CODE CONTRIBUTION

ADHARSH RENGARAJAN & XINDUO FAN

OpenAIService.java

```
public class OpenAIService {

    public String getScoreAndFeedback(String resumeContent, String jobDescription) throws Exception {
        WordReader word=new WordReader();
        String resumeContentTest=word.getResumeContent("src/main/java/com/edu/neu/csye6200/resume/sampleResume.docx");
        System.out.println(resumeContentTest);

        String finalPrompt = String.format("""
                    Your task is to provide a resume score and feedback in this exact format: score#feedback
        Rules:
        1. Score must be between 0-100 with up to 2 decimal places
        2. Feedback must be exactly 20 words
        3. Use only a single # to separate score and feedback
        4. No additional text or sections allowed
        5. Focus on technical abilities and job suitability

        Example response:
        90.25#The candidate's profile is specialised in Java Springboot. His internship in full-stack development could be a major factor to consider

        Resume to analyze:
        %s

        Job Description:
        %s

        Provide your score#feedback response:""", resumeContent, jobDescription);

        String jsonBody = String.format("""
            {
                "model": "meta-llama/llama-3.1-70b-instruct:free",
                "messages": [
                    {
                        "role": "user",
                        "content": "%s"
                    }
                ],
                "temperature":0.56
            }""", finalPrompt.replace("\"", "\\\"").replace("\n", "\\n").replace("\r", "\\r"));

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(BASE_URL + "/chat/completions"))
            .header("Content-Type", "application/json")
```

ScanResume.java

```java
package edu.neu.csye6200.services;

import java.io.IOException;

interface ScanResume {
        public String getResumeContent(String fileLocation) throws IOException;
}
```

ScanResumeFactory.java

```java
package edu.neu.csye6200.services;

import java.io.IOException;

import edu.neu.csye6200.models.Candidate;

public class ScanResumeFactory {

        public String getResumeContent(String file) throws IOException
        {
                if(file.endsWith(".docx"))
                {
                        WordReader reader = new WordReader();
                        return reader.getResumeContent(file);
                }
                else if(file.endsWith(".pdf"))
                {
                        PdfReader reader = new PdfReader();
                        return reader.getResumeContent(file);
                }
                else {
                        return "";
                }
        }

}
```

PdfReader.java

```java
import java.io.File;
import java.io.IOException;

import org.apache.pdfbox.Loader;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;

class PdfReader implements ScanResume{

	@Override
	public String getResumeContent(String fileLocation) throws IOException {
		String resumeContent="";
		PDDocument doc = new PDDocument();
		try
		{
		   File file=new File("src/main/java/com/edu/neu/csye6200/resume/example_resume3.pdf");
		   PDFTextStripper stripper=new PDFTextStripper();
		   doc = Loader.loadPDF(file);
		   resumeContent=stripper.getText(doc);
		   System.out.println("Working");
		}
		finally
		{
		   if( doc != null )
		   {
		      doc.close();
		   }
		}
		return resumeContent;
	}

}
```

WordReader.java

```java
package edu.neu.csye6200.services;

import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.poifs.filesystem.POIFSFileSystem;
import org.apache.poi.hpsf.DocumentSummaryInformation;
import org.apache.poi.hwpf.HWPFDocument;
import org.apache.poi.hwpf.extractor.WordExtractor;
import org.apache.poi.xwpf.extractor.XWPFWordExtractor;
import org.apache.poi.xwpf.usermodel.XWPFDocument;

class WordReader implements ScanResume {

    @Override
    public String getResumeContent(String fileLocation) throws IOException {
        String filePath = "src/main/java/com/edu/neu/csye6200/resume/sampleResume.docx";
        String resumeContent = "";

        try (FileInputStream fis = new FileInputStream(filePath)) {
            try {
                XWPFDocument document = new XWPFDocument(fis);
                XWPFWordExtractor extractor = new XWPFWordExtractor(document);
                resumeContent = extractor.getText();
                extractor.close();
                document.close();
            } catch (org.apache.poi.openxml4j.exceptions.OLE2NotOfficeXmlFileException ole2Exception) {
                fis.getChannel().position(0);
                POIFSFileSystem fs = new POIFSFileSystem(fis);
                HWPFDocument document = new HWPFDocument(fs);
                WordExtractor extractor = new WordExtractor(document);
                resumeContent = extractor.getText();
                extractor.close();
                document.close();
                fs.close();
            }
        }

        return resumeContent;
    }
}
```

