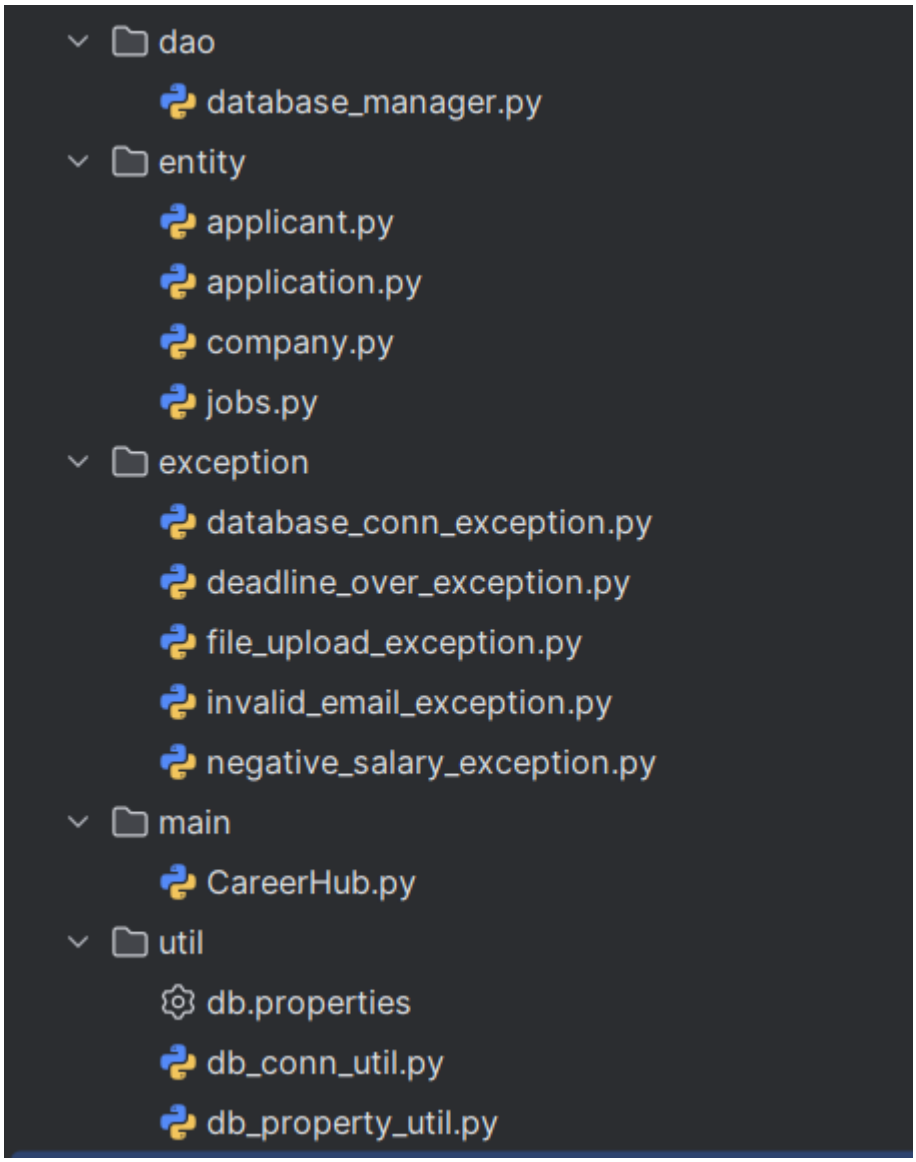
CODING CHALLENGE 2

CAREER HUB

SUBMITTED BY – ANUSHA P

The following Directory structure is to be followed in the application.



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

Companies Table:

```
create table companies (
    company_id int auto_increment primary key,
    company_name varchar(255),
    location varchar(255))
```

Jobs Table:

```
create table jobs (
    job_id int auto_increment primary key,
    company_id int,
    job_title varchar(255),
    job_description text,
    job_location varchar(255),
    salary decimal(10,2),
    job_type varchar(50),
    posted_date datetime,
    deadline datetime,
    foreign key (company_id) references companies(company_id))
```

Applicants Table:

```
create table applicants (
    applicant_id int auto_increment primary key,
    first_name varchar(255),
    last_name varchar(255),
    email varchar(255),
    phone varchar(20),
    resume text,
    experience int )
```

