**JAVA CODING CHALLENGE**

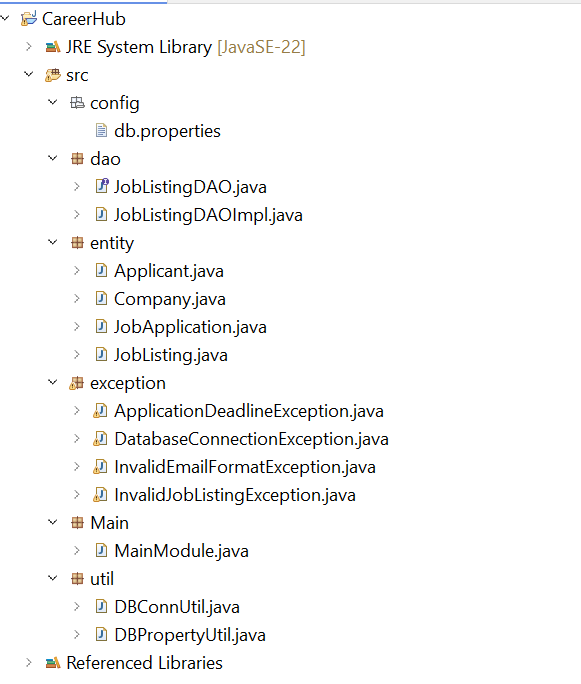
Coding Challenges: CareerHub, The Job Board

BY,

POOJA SRI B

Mail ID: [poojasribk@gmail.com](mailto:poojasribk@gmail.com)

**DIRECTORY STRUCTURE:**

****

**1. JobListingClass**

**File:** JobListingClass.java

This class represents a job listing in the CareerHub system. It contains details about a particular job posting, including the job ID, company ID, job title, description, location, salary, type, and the date it was posted. The constructor initializes these details when a new job listing is created, and the class provides getter and setter methods for each attribute to enable access and modification of these details.

**Key Attributes:**

* **jobID:** A unique identifier for the job.
* **companyID:** The unique identifier for the company posting the job.
* **jobTitle:** The title of the job being offered.
* **jobDescription:** A description of the job duties and responsibilities.
* **jobLocation:** The location of the job.
* **salary:** The salary being offered for the job.
* **jobType:** The type of job (e.g., full-time, part-time).
* **postedDate:** The date when the job was posted.
* **companyName:** The name of the company offering the job.

**Constructor:**

The constructor accepts all the attributes and initializes them when a new object of JobListingClass is created.

**MYCODING:**

**package** entity;

**import** java.time.LocalDate;

**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** LocalDate postedDate;

// Getters and Setters

**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** LocalDate getPostedDate() {

**return** postedDate;

}

**public** **void** setPostedDate(LocalDate postedDate) {

**this**.postedDate = postedDate;

}

@Override

**public** String toString() {

**return** "Job ID: " + jobID + ", " +

"Company ID: " + companyID + ", " +

"Job Title: " + jobTitle + ", " +

"Job Description: " + jobDescription + ", " +

"Job Location: " + jobLocation + ", " +

"Salary: $" + salary + ", " +

"Job Type: " + jobType + ", " +

"Posted Date: " + postedDate;

}

}

**2. Company**

**File:** Company.java

This class represents a company in the Career hub system. Each company has a unique ID, a name, and a location. The constructor initializes these details, and getter and setter methods allow interaction with the class fields.

**Key Attributes:**

* **companyID:** A unique identifier for the company.
* **companyName:** The name of the company.
* **location:** The location where the company operates.

**MYCODE:**

**package entity;**

**public class Company {**

**private int companyID;**

**private String companyName;**

**private String location;**

**// Getters and Setters**

**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;**

**}**

**}**

**3. Applicant**

**File:** Applicant.java

This class represents an applicant in the CareerHub system. It includes personal details like the applicant’s ID, first name, last name, email, phone number, and resume. These attributes are essential for storing the applicant's information in the system.

