**Coding Challenges: CareerHub, The Job Board**

**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.

**Package entity**

**JobListing.java**

|  |
| --- |
| **package entity;**  **import java.time.LocalDateTime;**  **public class JobListing {**  **private int jobID;**  **private int companyID;**  **private String jobTitle;**  **private String jobDescription;**  **private String jobLocation;**  **private double salary;**  **private String jobType;**  **private LocalDateTime postedDate;**  **public JobListing(int jobID, int companyID, String jobTitle, String jobDescription, String jobLocation, double salary, String jobType, LocalDateTime postedDate) {**  **this.jobID = jobID;**  **this.companyID = companyID;**  **this.jobTitle = jobTitle;**  **this.jobDescription = jobDescription;**  **this.jobLocation = jobLocation;**  **this.salary = salary;**  **this.jobType = jobType;**  **this.postedDate = postedDate;**  **}**  **// Getters, Setters, toString()**  **public int getJobID() {**  **return jobID;**  **}**  **public void setJobID(int jobID) {**  **this.jobID = jobID;**  **}**  **public int getCompanyID() {**  **return companyID;**  **}**  **public void setCompanyID(int companyID) {**  **this.companyID = companyID;**  **}**  **public String getJobTitle() {**  **return jobTitle;**  **}**  **public void setJobTitle(String jobTitle) {**  **this.jobTitle = jobTitle;**  **}**  **public String getJobDescription() {**  **return jobDescription;**  **}**  **public void setJobDescription(String jobDescription) {**  **this.jobDescription = jobDescription;**  **}**  **public String getJobLocation() {**  **return jobLocation;**  **}**  **public void setJobLocation(String jobLocation) {**  **this.jobLocation = jobLocation;**  **}**  **public double getSalary() {**  **return salary;**  **}**  **public void setSalary(double salary) {**  **this.salary = salary;**  **}**  **public String getJobType() {**  **return jobType;**  **}**  **public void setJobType(String jobType) {**  **this.jobType = jobType;**  **}**  **public LocalDateTime getPostedDate() {**  **return postedDate;**  **}**  **public void setPostedDate(LocalDateTime postedDate) {**  **this.postedDate = postedDate;**  **}**    } |

**Company.java**

|  |
| --- |
| **package entity;**  **public class Company {**  **private int companyID;**  **private String companyName;**  **private String location;**  **public Company(int companyID, String companyName, String location) {**  **this.companyID = companyID;**  **this.companyName = companyName;**  **this.location = location;**  **}**    **public int getCompanyID() {**  **return companyID;**  **}**  **public void setCompanyID(int companyID) {**  **this.companyID = companyID;**  **}**  **public String getCompanyName() {**  **return companyName;**  **}**  **public void setCompanyName(String companyName) {**  **this.companyName = companyName;**  **}**  **public String getLocation() {**  **return location;**  **}**  **public void setLocation(String location) {**  **this.location = location;**  **}**    **}** |

**Applicant.java**

|  |
| --- |
| **package entity;**  **public class Applicant {**  **private int applicantID;**  **private String firstName;**  **private String lastName;**  **private String email;**  **private String phone;**  **private String resume;**  **public Applicant(int applicantID, String firstName, String lastName, String email, String phone, String resume) {**  **this.applicantID = applicantID;**  **this.firstName = firstName;**  **this.lastName = lastName;**  **this.email = email;**  **this.phone = phone;**  **this.resume = resume;**  **}**  **// Getters, Setters, toString()**  **public int getApplicantID() {**  **return applicantID;**  **}**  **public void setApplicantID(int applicantID) {**  **this.applicantID = applicantID;**  **}**  **public String getFirstName() {**  **return firstName;**  **}**  **public void setFirstName(String firstName) {**  **this.firstName = firstName;**  **}**  **public String getLastName() {**  **return lastName;**  **}**  **public void setLastName(String lastName) {**  **this.lastName = lastName;**  **}**  **public String getEmail() {**  **return email;**  **}**  **public void setEmail(String email) {**  **this.email = email;**  **}**  **public String getPhone() {**  **return phone;**  **}**  **public void setPhone(String phone) {**  **this.phone = phone;**  **}**  **public String getResume() {**  **return resume;**  **}**  **public void setResume(String resume) {**  **this.resume = resume;**  **}**      **}** |

