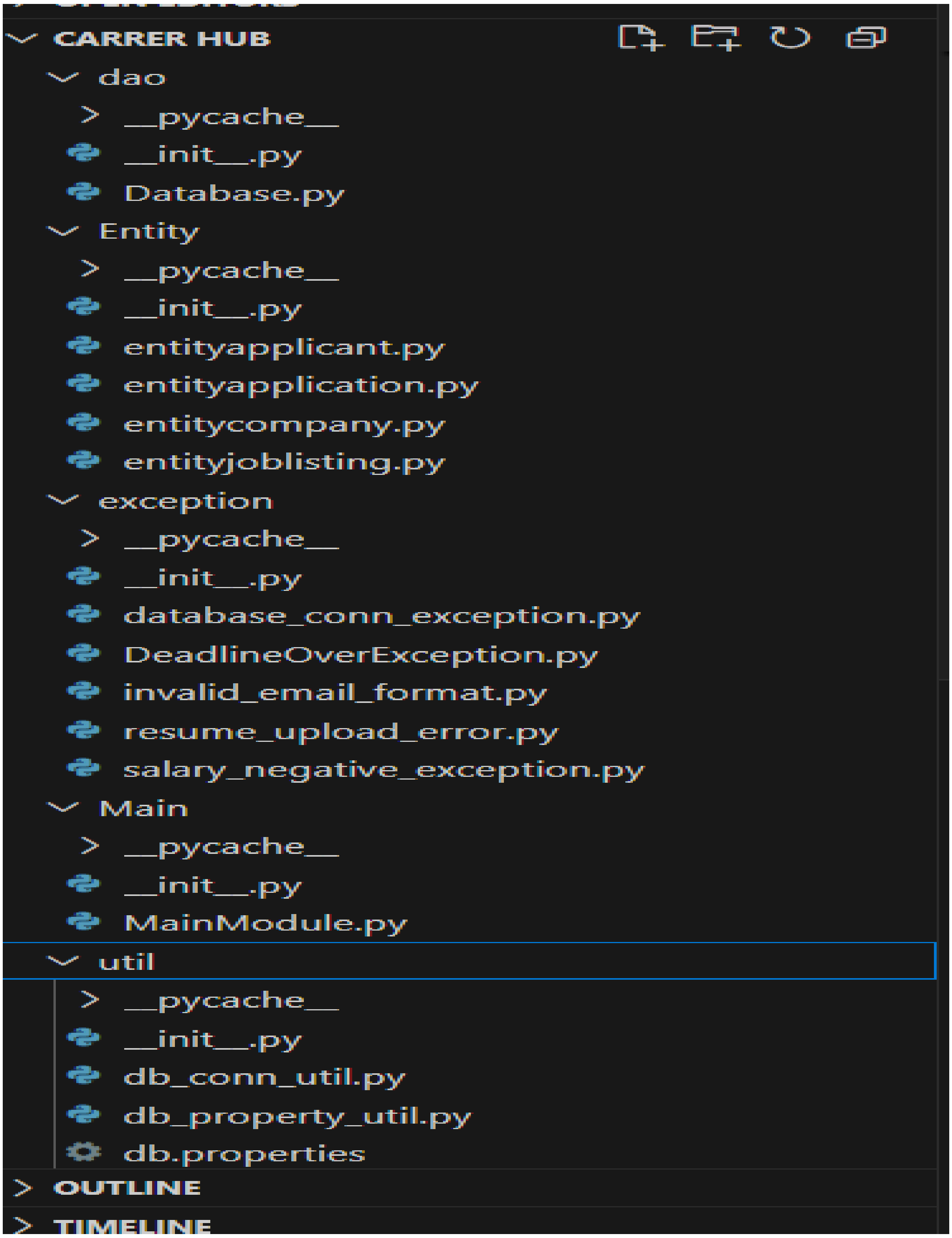
Coding challenges :
CAREER HUB



```
∨ CARRER HUB
  ∨ dao
    > __pycache__
      __init__.py
      Database.py
  ∨ Entity
    > __pycache__
      __init__.py
      entityapplicant.py
      entityapplication.py
      entitycompany.py
      entityjoblisting.py
  ∨ exception
    > __pycache__
      __init__.py
      database_conn_exception.py
      DeadlineOverException.py
      invalid_email_format.py
      resume_upload_error.py
      salary_negative_exception.py
  ∨ Main
    > __pycache__
      __init__.py
      MainModule.py
  ∨ util
    > __pycache__
      __init__.py
      db_conn_util.py
      db_property_util.py
      db.properties
> OUTLINE
> TIMELINE
```