**Key Attributes:**

* **applicantID:** A unique identifier for the applicant.
* **firstName:** The applicant's first name.
* **lastName:** The applicant's last name.
* **email:** The applicant's email address.
* **phone:** The applicant’s phone number.
* **resume:** The applicant's resume, typically stored as a file or a link.

**MYCODE:**

**package** entity;

**public** **class** Applicant {

**private** **int** applicantID;

**private** String firstName;

**private** String lastName;

**private** String email;

**private** String phone;

**private** String resume;

// Getters and Setters

**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;

}

}

**4. Jobapplications**

**File:** Jobapplications.java

This class represents an application submitted by an applicant for a specific job. It includes the application ID, job ID, applicant ID, application date, and a cover letter. The constructor initializes these attributes, and getter and setter methods allow access to and modification of the attributes.

**Key Attributes:**

* **applicationID:** A unique identifier for the application.
* **jobID:** The ID of the job for which the applicant has applied.
* **applicantID:** The ID of the applicant submitting the application.
* **applicationDate:** The date when the application was submitted, automatically set to the current date.
* **coverLetter:** The cover letter provided by the applicant as part of their application.

MYCODING:

**package** entity;

**import** java.time.LocalDate;

**public** **class** JobApplication {

**private** **int** applicationID;

**private** **int** jobID;

**private** **int** applicantID;

**private** LocalDate applicationDate;

**private** String coverLetter;

// Getters and Setters

**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** LocalDate getApplicationDate() {

**return** applicationDate;

}

**public** **void** setApplicationDate(LocalDate applicationDate) {

**this**.applicationDate = applicationDate;

}

**public** String getCoverLetter() {

**return** coverLetter;

}

**public** **void** setCoverLetter(String coverLetter) {

**this**.coverLetter = coverLetter;

}

}

**5. Main (Driver Class)**

**File:** Main.java

This is the driver class where the main execution of the CareerHub system starts. It serves as a test class that interacts with the DAO (Data Access Object) implementation. In this class, the following actions are performed:

* A new job listing is added to the system.
* All job listings are fetched and displayed.
* An existing job listing's salary is updated.
* A job listing is deleted from the system.

This class serves as the starting point to demonstrate how the data is being handled in the CareerHub system. It uses the JobListingDaoImplementation to interact with the database and perform CRUD operations on job listings.

MYCODING:

**package** Main;

**import** java.util.Scanner;

**import** dao.\*;

**import** entity.\*;

**import** exception.\*;

**public** **class** MainModule {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

JobListingDAO jobListingDAO = **new** JobListingDAOImpl();

**while** (**true**) {

System.***out***.println("\n=== Job Board Menu ===");

System.***out***.println("1. Insert Job Listing");

System.***out***.println("2. Show All Job Listings");

System.***out***.println("31. Exit");

System.***out***.print("Enter your choice: ");

**int** choice = scanner.nextInt();

scanner.nextLine(); // Consume newline character

**switch** (choice) {

**case** 1:

// Insert Job Listing

System.***out***.println("\nEnter Job Details:");

System.***out***.print("Company ID: ");

**int** companyID = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.***out***.print("Job Title: ");

String jobTitle = scanner.nextLine();

System.***out***.print("Job Description: ");

String jobDescription = scanner.nextLine();

System.***out***.print("Job Location: ");

String jobLocation = scanner.nextLine();

System.***out***.print("Salary: ");

**double** salary = scanner.nextDouble();

scanner.nextLine(); // Consume newline

System.***out***.print("Job Type (Full-time/Part-time): ");

String jobType = scanner.nextLine();

JobListing jobListing = **new** JobListing();

jobListing.setCompanyID(companyID);

jobListing.setJobTitle(jobTitle);

jobListing.setJobDescription(jobDescription);

jobListing.setJobLocation(jobLocation);

jobListing.setSalary(salary);

jobListing.setJobType(jobType);

