CASE STUDY CAREER HUB:

Creation of Classes:

• 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: Retrieves a list of applicants who have applied for the job

```
class jobListing:

def__init__(self,job_id,company_id,job_title,job_description,job_location,salary,job_type,posted_dat
e):

    self.job_id=job_id

    self.company_id=company_id

    self.job_title=job_title

    self.job_description=job_description

    self.job_location=job_location

    self.salary=salary

    self.job_type=job_type

    self.posted_date=posted_date

def apply(self, applicant_id, cover_letter):

  application = {

    'applicant_id': applicant_id,

    'cover_letter': cover_letter,

    'application_date': datetime.now()

  }

  self.applicants.append(application)

  print(f"Application submitted for job '{self.job_title}' by applicant ID {applicant_id}.")
```

```
def getApplicants(self):
    applicant_ids = [app['applicant_id'] for app in self.applicants]
    return applicant_ids
```

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.

```
class company(jobListing):
    def __init__(self,company_id,company_name,location):
        self.company_id=company_id
        self.company_name=company_name
        self.location=location
        self.jobs=[]
    def postJob(self,job_title,job_description,salary,job_type):
        self.job_title=job_title
        self.job_description=job_description
        self.salary=salary
        self.job_type=job_type
        list1=[self.job_title,self.job_description,self,salary,self.job_type]
        jobs.append(list1)
    def getJobs(self):
        print(self.jobs)
```

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

```
class Applicant(jobListing):
    def __init__(self,applicant_id,first_name,last_name,email,phone,resume):
        self.applicant_id = applicant_id
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.phone = phone
        self.resume = resume
        self.cover_letter = " "
    def CreateProfile(self,email,first_name,last_name):
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
    def ApplyForJob(self,job_id,cover_letter):
        self.job_id=job_id
        self.cover_letter=cover_letter
```

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

```
class jobApplication(Applicant,jobListing):
    def __init__(self,application_id,job_id,applicant_id,application_date,cover_letter):
        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
```

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<Job

class DatabaseManager:

```
  def __init__(self, host, database, user, password):
      self.host = host
      self.database = database
      self.user = user
      self.password = password
   def initializeDatabase(self):
      try:
         connection = mysql.connector.connect(
            host='localhost',
            database='exam',
            user='root',
            password='Vishnu@542'
         )
         if connection.is_connected():
            print('Connected to MySQL database')
      except mysql.connector.Error as e:
         print(f"Error Connecting the database")
   def insertJobListings(self,jobListing):
      connection = mysql.connector.connect(
         host='localhost',
         database='exam',
         user='root',
         password='Vishnu@542'
      )
      query="insert into jobs values(%s,%s,%s,%s,%s,%s,%s,%s)"

data=(jobListing.job_id,jobListing.company_id,jobListing.job_title,jobListing.job_description,jobListin
g.job_location,jobListing.salary,jobListing.job_type,jobListing.posted_date)
      cursor=connection.cursor()
      cursor.execute(query,data)
      try:
         q1="select salary from jobs"
```

```
        cursor.execute(q1)
        salaries=cursor.fetchall()
        avg_salaries=dm.calculate_average_salary(salaries)
        print(f"average salaries {avg_salaries}")
    except exception1.NegativeSalaryError as e:
        print(e)
    print("values inserted")
    connection.commit()
    cursor.close()
def insertcompany(self,company):
    connection = mysql.connector.connect(
        host='localhost',
        database='exam',
        user='root',
        password='Vishnu@542'
    )
    query="insert into companies values(%s,%s,%s)"
    cursor=connection.cursor()
    data=(company.company_id,company.company_name,company.location)
    cursor.execute(query,data)
    connection.commit()
    cursor.close()
    print("values entered into companies tables")
def insertApplicant(self,Applicant):
    connection = mysql.connector.connect(
        host='localhost',
        database='exam',
        user='root',
        password='Vishnu@542'
    )
```

```
data=(Applicant.applicant_id,Applicant.first_name,Applicant.last_name,Applicant.email,Applicant.phone,Applicant.resume)
        try:
            query="insert into applicants values(%s,%s,%s,%s,%s,%s)"
            dm.email_check(Applicant.email)
        except exception1.InvalidEmailError as e:
            print(e)
        cursor=connection.cursor()
        cursor.execute(query,data)
        connection.commit()
        cursor.close()
        print("values entered into applicants")
    def insertJobApplications(self,jobApplication):
        connection = mysql.connector.connect(
            host='localhost',
            database='exam',
            user='root',
            password='Vishnu@542'
        )
        query="insert into applications values(%s,%s,%s,%s,%s)"

data=(jobApplication.application_id,jobApplication.job_id,jobApplication.applicant_id,jobApplication.application_date,jobApplication.cover_letter)
        cursor=connection.cursor()
        cursor.execute(query,data)
        connection.commit()
        cursor.close()
        print("values entered into applications")
    def getJobListings(self):
        connection = mysql.connector.connect(
            host='localhost',
```

```
        database='exam',
        user='root',
        password='Vishnu@542'
    )
    cursor=connection.cursor()
    query="select * from jobs"
    cursor.execute(query)
    joblist=cursor.fetchall()
    for x in joblist:
        print(x)
def getCompanies(self):
    connection  = mysql.connector.connect(
        host='localhost',
        database='exam',
        user='root',
        password='Vishnu@542'
    )
    cursor = connection.cursor()
    query = "select * from companies"
    cursor.execute(query)
    joblist = cursor.fetchall()
    for x in joblist:
        print(x)
def getApplicants(self):
    connection  = mysql.connector.connect(
        host='localhost',
        database='exam',
        user='root',
        password='Vishnu@542'
    )
    cursor=connection.cursor()
```

```
        cursor.execute("select  * from applicants")
        applicants=cursor.fetchall()
        for x in applicants:
            print(x)
    def getApplications(self):
        connection  = mysql.connector.connect(
            host='localhost',
            database='exam',
            user='root',
            password='Vishnu@542'
        )
        cursor=connection.cursor()
        cursor.execute("select  * from applications")
        applications=cursor.fetchall()
        for x in applications:
            print(x)
```