**JobApplication.java**

|  |
| --- |
| **package entity;**  **import java.time.LocalDateTime;**  **public class JobApplication {**  **private int applicationID;**  **private int jobID;**  **private int applicantID;**  **private LocalDateTime applicationDate;**  **private String coverLetter;**  **public JobApplication(int applicationID, int jobID, int applicantID, LocalDateTime applicationDate, String coverLetter) {**  **this.applicationID = applicationID;**  **this.jobID = jobID;**  **this.applicantID = applicantID;**  **this.applicationDate = applicationDate;**  **this.coverLetter = coverLetter;**  **}**  **// Getters, Setters, toString()**  **public int getApplicationID() {**  **return applicationID;**  **}**  **public void setApplicationID(int applicationID) {**  **this.applicationID = applicationID;**  **}**  **public int getJobID() {**  **return jobID;**  **}**  **public void setJobID(int jobID) {**  **this.jobID = jobID;**  **}**  **public int getApplicantID() {**  **return applicantID;**  **}**  **public void setApplicantID(int applicantID) {**  **this.applicantID = applicantID;**  **}**  **public LocalDateTime getApplicationDate() {**  **return applicationDate;**  **}**  **public void setApplicationDate(LocalDateTime applicationDate) {**  **this.applicationDate = applicationDate;**  **}**  **public String getCoverLetter() {**  **return coverLetter;**  **}**  **public void setCoverLetter(String coverLetter) {**  **this.coverLetter = coverLetter;**  **}**    **}** |

**Package dao**

**CareerHubDAO.java**

|  |
| --- |
| **package dao;**  **import java.util.List;**  **import entity.\*;**  **public interface CareerHubDAO {**  **void insertCompany(Company company);**  **void insertJobListing(JobListing job);**  **void insertApplicant(Applicant applicant);**  **void insertJobApplication(JobApplication application);**  **List<JobListing> getJobListings();**    **List<Applicant> getApplicants();**  **List<JobListing> searchJobsBySalaryRange(double minSalary, double maxSalary);**  **}** |