**try** {

jobListingDAO.insertJobListing(jobListing);

System.***out***.println("Job Listing Inserted Successfully!");

} **catch** (DatabaseConnectionException e) {

System.***out***.println("Error: " + e.getMessage());

}

**break**;

**case** 2:

// Show All Job Listings

**try** {

**var** jobListings = jobListingDAO.getAllJobListings();

System.***out***.println("\n=== Job Listings ===");

**for** (JobListing listing : jobListings) {

System.***out***.println(listing);

}

} **catch** (DatabaseConnectionException e) {

System.***out***.println("Error: " + e.getMessage());

}

**break**;

**case** 3:

// Exit

System.***out***.println("Exiting...");

scanner.close();

**return**;

**default**:

System.***out***.println("Invalid choice! Please try again.");

}

}

}

}

**6. DBproperty**

**File:** DBproperty.java

This class handles the configuration and connection string for connecting to the database. It reads database properties (host, port, user, password, etc.) from a db.properties file and constructs a JDBC connection string. The getConnectionString() method is used to read the properties and return the connection URL for the MySQL database.

**Method:**

* **getConnectionString:** Reads the properties from the file and returns a MySQL connection string to be used for connecting to the database.

**MYCODING:**

**package util;**

**import java.io.FileInputStream;**

**import java.io.IOException;**

**import java.io.InputStream;**

**import java.util.Properties;**

**public class DBPropertyUtil {**

**private static final String *propertyfile* = "src/config/db.properties"; // Path to your db.properties file**

**// Method to load database properties from the db.properties file**

**public static Properties loadProperties() {**

**Properties properties = new Properties();**

**try (InputStream input = new FileInputStream(*propertyfile*)) {**

**properties.load(input);**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**return properties;**

**}**

**}**

**DBCONUTIL CLASS:**

**package** util;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.util.Properties;

**import** exception.DatabaseConnectionException;

**public** **class** DBConnUtil {

// Method to establish a connection with the database

**public** **static** Connection getConnection() **throws** DatabaseConnectionException {

Properties properties = DBPropertyUtil.*loadProperties*(); // Load the database properties

String dbURL = properties.getProperty("db.url");

String dbUser = properties.getProperty("db.user");

String dbPassword = properties.getProperty("db.password");

**try** {

// Load the database driver (for MySQL)

Class.*forName*("com.mysql.cj.jdbc.Driver");

**return** DriverManager.*getConnection*(dbURL, dbUser, dbPassword);

} **catch** (ClassNotFoundException | SQLException e) {

**throw** **new** DatabaseConnectionException("Error connecting to the database", e);

}

}

// Method to close the database connection

**public** **static** **void** closeConnection(Connection connection) {

**if** (connection != **null**) {

**try** {

connection.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

}

}

}

**7. JobListingDAO Interface**

**File:** JobListingDAO.java

This interface defines the basic operations that can be performed on job listings in the CareerHub system. It includes methods for adding, retrieving, updating, and deleting job listings. The implementation of these methods will interact with the database to perform the required operations.

**Methods:**

* **addJobListing:** Adds a new job listing to the database.
* **getAllJobListings:** Retrieves a list of all job listings.
* **getJobById:** Retrieves a job listing by its ID.
* **updateJobListing:** Updates an existing job listing.
* **deleteJobListing:** Deletes a job listing based on its ID.

**MYCODING:**

**package dao;**

**import entity.JobListing;**

**import exception.DatabaseConnectionException;**

**import java.util.List;**

**public interface JobListingDAO {**

**void insertJobListing(JobListing jobListing) throws DatabaseConnectionException;**

**List<JobListing> getAllJobListings() throws DatabaseConnectionException;**

**}**

**8. JobListingDaoImplementation**

**File:** JobListingDaoImplementation.java

This class implements the JobListingDAO interface and provides the actual database interaction for job listings. It uses JDBC to connect to the database and execute SQL queries for adding, retrieving, updating, and deleting job listings.