Applications Table:

```
create table if not exists applications (
    application_id int auto_increment primary key,
    job_id int,
    applicant_id int,
    application_date datetime,
    cover_letter text,
    foreign key (job_id) references jobs(job_id),
    foreign key (applicant_id) references applicants(applicant_id)
)
```

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

Jobs class:

```
class Jobs:  3 usages  AnushaPraba
    def __init__(self,company_id,job_title,job_desc,job_location,salary,job_type,posted_date,deadline,job_id=None):
        self.job_id=job_id
        self.company_id=company_id
        self.job_title=job_title
        self.job_desc=job_desc
        self.job_location=job_location
        self.salary=salary
        self.job_type=job_type
        self.posted_date=posted_date
        self.deadline=deadline
```

Company Class:

```
class Company:  6 usages  AnushaPraba
    def __init__(self,company_name,location,company_id=None):
        self.company_name=company_name
        self.location=location
        self.company_id=company_id
```

Applicants Class:

```
class Applicant:  7 usages  AnushaPraba
    def __init__(self, first_name, last_name, email, phone, resume,experience, applicant_id=None):
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.phone = phone
        self.resume = resume
        self.experience = experience
        self.applicant_id = applicant_id
```

Applications Class:

```
class Application:  5 usages  AnushaPraba
    def __init__(self, job_id, applicant_id,cover_letter,application_date,application_id=None):
        self.application_id = application_id
        self.job_id = job_id
        self.applicant_id = applicant_id
        self.application_date = application_date
        self.cover_letter = cover_letter
```

DatabaseManager Class:

Methods:

InitializeDatabase(): Initializes the database schema and tables.

```
def initialize_database(self):  1 usage  AnushaPraba
    self.cursor.execute("""
    CREATE TABLE IF NOT EXISTS Companies (
        company_id INT AUTO_INCREMENT PRIMARY KEY,
        company_name VARCHAR(255),
        location VARCHAR(255)
    )""")

    self.cursor.execute("""
    CREATE TABLE IF NOT EXISTS Jobs (
        job_id INT AUTO_INCREMENT PRIMARY KEY,
        company_id INT,
        job_title VARCHAR(255),
        job_description TEXT,
        job_location VARCHAR(255),
        salary DECIMAL(10,2),
        job_type VARCHAR(50),
        posted_date DATETIME,
        deadline DATETIME,
        FOREIGN KEY (company_id) REFERENCES Companies(company_id)
    )""")
```

```
self.cursor.execute("""
CREATE TABLE IF NOT EXISTS Applicants (
    applicant_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255),
    phone VARCHAR(20),
    resume TEXT
)""")

self.cursor.execute("""
CREATE TABLE IF NOT EXISTS Applications (
    application_id INT AUTO_INCREMENT PRIMARY KEY,
    job_id INT,
    applicant_id INT,
    application_date DATETIME,
    cover_letter TEXT,
    FOREIGN KEY (job_id) REFERENCES JobListings(job_id),
    FOREIGN KEY (applicant_id) REFERENCES Applicants(applicant_id)
)""")

self.conn.commit()
```

InsertJobListing(job: JobListing): Inserts a new job listing into the "Jobs" table.

```
def insert_job(self, company_id, job_title, description, location, salary, job_type, deadline):  1 usage  AnushaPraba
    query = """
            INSERT INTO Jobs (company_id, jobtitle, job_description, job_location, salary, job_type, posted_date, application_deadline)
            VALUES (%s, %s, %s, %s, %s, %s, NOW(),%s)
        """
    self.cursor.execute(query, (company_id, job_title, description, location, salary, job_type, deadline))
    self.conn.commit()
    print("Job posted successfully.")
```

InsertCompany(company: Company): Inserts a new company into the "Companies" table.

```
def insert_company(self, company: Company):  1 usage  AnushaPraba *
    self.cursor.execute("""
    INSERT INTO Companies (company_name, location) VALUES (%s, %s)
    """, (company.company_name, company.location))
    self.conn.commit()
```

InsertApplicant(applicant: Applicant): Inserts a new applicant into the "Applicants" table.

```
def insert_applicant(self, applicant:Applicant):  1 usage  AnushaPraba
    try:
        Applicant.validate_email(applicant.email)  # Email format validation
        self.cursor.execute("""
                INSERT INTO Applicants (first_name, last_name, email, phone, resume)
                VALUES (%s, %s, %s, %s, %s)
                """, (
        applicant.first_name, applicant.last_name, applicant.email, applicant.phone, applicant.resume))
        self.conn.commit()
        applicant.applicant_id = self.cursor.lastrowid
        print("Applicant profile created successfully.")
    except InvalidEmailException as e:
        raise e
```