Exception Handling:

Invalid Email Format Handling:

o 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:

o 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:

o 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:

o 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:

o 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

```
class InvalidEmailError(Exception):
    def __init__(self, message="Invalid email format. Please enter a valid email address."):
        self.message = message
        print(self.message)

class NegativeSalaryError(Exception):
    def __init__(self, message="Invalid salary value. Salary should be non-negative."):
        self.message = message
        super().__init__(self.message)

class FileUploadError(Exception):
    def __init__(self, message="Error uploading file."):
        self.message = message
        super().__init__(self.message)
```

```
class DeadlineExceededError(Exception):
    def __init__(self, message="Application deadline exceeded."):
        self.message = message
        super().__init__(self.message)

functions in databaseManager:
    def email_check(self,email):
        if "@" not in email or "." not in email:
            raise exception1.InvalidEmailError()
    def upload_file(file):
        try:
            with open(file, 'rb') as f:
                # Upload the file
                print(f"File {file} uploaded successfully.")
        except FileNotFoundError:
            raise exception1.FileUploadError("File not found.")
        except Exception as e:
            raise exception1.FileUploadError(f"File upload failed: {e}")

    def check_deadline(application_date, deadline):
        if application_date > deadline:
            raise exception1.DeadlineExceededError()

    def calculate_average_salary(salaries):
        total_salary = sum(salaries)
        count = len(salaries)
        if count == 0:
            return 0
        for salary in salaries:
            if salary < 0:
                raise exception1.NegativeSalaryError()
```

return total_salary / count

4.Database Connectivity:

Basic connection:

```
connection = mysql.connector.connect(
    host='localhost',
    database='exam',
    user='root',
    password='Vishnu@542')
```