**Methods:**

* **addJobListing:** Executes an INSERT SQL query to add a new job listing to the job\_listings table.
* **getAllJobListings:** Executes a SELECT SQL query to fetch all job listings from the job\_listings table and returns a list of JobListingClass objects.
* **getJobById:** Executes a SELECT SQL query to fetch a specific job listing by its ID.
* **updateJobListing:** Executes an UPDATE SQL query to modify the details of an existing job listing.
* **deleteJobListing:** Executes a DELETE SQL query to remove a job listing from the job\_listings table.

**Mycoding:**

**package** dao;

**import** entity.JobListing;

**import** exception.DatabaseConnectionException;

**import** java.util.List;

**import** java.util.ArrayList;

**public** **class** JobListingDAOImpl **implements** JobListingDAO {

**private** **static** List<JobListing> *jobListings* = **new** ArrayList<>();

@Override

**public** **void** insertJobListing(JobListing jobListing) **throws** DatabaseConnectionException {

// Simulate inserting to a database

*jobListings*.add(jobListing);

System.***out***.println("Job Listing Inserted into Database.");

}

@Override

**public** List<JobListing> getAllJobListings() **throws** DatabaseConnectionException {

**return** *jobListings*; // Return the list of job listings (this can be modified to fetch from a database)

}

}

**EXCEPTION CLASSES:**

**ApplicationDeadlineException**

**package exception;**

**public class ApplicationDeadlineException extends Exception {**

**public ApplicationDeadlineException(String message) {**

**super(message);**

**}**

**}**

**DatabaseConnectionException**

**package exception;**

**public class DatabaseConnectionException extends Exception {**

**public DatabaseConnectionException(String message) {**

**super(message);**

**}**

**public DatabaseConnectionException(String message, Throwable cause) {**

**super(message, cause);**

**}**

**}**

**InvalidEmailFormatException**

**package exception;**

**public class InvalidEmailFormatException extends Exception {**

**public InvalidEmailFormatException(String message) {**

**super(message);**

**}**

**}**

**InvalidJobListingException**

**package exception;**

**public class InvalidJobListingException extends Exception {**

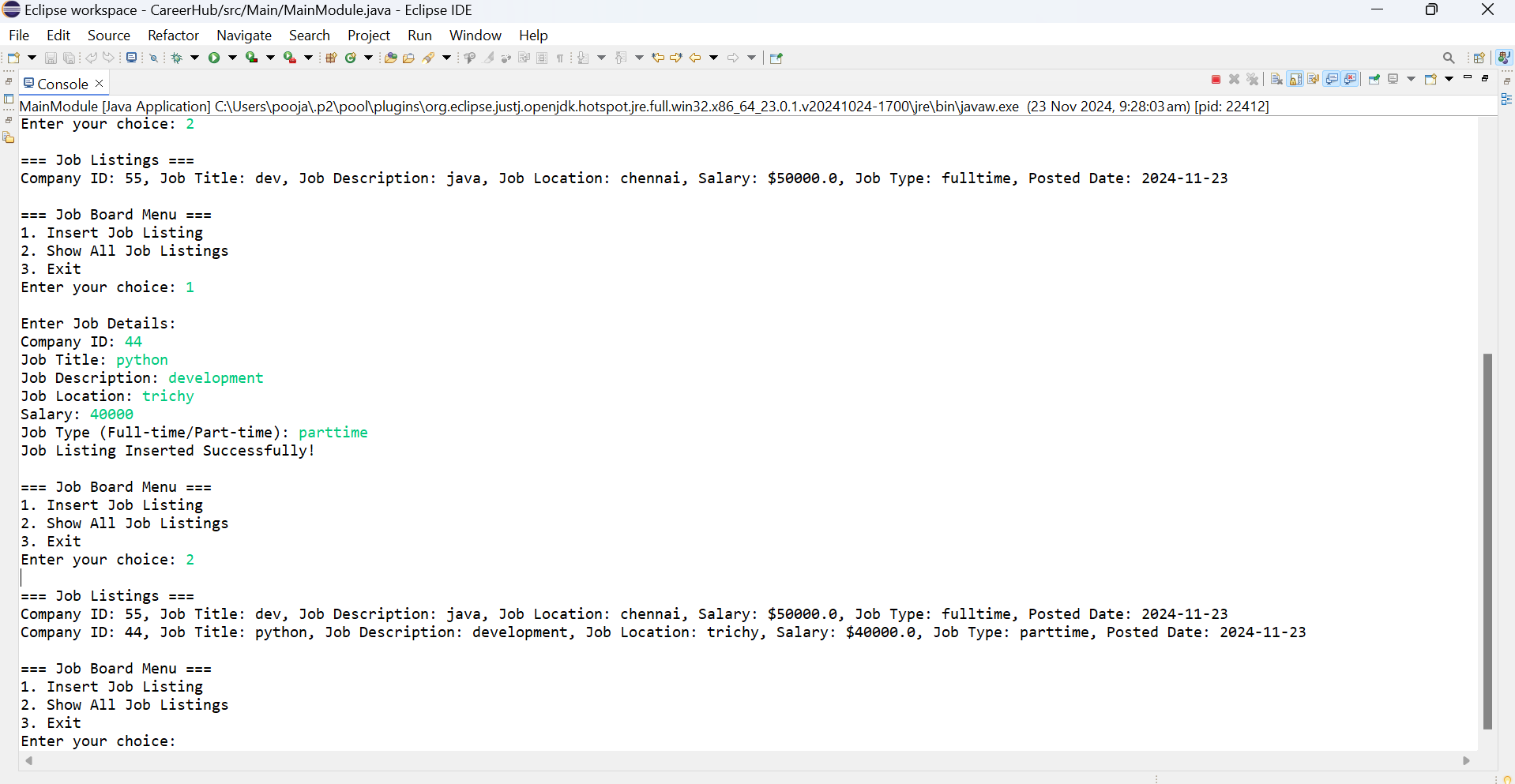