**CareerHubDAOImpl.java**

|  |
| --- |
| **package dao;**  **import entity.\*;**  **import util.DBConnUtil;**  **import exception.\*;**  **import java.sql.\*;**  **import java.util.\*;**  **public class CareerHubDAOImpl implements CareerHubDAO {**  **Connection conn = DBConnUtil.getDbConnection();**  **public void insertCompany(Company company) {**  **try (PreparedStatement ps = conn.prepareStatement("INSERT INTO Companies VALUES (?, ?, ?)")) {**  **ps.setInt(1, company.getCompanyID());**  **ps.setString(2, company.getCompanyName());**  **ps.setString(3, company.getLocation());**  **ps.executeUpdate();**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error inserting company", e);**  **}**  **}**  **public void insertJobListing(JobListing job) {**  **try (PreparedStatement ps = conn.prepareStatement("INSERT INTO Jobs VALUES (?, ?, ?, ?, ?, ?, ?, ?)");) {**  **ps.setInt(1, job.getJobID());**  **ps.setInt(2, job.getCompanyID());**  **ps.setString(3, job.getJobTitle());**  **ps.setString(4, job.getJobDescription());**  **ps.setString(5, job.getJobLocation());**  **ps.setDouble(6, job.getSalary());**  **ps.setString(7, job.getJobType());**  **ps.setTimestamp(8, Timestamp.valueOf(job.getPostedDate()));**  **ps.executeUpdate();**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error inserting job listing", e);**  **}**  **}**  **public void insertApplicant(Applicant applicant) {**  **try (PreparedStatement ps = conn.prepareStatement("INSERT INTO Applicants VALUES (?, ?, ?, ?, ?, ?)");) {**  **ps.setInt(1, applicant.getApplicantID());**  **ps.setString(2, applicant.getFirstName());**  **ps.setString(3, applicant.getLastName());**  **ps.setString(4, applicant.getEmail());**  **ps.setString(5, applicant.getPhone());**  **ps.setString(6, applicant.getResume());**  **ps.executeUpdate();**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error inserting applicant", e);**  **}**  **}**  **public void insertJobApplication(JobApplication application) {**  **try (PreparedStatement ps = conn.prepareStatement("INSERT INTO Applications VALUES (?, ?, ?, ?, ?)");) {**  **ps.setInt(1, application.getApplicationID());**  **ps.setInt(2, application.getJobID());**  **ps.setInt(3, application.getApplicantID());**  **ps.setTimestamp(4, Timestamp.valueOf(application.getApplicationDate()));**  **ps.setString(5, application.getCoverLetter());**  **ps.executeUpdate();**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error inserting application", e);**  **}**  **}**  **public List<JobListing> getJobListings() {**  **List<JobListing> jobs = new ArrayList<>();**  **try (Statement stmt = conn.createStatement();**  **ResultSet rs = stmt.executeQuery("SELECT \* FROM Jobs")) {**  **while (rs.next()) {**  **JobListing job = new JobListing(**  **rs.getInt("JobID"),**  **rs.getInt("CompanyID"),**  **rs.getString("JobTitle"),**  **rs.getString("JobDescription"),**  **rs.getString("JobLocation"),**  **rs.getDouble("Salary"),**  **rs.getString("JobType"),**  **rs.getTimestamp("PostedDate").toLocalDateTime()**  **);**  **jobs.add(job);**  **}**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error retrieving job listings", e);**  **}**  **return jobs;**  **}**  **public List<JobListing> searchJobsBySalaryRange(double minSalary, double maxSalary) {**  **List<JobListing> jobs = new ArrayList<>();**  **try (PreparedStatement ps = conn.prepareStatement("SELECT \* FROM Jobs WHERE Salary BETWEEN ? AND ?")) {**  **ps.setDouble(1, minSalary);**  **ps.setDouble(2, maxSalary);**  **ResultSet rs = ps.executeQuery();**  **while (rs.next()) {**  **JobListing job = new JobListing(**  **rs.getInt("JobID"),**  **rs.getInt("CompanyID"),**  **rs.getString("JobTitle"),**  **rs.getString("JobDescription"),**  **rs.getString("JobLocation"),**  **rs.getDouble("Salary"),**  **rs.getString("JobType"),**  **rs.getTimestamp("PostedDate").toLocalDateTime()**  **);**  **jobs.add(job);**  **}**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error searching job by salary range", e);**  **}**  **return jobs;**  **}**  **public List<Applicant> getApplicants() {**    **List<Applicant> apps = new ArrayList<>();**  **try (Statement stmt = conn.createStatement();**  **ResultSet rs = stmt.executeQuery("SELECT \* FROM applicants")) {**  **while (rs.next()) {**  **Applicant app = new Applicant(**  **rs.getInt("ApplicantID"),**  **rs.getString("FirstName"),**  **rs.getString("LastName"),**  **rs.getString("Email"),**  **rs.getString("Phone"),**  **rs.getString("Resume")**    **);**  **apps.add(app);**  **}**  **} catch (SQLException e) {**  **throw new DBConnectionException("Error retrieving job listings", e);**  **}**      **return apps;**  **}**  **}** |

**Package exception**

**ApplicationDeadlineException.java**

|  |
| --- |
| **package exception;**  **public class ApplicationDeadlineException extends RuntimeException {**    **private static final long serialVersionUID = 1L;**  **public ApplicationDeadlineException(String message) {**  **super(message);**  **}**  **}** |

**DBConnectionException.java**

|  |
| --- |
| **package exception;**  **public class DBConnectionException extends RuntimeException {**    **private static final long serialVersionUID = 1L;**  **public DBConnectionException(String message, Throwable cause) {**  **super(message, cause);**  **}**  **}** |

**InvalidEmailException.java**

|  |
| --- |
| **package exception;**  **public class InvalidEmailException extends RuntimeException {**    **private static final long serialVersionUID = 1L;**  **public InvalidEmailException(String message) {**  **super(message);**  **}**  **}** |

**NegativeSalaryException.java**

|  |
| --- |
| **package exception;**  **public class NegativeSalaryException extends RuntimeException {**    **private static final long serialVersionUID = 1L;**  **public NegativeSalaryException(String message) {**  **super(message);**  **}**  **}** |

**ResumeUploadException.java**

|  |
| --- |
| **package exception;**  **public class ResumeUploadException extends RuntimeException {**    **private static final long serialVersionUID = 1L;**  **public ResumeUploadException(String message) {**  **super(message);**  **}**  **}** |