Backend Database And Tables:

```
mysql> use exam;
Database changed
mysql> show tables;
+----------------+
| Tables_in_exam |
+----------------+
| applicants     |
| applications   |
| companies      |
| jobs           |
+----------------+
4 rows in set (0.03 sec)
```

```
mysql> desc jobs;
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| job_id          | int           | NO   | PRI | NULL    |       |
| company_id      | int           | YES  | MUL | NULL    |       |
| job_title       | varchar(30)   | YES  |     | NULL    |       |
| job_description | tinytext      | YES  |     | NULL    |       |
| job_location    | varchar(30)   | YES  |     | NULL    |       |
| salary          | decimal(10,2) | YES  |     | NULL    |       |
| job_type        | varchar(30)   | YES  |     | NULL    |       |
| posted_date     | date          | YES  |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
8 rows in set (0.01 sec)

mysql> desc companies
    -> ;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| company_id   | int         | NO   | PRI | NULL    |       |
| company_name | varchar(30) | YES  |     | NULL    |       |
| Location     | varchar(30) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> desc applications;
+------------------+------+------+-----+---------+-------+
| Field            | Type | Null | Key | Default | Extra |
+------------------+------+------+-----+---------+-------+
| application_id   | int  | NO   | PRI | NULL    |       |
| job_id           | int  | YES  | MUL | NULL    |       |
| applicant_id     | int  | YES  | MUL | NULL    |       |
| application_date | date | YES  |     | NULL    |       |
| cover_letter     | text | YES  |     | NULL    |       |
+------------------+------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> desc applications;
+------------------+------+------+-----+---------+-------+
| Field            | Type | Null | Key | Default | Extra |
+------------------+------+------+-----+---------+-------+
| application_id   | int  | NO   | PRI | NULL    |       |
| job_id           | int  | YES  | MUL | NULL    |       |
| applicant_id     | int  | YES  | MUL | NULL    |       |
| application_date | date | YES  |     | NULL    |       |
| cover_letter     | text | YES  |     | NULL    |       |
+------------------+------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Insertion into databse using python

Object creation:

```
invalidEmail                                                    2/2
355
356  dm=DatabaseManager( host: "localhost", database: "exam", user: "root", password: "Vishnu@542")
357
358  j1=jobListing( job_id: 16, company_id: 4, job_title: "support", job_description: "na", job_location: "delhi", salary: 57000, job_typ
359  j1.getApplicants()
360  c1=company( company_id: 10, company_name: "zebronics", location: "canada")
361  ap1=Applicant( applicant_id: 509, first_name: "devendra", last_name: "singhal", email: "asdf@gamil.com", phone: 7236541897,
362  ap2=jobApplication( application_id: 213, job_id: 111, applicant_id: 508, application_date: "2020/05/10", cover_letter: "cv2")
363  dm.insertJobListings(j1)
364  dm.insertApplicant(ap1)
365  dm.insertJobApplications(ap2)
366  dm.insertcompany(c1)
367
```

o/p:

```
C:\Users\Lenovo\PycharmProjects\casestudy\.venv\Scripts\python.exe C:\Users\Lenovo\AppData\Roaming\JetBrains\PyCharmCE2024.1\scratches\main.py
values inserted
values entered into applicants
values entered into applications
values entered into companies tables

Process finished with exit code 0
```

Functions:

```
def get_job_listings(self):
    try:
        connection = mysql.connector.connect(
            host=self.host,
            database=self.database,
            user=self.user,
            password=self.password
        )

        if connection.is_connected():
            cursor = connection.cursor()
            query = "SELECT JobTitle, CompanyName, Salary FROM jobs INNER JOIN companies ON jobs.CompanyID = companies.CompanyID"
            cursor.execute(query)
            job_listings = cursor.fetchall()
```

```
            for job in job_listings:
                print("Job Title:", job[0])
                print("Company Name:", job[1])
                print("Salary:", job[2])
                print()
            cursor.close()

        else:
            print('Connection to MySQL database failed')

    except mysql.connector.Error as e:
        print(f"Error retrieving job listings from MySQL database: {e}")

def create_applicant_profile(self, applicant):
    try:
        connection = mysql.connector.connect(
            host=self.host,
            database=self.database,
            user=self.user,
            password=self.password
        )
        if connection.is_connected():
            cursor = connection.cursor()
            query = "INSERT INTO applicants (FirstName, LastName, Email, Phone, Resume) VALUES
(%s, %s, %s, %s, %s)"
            data = (applicant.first_name, applicant.last_name, applicant.email, applicant.phone,
applicant.resume)
            cursor.execute(query, data)
            connection.commit()
            print("Applicant profile created successfully")
            cursor.close()
        else:
```

```
            print('Connection to MySQL database failed')
        except mysql.connector.Error as e:
            print(f"Error creating applicant profile in MySQL database: {e}")

        def submit_job_application(self, job_id, applicant_id, application_date, cover_letter):
            try:
                connection = mysql.connector.connect(
                    host=self.host,
                    database=self.database,
                    user=self.user,
                    password=self.password
                )

                if connection.is_connected():
                    cursor = connection.cursor()
                    query = "INSERT INTO job_applications (JobID, ApplicantID, ApplicationDate,
CoverLetter) VALUES (%s, %s, %s, %s)"
                    data = (job_id, applicant_id, application_date, cover_letter)
                    cursor.execute(query, data)
                    connection.commit()
                    print("Job application submitted successfully")
                    cursor.close()
                else:
                    print('Connection to MySQL database failed')

            except mysql.connector.Error as e:
                print(f"Error submitting job application in MySQL database: {e}")
    def post_job_listing(self, company_id, job_title, job_description, job_location, salary, job_type,
posted_date):
        try:
            connection = mysql.connector.connect(
                host=self.host,
```