**public InvalidJobListingException(String message) {**

**super(message);**

**}**

**}**

**SAMPLE OUTPUT:**

****

**OUTPUT FROM DATABASE:**

**A screenshot of a computer

Description automatically generated**

*// entity/model/JobListing.java*

package entity;

import java.time.LocalDateTime;

import java.util.List;

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;

private LocalDateTime applicationDeadline;

*// Constructor*

public JobListing(int jobId, int companyId, String jobTitle, String jobDescription,

String jobLocation, double salary, String jobType) {

this.jobId = jobId;

this.companyId = companyId;

this.jobTitle = jobTitle;

this.jobDescription = jobDescription;

this.jobLocation = jobLocation;

this.salary = salary;

this.jobType = jobType;

this.postedDate = LocalDateTime.now();

this.applicationDeadline = LocalDateTime.now().plusDays(30); *// Default 30 days deadline*

}

*// Getters and Setters*

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 LocalDateTime getApplicationDeadline() { return applicationDeadline; }

public void setApplicationDeadline(LocalDateTime deadline) { this.applicationDeadline = deadline; }

}

*// entity/model/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;

}

*// Getters and Setters*

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; }

}

*// entity/model/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 and Setters*

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; }

}

*// entity/model/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,

String coverLetter) {

this.applicationId = applicationId;

this.jobId = jobId;

this.applicantId = applicantId;

this.applicationDate = LocalDateTime.now();

this.coverLetter = coverLetter;

}

*// Getters and Setters*

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 String getCoverLetter() { return coverLetter; }

public void setCoverLetter(String coverLetter) { this.coverLetter = coverLetter; }