**Package main**

**MainModule.java**

|  |
| --- |
| **package main;**  **import dao.\*;**  **import entity.\*;**  **import java.util.\*;**  **import java.time.LocalDateTime;**  **public class MainModule {**  **public static void main(String[] args) {**  **Scanner scanner = new Scanner(System.in);**  **CareerHubDAO service = new CareerHubDAOImpl();**  **int choice;**  **do {**  **System.out.println("\n--- CareerHub Job Board ---");**  **System.out.println("1. Add Company");**  **System.out.println("2. Add Job Listing");**  **System.out.println("3. Add Applicant");**  **System.out.println("4. Apply for Job");**  **System.out.println("5. View Job Listings");**  **System.out.println("6. View Applicants");**  **System.out.println("7. Search Jobs by Salary Range");**  **System.out.println("0. Exit");**  **System.out.print("Enter choice: ");**  **choice = scanner.nextInt();**  **scanner.nextLine();**  **switch (choice) {**  **case 1:**  **System.out.print("Enter Company ID: ");**  **int cid = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter Company Name: ");**  **String cname = scanner.nextLine();**  **System.out.print("Enter Location: ");**  **String cloc = scanner.nextLine();**  **Company company = new Company(cid, cname, cloc);**  **service.insertCompany(company);**  **System.out.println("Company added successfully.");**  **break;**  **case 2:**  **System.out.print("Enter Job ID: ");**  **int jid = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter Company ID: ");**  **int jcID = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter Job Title: ");**  **String jtitle = scanner.nextLine();**  **System.out.print("Enter Job Description: ");**  **String jdesc = scanner.nextLine();**  **System.out.print("Enter Job Location: ");**  **String jloc = scanner.nextLine();**  **System.out.print("Enter Salary: ");**  **double jsalary = scanner.nextDouble(); scanner.nextLine();**  **System.out.print("Enter Job Type: ");**  **String jtype = scanner.nextLine();**  **JobListing job = new JobListing(jid, jcID, jtitle, jdesc, jloc, jsalary, jtype, LocalDateTime.now());**  **service.insertJobListing(job);**  **System.out.println("Job listing added successfully.");**  **break;**  **case 3:**  **System.out.print("Enter Applicant ID: ");**  **int aid = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter First Name: ");**  **String fname = scanner.nextLine();**  **System.out.print("Enter Last Name: ");**  **String lname = scanner.nextLine();**  **System.out.print("Enter Email: ");**  **String email = scanner.nextLine();**  **System.out.print("Enter Phone: ");**  **String phone = scanner.nextLine();**  **System.out.print("Enter Resume Info: ");**  **String resume = scanner.nextLine();**  **Applicant applicant = new Applicant(aid, fname, lname, email, phone, resume);**  **service.insertApplicant(applicant);**  **System.out.println("Applicant profile created.");**  **break;**  **case 4:**  **System.out.print("Enter Application ID: ");**  **int appid = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter Job ID: ");**  **int ajob = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter Applicant ID: ");**  **int aapp = scanner.nextInt(); scanner.nextLine();**  **System.out.print("Enter Cover Letter: ");**  **String cover = scanner.nextLine();**  **JobApplication application = new JobApplication(appid, ajob, aapp, LocalDateTime.now(), cover);**  **service.insertJobApplication(application);**  **System.out.println("Application submitted.");**  **break;**  **case 5:**  **List<JobListing> jobs = service.getJobListings();**  **for (JobListing jl : jobs) {**  **System.out.println(jl.getCompanyID()+" "+jl.getJobDescription()+" "+jl.getJobID()+" "+jl.getJobLocation()+" "+jl.getJobTitle()+" "+jl.getJobTitle()+" "+jl.getSalary());**  **}**  **break;**  **case 6:**  **List<Applicant> applicants = service.getApplicants();**  **for (Applicant a : applicants) {**  **System.out.println(a.getApplicantID()+" "+a.getFirstName()+" "+a.getEmail());**  **}**  **break;**  **case 7:**  **System.out.print("Enter minimum salary: ");**  **double minSalary = scanner.nextDouble();**  **System.out.print("Enter maximum salary: ");**  **double maxSalary = scanner.nextDouble();**  **scanner.nextLine();**  **List<JobListing> salaryJobs = service.searchJobsBySalaryRange(minSalary, maxSalary);**  **for (JobListing jl : salaryJobs) {**  **System.out.println(jl.getCompanyID()+" "+jl.getJobDescription()+" "+jl.getJobID()+" "+jl.getJobLocation()+" "+jl.getJobTitle()+" "+jl.getJobTitle()+" "+jl.getSalary());**  **}**  **break;**  **case 0:**  **System.out.println("Exiting ...");**  **break;**  **default:**  **System.out.println("Invalid choice. Try again.");**  **}**  **} while (choice != 0);**  **scanner.close();**  **}**  **}** |

