Cheatsheet: Designing and Testing Overview for Resume Analyzer Project

1. Designing Overview

Unified Modeling Language (UML) Diagrams

UML is a standardized general-purpose modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. For the Resume Analyzer project, we utilized two primary types of UML diagrams:

1.1. Use Case Diagram

A Use Case Diagram illustrates the different ways users interact with a system. It shows the relationships between actors (users or external systems) and use cases (functions or services provided by the system). It helps in understanding the functional requirements of the system from an external perspective.

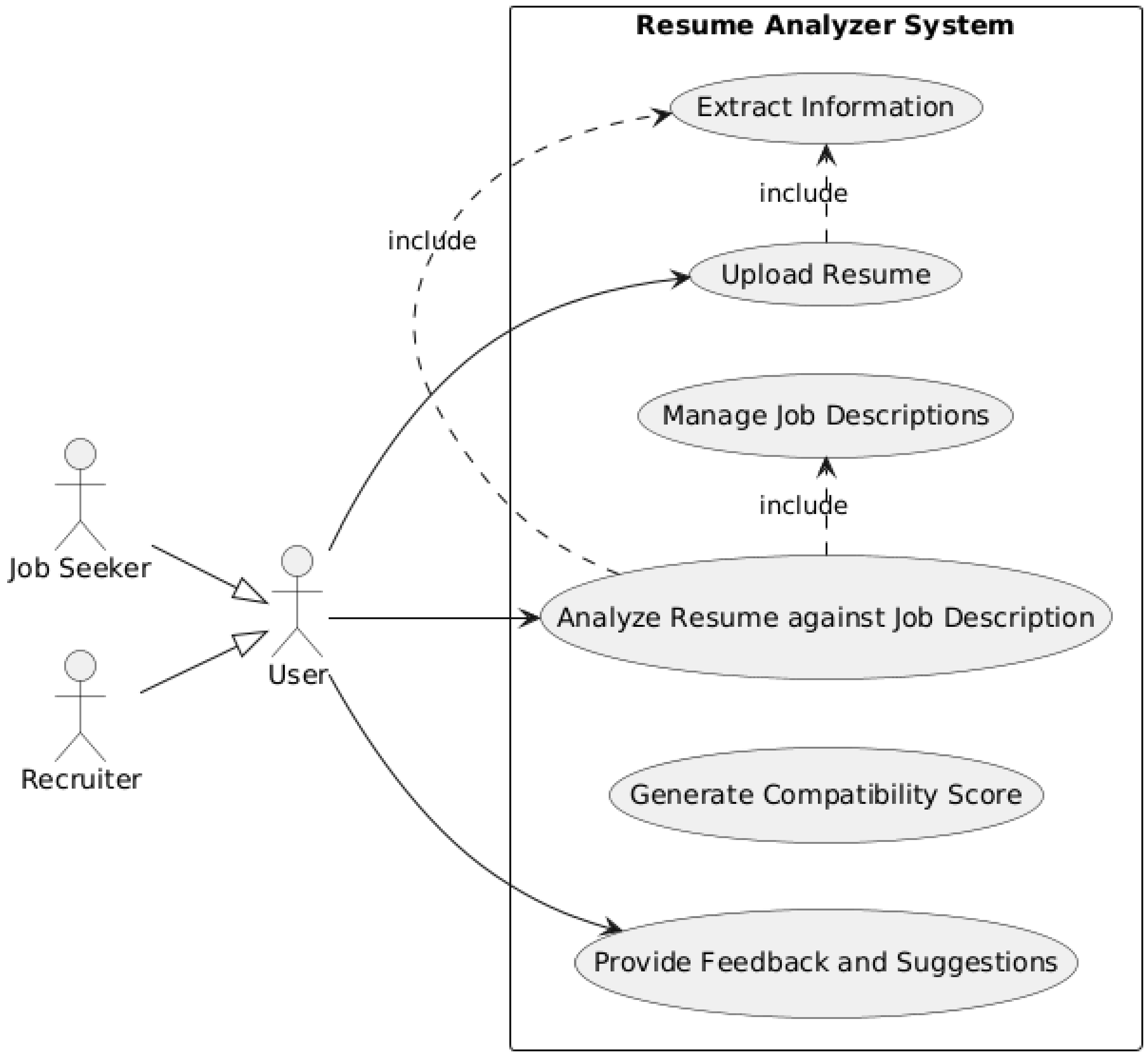
Description for Resume Analyzer:

Our Use Case Diagram for the Resume Analyzer project (as shown in `use_case_diagram.png`) depicts the main actors: 'User', 'Job Seeker', and 'Recruiter'. The key use cases include:

- **Upload Resume**: Allows users to submit their resume files.
- **Extract Information**: System processes the uploaded resume to pull out relevant data.
- **Analyze Resume against Job Description**: Compares the resume content with a provided job description.
- **Generate Compatibility Score**: Calculates a score indicating the match between the resume and job description.

- **Provide Feedback and Suggestions**: Offers personalized advice for resume improvement.
- **Manage Job Descriptions**: Allows recruiters to input and manage job descriptions.

This diagram helps in visualizing how different types of users interact with the core functionalities of the Resume Analyzer system.



1.2. Component Diagram (Backend Architecture)

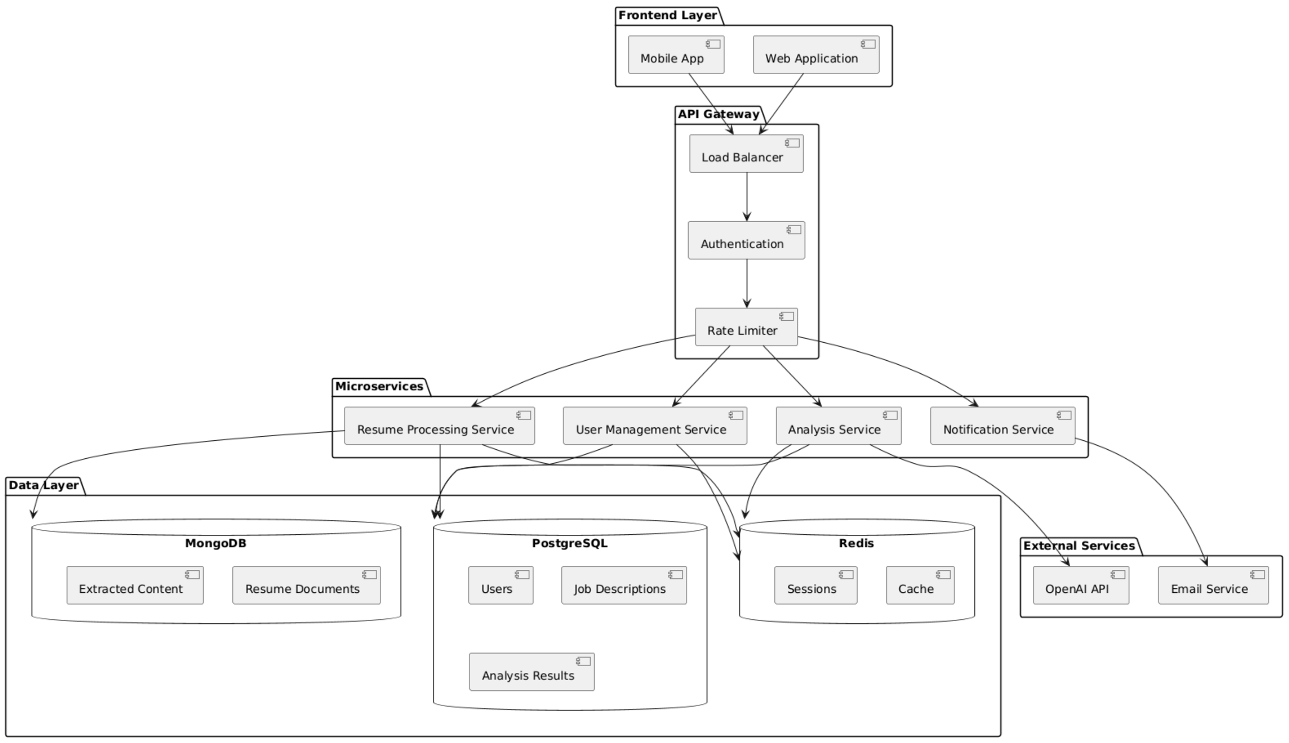
A Component Diagram shows the structural relationships between components of a system. It provides a high-level view of the system's architecture, illustrating how different parts of the system interact with each other. In our case, we used it to represent the backend architecture.

