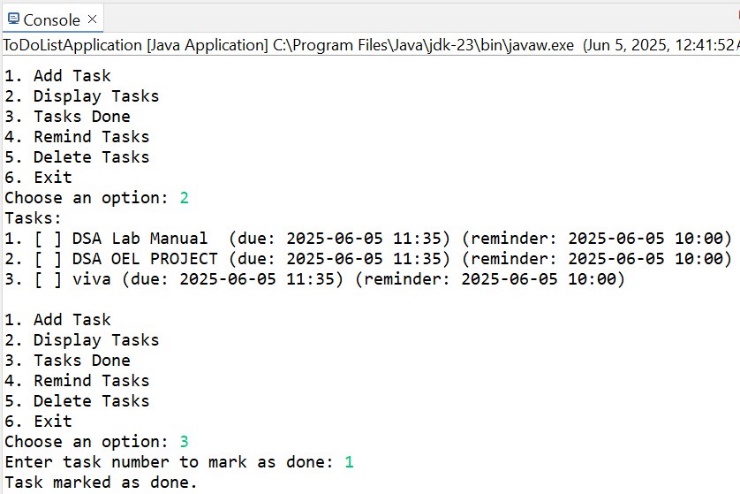
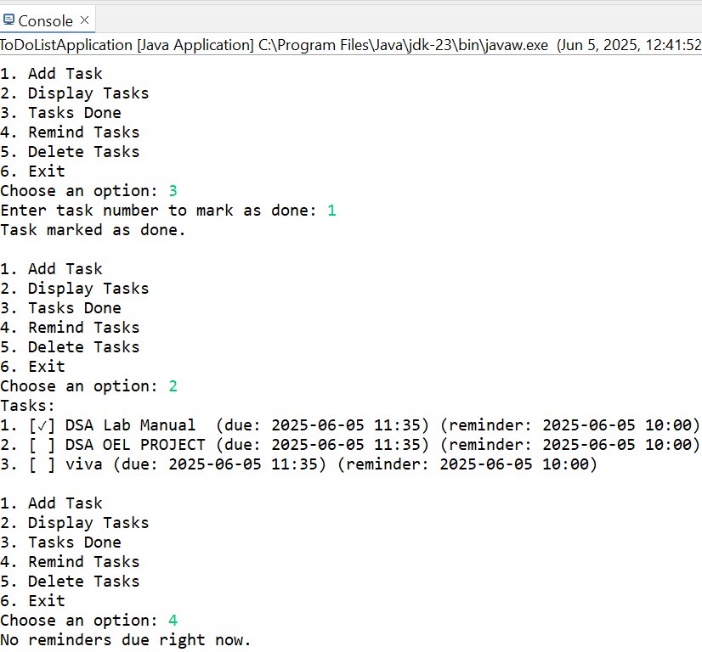
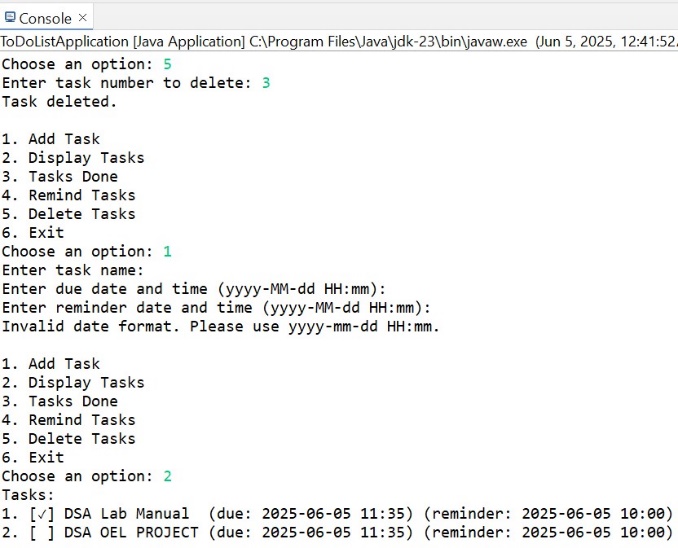
}

}

}

}

# 10. **Results**

The application successfully adds, displays, deletes, and completes tasks via command-line interface. Reminder checking also works as expected.