```
            database=self.database,
            user=self.user,
            password=self.password
        )

        if connection.is_connected():
            cursor = connection.cursor()
            query = "INSERT INTO jobs (CompanyID, JobTitle, JobDescription, JobLocation, Salary,
JobType, PostedDate) VALUES (%s, %s, %s, %s, %s, %s, %s)"
            data = (company_id, job_title, job_description, job_location, salary, job_type, posted_date)
            cursor.execute(query, data)
            connection.commit()
            print("Job listing posted successfully")
            cursor.close()

        else:
            print('Connection to MySQL database failed')

    except mysql.connector.Error as e:
        print(f"Error posting job listing in MySQL database: {e}")

def search_job_listings_by_salary_range(self, min_salary, max_salary):
    try:
        connection = mysql.connector.connect(
            host=self.host,
            database=self.database,
            user=self.user,
            password=self.password
        )

        if connection.is_connected():
```

```
        cursor = connection.cursor()
        query = "SELECT JobTitle, CompanyName, Salary FROM jobs INNER JOIN companies ON
jobs.CompanyID = companies.CompanyID WHERE Salary BETWEEN %s AND %s"
        data = (min_salary, max_salary)
        cursor.execute(query, data)
        job_listings = cursor.fetchall()
        for job in job_listings:
            print("Job Title:", job[0])
            print("Company Name:", job[1])
            print("Salary:", job[2])
            print()
        cursor.close()

    else:
        print('Connection to MySQL database failed')

except mysql.connector.Error as e:
    print(f"Error searching job listings by salary range in MySQL database: {e}")
```

Databases before insertion:

```
mysql> select * from applications;
+----------------+--------+--------------+------------------+--------------+
| application_id | job_id | applicant_id | application_date | cover_letter |
+----------------+--------+--------------+------------------+--------------+
|            201 |    101 |          501 | 2023-05-15       | coverletter1 |
|            202 |    102 |          502 | 2023-06-20       | coverletter2 |
|            203 |    103 |          503 | 2023-07-10       | coverletter3 |
|            204 |    101 |          501 | 2023-08-05       | coverletter4 |
|            205 |    102 |          503 | 2023-09-12       | coverletter5 |
|            206 |    104 |          503 | 2023-10-18       | coverletter6 |
|            207 |    107 |          501 | 2023-11-25       | coverletter7 |
|            208 |    108 |          501 | 2023-12-30       | coverletter8 |
|            209 |    109 |          503 | 2024-01-10       | coverletter9 |
|            210 |    110 |          503 | 2024-02-22       | coverletter10|
|            211 |    102 |          501 | 2023-05-10       | hello        |
+----------------+--------+--------------+------------------+--------------+
11 rows in set (0.00 sec)

mysql> select * from jobs;
+--------+------------+-----------+-----------------------+--------------+----------+-----------+-------------+
| job_id | company_id | job_title | job_description       | job_location | salary   | job_type  | posted_date |
+--------+------------+-----------+-----------------------+--------------+----------+-----------+-------------+
|      1 |          2 | developer | developing            | hyd          |  10000.00 | full time | 2022-09-09  |
|      2 |          7 | developer | developing            | pune         | 100000.00 | part time | 2002-05-04  |
|     51 |          7 | developer | developing            | pune         | 100000.00 | part time | 2002-05-04  |
|    101 |          1 | Developer | developing software   | pune         |  50000.00 | full-time | 2022-04-12  |
|    102 |          1 | Testing   | testing software      | pune         |  20000.00 | part-time | 2023-06-12  |
|    103 |          2 | developer | Developing software   | cityX        |  80000.00 | full-time | 2019-01-01  |
|    104 |          2 | Testing   | testing software      | hyderabad    |  20000.00 | full-time | 2022-12-27  |
|    105 |          3 | Testing   | testing software      | Hyderabad    |  60000.00 | part-time | 2022-03-01  |
|    106 |          3 | developer | developing software   | bangalore    |  70000.00 | part-time | 2023-07-11  |
|    107 |          4 | developer | developing software   | cityX        |  45000.00 | full-time | 2023-06-12  |
|    108 |          4 | Testing   | testing software      | bhuvaneswar  |  30000.00 | part-time | 2022-02-01  |
|    109 |          5 | developer | developing software   | bhuvaneswar  |  25000.00 | full-time | 2022-09-17  |
|    110 |          5 | Testing   | testing software      | vizag        |  55000.00 | part-time | 2023-07-17  |
|    111 |          2 | developer | developing            | hyd          |  10000.00 | full time | 2022-09-09  |
|    501 |          7 | developer | developing            | pune         | 100000.00 | part time | 2002-05-04  |
+--------+------------+-----------+-----------------------+--------------+----------+-----------+-------------+
```

Database after insertion :