GetJobListings(): List<JobListing>: Retrieves a list of all job listings.

```
def get_jobs(self) -> List[Jobs]:  1 usage  AnushaPraba
    self.cursor.execute("SELECT * FROM Jobs")
    rows = self.cursor.fetchall()
    return [Jobs(*row[1:], job_id=row[0]) for row in rows]
```

InsertJobApplication(application: JobApplication): Inserts a new job application into the "Applications" table.

```
def insert_application(self, applicant_id, job_id, cover_letter):  1 usage  AnushaPraba *
    self.cursor.execute("SELECT application_deadline FROM Jobs WHERE job_id = %s", (job_id,))
    deadline_result = self.cursor.fetchone()

    if deadline_result and deadline_result[0] and datetime.now() > deadline_result[0]:
        raise DeadlinePassedException("Application deadline has passed.")

    self.cursor.execute("""
            INSERT INTO Applications (applicant_id, job_id, application_date, cover_letter)
            VALUES (%s, %s, %s, %s)
        """, (applicant_id, job_id, datetime.now(), cover_letter))
    self.conn.commit()
    print("Application submitted successfully.")
```

GetCompanies(): List<Company>: Retrieves a list of all companies.

```
def get_companies(self) -> List[Company]:  1 usage  AnushaPraba
    self.cursor.execute("SELECT * FROM Companies")
    rows = self.cursor.fetchall()
    return [Company(*row) for row in rows]
```

GetApplicants(): List<Applicant>: Retrieves a list of all applicants.

```
def get_applicants(self) -> List[Applicant]:  1 usage  AnushaPraba
    self.cursor.execute("SELECT * FROM Applicants")
    rows = self.cursor.fetchall()
    return [Applicant(
        applicant_id=row[0],
        first_name=row[1],
        last_name=row[2],
        email=row[3],
        phone=row[4],
        resume=row[5],
        experience=row[6]
    ) for row in rows]
```

GetApplicationsForJob(jobID: int): List<JobApplication>: Retrieves a list of job applications for a specific job listing.

```
def get_applications_for_job(self, job_id: int) -> List[Application]:  1 usage  AnushaPraba
    self.cursor.execute("SELECT * FROM applications WHERE job_id = %s", (job_id,))
    rows = self.cursor.fetchall()
    return [ Application(
        application_id=row[0],
        job_id=row[1],
        applicant_id=row[2],
        application_date=row[3],
        cover_letter=row[4]
        ) for row in rows]
```

Exceptions handling

Create and implement the following exceptions in your application.

Invalid Email Format Handling:

```
class InvalidEmailException(Exception):  6 usages  AnushaPraba
    def __init__(self, message="Invalid email format."):  AnushaPraba
        super().__init__(message)
```

Salary Calculation Handling:

```
class NegativeSalaryException(Exception):  4 usages  AnushaPraba
    def __init__(self, message="Salary cannot be negative."):  AnushaPraba
        super().__init__(message)
```

Application Deadline Handling:

```
class DeadlinePassedException(Exception):  4 usages  AnushaPraba
    def __init__(self, message="Deadline has passed. Application cannot be submitted."):
        super().__init__(message)
```

Database Connection Handling:

```
class DatabaseConnectionException(Exception):  3 usages  AnushaPraba
    def __init__(self, message="Failed to connect to the database."):
        super().__init__(message)
```