WORKFLOW STRUCTURE

1.Create and implement the class and the structure in the application.(under entity package)

JOB CLASS

```
class Jobs:
    def __init__(self, job_id, job_title, job_description, job_location, salary, job_type, posted_date):
        self.job_id = job_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 __str__(self):
        return f"[{self.job_id}] {self.job_title} | {self.job_location} | ₹{self.salary} | {self.job_type} | Posted: {self.posted_date.strftime('%Y-%m-%d')}"
```

COMPANY CLASS

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

  def __str__(self):
    return f"[{self.company_id}] {self.company_name} - {self.location}"

  def __repr__(self):
    return self.__str__()
```

APPLICATION CLASS

```
class Application:
  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
```

APPLICANT CLASS

```
class Applicant:
    def __init__(self, first_name, last_name, email, phone, resume,
experience, applicant_id=None):
        try:
            validate_email(email)
        except InvalidEmailFormat as e:
            raise InvalidEmailFormat(e)

        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
```

2. DatabaseManager Class under dao

```
import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__),
'..')))
from typing import List
from util.db_conn_util import DBConnUtil
from Entity.entityjoblisting import Jobs
from Entity.entitycompany import Company
from Entity.entityapplicant import Applicant
from Entity.entityapplication import Application
from datetime import datetime

from exception.DeadlineOverException import DeadlineOverException
from exception.salary_negative_exception import SalaryNegativeException
from exception.invalid_email_format import InvalidEmailFormat
from exception.database_conn_exception import DatabaseConnException

class DatabaseManager:
    def __init__(self):
        try:
            self.conn = DBConnUtil.get_connection("util/db.properties")
            self.cursor = self.conn.cursor()
        except Exception as e:
            raise DatabaseConnException(str(e))

    def initialize_database(self):
        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,
   jobtitle VARCHAR(255),
   job_description TEXT,
   job_location VARCHAR(255),
   salary DECIMAL(10,2),
   job_type VARCHAR(50),
   posted_date DATETIME,
   application_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,
   experience INT
)""")
```

```
    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 Jobs(job_id),
      FOREIGN KEY (applicant_id) REFERENCES Applicants(applicant_id)
    )""")

    self.conn.commit()
```

Insert JobListing (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):
    if salary < 0:
      raise SalaryNegativeException()
    self.cursor.execute("""
      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)
    """, (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):

        self.cursor.execute("""

        INSERT INTO Companies (company_name, location) VALUES (%s, %s)

        """, (company.company_name, company.location))

        self.conn.commit()

        company.company_id = self.cursor.lastrowid
```

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

```
  def insert_applicant(self, applicant: Applicant):

        try:

            applicant.validate_email(applicant.email)

            self.cursor.execute("""

                INSERT INTO Applicants (first_name, last_name, email, phone,
resume, experience)

                VALUES (%s, %s, %s, %s, %s, %s)

            """, (applicant.first_name, applicant.last_name, applicant.email,

                  applicant.phone, applicant.resume, applicant.experience))

            self.conn.commit()

            applicant.applicant_id = self.cursor.lastrowid
            print("Applicant profile created.")
        except InvalidEmailFormat as e:
            raise e
```

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

```
      def get_applications_for_job(self, job_id: int) -> List[Application]:
      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 ]
```

CREATING EXCEPTION(under exception package)

1. database_conn_exception

```
class DatabaseConnException(Exception):
    def __init__(self, message="❌ Could not connect to the database."):
        super().__init__(message)
```

2. DeadlineOverException

```
class DeadlineOverException(Exception):
    def __init__(self, message="❌ Application deadline is over. You can't apply for this job."):
        super().__init__(message)
```

3. invalid_email_format

```
class InvalidEmailFormat(Exception):
    def __init__(self, message="Invalid email format. Must contain '@' and a domain."):
        super().__init__(message)
```

4. resume_upload_error

```
class ResumeUploadError(Exception):
    def __init__(self, message="Resume upload failed."):
        super().__init__(message)
```

