1

A

PROJECT REPORT

ON

DATABASE MANAGEMENT SYSTEM OF HOME RENOVATION PLANNE

submitted in partial fulfil of the requirement for ­­­­­­­­award of the Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE SPECIALIZED IN ARTIFICIAL INTELLIGENCE

AND

MACHINE LEARNING

## By

**K.NIKITHA 21AG1A6696**

**M.SREEJA 21AG1A6696**

**P.KAVYA 21AG1A66A6**

Under the guidance of

## Mr.Krishnanandh

Professor



**DEPARTMENT OF COMPUTER SCIENCE** -**AIML**

**ACE**

**Engineering College**

UGC AUTONOMOUS INSTITUTION

### Accredited with NBA & NAAC „A‟ Grade

(Approved by AICTE and Affiliated to Jawaharlal Nehru Technological University, Hyderabad) **Ankushapur (V), Ghatkesar (M), Medchal– Malkajgiri Dist.**

**501 301 T.S. In**

2

**ACKNOWLEDGEMENT**

We would like to thank our guide **Mr.Krishnanadh** ,**Professor** for providing valuable guidance throughout this project.

We would like to express our gratitude towards **Mr.T.Srinivas Rao, Associate Professor** & Head of the Training and Placements Department, ACE Engineering College, Ghatkesar, Hyderabad for allowing us to do this project.

We would like to thank **Dr. BL RAJU** Sir, **Principal**, of **ACE Engineering College**, Ghatkesar, Hyderabad for giving an opportunity to carry out the project.

We would like to express our sincere thanks to the **Management of ACE Engineering College** for their kind cooperation and timely help during the course of project.

We would like to express our sincere thanks to all Staff Members of **Training and Placement** department, ACE Engineering College for their kind cooperation and timely help during the course of our major project.

We would also express our deep sense of gratitude to our beloved parents for their concern and anxiety with regards to the successful completion of this project.

### 

**K.NIKITHA 21AG1A6696**

**M.SREEJA 21AG1A6699**

**P.KAVYA 21AG1A66A6**

**ABSTRACT**

3

The Home Renovation Planner Project aims to develop a comprehensive database system to assist homeowners, contractors, and designers in managing home renovation projects efficiently. Leveraging SQL Server, we’ll create a robust solution that allows users to track project details, costs, timelines, and materials.

**Key Features:**

**Project Management:** Users can create and manage renovation projects, including project names, descriptions, and associated properties.

**Task Tracking:** The system will enable users to define renovation tasks (e.g., painting, flooring, plumbing) and assign them to specific contractors or team members.

**Budget Management:** Users can set project budgets, track expenses, and receive alerts when costs exceed predefined limits.

**Material Inventory:** Maintain a catalog of construction materials, their prices, and suppliers.

**Schedule:** Display project timelines, milestones, and critical dates.

**User Authentication and Authorization:** Secure access to project data based on user roles (e.g., homeowner, contractor, administrator).

**Technical Implementation:**

**Database Design:** We’ll create tables for project, task, material, client, contractor, budget, schedule, invoice, review, warranty. Relationships (e.g., one-to-many) will be established.

**Views:** Utilize SQL Server views to simplify data presentation and restrict access to sensitive information.

**Stored Procedures:** Develop stored procedures for common actions (e.g., adding tasks, updating budgets).

**Security Measures:** Apply appropriate permissions to ensure data integrity and prevent unauthorized access.

By abstracting core data access through views, stored procedures, and UDFs, we maintain flexibility for future enhancements while minimizing security risks. This project will empower homeowners and professionals alike, streamlining the home renovation process.

4

**INDEX**

**S.NO CONTENT PAGE NO**

1. TITLE……………………………………… 1

2. INTRODUCTION…………………………. 2

3. ABSTRACT………………………………. 3

4. FUNCTIONAL REQUIREMENT ………... 5-6

5. ER DIAGRAM…………………………….. 7-9

6. QUERIES TO CREATE DATABASE…….. 10-15

7. UML ATTRIBUTE AND …………………. 16

METHOD DIAGRAM

8. ALL CLASS JAVA CODE FILES………… 17-23

9. CHALLENGES LIST…………………….. 23

**5**

**FUNCTIONAL REQUIREMENTS**

Below are some functional requirements categorized by different aspects of the home renovation process:

**Project Management**

* **Create Project:** The system should allow users to create a new renovation project with details such as project name, address, start date, and estimated end date.
* **Task Management:** Users should be able to create, assign, and track tasks related to the renovation. Each task should include a description, due date, assigned personnel, and status.
* **Milestone Tracking:** The system should support the creation and tracking of major milestones in the renovation project, such as "Demolition Complete" or "Plumbing Installed."