}

*// exception/InvalidEmailException.java*

package exception;

public class InvalidEmailException extends Exception {

public InvalidEmailException(String message) {

super(message);

}

}

*// exception/InvalidSalaryException.java*

package exception;

public class InvalidSalaryException extends Exception {

public InvalidSalaryException(String message) {

super(message);

}

}

*// exception/FileUploadException.java*

package exception;

public class FileUploadException extends Exception {

public FileUploadException(String message) {

super(message);

}

}

*// exception/ApplicationDeadlineException.java*

package exception;

public class ApplicationDeadlineException extends Exception {

public ApplicationDeadlineException(String message) {

super(message);

}

}

*// exception/DatabaseException.java*

package exception;

public class DatabaseException extends Exception {

public DatabaseException(String message) {

super(message);

}

}

*// util/DBPropertyUtil.java*

package util;

import java.io.FileInputStream;

import java.util.Properties;

public class DBPropertyUtil {

public static String getConnectionString(String propertyFileName) throws Exception {

Properties props = new Properties();

try (FileInputStream fis = new FileInputStream(propertyFileName)) {

props.load(fis);

return String.format("jdbc:mysql://%s:%s/%s",

props.getProperty("db.host"),

props.getProperty("db.port"),

props.getProperty("db.name"));

}

}

}

*// util/DBConnUtil.java*

package util;

import java.sql.Connection;

import java.sql.DriverManager;

public class DBConnUtil {

public static Connection getConnection(String connectionString) throws Exception {

return DriverManager.getConnection(connectionString);

}

}

*// dao/IJobBoardService.java*

package dao;

import entity.\*;

import exception.\*;

import java.util.List;

public interface IJobBoardService {

void createApplicantProfile(Applicant applicant) throws InvalidEmailException, DatabaseException;

void postJob(JobListing job) throws InvalidSalaryException, DatabaseException;

void submitApplication(JobApplication application) throws ApplicationDeadlineException, DatabaseException;

List<JobListing> getJobListings() throws DatabaseException;

List<JobListing> getJobsByCompany(int companyId) throws DatabaseException;

List<JobListing> getJobsBySalaryRange(double minSalary, double maxSalary) throws DatabaseException;

void uploadResume(int applicantId, String filePath) throws FileUploadException;

}

*// dao/JobBoardServiceImpl.java*

package dao;

import entity.\*;

import exception.\*;

import util.\*;

import java.sql.\*;

import java.util.\*;

import java.nio.file.\*;

import java.time.LocalDateTime;

import java.util.regex.Pattern;

public class JobBoardServiceImpl implements IJobBoardService {

private Connection conn;

private static final String EMAIL\_REGEX = "^[A-Za-z0-9+\_.-]+@(.+)$";

public JobBoardServiceImpl(String propertyFile) throws DatabaseException {

try {

String connString = DBPropertyUtil.getConnectionString(propertyFile);

conn = DBConnUtil.getConnection(connString);

} catch (Exception e) {

throw new DatabaseException("Failed to initialize database connection: " + e.getMessage());

}

}

@Override

public void createApplicantProfile(Applicant applicant) throws InvalidEmailException, DatabaseException {

if (!Pattern.compile(EMAIL\_REGEX).matcher(applicant.getEmail()).matches()) {

throw new InvalidEmailException("Invalid email format: " + applicant.getEmail());

}

String sql = "INSERT INTO Applicants (firstName, lastName, email, phone, resume) VALUES (?, ?, ?, ?, ?)";

try (PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setString(1, applicant.getFirstName());

stmt.setString(2, applicant.getLastName());

stmt.setString(3, applicant.getEmail());

stmt.setString(4, applicant.getPhone());

stmt.setString(5, applicant.getResume());

stmt.executeUpdate();

} catch (SQLException e) {

throw new DatabaseException("Failed to create applicant profile: " + e.getMessage());

}

}

@Override

public void postJob(JobListing job) throws InvalidSalaryException, DatabaseException {

if (job.getSalary() < 0) {

throw new InvalidSalaryException("Salary cannot be negative: " + job.getSalary());