5. salary_negative_exception

```
class SalaryNegativeException(Exception):
    def __init__(self, message="❌ Salary cannot be negative."):
```

```
    super().__init__(message)
```

CREATING DATABASE CONNECTIVITY (under util package)

db_conn_util.py

```
import sys
import os
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
import mysql.connector
from exception.database_conn_exception import DatabaseConnException
from util.db_property_util import DBPropertyUtil


class DBConnUtil:
  @staticmethod
  def get_connection(prop_file_name: str):
    try:
      conn_str = DBPropertyUtil.get_connection_string(prop_file_name)

      conn_params = {}
      for item in conn_str.split(';'):
        if '=' in item:
          key, value = item.split('=', 1)
          conn_params[key.strip()] = value.strip()
```

```
            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 DatabaseConnException(f"❌ Database connection error:
{err}")
        except Exception as e:
            raise DatabaseConnException(f"❌ Unexpected error: {e}")
```

db_property_util.py

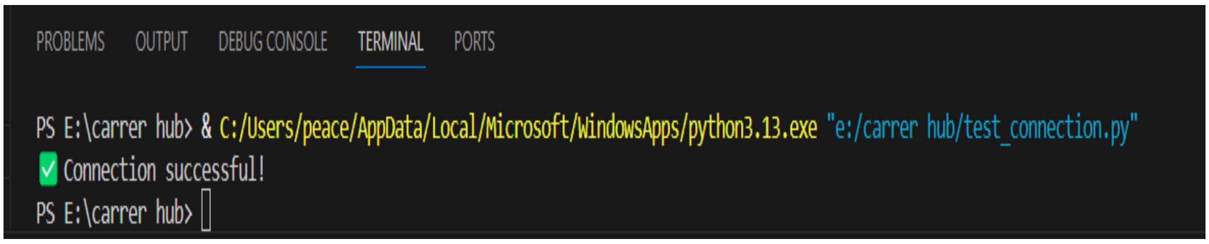
```
class DBPropertyUtil:
    @staticmethod
    def get_connection_string(prop_file_name: str) -> str:
        props = {}
        try:
            with open(prop_file_name, 'r') as file:
                for line in file:
                    line = line.strip()
                    if line and not line.startswith('#'):
                        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
```

db.properties

```
host=localhost
user=root
password=root
port=3306
database=careerhub
```

test_connection.py

```
import sys

import os


# This makes sure Python can find the util folder

sys.path.append(os.path.abspath(os.path.dirname(__file__)))


from util.db_conn_util import DBConnUtil


# Use the connection method and test

conn = DBConnUtil.get_connection("util/db.properties")

if conn:

    print("✅ Connection successful!")

    conn.close()

else:

    print("❌ Connection failed.")
```



```
PS E:\carrer hub> & C:/Users/peace/AppData/Local/Microsoft/WindowsApps/python3.13.exe "e:/carrer hub/test_connection.py"
✅ Connection successful!
PS E:\carrer hub> 
```