Description for Resume Analyzer:

Our Backend Architecture Diagram (as shown in `backend_architecture.png`) illustrates the microservices-based architecture of the Resume Analyzer. Key components include:

- **Frontend Layer**: Represents the user-facing web and mobile applications.
- **API Gateway**: Handles request routing, authentication, and rate limiting.
- **Microservices**: Dedicated services like Resume Processing, Analysis, User Management, and Notification services.
- **External Services**: Integration with OpenAI API for AI capabilities and an Email Service for notifications.
- **Data Layer**: Comprises PostgreSQL for structured data (users, job descriptions, analysis results), MongoDB for document storage (resumes, extracted content), and Redis for caching and sessions.

This diagram clarifies the flow of data and interactions between different backend services, highlighting the modular and scalable nature of the system.



2. Testing Overview

Testing is a crucial phase in the Software Development Life Cycle (SDLC) to ensure the quality, reliability, and performance of the software. For the Resume Analyzer project, various types of test cases were considered and applied:

2.1. Types of Test Cases Used

2.1.1. Functional Testing

Functional testing verifies that each function of the software application operates in conformance with the functional requirements. It focuses on the system's behavior, ensuring that it does what it's supposed to do.

Description for Resume Analyzer:

- **Resume Upload Functionality (TC001)**: Tests if users can successfully upload resumes in supported formats (PDF, DOCX) and receive appropriate confirmation.
- **Resume Analysis with Job Description (TC002)**: Verifies that the system correctly analyzes a resume against a job description, generates an accurate compatibility score, identifies matched skills, and provides relevant suggestions.
- **Invalid File Format Upload (TC003)**: Checks if the system properly handles unsupported file formats by displaying an error message.

2.1.2. Integration Testing

Integration testing tests the interfaces between components or modules to ensure that they work together as expected. It identifies defects that may arise when different parts of the system are combined.

Description for Resume Analyzer:

- **Frontend-Backend Communication**: Testing if the frontend can successfully send resume files and job descriptions to the backend API and receive analysis results.
- **Microservices Interaction**: Verifying that the Resume Processing Service correctly passes extracted data to the Analysis Service, and that the Analysis Service can communicate with the OpenAI API.
- **Database Connectivity**: Ensuring that all services can read from and write to PostgreSQL and MongoDB databases without issues.

2.1.3. UI/UX Testing

UI/UX testing focuses on the user interface and user experience to ensure that the application is intuitive, easy to use, and visually appealing. It covers aspects like layout, navigation, responsiveness, and overall user satisfaction.

Description for Resume Analyzer:

- **Landing Page Responsiveness**: Checking if the landing page adapts correctly to different screen sizes (desktop, tablet, mobile).
- **Dashboard Layout and Interactivity**: Verifying that the dashboard elements (resume preview, score meter, skill charts, input fields) are correctly displayed and interactive.
- **Navigation Flow**: Ensuring smooth transitions between the landing page and the dashboard, and that all buttons and links function as expected.

2.1.4. Performance Testing

Performance testing evaluates the speed, responsiveness, and stability of a computer, network, software program, or device under a particular workload. It ensures the system can handle expected user loads.

Description for Resume Analyzer:

- **Resume Analysis Response Time**: Measuring the time taken for the system to analyze a resume and return results, aiming for less than 3 seconds as per non-functional requirements.
- **Concurrent User Handling**: Testing the system's ability to process multiple resume analysis requests simultaneously without degradation in performance.
- **API Latency**: Monitoring the response times of various backend API endpoints under different load conditions.

2.1.5. Security Testing

Security testing identifies vulnerabilities in the system and ensures that data and functionalities are protected from unauthorized access or malicious attacks. It is critical for applications handling sensitive user data.

Description for Resume Analyzer:

- **User Authentication and Authorization**: Testing the login and registration processes to ensure only authorized users can access their data and specific functionalities.
- **Data Encryption**: Verifying that resume files and personal information are encrypted during transit and at rest.
- **Input Validation**: Checking for vulnerabilities like SQL injection or cross-site scripting (XSS) in user inputs (e.g., job description text area).
- **Access Control**: Ensuring that users can only access their own resumes and analysis results, and not those of other users.