**Prototype Spike / Proof of Concept (PoC) for Risky Components in HR Assistant Software**

**1. Resume Screening Module**

**1. Component Description:**  
The Resume Screening Module is a key part of the HR Assistant Software. It is responsible for automatically extracting and identifying key information such as skills, experience, and keywords from uploaded resumes. This helps in shortlisting candidates for interview without manual review.

**2. Risk Identified:**  
The primary risk was whether resumes in PDF or DOCX formats could be accurately parsed and screened automatically. Resumes vary in layout and structure, which makes consistent data extraction challenging.

**3. Prototype Implementation:**  
A lightweight Python script was created using the following libraries:

* PyMuPDF to extract text from PDF resumes.
* spaCy NLP library for named entity recognition.
* A simple keyword matching logic to detect skills (e.g., Python, Java, SQL).

**Sample Code Snippet:**

import fitz # PyMuPDF

import spacy

nlp = spacy.load("en\_core\_web\_sm")

doc = fitz.open("sample\_resume.pdf")

text = "".join([page.get\_text() for page in doc])

doc = nlp(text)

skills = ["Python", "Java", "SQL", "React"]

found = [skill for skill in skills if skill.lower() in text.lower()]

print("Extracted Skills:", found)

**4. Result:**  
The prototype successfully extracted text from resumes and identified relevant skills using keyword matching. It worked correctly for simple formatted resumes and proved that integrating this logic into the HR system is feasible.

**5. Decision / Conclusion:**  
This PoC validates that automated resume screening can be implemented in the HR Assistant Software. For now, keyword-based filtering will be used. In future enhancements, this can be extended with machine learning models for advanced semantic understanding.

**2. JWT-Based Authentication**

**Component Description:** This module manages secure login/logout functionality using JSON Web Tokens (JWT).

**Risk Identified:** The team needed to validate whether token-based authentication would properly protect routes and maintain user session integrity in a multi-role environment (HR, Employee, Candidate).

**Prototype Implementation:** A simple Express.js API was created with login functionality issuing JWT tokens. Protected routes were tested using token validation middleware.

**Result:** Protected routes successfully allowed/denied access based on token presence and validity.

**Conclusion:** JWT is a suitable method for user authentication in the HR system and supports role-based access control.

**3. Leave Application CRUD API**

**Component Description:** This feature allows employees to submit leave requests that are stored in the database and later processed by HR.

**Risk Identified:** The risk was whether the frontend and backend could correctly communicate with the database to perform basic Create (POST) and Read (GET) operations using REST APIs.

**Prototype Implementation:** A test POST /leave API was created using Express.js, and a sample leave request form submitted data to MySQL. Basic validations were implemented.

**Result:** Leave records were successfully saved in and retrieved from the database.

**Conclusion:** The leave management module is technically feasible and will be expanded with approval workflows.