```
package projects.dao;

import provided.util.DaoBase;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import projects.exception.DbException;
import projects.entity.Material;
import projects.entity.Project;
import projects.entity.Step;

import java.sql.SQLException;
import java.util.LinkedList;
import java.util.List;
import java.util.Objects;
import java.util.Optional;

import jdk.jfr.Category;



public class ProjectDao extends DaoBase {
        private static final String CATEGORY_TABLE = "category";
        private static final String MATERIAL_TABLE = "material";
        private static final String PROJECT_TABLE = "project";
        private static final String PROJECT_CATEGORY_TABLE = "project_category";
        private static final String STEP_TABLE = "step";

public Project insertProject(Project project) {
        String sql = ""
                        + "INSERT INTO " + PROJECT_TABLE + " "
                        + "(project_name, estimated_hours, actual_hours, difficulty,
notes)"
                        + " VALUES "
                        + "(?, ?, ?, ?, ?)";
        try(Connection conn = DbConnection.getConnection()){
                startTransaction(conn);

                try(PreparedStatement stmt = conn.prepareStatement(sql)){
                        setParameter(stmt, 1, project.getProjectName(),
String.class);
                        setParameter(stmt, 2, project.getEstimatedHours(),
BigDecimal.class);
                        setParameter(stmt, 3, project.getActualHours(),
BigDecimal.class);
                        setParameter(stmt, 4, project.getDifficulty(),
Integer.class);
                        setParameter(stmt, 5, project.getNotes(), String.class);
```

```
                        stmt.executeUpdate();
                        Integer projectID = getLastInsertId(conn, PROJECT_TABLE);
                        commitTransaction(conn);

                        project.setProjectId(projectID);
                        return project;
                }
                catch(Exception e) {
                        rollbackTransaction(conn);
                        throw new DbException(e);
                }
        }
        catch(SQLException e) {
                throw new DbException(e);
        }
}

public List<Project> fetchAllProjects() {
        String sql = "SELECT * FROM" + PROJECT_TABLE + " ORDER BY project_name";
        try(Connection conn = DbConnection.getConnection()) {
                startTransaction(conn);

                try(PreparedStatement stmt = conn.prepareStatement(sql)){
                        try(ResultSet rs = stmt.executeQuery()){
                                List<Project> projects = new LinkedList<>();

                                while(rs.next()) {
                                        projects.add(extract(rs, Project.class));
                                }
                                return projects;
                        }
                }
                catch(Exception e) {
                        rollbackTransaction(conn);
                        throw new DbException (e);
                }
        }
        catch(SQLException e) {
                throw new DbException(e);
        }
}
public Optional<Project> fetchProjectById(Integer projectId){
        String sql = "SELECT * FROM" + PROJECT_TABLE + "WHERE project_id = ?";

        try(Connection conn = DbConnection.getConnection()) {
                startTransaction(conn);

                try {
                        Project project = null;
```

```
                        try(PreparedStatement stmt = conn.prepareStatement(sql)){
                                setParameter(stmt, 1, projectId, Integer.class);

                                try(ResultSet rs = stmt.executeQuery()) {
                                        if(rs.next()) {
                                                project = extract(rs,
Project.class);
                                        }
                                }
                        }
                        if(Objects.nonNull(project)) {

project.getMaterials().addAll(fetchMaterialsForProject(conn, projectId));
                                project.getSteps().addAll(fetchStepsForProject(conn,
projectId));

project.getCategories().addAll(fetchCategoriesForProject(conn, projectId));
                        }
                        commitTransaction(conn);
                        return Optional.ofNullable(project);
                }
                catch(Exception e) {
                        rollbackTransaction(conn);
                        throw new DbException(e);
                }
        }
        catch(SQLException e) {
                throw new DbException(e);
                }
}


private List<Category> fetchCategoriesForProject(Connection conn,Integer projectId)
throws SQLException {
        // @formatter:off
        String sql = " "
                        + "SELECT c. * FROM " + CATEGORY_TABLE + " c "
                        + "JOIN " + PROJECT_CATEGORY_TABLE + " pc USING
(category_id) "
                        + "WHERE project_id = ?";
        // @formatter:on

        try(PreparedStatement stmt = conn.prepareStatement(sql)) {
                setParameter(stmt, 1, projectId, Integer.class);

                try(ResultSet rs = stmt.executeQuery()){
                        List<Category> categories = new LinkedList<>();

                        while(rs.next()) {
                                categories.add(extract(rs,Category.class));
```

```
                        }
                        return categories;
                }

        }

}



private List<Step> fetchStepsForProject(Connection conn, Integer projectId) throws
SQLException {
        String sql = "SELECT * FROM " + STEP_TABLE + " WHERE project_id = ? ";

        try(PreparedStatement stmt = conn.prepareStatement(sql)) {
                setParameter(stmt, 1, projectId, Integer.class);

                try(ResultSet rs = stmt.executeQuery()){
                        List<Step> steps = new LinkedList<>();

                        while(rs.next()) {
                                steps.add(extract(rs, Step.class));



                        }
                        return steps;
                }
        }
}

private List<Material> fetchMaterialsForProject (Connection conn, Integer projectId)
throws SQLException {
                        String sql = "SELECT * FROM " + MATERIAL_TABLE + " WHERE
project_id = ? ";

                        try(PreparedStatement stmt = conn.prepareStatement(sql)) {
                                setParameter(stmt, 1, projectId, Integer.class);

                                try(ResultSet rs = stmt.executeQuery()){
                                        List<Material> materials = new
LinkedList<>();

                                        while(rs.next()) {
                                                materials.add(extract(rs,
Material.class));
```

```
                                        }
                                        return materials;
                                    }
                                }
}
}
```