```
mysql> select * from applications;
+----------------+--------+--------------+------------------+---------------+
| application_id | job_id | applicant_id | application_date | cover_letter  |
+----------------+--------+--------------+------------------+---------------+
|            201 |    101 |          501 | 2023-05-15       | coverletter1  |
|            202 |    102 |          502 | 2023-06-20       | coverletter2  |
|            203 |    103 |          503 | 2023-07-10       | coverletter3  |
|            204 |    101 |          501 | 2023-08-05       | coverletter4  |
|            205 |    102 |          503 | 2023-09-12       | coverletter5  |
|            206 |    104 |          503 | 2023-10-18       | coverletter6  |
|            207 |    107 |          501 | 2023-11-25       | coverletter7  |
|            208 |    108 |          501 | 2023-12-30       | coverletter8  |
|            209 |    109 |          503 | 2024-01-10       | coverletter9  |
|            210 |    110 |          503 | 2024-02-22       | coverletter10 |
|            211 |    102 |          501 | 2023-05-10       | hello         |
|            213 |    111 |          508 | 2020-05-10       | cv2           |
+----------------+--------+--------------+------------------+---------------+
12 rows in set (0.00 sec)
```

```
mysql> select * from jobs;
+--------+------------+-----------+--------------------+--------------+-----------+-----------+-------------+
| job_id | company_id | job_title | job_description    | job_location | salary    | job_type  | posted_date |
+--------+------------+-----------+--------------------+--------------+-----------+-----------+-------------+
|      1 |          2 | developer | developing         | hyd          |  10000.00 | full time | 2022-09-09  |
|      2 |          7 | developer | developing         | pune         | 100000.00 | part time | 2002-05-04  |
|     15 |          4 | support   | na                 | delhi        |  57000.00 | full time | 2010-06-10  |
|     16 |          4 | support   | na                 | delhi        |  57000.00 | full time | 2010-06-10  |
|     51 |          7 | developer | developing         | pune         | 100000.00 | part time | 2002-05-04  |
|    101 |          1 | Developer | developing software| pune         |  50000.00 | full-time | 2022-04-12  |
|    102 |          1 | Testing   | testing software   | pune         |  20000.00 | part-time | 2023-06-12  |
|    103 |          2 | developer | Developing software| cityX        |  80000.00 | full-time | 2019-01-01  |
|    104 |          2 | Testing   | testing software   | hyderabad    |  20000.00 | full-time | 2022-12-27  |
|    105 |          3 | Testing   | testing software   | Hyderabad    |  60000.00 | part-time | 2022-03-01  |
|    106 |          3 | developer | developing software| bangalore    |  70000.00 | part-time | 2023-07-11  |
|    107 |          4 | developer | developing software| cityX        |  45000.00 | full-time | 2023-06-12  |
|    108 |          4 | Testing   | testing software   | bhuvaneswar  |  30000.00 | part-time | 2022-02-01  |
|    109 |          5 | developer | developing software| bhuvaneswar  |  25000.00 | full-time | 2022-09-17  |
|    110 |          5 | Testing   | testing software   | vizag        |  55000.00 | part-time | 2023-07-17  |
|    111 |          2 | developer | developing         | hyd          |  10000.00 | full time | 2022-09-09  |
|    501 |          7 | developer | developing         | pune         | 100000.00 | part time | 2002-05-04  |
+--------+------------+-----------+--------------------+--------------+-----------+-----------+-------------+
17 rows in set (0.00 sec)
```

Email verification;

We missed giving the @symbol

```
ap1=Applicant( applicant_id: 521, first_name: "raju", last_name: "krishna", email: "asdfgamil.com", phone: 7236541897, resu
```

o/p:

```
C:\Users\Lenovo\PycharmProjects\casestudy\.venv\Scripts\python.exe C:\Users\Lenovo\AppData\Roaming\JetBrains\PyCharmCE2024.1\scratches\main.py
Invalid email format. Please enter a valid email address.
```

Data retrieval functions:

```
dm=DatabaseManager( host: "localhost", database: "exam", user: "root", password: "Vishnu@542")

dm.getJobListings()
```

o/p:

```
C:\Users\Lenovo\PycharmProjects\casestudy\.venv\Scripts\python.exe C:\Users\Lenovo\AppData\Roaming\JetBrains\PyCharmCE2024.1\scratches\main.py
(1, 2, 'developer', 'developing', 'hyd', Decimal('10000.00'), 'full time', datetime.date(2022, 9, 9))
(2, 7, 'developer', 'developing', 'pune', Decimal('100000.00'), 'part time', datetime.date(2002, 5, 4))
(15, 4, 'support', 'na', 'delhi', Decimal('57000.00'), 'full time', datetime.date(2010, 6, 10))
(16, 4, 'support', 'na', 'delhi', Decimal('57000.00'), 'full time', datetime.date(2010, 6, 10))
(51, 7, 'developer', 'developing', 'pune', Decimal('100000.00'), 'part time', datetime.date(2002, 5, 4))
(101, 1, 'Developer', 'developing software', 'pune', Decimal('50000.00'), 'full-time', datetime.date(2022, 4, 12))
(102, 1, 'Testing', 'testing software', 'pune', Decimal('20000.00'), 'part-time', datetime.date(2023, 6, 12))
(103, 2, 'developer', 'Developing software', 'cityX', Decimal('80000.00'), 'full-time', datetime.date(2019, 1, 1))
(104, 2, 'Testing', 'testing software', 'hyderabad', Decimal('20000.00'), 'full-time', datetime.date(2022, 12, 27))
(105, 3, 'Testing', 'testing software', 'Hyderabad', Decimal('60000.00'), 'part-time', datetime.date(2022, 3, 1))
```

```
dm=DatabaseManager("localhost","exam","root","Vishnu@542")
dm.getApplications()
```

```
C:\Users\Lenovo\PycharmProjects\casestudy\.venv\Scripts\python.exe C:\Users\Lenovo\AppData\Roaming\JetBrains\PyCharmCE2024.1\scratches\main.py
(201, 101, 501, datetime.date(2023, 5, 15), 'coverletter1')
(202, 102, 502, datetime.date(2023, 6, 20), 'coverletter2')
(203, 103, 503, datetime.date(2023, 7, 10), 'coverletter3')
(204, 101, 501, datetime.date(2023, 8, 5), 'coverletter4')
(205, 102, 503, datetime.date(2023, 9, 12), 'coverletter5')
(206, 104, 503, datetime.date(2023, 10, 18), 'coverletter6')
(207, 107, 501, datetime.date(2023, 11, 25), 'coverletter7')
(208, 108, 501, datetime.date(2023, 12, 30), 'coverletter8')
(209, 109, 503, datetime.date(2024, 1, 10), 'coverletter9')
(210, 110, 503, datetime.date(2024, 2, 22), 'coverletter10')
```

```
dm.getApplicants()
```

```
C:\Users\Lenovo\PycharmProjects\casestudy\.venv\Scripts\python.exe C:\Users\Lenovo\AppData\Roaming\JetBrains\PyCharmCE2024.1\scratches\main.py
(501, 'vishnu', 'charan', 'vcptes44@gmail.com', '1234574337', 'resume1')
(502, 'soma', 'shekar', 'shek@gmail.com', '9495743371', 'resume2')
(503, 'satya sai', 'pavan', 'pavan@gmail.com', '9848652314', 'resume3')
(504, 'raja', 'kumar', 'r123@gmail.com', '1234567890', 'resumeee')
(507, 'sai', 'krishna', 'abc@email,com', '4236589745', 'hi')
(508, 'devendra', 'singhal', 'asdf@gamil.com', '7236541897', 'resume2')
(509, 'devendra', 'singhal', 'asdf@gamil.com', '7236541897', 'resume2')
(510, 'ravindra', 'sharma', 'asdc@gmail.com', '54698745', 'resumeee')
(511, 'ravindra', 'sharma', 'asdc@gmailcom', '54698745', 'resumeee')
(521, 'raju', 'krishna', 'asdfgamil.com', '7236541897', 'resume2')
```