JAYANTH MANI

Display.java

```java
package edu.neu.csye6200.services;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

import edu.neu.csye6200.models.Candidate;

/*
1. Class and Object:

Classes are defined for Candidate, HiringManager, and Display. Objects are created from these classes,
representing individual candidates.
The Candidate class represents each candidate with properties like candidateFirstName, candidateLastName, etc.

2. Inheritance:

The Candidate and HiringManager classes inherit from the Person class. This demonstrates inheritance, allowing common
properties like username, password, etc., to be shared.

3. Encapsulation:

Data members in the Candidate class are private (not shown here, but inferred from typical OOP best practices), and
they are accessed via public getters and setters. This ensures encapsulation, which protects the internal state of
the object.

4. Polymorphism:

Method Overriding: The Comparator interface is used to create a custom sorting mechanism, demonstrating method overriding
in an anonymous inner class in the sorting options.

Lambda Expressions: The use of lambda expressions in sorting (e.g., sorting by score) shows a form of polymorphism by
providing different implementations for sorting behavior.

5. Abstraction:

The use of abstract classes or interfaces such as Comparator abstracts away the implementation details of sorting. For
```

```java
public class Display {
    public static void displayCandidates(List<Candidate> candidates, List<Candidate> openApiCandidates, String jobDescription) {
        try (Scanner scanner = new Scanner(System.in)) {
            // Merging the two candidate lists
            List<Candidate> allCandidates = new ArrayList<>(candidates);
            allCandidates.addAll(openApiCandidates); // need to check with team - confused about this

            // Sort candidates by openapi score (candidateScore)
            allCandidates.sort((c1, c2) -> Integer.compare(c2.getCandidateScore(), c1.getCandidateScore()));

            System.out.println("Select an option to sort/filter candidates: Enter a Valid Number");
            System.out.println("1. Sort by First Name (Comparator)");
            System.out.println("2. Sort by Last Name (Stream API)");
            System.out.println("3. Sort by OpenAPI Score (Lambda)");
            System.out.println("4. Sort by Age (Comparable)");
            System.out.println("5. Update Job Description");
            int choice = scanner.nextInt();
            scanner.nextLine();  // Consume newline

            switch (choice) {
                case 1:
                    allCandidates.sort(Comparator.comparing(Candidate::getCandidateFirstName));
                    break;
                case 2:
                    allCandidates = allCandidates.stream()
                            .sorted(Comparator.comparing(Candidate::getCandidateLastName))
                            .collect(Collectors.toList());
                    break;
                case 3:
                    allCandidates.sort((c1, c2) -> Integer.compare(c2.getCandidateScore(), c1.getCandidateScore()));
                    break;
                case 4:
                    Collections.sort(allCandidates, new Comparator<Candidate>() {
                        @Override
                        public int compare(Candidate c1, Candidate c2) {
                            return Integer.compare(c1.getCandidateAge(), c2.getCandidateAge());
                        }
                    });
                    break;
                case 5:
                    System.out.println("Enter new job description:");
                    String newJobDescription = scanner.nextLine();
```

Zuoyu Wang & Sai Kalyan Burra

Person.java

```
ublic class Person {
        String username;
        String password;
        String type;
         long number;
        String emailId;

        // Constructor
        public Person(String username, String password, String type, long number, String emailId) {
                this.username = username;
                this.password = password;
                this.type = type;
                this.number = number;
                this.emailId = emailId;
        }


        //Setters and Getters
        public String getUsername() {
                return username;
        }


        public void setUsername(String username) {
                this.username = username;
        }


        public String getPassword() {
                return password;
        }


        public void setPassword(String password) {
                this.password = password;
        }


        public String getType() {
                return type;
```

Candidate.java

```java
public class Candidate extends Person {
	String candidateFirstName;
	String candidateLastName;
	String candidateLocation;
	int candidateAge;
	int candidateId;
	int candidateScore;
	String candidateFeedBack;
	String jobDesription;
	int getCandidateId;


	//Constructor
	public Candidate(String username, String password, String type, long number, String emailId,
			String candidateFirstName, String candidateLastName, String candidateLocation, int candidateAge,
			int candidateId, int candidateScore, String candidateFeedBack, String jobDesription, int getCandidateId) {
		super(username, password, type, number, emailId);
		this.candidateFirstName = candidateFirstName;
		this.candidateLastName = candidateLastName;
		this.candidateLocation = candidateLocation;
		this.candidateAge = candidateAge;
		this.candidateId = candidateId;
		this.candidateScore = candidateScore;
		this.candidateFeedBack = candidateFeedBack;
		this.jobDesription = jobDesription;
		this.getCandidateId = getCandidateId;
	}


	//Setters and Getters
	public String getCandidateFirstName() {
		return candidateFirstName;
	}


	public String getCandidateLastName() {
		return candidateLastName;
	}


	public String getCandidateLocation() {
		return candidateLocation;
```

HiringManager.java

```java
public class HiringManager extends Person {

	String jobDescription;
	String jobTitle;
	String jobLocation;

	//Constructor
	public HiringManager(String username, String password, String type, long number, String emailId,
			String jobDescription, String jobTitle, String jobLocation) {
		super(username, password, type, number, emailId);
		this.jobDescription = jobDescription;
		this.jobTitle = jobTitle;
		this.jobLocation = jobLocation;
	}

	//Setters and Getters
	public String getJobDescription() {
		return jobDescription;
	}

	public void setJobDescription(String jobDescription) {
		this.jobDescription = jobDescription;
	}

	public String getJobTitle() {
		return jobTitle;
	}

	public void setJobTitle(String jobTitle) {
		this.jobTitle = jobTitle;
	}

	public String getJobLocation() {
		return jobLocation;
	}

	public void setJobLocation(String jobLocation) {
		this.jobLocation = jobLocation;
	}
```

Database.java

```java
public abstract class Database implements Validation {
    public boolean connectToDatabase() {
        System.out.println("Connecting to database...");
        return true;
    }
```

Validation.java

```java
public interface  Validation {
        static void login(String username, String password) {}
    }
```

Login.java

```java
public class Login extends Database {
    private static String username;
    private static String password;

    public static void login(String user, String pass) {
        username = user;
        password = pass;
        if (username.equals("admin") && password.equals("admin")) {
            System.out.println("Login successful!");
        } else {
            System.out.println("Invalid credentials!");
        }
    }
}
```