**Package util**

**DBConnUtil.java**

|  |
| --- |
| **package util;**  **import java.io.IOException;**  **import java.sql.Connection;**  **import java.sql.DriverManager;**  **import java.sql.SQLException;**  **public class DBConnUtil {**    **private static final String fileName="C:\\Users\\ELCOT\\Workspace\\CareerHub1\\src\\db.properties";**  **public static Connection getDbConnection() {**  **Connection con=null;**  **String connString=null;**  **try {**  **connString=DBPropertyUtil.getConnectionString(fileName);**    **}catch (IOException e) {**  **System.out.println("Connection String Creation Failed");**    **e.printStackTrace();**  **}**  **if(connString!=null)**  **{**  **try**  **{**  **con=DriverManager.getConnection(connString);**    **}**  **catch (SQLException e)**  **{**  **System.out.println("Error While Establishing DBConnection........");**  **e.printStackTrace();**  **}**  **}**  **return con;**  **}**  **}** |

**DBPropertyUtil.java**

|  |
| --- |
| **package util;**  **import java.io.FileInputStream;**  **import java.io.IOException;**  **import java.util.Properties;**  **public class DBPropertyUtil {**      **public static String getConnectionString(String fileName)throws IOException {**    **String connStr=null;**  **Properties props=new Properties();**  **FileInputStream fis=new FileInputStream(fileName);**    **props.load(fis);**  **String user=props.getProperty("user");**  **String password=props.getProperty("password");**  **String protocol=props.getProperty("protocol");**  **String system=props.getProperty("system");**  **String database=props.getProperty("database");**  **String port=props.getProperty("port");**  **connStr=protocol+"//"+system+":"+port+"/"+database+"?user="+user+"&password="+password;**  **return connStr;**  **}**      **}** |

**db.properties**

|  |
| --- |
| **user=root**  **password=@Tamil9988**  **protocol=jdbc:mysql:**  **system=localhost**  **database=CareerHub**  **port=3306** |

DATABASE CREATED DURING SQLCODING CHALLENGE:

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS CareerHub;  USE CareerHub;  CREATE TABLE IF NOT EXISTS Companies (  CompanyID INT PRIMARY KEY,  CompanyName VARCHAR(255) NOT NULL,  Location VARCHAR(255) NOT NULL  );  SELECT \* FROM Companies;  CREATE TABLE IF NOT EXISTS Jobs (  JobID INT PRIMARY KEY,  CompanyID INT,  JobTitle VARCHAR(255) NOT NULL,  JobDescription TEXT NOT NULL,  JobLocation VARCHAR(255) NOT NULL,  Salary DECIMAL(10,2),  JobType VARCHAR(50) NOT NULL,  PostedDate DATETIME DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID) ON DELETE CASCADE  );  CREATE TABLE IF NOT EXISTS Applicants (  ApplicantID INT PRIMARY KEY,  FirstName VARCHAR(100) NOT NULL,  LastName VARCHAR(100) NOT NULL,  Email VARCHAR(255) UNIQUE NOT NULL,  Phone VARCHAR(20) NOT NULL,  Resume TEXT NOT NULL  );  CREATE TABLE IF NOT EXISTS Applications (  ApplicationID INT PRIMARY KEY,  JobID INT,  ApplicantID INT,  ApplicationDate DATETIME DEFAULT CURRENT\_TIMESTAMP,  CoverLetter TEXT,  FOREIGN KEY (JobID) REFERENCES Jobs(JobID) ON DELETE CASCADE,  FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID) ON DELETE CASCADE  );  INSERT INTO Companies (CompanyID, CompanyName, Location) VALUES  (1, 'Hexaware', 'France'),  (2, 'Microsoft', 'Pune'),  (3, 'Amazon', 'Chennai'),  (4, 'Facebook', 'Europe'),  (5, 'Apple', 'Paris'),  (6, 'Hexaware', 'Chennai'),  (7, 'Microsoft', 'New York'),  (8, 'Cognizant', 'Banglore'),  (9, 'Facebook', 'Chicago'),  (10, 'Apple', 'Germany');  INSERT INTO Jobs (JobID, CompanyID, JobTitle, JobDescription, JobLocation, Salary, JobType, PostedDate) VALUES  (1, 1, 'Software Engineer', 'Develop and maintain software.', 'France', 120000.00, 'Contract', NOW()),  (2, 2, 'Software Engineer', 'Develop applications.', 'Pune', 115000.00, 'Full-time', NOW()),  (3, 3, 'Data Scientist', 'Analyze data.', 'Chennai', 110000.00, 'Full-time', NOW()),  (4, 4, 'Cloud Engineer', 'Manage cloud solutions.', 'Europe', 118000.00, 'Full-time', NOW()),  (5, 5, 'UI/UX Designer', 'Design user interfaces.', 'Paris', 65000.00, 'Part-time', NOW()),  (6, 6, 'Software Engineer', 'Write backend code.', 'Chennai', 122000.00, 'Full-time', NOW()),  (7, 7, 'Cloud Engineer', 'Deploy cloud applications.', 'New York', 77000.00, 'Full-time', NOW()),  (8, 8, 'Data Scientist', 'Analyze business data.', 'Banglore', 112000.00, 'Full-time', NOW()),  (9, 9, 'Software Developer', 'Develop system software.', 'Chicago', 119000.00, 'Part-time', NOW()),  (10, 10, 'UI/UX Designer', 'Improve user experience.', 'Germany', 80000.00, 'Contract', NOW());  INSERT INTO Applicants (ApplicantID, FirstName, LastName, Email, Phone, Resume) VALUES  (1, 'Aara', 'Johnson', 'aara@example.com', '1234567890', 'Resume\_Aara'),  (2, 'Babu', 'Smith', 'babu@example.com', '1234567891', 'Resume\_Babu'),  (3, 'Durga', 'Devi', 'durgadevi@example.com', '1234567892', 'Resume\_Durga'),  (4, 'Ram', 'Kumara', 'ramkumar@example.com', '1234567893', 'Resume\_Ram'),  (5, 'Smriti', 'Davis', 'davis@example.com', '1234567894', 'Resume\_Smriti'),  (6, 'Arun', 'Johnson', 'arunjohn@example.com', '1234567895', 'Resume\_Arun'),  (7, 'Bob', 'Smith', 'bobsmithp@example.com', '1234567896', 'Resume\_Bob\_Dup'),  (8, 'Frank', 'Miller', 'frank@example.com', '1234567897', 'Resume\_Frank'),  (9, 'Durga', 'Devi', 'durga\_dup@example.com', '1234567898', 'Resume\_Durga\_devi'),  (10, 'Jack', 'Taylor', 'jack@example.com', '1234567899', 'Resume\_Jack');  INSERT INTO Applications (ApplicationID, JobID, ApplicantID, ApplicationDate, CoverLetter) VALUES  (1, 1, 1, NOW(), 'Cover Letter for Software Engineer at Hexaware'),  (2, 2, 1, NOW(), 'Cover Letter for Software Engineer at Microsoft'),  (3, 3, 2, NOW(), 'Cover Letter for Data Scientist at Amazon'),  (4, 1, 3, NOW(), 'Cover Letter for Data Scientist at Amazon'),  (5, 4, 4, NOW(), 'Cover Letter for Cloud Engineer at Facebook'),  (6, 5, 5, NOW(), 'Cover Letter for UI/UX Designer at Apple'),  (7, 8, 6, NOW(), 'Cover Letter for Software Engineer at Cognizant'),  (8, 7, 7, NOW(), 'Cover Letter for Cloud Engineer at Microsoft (NY)'),  (9, 6, 8, NOW(), 'Cover Letter for Data Scientist at Hexaware (Chennai)'),  (10, 9, 9, NOW(), 'Cover Letter for Data Scientist at Facebook (Chicago)'),  (11, 10, 10, NOW(), 'Cover Letter for UI/UX Designer at Apple (Germany)'),  (12, 2, 5, NOW(), 'Cover Letter for Software Engineer at Microsoft'),  (13, 4, 3, NOW(), 'Cover Letter for Cloud Engineer at Facebook'),  (14, 5, 1, NOW(), 'Cover Letter for UI/UX Designer at Apple'),  (15, 10, 7, NOW(), 'Cover Letter for UI/UX Designer at Apple (Paris)'); |