Coding Challenges: CareerHub, The Job Board

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 application implemented with a strong focus on SQL, control flow statements, loops, arrays, collections, exception handling, database interaction.
- 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/Absract Class to showcase functionalities.
        - Create the implementation class for the above Interface/Absract Class 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.

Problem Statement:

A Job Board scenario is a digital platform or system that facilitates the process of job searching and recruitment. In this scenario, various stakeholders, such as job seekers, companies, and recruiters, use the platform to post, search for, and apply to job opportunities.

Create SQL Schema from the application, use the class attributes for table column names.

1.Create and implement the mentioned class and the structure in your application.

JobListing Class:
Attributes:

- JobID (int): A unique identifier for each job listing.
- CompanyID (int): A reference to the company offering the job.
- JobTitle (string): The title of the job.
- JobDescription (string): A detailed description of the job.
- JobLocation (string): The location of the job.
- Salary (decimal): The salary offered for the job.
- JobType (string): The type of job (e.g., Full-time, Part-time, Contract).
- PostedDate (DateTime): The date when the job was posted.

Methods:

- Apply(applicantID: int, coverLetter: string): Allows applicants to apply for the job by providing their ID and a cover letter.
- GetApplicants(): List<Applicant>: Retrieves a list of applicants who have applied for the job.

Company Class:

Attributes:

- CompanyID (int): A unique identifier for each company.
- CompanyName (string): The name of the hiring company.
- Location (string): The location of the company.

Methods:

- PostJob(jobTitle: string, jobDescription: string, jobLocation: string, salary: decimal, jobType: string): Allows a company to post a new job listing.
- GetJobs(): List<JobListing>: Retrieves a list of job listings posted by the company.

Applicant Class:

Attributes:

- ApplicantID (int): A unique identifier for each applicant.
- FirstName (string): The first name of the applicant.
- LastName (string): The last name of the applicant.
- Email (string): The email address of the applicant.
- Phone (string): The phone number of the applicant.
- Resume (string): The applicant's resume or a reference to the resume file.

Methods:

- CreateProfile(email: string, firstName: string, lastName: string, phone: string): Allows applicants to create a profile with their contact information.
- ApplyForJob(jobID: int, coverLetter: string): Enables applicants to apply for a specific job listing.

JobApplication Class:

Attributes:

- ApplicationID (int): A unique identifier for each job application.
- JobID (int): A reference to the job listing.
- ApplicantID (int): A reference to the applicant.
- ApplicationDate (DateTime): The date and time when the application was submitted.
- CoverLetter (string): The cover letter submitted with the application.

2.DatabaseManager Class:

Methods:

- InitializeDatabase(): Initializes the database schema and tables.
- InsertJobListing(job: JobListing): Inserts a new job listing into the "Jobs" table.
- InsertCompany(company: Company): Inserts a new company into the "Companies" table.
- InsertApplicant(applicant: Applicant): Inserts a new applicant into the "Applicants" table.
- InsertJobApplication(application: JobApplication): Inserts a new job application into the "Applications" table.
- GetJobListings(): List<JobListing>: Retrieves a list of all job listings.
- GetCompanies(): List<Company>: Retrieves a list of all companies.
- GetApplicants(): List<Applicant>: Retrieves a list of all applicants.
- GetApplicationsForJob(jobID: int): List<JobApplication>: Retrieves a list of job applications for a specific job listing.

3.Exceptions handling
Create and implement the following exceptions in your application.

- Invalid Email Format Handling:
    - In the Job Board application, during the applicant registration process, users are required to enter their email addresses. Write a program that prompts the user to input an email address and implement exception handling to ensure that the email address follows a valid format (e.g., contains "@" and a valid domain). If the input is not valid, catch the exception and display an error message. If it is valid, proceed with registration.
- Salary Calculation Handling:
    - Create a program that calculates the average salary offered by companies for job listings. Implement exception handling to ensure that the salary values are non-negative when computing the average. If any salary is negative or invalid, catch the exception and display an error message, indicating the problematic job listings.

- File Upload Exception Handling:
    - In the Job Board application, applicants can upload their resumes as files. Write a program that handles file uploads and implements exception handling to catch and handle potential errors, such as file not found, file size exceeded, or file format not supported. Provide appropriate error messages in each case.
- Application Deadline Handling:
    - Develop a program that checks whether a job application is submitted before the application deadline. Implement exception handling to catch situations where an applicant tries to submit an application after the deadline has passed. Display a message indicating that the application is no longer accepted.
- Database Connection Handling:
    - In the Job Board application, database connectivity is crucial. Create a program that establishes a connection to the database to retrieve job listings. Implement exception handling to catch database-related exceptions, such as connection errors or SQL query errors. Display appropriate error messages and ensure graceful handling of these exceptions.

4.Database Connectivity
Create and implement the following tasks in your application.

- **Job Listing Retrieval:** Write a program that connects to the database and retrieves all job listings from the "Jobs" table. Implement database connectivity using Entity Framework and display the job titles, company names, and salaries.
- **Applicant Profile Creation:** Create a program that allows applicants to create a profile by entering their information. Implement database connectivity to insert the applicant's data into the "Applicants" table. Handle potential database-related exceptions.
- **Job Application Submission:** Develop a program that allows applicants to apply for a specific job listing. Implement database connectivity to insert the job application details into the "Applications" table, including the applicant's ID and the job ID. Ensure that the program handles database connectivity and insertion exceptions.
- **Company Job Posting:** Write a program that enables companies to post new job listings. Implement database connectivity to insert job listings into the "Jobs" table, including the company's ID. Handle database-related exceptions and ensure the job posting is successful.
- **Salary Range Query:** Create a program that allows users to search for job listings within a specified salary range. Implement database connectivity to retrieve job listings that match the user's criteria, including job titles, company names, and salaries. Ensure the program handles database connectivity and query exceptions.