**Budget and Cost Management**

* **Expense Tracking:** The system should allow users to log expenses, categorize them (e.g., materials, labor), and track them against the budget.
* **Budget Creation:** Users should be able to set an initial budget for the renovation project.
* **Budget Updates:** Users should be able to update the budget as needed, with a log of changes for tracking purposes.
* **Financial Reports:** Generate reports showing budget vs. actual expenses, and highlight any variances.

**Scheduling and Timeline**

* **Calendar Integration:** The system should integrate with a calendar to show the timeline of tasks and milestones.
* **Gantt Chart:** Provide a Gantt chart view to visualize the project schedule, dependencies, and progress.

**Resource Management**

* **Contractor Management:** Store and manage information about contractors, including contact details, availability, and assigned tasks.
* **Material Inventory:** Track materials required for the project, including quantities, costs, and current stock levels.

6

* **Resource Allocation:** Assign resources (both human and material) to specific tasks and track their usage.

**Communication and Collaboration**

* **Messaging System:** Enable communication between project stakeholders, such as homeowners, contractors, and suppliers.
* **Document Sharing:** Allow users to upload, share, and access documents related to the project, such as blueprints, permits, and invoices.
* **Commenting:** Users should be able to comment on tasks, milestones, and documents to facilitate discussion and clarification.

**Quality Control and Inspections**

* **Inspection Scheduling:** Schedule inspections for different stages of the renovation (e.g., after plumbing, electrical work).
* **Inspection Reports:** Log inspection reports, including findings, recommendations, and any required follow-up actions.
* **Quality Checklists:** Provide checklists for different phases of the renovation to ensure quality standards are met.

**User Roles and Permissions**

* **Role-Based Access:** Define different user roles (e.g., homeowner, project manager, contractor) with specific permissions.
* **Access Control:** Control access to different parts of the system based on user roles, ensuring sensitive information is only accessible to authorized users.

**Reporting and Analytics**

* **Progress Reports:** Generate regular reports on project progress, highlighting completed tasks, pending tasks, and any delays.
* **Cost Analysis:** Provide detailed cost analysis reports to help users understand where the budget is being spent.

**System Integration**

* **Third-Party Tools:** Integrate with third-party tools for accounting, calendar management, and communication.
* **Mobile Access:** Provide a mobile application or mobile-friendly interface for users to access the system on the go.
* **Backup and Restore:** Ensure the system can back up project data regularly and restore it in case of data loss.

7

**HOME RENOVATION PLANNER**

**ENTITY-RELATIONSHIP ATTRIBUTES:**

The main entities are Project, Contractor, Client, Task, Budget, Schedule, Material, and Warranty:

**Description of the entities and their relationships:**

● A project can have multiple tasks (1 to many with Task).

● A contractor can work on multiple projects (many to many with Project).

● A client can initiate multiple projects (1 to many with Project). ● A task belongs to one project (many to 1 with Project).

● A project has one budget (1 to 1 with Project).

● A task has one schedule (1 to 1 with Task).



● A project can have multiple materials (1 to many with Material). ● Amaterial can have one warranty (1 to 1 with Material)



❖ **Attributes:**

➢ Project:



● project\_name

● start\_date



● end\_date



* client\_id (FK)



* Budget\_id

➢ Contractor:



● contractor\_id (PK)

● contractor\_name

● contact\_info

8

➢ Client:

● client\_id (PK)

● client\_name

● contact\_info

➢ Task:

● task\_id (PK)

● task\_name

● project\_id (FK)

● contractor\_id (FK)

➢ Budget:

● budget\_id (PK)

● project\_id (FK)

● total\_budget

➢ Schedule:

schedule\_id (PK)



● project\_id (FK)

● task\_id (FK)

● start\_date

● End\_date

➢ Material :

● material\_id (PK)

● material\_name

● quantity

● project\_id (FK)

➢ Warranty:

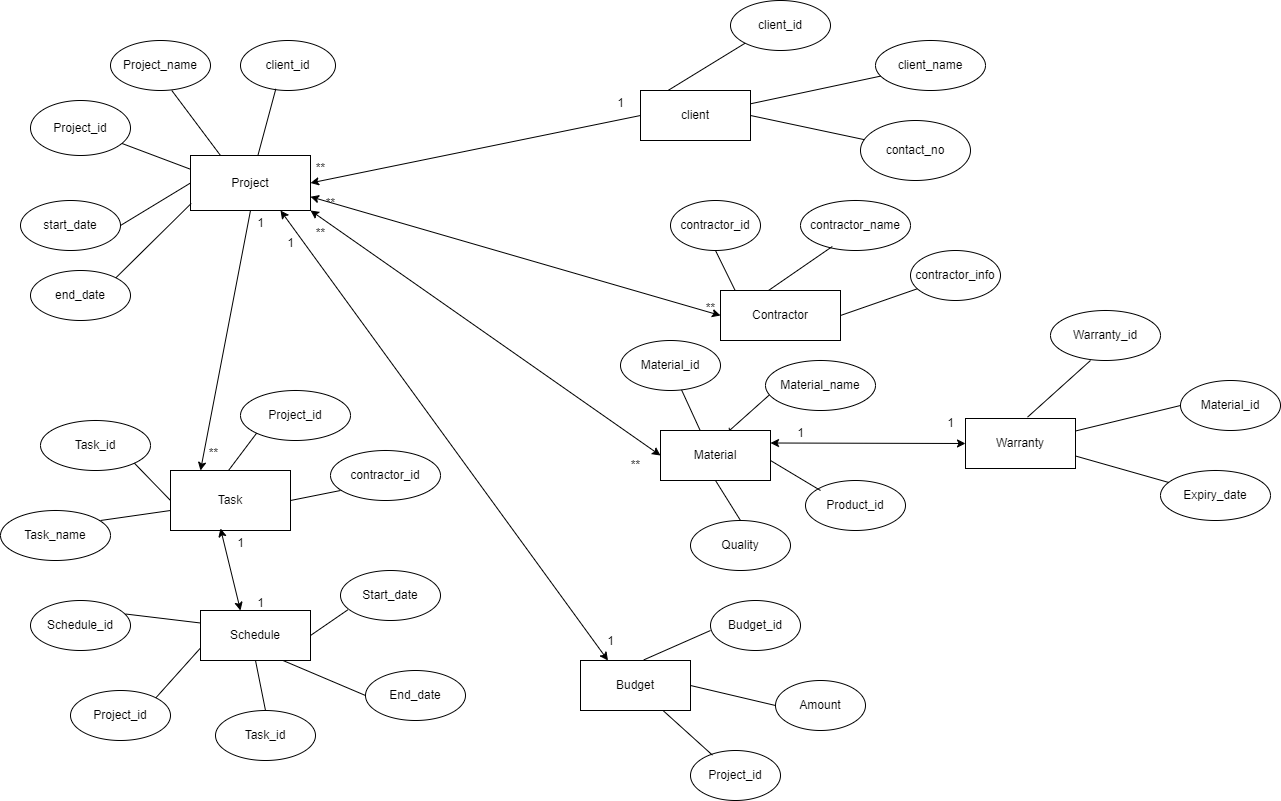
● warranty\_id (PK)

● project\_id (FK)

● warranty\_period

9

**ER-DIAGRAM**



10

**QUERIES TO CREATE TABLE DATABASE**

* create database homerenovation;

use homerenovation;

* create table project(project\_idint(10) primary key,project\_namevarchar(50), start\_end datetime,end\_datedatetime,client\_id int(20),budget\_id int(15));

* insert into project values(1,"home renovation",'2024-04- 13 7:30:00','2024-08-13 7:30:00',211,321);

insert into project values(2,"building",'2024-01-01 8:00:00','2024-05-01 8:00:00',212,322); insert into project values(3,"cardboard",'2024-05-24 12:34:45','2024-09-213 14:30:47',213,323);

insert into project values(4,"new home",'2024-06-21 14:30:24','2024-11-23 17:30:00',214,324);

insert into project values(5,"renowed home",'2024-02-24 12:30:23','2024-12-20 12:00:00',215,325);

insert into project values(6,"tiles",'2024-04-10 7:00:00','2024-08-17 7:30:00',216,326);

insert into project values(7,"wall design",'2024-02-13 12:30:00','2024-04-13 16:30:00',217,327);

insert into project values(8,"interior",'2024-04-03 9:50:00','2024-12-23 8:40:00',218,328);

insert into project values(9,"dream house",'2024-04-09 4:50:00','2024-011-13 22:00:00',219,329);

insert into project values(10,"home",'2024-06-24 23:30:00','2024-08-23 6:30:00',220,330);

select \* from project;