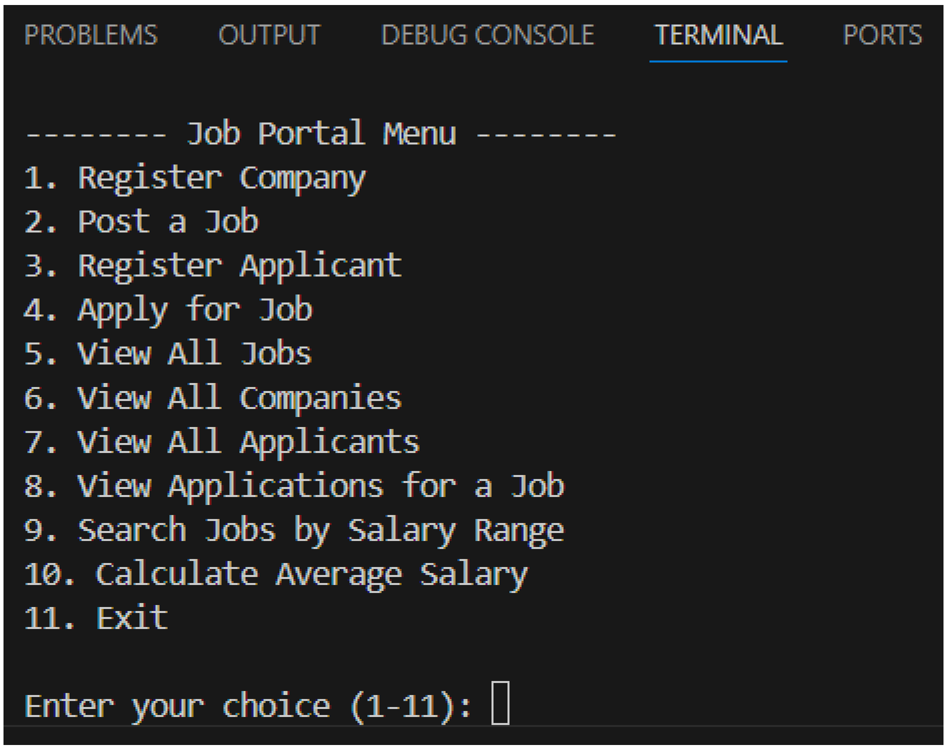
CREATING MAIN MODULE (UNDER MAIN PACKAGE)

```
import sys

import os

sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__),
'..')))


from dao.Database import DatabaseManager

from Entity.entitycompany import Company

from Entity.entityapplicant import Applicant

from Entity.entityjoblisting import Jobs

from Entity.entityapplication import Application


from exception.invalid_email_format import InvalidEmailFormat

from exception.salary_negative_exception import SalaryNegativeException

from exception.DeadlineOverException import DeadlineOverException

from datetime import datetime

from datetime import datetime


def main():

    db = DatabaseManager()

    db.initialize_database()



    while True:

        print("""
-------- 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
        """)

        choice = input("Enter your choice (1-11): ")

        try:
            if choice == "1":
                name = input("Enter company name: ")
                location = input("Enter company location: ")
                company = Company(company_name=name, location=location)
                db.insert_company(company)
                print(f"Company {name} registered successfully.")

            elif choice == "2":
```

```
            company_id = int(input("Enter company ID: "))

            title = input("Enter job title: ")

            description = input("Enter job description: ")

            location = input("Enter job location: ")

            salary = float(input("Enter job salary: "))

            job_type = input("Enter job type (full time,part time,contract): ")

            deadline = input("Enter application deadline (YYYY-MM-DD
HH:MM:SS): ")

            deadline = datetime.strptime(deadline, "%Y-%m-%d %H:%M:%S")

            db.insert_job(company_id, title, description, location, salary,
job_type, deadline)


        elif choice == "3":

            first_name = input("Enter first name: ")

            last_name = input("Enter last name: ")

            email = input("Enter email: ")

            phone = input("Enter phone number: ")

            resume = input("Enter resume details: ")

            experience = int(input("Enter years of experience: "))

            applicant = Applicant(first_name=first_name,
last_name=last_name, email=email, phone=phone, resume=resume,
experience=experience)

            db.insert_applicant(applicant)



        elif choice == "4":
```

