Case Study on project Management System

Instructions

- Project submissions should be done through the partcipants' Github repository, and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a comprehensive **Project management** implemented with a strong focus on **SQL**, **control flow statements, loops, arrays, collections, exception handling**, **database interaction** and **Unit Testing**.
- Follow **object-oriented principles** throughout the project. Use classes and objects to model real-world entities, **encapsulate data and behavior**, and **ensure code reusability**.
- Throw **user defined exceptions** from corresponding methods and handled**.**
- The following **Directory structure** is to be followed in the application.
    - **entity/model**
        - Create entity classes in this package. All entity class should not have any business logic.
    - **dao**
        - Create Service Provider interface to showcase functionalities.
        - Create the implementation class for the above interface with db interaction.
    - **exception**
        - Create user defined exceptions in this package and handle exceptions whenever needed.
    - **util**
        - Create a **DBPropertyUtil** class with a static function which takes property file name as parameter and returns connection string.
        - Create a **DBConnUtil** class which holds **static method** which takes connection string as parameter file and returns **connection object(Use method defined in DBPropertyUtil class to get the connection String)**.
    - **main**
        - Create a class MainModule and demonstrate the functionalities in a menu driven application.

Key Functionalities:

1. **Project Management:** manager can add, view, update, and delete projects
2. **Project allocations: employees can be alloted to various projects**
3. **Taks addition:** Tasks can be added to each project with allocation date and deadline date
4. **Reports Generation:** Users can generate reports for their expenses over specific time periods.
5. **Database Connectivity:** Data will be stored in a relational database to ensure persistence.

Create following tables in SQL Schema with appropriate class and write the unit test case for the Project Management application.

Schema Design:

Employee:

- id (Primary Key)

- name
- Designation
- Gender
- Salary
- Project_id((Foreign Key referencing Projects table)

Project

- Id (Primary Key)
- ProjectName
- Description
- Start date
- Status(started/dev/build/test/deployed)

Task:

- task_id (Primary Key)
- task_name
- project_id Foreign Key referencing Projects table)
- employee id Foreign Key referencing Employee table)
- Status (Assigned, started, completed)

Explanation:

- The **Employee** table stores information about details of employees
- The **Project** table contains details project and optional description.
- The **Task** table stores details of task and data about which project it comes under and employees assigned with that taks along with status .

Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)

2. **Service Provider Interface/Abstract class:**
   Keep the interfaces and implementation classes in package dao
   - Define **IProjectRepository** interface/abstract class with methods for adding/removing products to/from the cart and placing orders. The following methods will interact with database.
     1. **createEmployee()**
        parameter: **Employee** emp
        return type: boolean
     2. **createProject()**
        parameter: Project pj
        return type: Boolean
     3. **createTask()**
        Parameter: Project pj
        Return type: Boolean
     4. assignProjectToEmployee()
        Parameter: int projectId, int employeeId
        Return type: Boolean

     5.AssigntaskInProjecttoEmployee()

Parameter: int taskid, int projectid, int employeeId
Return type: Boolean

6. **deleteEmployee()**
   parameter: userId
   return type: boolean
   1. **deleteProject()**
      parameter: projectId
      return type: boolean
   2. **getAllTasks()** list all Taks in a project assigned to an employeee
      parameter: **empId,projectId**
      return type: list of expenes
7. Implement the above interface in a class called **ProjectRepositoryImpl in package dao**.

Connect your application to the SQL database:

8. Write code to establish a connection to your SQL database.
   - Create a utility class **DBConnection** in a package **util** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.
   - Connection properties supplied in the connection string should be read from a property file.
   1. Create a utility class **PropertyUtil** which contains a static method named **getPropertyString()** which reads a property file containing connection details like hostname, dbname, username, password, port number and returns a connection string.
9. Create the exceptions in package **myexceptions** and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,
   1. **EmployeeNotFoundException**: throw this exception when user enters an invalid user id which doesn't exist in db
   2. **ProjectNotFoundException**: throw this exception when user enters an invalid product id which doesn't exist in db
10. Create class named **ProjectApp** with main method in app Trigger all the methods in service implementation class by user choose operation from the following menu.
    1. Add Employee.
    2. Add Project.
    3. Add Task
    4. Assign project to employee
    5. Assign task within a project to employee
    6. Delete Employee.
    7. Delete task
    8. List all projects assigned with tasks to an employeee

Unit Testing

11. Create Unit test cases for **Project System** essential to ensure the reliability of your system. Following questions to guide the creation of Unit test cases:
    1. Write test case to test if employee created successfully
    2. Write test case to test if task is created successfully.

3. Write test case to test search for projects and taskas assigned to employee
4. write test case to test if the exceptions are thrown correctly based on scenario