11

* create table contractor(contractor\_id int(10) primary key,contractor\_name varchar(40),contact\_info int);

insert into contractor values(1,"Akhil",987654321);

insert into contractor values(2,"swetha",897654362);

insert into contractor values(3,"pranathi",786543219);

insert into contractor values(4,"kavya",984563721);

insert into contractor values(5,"swathi",997654321);

insert into contractor values(6,"Harsha",897654321);

insert into contractor values(7,"Abhilash",78906543);

insert into contractor values(8,"Rahul",77654321);

insert into contractor values(9,"nikhil",987655678);

insert into contractor values(10,"Nikitha",988765890);

select \* from contractor;

* create table Clients(client\_id int primary key,client\_name varchar(50),contact\_info int);

insert into Clients values(211,"Bhanu",89765);

insert into Clients values(212,"Yamini",876533);

insert into Clients values(213,"Bhavana",9886645);

insert into Clients values(214,"Shruthi",676448);

insert into Clients values(215,"Abhi",775298);

insert into Clients values(216,"Yash",894873);

insert into Clients values(217,"Sreeja",75380);

insert into Clients values(218,"Anil",83987497);

insert into Clients values(219,"vedha",544276);

insert into Clients values(220,"chinni",9855328);

select \* from Clients;

* create table Task(task\_id int(10) primary key,task\_name varchar(50),project\_id int(10),foreign key(project\_id) references project(project\_id),contractor\_id

12

int(10),foreign key(contractor\_id) references contractor(contractor\_id));

select \* from Task;

insert into Task values(10,"A",1,1);

insert into Task values(20,"b",2,2);

insert into Task values(30,"c",3,3);

insert into Task values(40,"d",4,4);

insert into Task values(50,"e",5,5);

insert into Task values(60,"f",6,6);

insert into Task values(70,"g",7,7);

insert into Task values(80,"h",1,8);

insert into Task values(90,"A",2,3);

insert into Task values(100,"h",4,6);

select \* from Task;

* create table budget(budget\_id int(10) primary key,project\_id int(20),foreign key(project\_id) references project(project\_id),total\_budget int(20));

insert into budget values(321,1,10000);

insert into budget values(322,1,20000);

insert into budget values(323,2,30000);

insert into budget values(324,3,40000);

insert into budget values(325,4,50000);

insert into budget values(326,5,60000);

insert into budget values(327,6,70000);

insert into budget values(328,7,80000);

insert into budget values(329,8,90000);

insert into budget values(330,3,90000);

select \* from budget;

13

* create table Schedules(schedule\_id int(10) primary key,project\_id int(10),foreign key(project\_id) references project(project\_id),task\_id int(10),foreign key(task\_id) references Task(task\_id),start\_date date,end\_date date);

insert into Schedules values(101,1,20,'2023-01-11','2024-01-11');

insert into Schedules values(102,1,30,'2023-01-12','2024-01-21');

insert into Schedules values(103,1,40,'2023-01-15','2024-01-28');

insert into Schedules values(104,1,50,'2023-01-15','2024-01-15');

insert into Schedules values(105,1,60,'2023-01-17','2024-01-19');

insert into Schedules values(106,1,70,'2023-01-18','2024-01-13');

insert into Schedules values(107,1,80,'2023-01-12','2024-01-14');

insert into Schedules values(108,1,90,'2023-01-19','2024-01-17');

insert into Schedules values(109,1,10,'2023-01-18','2024-01-12');

insert into Schedules values(110,1,20,'2023-01-11','2024-01-16');

select \* from Schedules;

* create table material(material\_id int(10) primary key,material\_name varchar(50), quantity varchar(20),project\_id int(10),foreign key(project\_id) references project(project\_id));

insert into material values(301,"mud","10kg",1);

insert into material values(302,"bricks","50kg",2);