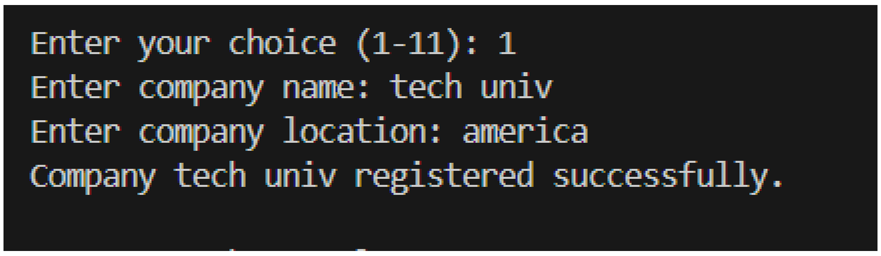
```
        applicant_id = int(input("Enter applicant ID: "))
        job_id = int(input("Enter job ID to apply for: "))
        cover = input("Enter cover letter: ")
        db.insert_job_application(applicant_id, job_id, cover)

    elif choice == "5":
        print("\n📄 Job Listings:")
        jobs = db.get_jobs()
        if not jobs:
            print("No jobs available.")
        else:
            for job in jobs:
                print(f"{job.job_id} | {job.job_title} | {job.job_location} |
₹{job.salary} | {job.job_type}")



    elif choice == "6":
        print("\n🏢 Companies:")
        companies = db.get_companies()
        if not companies:
           print("No companies registered.")
        else:
         for company in companies:
           print(company)
    elif choice == "7":
        applicants = db.get_applicants()
        for app in applicants:
            print(app)

    elif choice == "8":
        job_id = int(input("Enter job ID: "))
        applications = db.get_applications_for_job(job_id)
        for app in applications:
            print(f"Application ID: {app.application_id}, Applicant ID:
{app.applicant_id}, Date: {app.application_date}, Cover:
{app.cover_letter[:30]}...")

    elif choice == "9":
        min_sal = float(input("Enter minimum salary: "))
        max_sal = float(input("Enter maximum salary: "))
        results = db.get_jobs_by_salary_range(min_sal, max_sal)
        for title, company, salary in results:
            print(f"{title} at {company} - ₹{salary}")

    elif choice == "10":
        avg = db.calculate_average_salary()
        print(f"Average Salary: ₹{avg:.2f}")

    elif choice == "11":
        print("Exiting Job Portal. Goodbye!")
        db.close()
        break

      else:
        print("Invalid choice. Please enter a number from 1 to 11.")

    except InvalidEmailFormat as e:
      print(f"Email Error: {e}")
    except SalaryNegativeException as e:
      print(f"Salary Error: Negative salary found for job ID {e.job_id}.")
    except DeadlineOverException as e:
      print(f"Deadline Error: Application deadline has passed for job ID
{e.job_id}.")
    except ValueError as e:
      print("Input error. Please enter data in the correct format.")
    except Exception as e:
      print(f"Unexpected error: {e}")

if __name__ == "__main__":
  main()
```