}

String sql = "INSERT INTO Jobs (companyId, jobTitle, jobDescription, jobLocation, salary, jobType, postedDate) " +

"VALUES (?, ?, ?, ?, ?, ?, ?)";

try (PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, job.getCompanyId());

stmt.setString(2, job.getJobTitle());

stmt.setString(3, job.getJobDescription());

stmt.setString(4, job.getJobLocation());

stmt.setDouble(5, job.getSalary());

stmt.setString(6, job.getJobType());

stmt.setTimestamp(7, Timestamp.valueOf(job.getPostedDate()));

stmt.executeUpdate();

} catch (SQLException e) {

throw new DatabaseException("Failed to post job: " + e.getMessage());

}

}

@Override

public void submitApplication(JobApplication application) throws ApplicationDeadlineException, DatabaseException {

*// Check deadline*

try {

String sql = "SELECT applicationDeadline FROM Jobs WHERE jobId = ?";

try (PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, application.getJobId());

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

LocalDateTime deadline = rs.getTimestamp("applicationDeadline").toLocalDateTime();

if (LocalDateTime.now().isAfter(deadline)) {

throw new ApplicationDeadlineException("Application deadline has passed");

}

}

}

*// Submit application*

sql = "INSERT INTO Applications (jobId, applicantId, applicationDate, coverLetter) VALUES (?, ?, ?, ?)";

try (PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, application.getJobId());

stmt.setInt(2, application.getApplicantId());

stmt.setTimestamp(3, Timestamp.valueOf(application.getApplicationDate()));

stmt.setString(4, application.getCoverLetter());

stmt.executeUpdate();

}

} catch (SQLException e) {

throw new DatabaseException("Failed to submit application: " + e.getMessage());

}

}

@Override

public List<JobListing> getJobListings() throws DatabaseException {

List<JobListing> jobs = new ArrayList<>();

String sql = "SELECT \* FROM Jobs";

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

while (rs.next()) {

jobs.add(new JobListing(

rs.getInt("jobId"),

rs.getInt("companyId"),

rs.getString("jobTitle"),

rs.getString("jobDescription"),

rs.getString("jobLocation"),

rs.getDouble("salary"),

rs.getString("jobType")

));

}

} catch (SQLException e) {

throw new DatabaseException("Failed to retrieve job listings: " + e.getMessage());

}

return jobs;

}

@Override

public List<JobListing> getJobsByCompany(int companyId) throws DatabaseException {

List<JobListing> jobs = new ArrayList<>();

String sql = "SELECT \* FROM Jobs WHERE companyId = ?";

try (PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, companyId);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

jobs.add(new JobListing(

rs.getInt("jobId"),

rs.getInt("companyId"),

rs.getString("jobTitle"),

rs.getString("jobDescription"),

rs.getString("jobLocation"),

rs.getDouble("salary"),

rs.getString("jobType")

));

}

} catch (SQLException e) {

throw new DatabaseException("Failed to retrieve company jobs: " + e.getMessage());

}

return jobs;

}

@Override

public List<JobListing> getJobsBySalaryRange(double minSalary, double maxSalary) throws DatabaseException {

List<JobListing> jobs = new ArrayList<>();

String sql = "SELECT \* FROM Jobs WHERE salary BETWEEN ? AND ?";

try (PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setDouble(1, minSalary);

stmt.setDouble(2, maxSalary);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

jobs.add(new JobListing(

rs.getInt("jobId"),

rs.getInt("companyId"),

rs.getString("jobTitle"),

rs.getString("jobDescription"),

rs.getString("jobLocation"),

rs.getDouble("salary"),

rs.getString("jobType")

));

}

} catch (SQLException e) {

throw new DatabaseException("Failed to retrieve jobs by salary range: " + e.getMessage());

}

return jobs;

}

@Override