Sir Syed University of Engineering & Technology (SSUET)

Software Engineering Department

Course Name: Software Construction & Development (SWE-312)

Semester: 5th

Batch: 2022F

Section: B

PROJECT REPORT

Project Title: EMPLOYEE TRACKING SYSTEM

Submitted By:

STUDENTS NAMES	ROLL NUMBERS
M. Ali	**2022F-BSE-085**
Ali Jafri	**2022F-BSE-059**
Arsalan Imran	**2022F-BSE-069**
	2022F-BSE-

Submitted To: Miss Eman

TABLE OF CONTENTS

TEAM PROFILE

1-M. Ali (2022F-BSE-085)

(Development of project architecture and coding implementation. .**)**

2-Arsalan Imran (2022F-BSE-069)

(Testing and debugging to ensure smooth operation.**)**

3-Ali Jafri (2022F-BSE-059)

(Preparing project documentation, including reports, manuals, and user guides.**)**

4-Saeed(2022F-BSE-10)

(Designing user interfaces and ensuring an intuitive user experience. **)**

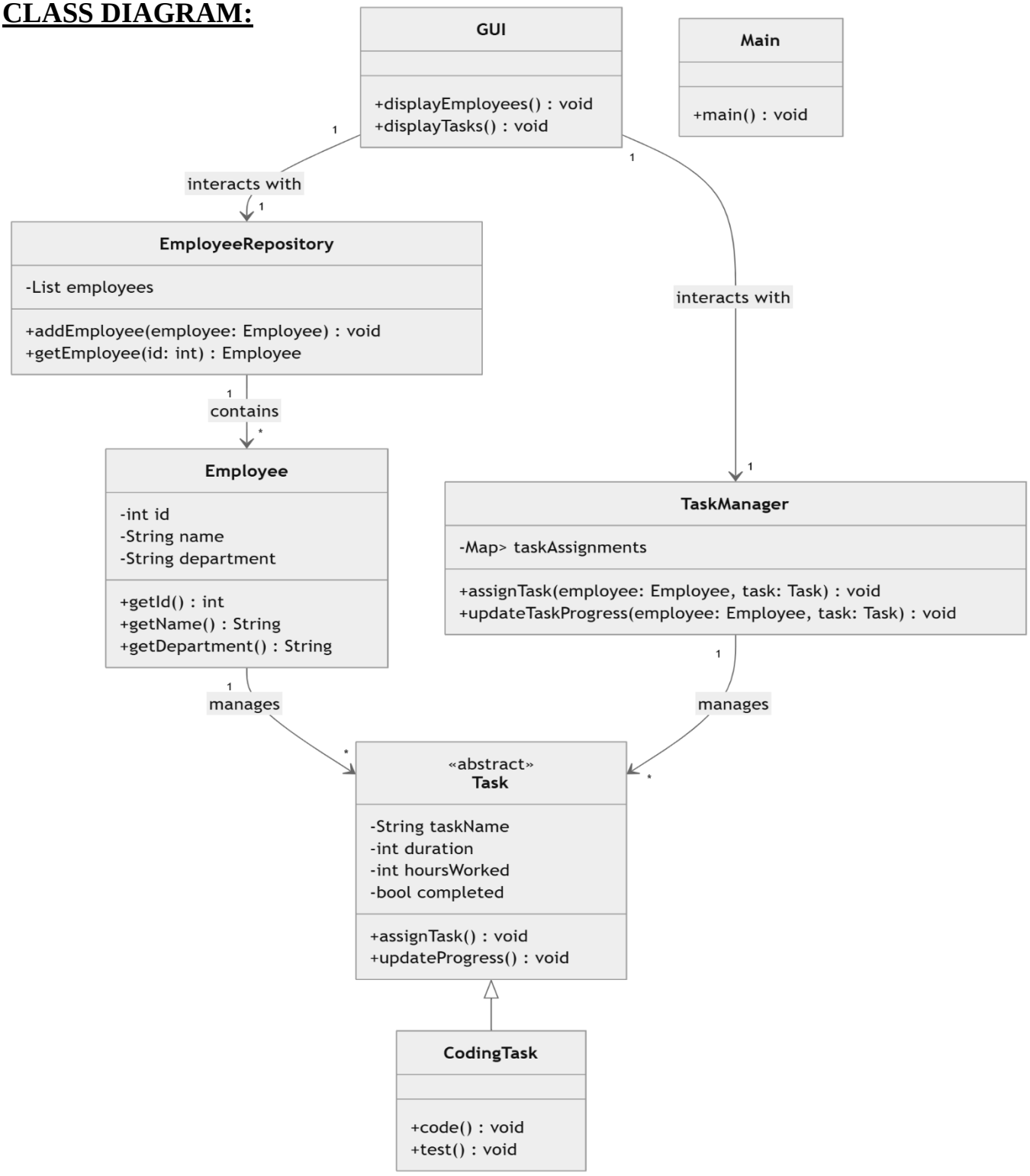
1. PROJECT DESCRIPTION:

The **Employee Tracking System** is a platform designed to manage employee tasks, monitor progress, and assess performance. It allows managers to assign tasks, track their completion, and analyze performance in real time. The system offers features like task progress tracking, employee management, and performance reporting to enhance productivity and decision-making.

OBJECTIVES:

1. **Employee Management & Task Assignment**: Centralize employee records and easily assign tasks.
2. **Task Tracking & Progress Monitoring**: Track task completion and update progress in real time.
3. **Performance Analysis**: Generate reports to evaluate employee performance and efficiency.
4. **Scalability & Multi-User Support**: Support growth with scalable features for multiple users.
5. **Task Customization & Prioritization**: Customize and prioritize tasks based on needs.
6. **Security & Data Privacy**: Ensure secure access and protect employee data.

CLASS DIAGRAM:



2.

3. THREAD SYNCHRONIZATION:

```
class TaskManager {
    private final Map<Employee, List<Task>> taskMap = new HashMap<>();

    public synchronized void assignTask(Employee employee, Task task) {
        taskMap.computeIfAbsent(employee, k -> new ArrayList<>()).add(task);
        System.out.println("Task assigned: " + task + " to " + employee);
    }

    public synchronized List<Task> getTasksForEmployee(Employee employee) {
        return taskMap.getOrDefault(employee, Collections.emptyList());
    }

    public synchronized void updateTaskProgress(Employee employee, Task task, int hours) {
        task.updateTaskProgress(hours);
    }
}
```

4. CONCEPT OF MUTABILITY & IMMUTABILITY:

```java
final class Employee {
    private final int id;
    private final String name;
    private final String department;

    public Employee(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getDepartment() {
        return department;
    }

    @Override
    public String toString() {
        return id + " - " + name + " (" + department + ")";
    }
```

5. EXCEPTIONAL HANDLING :

```
class EmployeeRepository<T> {
    private final List<Employee> employees = new ArrayList<>();

    public void addEmployee(Employee employee) {
        employees.add(employee);
    }

    public Employee findEmployeeById(int id) throws NoSuchElementException {
        return employees.stream()
                .filter(e -> e.getId() == id)
                .findFirst()
                .orElseThrow(() -> new NoSuchElementException("Employee not found"));
    }
}
```

6. UNIT TESTING:

Process order:

```
import static org.junit.Assert.*;
import org.junit.Test;

public class TaskTest {

    @Test
    public void testTaskProgressUpdate() {
        Task task = new CodingTask("Write Code", 10);  // Duration of 10 hours
        task.updateTaskProgress(5);  // Update with 5 hours

        assertEquals(5, task.getHoursWorked());  // Check if 5 hours were worked
        assertFalse(task.isCompleted());  // Task should not be completed yet

        task.updateTaskProgress(5);  // Update with another 5 hours
        assertTrue(task.isCompleted());  // Task should be completed
    }
}
```

7. DEADLOCK:

```
class DeadlockExample {
    private final Object employeeLock = new Object();
    private final Object taskLock = new Object();

    public void thread1Method() {
        synchronized (employeeLock) {
            System.out.println("Thread 1: Locked Employee");
            synchronized (taskLock) {
                System.out.println("Thread 1: Locked Task");
            }
        }
    }

    public void thread2Method() {
        synchronized (taskLock) {
            System.out.println("Thread 2: Locked Task");
            synchronized (employeeLock) {
                System.out.println("Thread 2: Locked Employee");
            }
        }
    }
}
```

8. ABSTRACT DATATYPE AND GENERICS (List):

List:

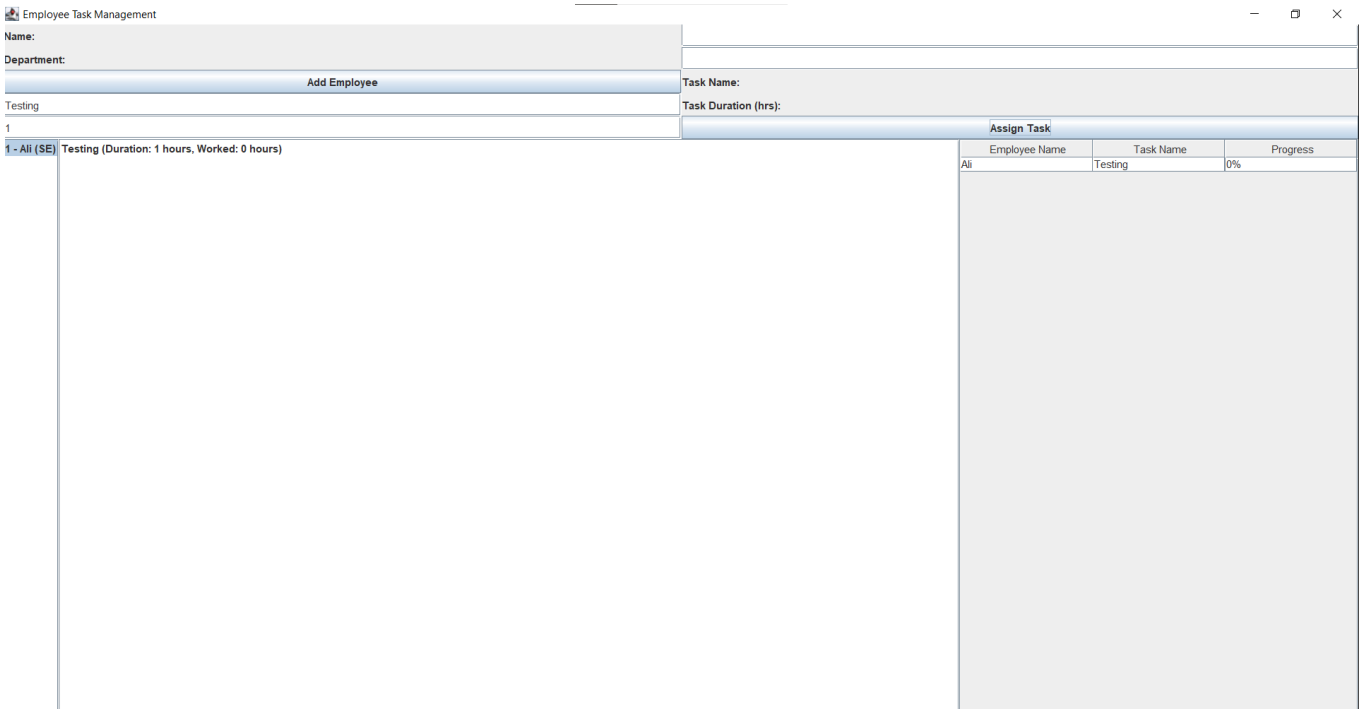
```
class EmployeeRepository<T> {
    private final List<Employee> employees = new ArrayList<>();  // List used for storing
}
```

Map:

```
// Map in TaskManager
class TaskManager {
    private final Map<Employee, List<Task>> taskMap = new HashMap<>();  // Map stores task

    public void assignTask(Employee employee, Task task) {
        taskMap.computeIfAbsent(employee, k -> new ArrayList<>()).add(task);
    }
}
```

9. USER GUIDE:


Employee Task Management
Name:
Department:
Add Employee
Testing
1
1 - Ali (SE)
Testing (Duration: 1 hours, Worked: 0 hours)
Task Name:
Task Duration (hrs):
Assign Task
Employee Name
Task Name
Progress
Ali
Testing
0%

10.FUTURE EXPANSION

For future growth and scalability of the Employee Tracking System, the following features could be considered:

1. **Mobile Application Integration**: Develop a mobile version of the platform for on-the-go access by employees, managers, and administrators.
2. **Advanced Analytics & Reporting**: Integrate data analytics to track employee performance, task efficiency, and team productivity over time.
3. **Real-Time Notifications**: Add real-time push notifications to notify employees and managers about task updates, deadlines, and status changes.
4. **Role-Based Access Control**: Introduce different user roles (e.g., Admin, Manager, Employee) with specific permissions to enhance security and usability.
5. **AI-Powered Task Prioritization**: Implement AI to automatically prioritize tasks based on deadlines, employee skills, and availability.
6. **Integration with Other Tools**: Support integration with other business tools like project management software (e.g., Jira, Asana) and communication platforms (e.g., Slack).
7. **Cloud-Based Storage**: Migrate the system to the cloud for better scalability, data backup, and remote access.
8. **Multi-Language Support**: Implement multi-language support to cater to a diverse workforce.