CARRER HUB PY FUNCTIONALITIES

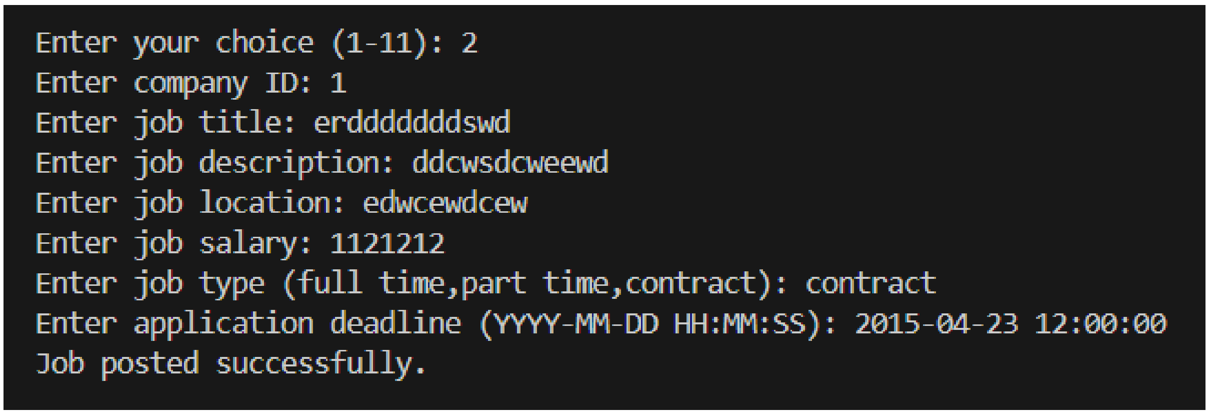


```
-------- Job Portal Menu --------
1. Register Company
2. Post a Job
3. Register Applicant
4. Apply for Job
5. View All Jobs
6. View All Companies
7. View All Applicants
8. View Applications for a Job
9. Search Jobs by Salary Range
10. Calculate Average Salary
11. Exit

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

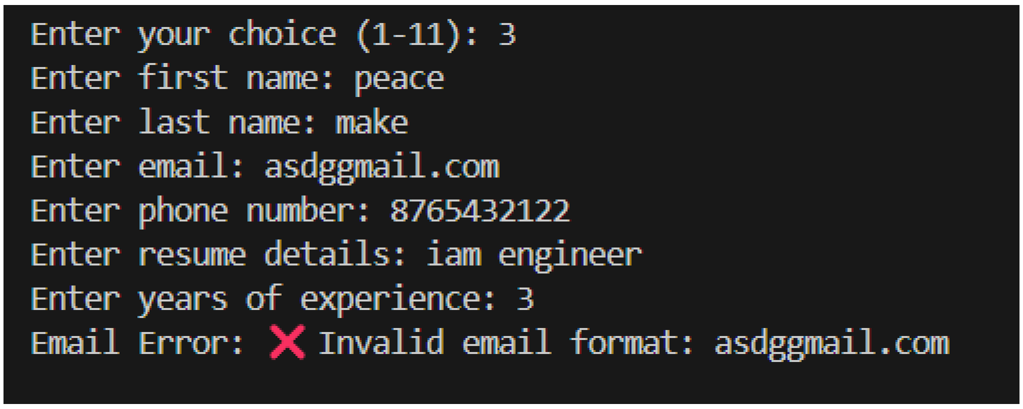
OUTPUTS



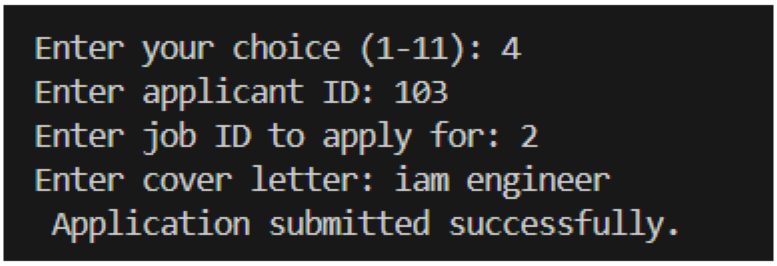
```
Enter your choice (1-11): 1
Enter company name: tech univ
Enter company location: america
Company tech univ registered successfully.
```



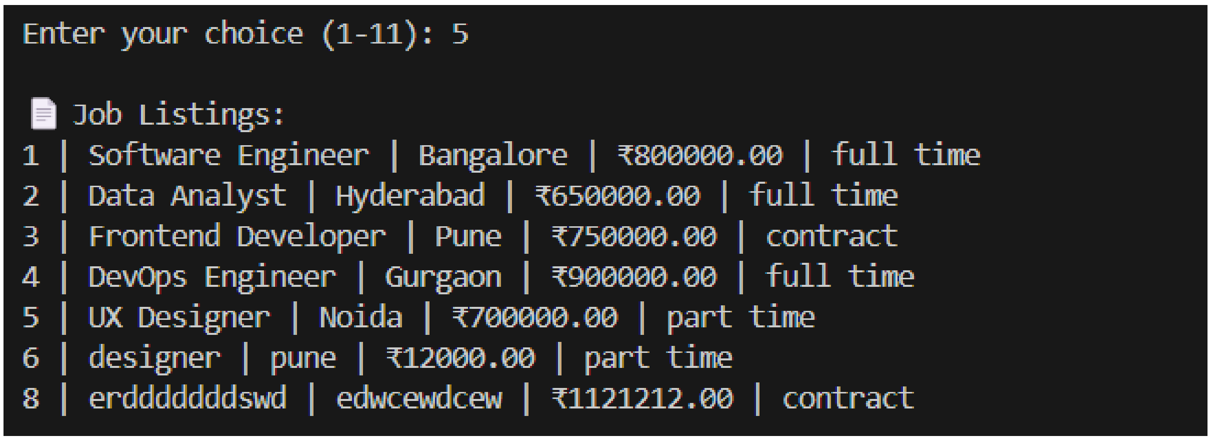
```
Enter your choice (1-11): 2
Enter company ID: 1
Enter job title: erddddddswd
Enter job description: ddcwsdcweewd