Database Connectivity

```python
class DBConnUtil:  2 usages  AnushaPraba
    @staticmethod  1 usage  AnushaPraba
    def get_connection(prop_file_name: str):
        try:
            # Get the full connection string
            conn_str = DBPropertyUtil.get_connection_string(prop_file_name)

            # Parse the connection string into a dictionary
            conn_params = {}
            for item in conn_str.split(';'):
                if '=' in item:
                    key, value = item.split( sep: '=',  maxsplit: 1)
                    conn_params[key.strip()] = value.strip()

            # Connect to the MySQL database
            conn = mysql.connector.connect(
                host=conn_params.get('host'),
                user=conn_params.get('user'),
                password=conn_params.get('password'),
                database=conn_params.get('database')
            )
            return conn

        except mysql.connector.Error as err:
            raise DatabaseConnectionException(err)

        except Exception as e:
            raise DatabaseConnectionException(f"Unexpected database error: {str(e)}")
```

```python
class DBPropertyUtil:  2 usages  AnushaPraba
    @staticmethod  1 usage  AnushaPraba
    def get_connection_string(prop_file_name: str) -> str:
        props = {}
        try:
            with open(prop_file_name, 'r') as file:
                for line in file:
                    line = line.strip()
                    if line and not line.startswith('#'):
                        key_value = line.split('=')
                        if len(key_value) == 2:
                            key, value = key_value
                            props[key.strip()] = value.strip()
        except FileNotFoundError:
            print(f"Property file '{prop_file_name}' not found.")
        except Exception as e:
            print(f"Error reading property file: {e}")

        # Build connection string from properties
        connection_string = (
            f"host={props.get('host')};"
            f"user={props.get('user')};"
            f"password={props.get('password')};"
            f"database={props.get('database')}"
        )
        return connection_string
```

Create and implement the following tasks in your application.

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.

```
def get_jobs_by_salary_range(self, min_salary, max_salary):  1 usage  AnushaPraba
    query = """
            SELECT j.jobtitle, c.company_name, j.salary
            FROM Jobs j
            JOIN Companies c ON j.company_id = c.company_id
            WHERE j.salary BETWEEN %s AND %s
        """
    self.cursor.execute(query, (min_salary, max_salary))
    return self.cursor.fetchall()
```

Careerhub.py functionalities:

```
-------- Job Portal Menu --------
1. Register Company
2. Post a Job
3. Register Applicant
4. Apply for Job
5. View All Jobs
6. View All Companies
7. View All Applicants
8. View Applications for a Job
9. Search Jobs by Salary Range
10. Calculate Average Salary
11. Exit
```

Outputs:

```
Enter your choice (1-11): 1
Enter company name: Hexaware
Enter company location: Chenna
Company Hexaware registered successfully.
```

```
Enter your choice (1-11): 2
Enter company ID: 5
Enter job title: Cloud Engineer
Enter job description: Deploy cloud-based solutions.
Enter job location: Pune
Enter job salary: 25000
Enter job type (full time,part time,contract): part time
Enter application deadline (YYYY-MM-DD HH:MM:SS): 2025-04-13 12:00:00
Job posted successfully.
```

```
Enter your choice (1-11): 3
Enter email: student@mail.com
Enter first name: Karan
Enter last name: Johar
Enter phone number: 8796879689
Enter resume content or file name: I am Karan Johar..I have completed my Bachelors in Computer Science Engineering.
Enter experience (years): 3
Applicant profile created successfully.
```

```
Enter your choice (1-11): 4
Enter applicant ID: 2
Enter job ID to apply for: 6
Enter cover letter: I am Varun...
Application submitted successfully.
```

```
Job Listing
Job ID       : 26
Title        : AI engineer
Company ID   : 2
Location     : Mumbai
Salary       : ₹250000.00
Job Type     : full time
Posted On    : 2025-04-09 21:07:47
Deadline     : 2025-04-13 00:00:00


Job Listing
Job ID       : 27
Title        : Cloud Engineer
Company ID   : 5
Location     : Pune
Salary       : ₹25000.00
Job Type     : part time
Posted On    : 2025-04-09 22:38:48
Deadline     : 2025-04-13 12:00:00
```

```
--- Company ---
Company ID    : Chennai
Name          : 1
Location      : Hexaware


--- Company ---
Company ID    : Bangalore
Name          : 2
Location      : Google
```

```
Enter your choice (1-11): 7

--- Applicant ---
Applicant ID : 1
Name         : Arun Kumar
Email        : arun.kumar@email.com
Phone        : 9876543200
Resume       : Software Engineer with expertise in Java and Python.
Experience   : 4


--- Applicant ---
Applicant ID : 2
Name         : Lakshmi Narayan
Email        : lakshmi.narayan@email.com
Phone        : 9876543201
Resume       : Data Scientist specializing in predictive analytics.
Experience   : 3
```

```
Enter your choice (1-11): 8
Enter job ID: 5
Application ID: 5, Applicant ID: 5, Date: 2025-03-25 14:36:53, Cover: Cybersecurity is my passion, a...
```

```
Enter your choice (1-11): 9
Enter minimum salary: 25000
Enter maximum salary: 50000
Cloud Engineer at TCS - ₹25000.00
```

```
Enter your choice (1-11): 10
Average Salary: ₹327962.96
```