14

insert into material values(303,"steals","10kg",3);

insert into material values(304,"plastic","40kg",4);

insert into material values(305,"taps","20kg",5);

insert into material values(306,"cement","30kg",6);

insert into material values(307,"iron","10kg",7);

insert into material values(308,"grills","60kg",7);

insert into material values(309,"carpet","40kg",8);

insert into material values(310,"stome","20kg",9);

select \* from material;

* create table warranty(waranty\_id int(10) primary key,project\_id int(10),foreign key(project\_id) references project(project\_id),warranty\_period varchar(20));

insert into warranty values(201,1,"1yr");

insert into warranty values(202,2,"6months");

insert into warranty values(203,6,"2yr");

insert into warranty values(204,4,"8months");

insert into warranty values(205,8,"2months");

insert into warranty values(206,9,"1month");

insert into warranty values(207,3,"3months");

insert into warranty values(208,8,"6months");

insert into warranty values(209,2,"7months");

insert into warranty values(210,6,"2months");

select \* from warranty;

* create table invoice(invoice\_id int(10) primary key,project\_id int(100), foreign key(project\_id) references project(project\_id),amount int(20) );

insert into invoice values(1,101,5000);

insert into invoice values(2,102,6000);

insert into invoice values(3,103,2000);

insert into invoice values(4,104,3000);

15

insert into invoice values(5,105,9000);

insert into invoice values(6,106,8000);

insert into invoice values(7,107,4000);

insert into invoice values(8,108,15000);

insert into invoice values(9,109,1000);

insert into invoice values(10,110,1200);

* create table review(review\_id int(10) primary key,project\_id int(100), foreign key(project\_id) references project(project\_id),rating int(10),comments varchar(50) );

insert into review values(1,101,10,"very excellent");

insert into review values(2,102,9,"excellent");

insert into review values(3,103,10,"very excellent");

insert into review values(4,104,1,"very bad");

insert into review values(5,105,8,"very good");

insert into review values(6,106,7,"good");

insert into review values(7,107,9,"excellent");

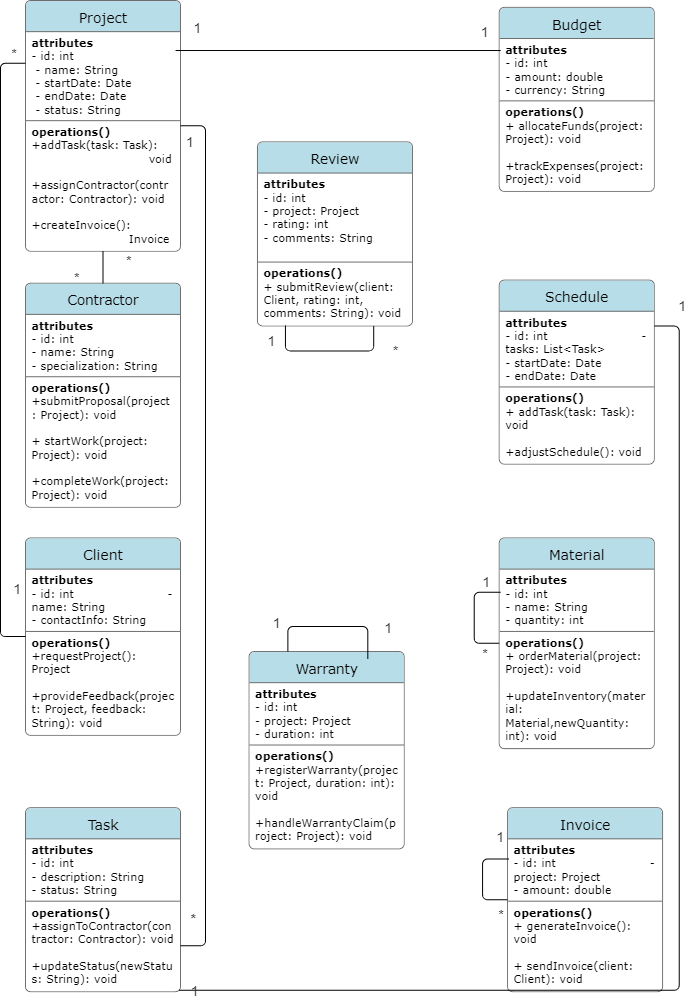
insert into review values(8,108,1,"very bad");

insert into review values(9,109,8,"very good");

insert into review values(10,110,7,"good");

16

­­­ **UML METHOD DIAGRAM**



17

**ALL CLASS JAVA CODE FILES**

* Database Connection Class:

This class manages the connection to the database.