Enter job location: edwcewdcew
Enter job salary: 1121212
Enter job type (full time,part time,contract): contract
Enter application deadline (YYYY-MM-DD HH:MM:SS): 2015-04-23 12:00:00
Job posted successfully.
```



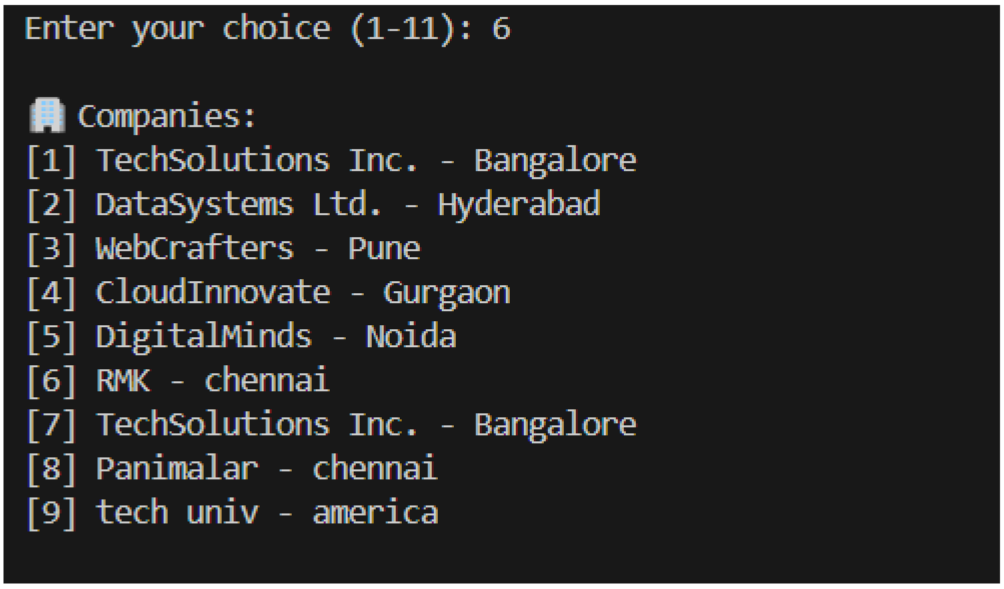
```
Enter your choice (1-11): 3
Enter first name: peace
Enter last name: make
Enter email: asdggmail.com
Enter phone number: 8765432122
Enter resume details: iam engineer
Enter years of experience: 3
Email Error: ❌ Invalid email format: asdggmail.com
```



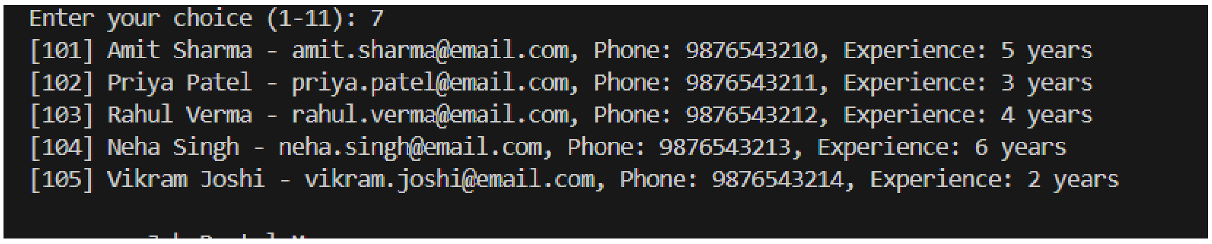
```
Enter your choice (1-11): 4
Enter applicant ID: 103
Enter job ID to apply for: 2
Enter cover letter: iam engineer
 Application submitted successfully.
```



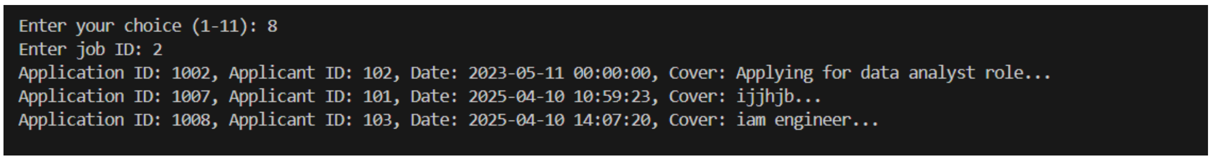
```
Enter your choice (1-11): 5