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final string

URL="jdbc:mysql://localhost:3306/home\_renovation";

private static final String USER = "username";

private static final String PASSWORD = "password";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

}

* Project Class

This class represents a renovation project.

public class Project {

private int id;

private String name;

private String description;

private double budget;

18

public Project(int id, String name, String description, double budget) {

this.id = id;

this.name = name;

this.description = description;

this.budget = budget;

}

// Getters and setters

}

* Task Class

This class represents a task within a project.

public class Task {

private int id;

private int projectId;

private String name;

private String description;

private String status;

public Task(int id, int projectId, String name, String description, String status) {

this.id = id;

this.projectId = projectId;

this.name = name;

this.description = description;

this.status = status;

}

// Getters and setters

19

}

* ProjectDAO Class

This class handles database operations for projects.

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class ProjectDAO {

public void addProject(Project project) throws SQLException {

String query = "INSERT INTO projects (name, description, budget) VALUES (?, ?, ?)";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, project.getName());

stmt.setString(2, project.getDescription());

stmt.setDouble(3, project.getBudget());

stmt.executeUpdate();

}

}

public List<Project> getAllProjects() throws SQLException {

List<Project> projects = new ArrayList<>();

String query = "SELECT \* FROM projects";

try (Connection connection = DatabaseConnection.getConnection();

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

while (rs.next()) {

20

Project project = new Project(

rs.getInt("id"),

rs.getString("name"),

rs.getString("description"),

rs.getDouble("budget")

);

projects.add(project);

}

}

return projects;

}

// Other CRUD operations

}

* TaskDAO Class

This class handles database operations for tasks.

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class TaskDAO {

public void addTask(Task task) throws SQLException {

String query = "INSERT INTO tasks (project\_id, name, description, status) VALUES (?, ?, ?, ?)";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setInt(1, task.getProjectId());

stmt.setString(2, task.getName());

21

stmt.setString(3, task.getDescription());

stmt.setString(4, task.getStatus());

stmt.executeUpdate();

}

}

public List<Task> getTasksByProjectId(int projectId) throws SQLException {

List<Task> tasks = new ArrayList<>();

String query = "SELECT \* FROM tasks WHERE project\_id = ?";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setInt(1, projectId);

try (ResultSet rs = stmt.executeQuery()) {

while (rs.next()) {

Task task = new Task(

rs.getInt("id"),

rs.getInt("project\_id"),

rs.getString("name"),

rs.getString("description"),

rs.getString("status")

);

tasks.add(task);

}

}

}

return tasks;

}

22

// Other CRUD operations

}

* Main Class

This class contains the main method and ties everything together.

import java.sql.SQLException;

import java.util.List;

public class Main {

public static void main(String[] args) {

ProjectDAO projectDAO = new ProjectDAO();

TaskDAO taskDAO = new TaskDAO();

try {

// Add a new project

Project project = new Project(0, "Kitchen Remodel", "Renovate the kitchen", 10000);

projectDAO.addProject(project);

// Fetch all projects

List<Project> projects = projectDAO.getAllProjects();

for (Project p : projects) {

System.out.println("Project ID: " + p.getId() + ", Name: " + p.getName());

// Add tasks to the project

Task task = new Task(0, p.getId(), "Buy appliances", "Purchase new kitchen appliances", "Pending");

taskDAO.addTask(task);

// Fetch tasks for the project

23

List<Task> tasks = taskDAO.getTasksByProjectId(p.getId());

for (Task t : tasks) {

System.out.println("\tTask ID: " + t.getId() + ", Name: " + t.getName());

}

}

} catch (SQLException e) {

e.printStackTrace();

       }

    }

}

**CHALLENGES LIST**

* Designing a normalised and efficient database schema to handle complex relationships like many-to-many (e.g., Project-Material , Project-Contractor, etc).
* Creating the tables in a specific order and inserting the values accordingly.
* Implementing and managing the various one-to-many and many-to-many relationships in Java objects and ensuring data integrity.

\* \* \*