📄 Job Listings:
1 | Software Engineer | Bangalore | ₹800000.00 | full time
2 | Data Analyst | Hyderabad | ₹650000.00 | full time
3 | Frontend Developer | Pune | ₹750000.00 | contract
4 | DevOps Engineer | Gurgaon | ₹900000.00 | full time
5 | UX Designer | Noida | ₹700000.00 | part time
6 | designer | pune | ₹12000.00 | part time
8 | erddddddswd | edwcewdcew | ₹1121212.00 | contract
```



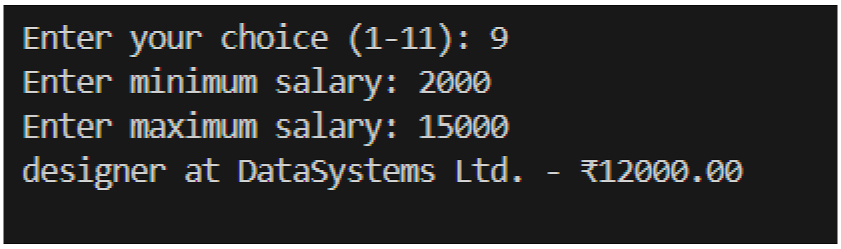
```
Enter your choice (1-11): 6

🏢 Companies:
[1] TechSolutions Inc. - Bangalore
[2] DataSystems Ltd. - Hyderabad
[3] WebCrafters - Pune
[4] CloudInnovate - Gurgaon
[5] DigitalMinds - Noida
[6] RMK - chennai
[7] TechSolutions Inc. - Bangalore
[8] Panimalar - chennai
[9] tech univ - america
```



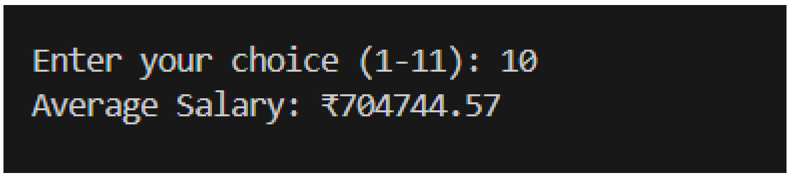
```
Enter your choice (1-11): 7
[101] Amit Sharma - amit.sharma@email.com, Phone: 9876543210, Experience: 5 years
[102] Priya Patel - priya.patel@email.com, Phone: 9876543211, Experience: 3 years
[103] Rahul Verma - rahul.verma@email.com, Phone: 9876543212, Experience: 4 years
[104] Neha Singh - neha.singh@email.com, Phone: 9876543213, Experience: 6 years
[105] Vikram Joshi - vikram.joshi@email.com, Phone: 9876543214, Experience: 2 years
```

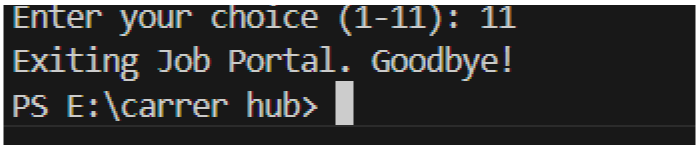


```
Enter your choice (1-11): 8
Enter job ID: 2
Application ID: 1002, Applicant ID: 102, Date: 2023-05-11 00:00:00, Cover: Applying for data analyst role...
Application ID: 1007, Applicant ID: 101, Date: 2025-04-10 10:59:23, Cover: ijjhjb...
Application ID: 1008, Applicant ID: 103, Date: 2025-04-10 14:07:20, Cover: iam engineer...
```



```
Enter your choice (1-11): 9
Enter minimum salary: 2000
Enter maximum salary: 15000
designer at DataSystems Ltd. - ₹12000.00
```


```
Enter your choice (1-11): 10
Average Salary: ₹704744.57
```


```
Enter your choice (1-11): 11
Exiting Job Portal. Goodbye!
PS E:\carrer hub>
```

MYSQL DATABASE AND TABLES

```
CREATE DATABASE careerhub;

USE careerhub;


CREATE TABLE companies (

  company_id INT PRIMARY KEY AUTO_INCREMENT,

  company_name VARCHAR(50),

  location VARCHAR(100)

);


CREATE TABLE jobs (

  job_id INT PRIMARY KEY AUTO_INCREMENT,

  company_id INT,

  FOREIGN KEY (company_id) REFERENCES companies(company_id),

  jobtitle VARCHAR(30),

  job_description TEXT,

  job_location VARCHAR(50),

  salary DECIMAL(15,2) DEFAULT 0.00,

  job_type ENUM('full time', 'part time', 'contract'),
  posted_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE applicants (
  applicant_id INT PRIMARY KEY,
  first_name VARCHAR(20),
  last_name VARCHAR(20),
  email VARCHAR(50),
  phone VARCHAR(15),
  resume TEXT,
  experience  INT
);

CREATE TABLE applications (
  application_id INT PRIMARY KEY,
  job_id INT,
  FOREIGN KEY (job_id) REFERENCES jobs(job_id),
  applicant_id INT,
  FOREIGN KEY (applicant_id) REFERENCES applicants(applicant_id),
  application_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  cover_letter  TEXT
);
```