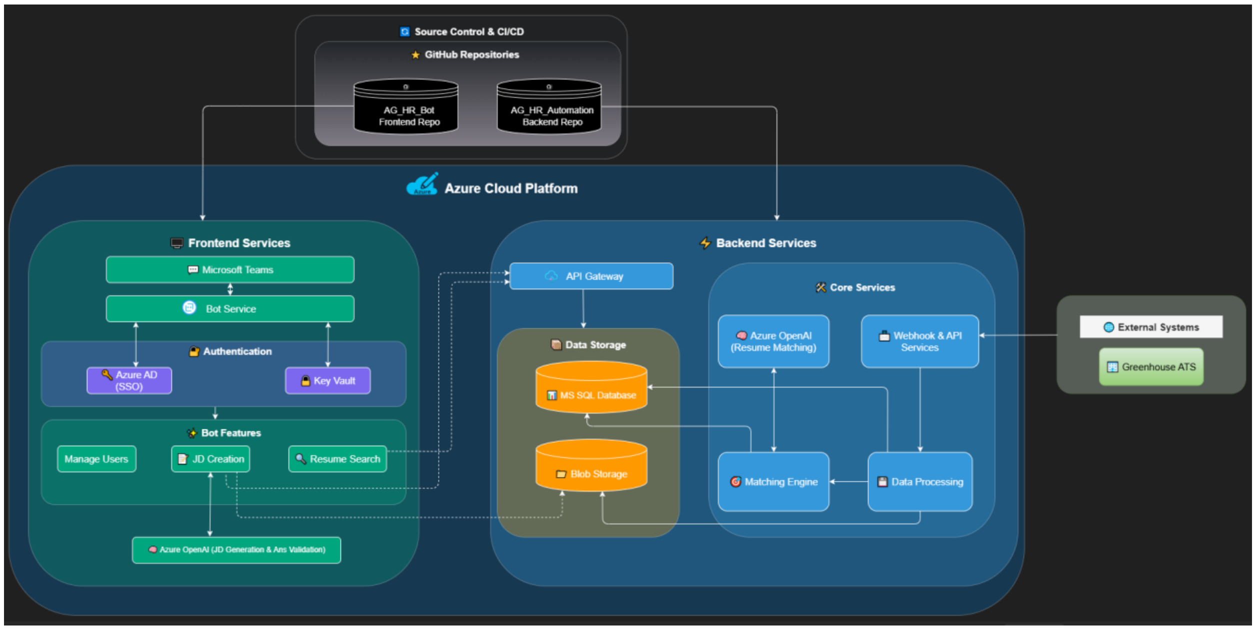
Technical Documentation: HR Automation System

System Architecture:



Overview

The HR Automation System is designed to streamline recruitment processes by integrating Microsoft Teams, Greenhouse ATS, Azure services, and AI-driven resume matching. The solution consists of two repositories:

1. **AG_HR_BOT / Frontend**- A Python-based chatbot using Microsoft Bot Framework to interact with users via MS Teams.
2. **AG_HR_Automation / Backend** - The backend service handling API interactions, Greenhouse ATS integration, and database management.

Architecture Overview

The system is divided into two primary components: **Frontend and Backend services**, both hosted on the **Azure Cloud Platform**.

Source Control & CI/CD

The application code is maintained in two GitHub repositories:

- **AG HR Bot Frontend Repo**: Contains the conversational interface and Teams integration components.
- **AG HR Automation Backend Repo**: Houses the core processing logic and integration services.

Frontend Services

The frontend layer handles user interactions and consists of several key components:

Microsoft Teams Integration

- **AG HR BOT** is implemented as a **Teams application**, providing a familiar interface for users across desktop, web, and mobile platforms.
- **Bot Service**: Azure Bot Service powers the conversational interface, enabling natural language interactions and structured workflows within the Teams environment.
- **Authentication Layer**:
  - **Azure AD (SSO)**: Implements Single Sign-On using **Azure Active Directory** to authenticate users based on their organizational credentials.
  - **Key Vault**: Securely stores authentication secrets, API keys, and other sensitive configuration information.
- **Bot Features**:
  - **Manage Users**: Administrative interface for controlling access to the bot.
  - **JD Creation**: User interface for creating and modifying job descriptions.
  - **Resume Search**: Interface for initiating CV matching against job requirements.
- **AI Integration (Frontend)**:

    - **Azure OpenAI** is leveraged for JD generation and user input validation, ensuring that job descriptions are properly formatted and contain all required information.

Backend Services

The backend provides the core business logic and data processing capabilities:

- **API Gateway**: Manages all API requests between frontend components and backend services, providing security, rate limiting, and request routing.
- **Data Storage**:
    - **Azure MSSQL Database**:
        - Stores structured data, including:
            - User information and access permissions
            - Job descriptions and their metadata
            - CV matching results and scoring data
    - **Blob Storage**:
        - Stores unstructured data such as:
            - Original JD documents in various formats
            - Resume/CV files for processing
            - Exported reports and documents
- **Core Services**:
    - **Azure OpenAI (Resume Matching)**: Utilizes advanced language models to analyze resume content against job requirements.
    - **Webhook & API Services**: Provides integration points for external systems and data sources.
    - **Matching Engine**: Contains the algorithms and business logic for comparing candidate qualifications to job requirements.
    - **Data Processing**: Prepares and transforms data for the matching engine, including document parsing and entity extraction.
- **External Systems Integration**:
    - **Greenhouse ATS**: Applicant Tracking System integration for accessing candidate information.

Application Workflow

1. Microsoft Teams System Setup & Requirements

To deploy and configure the bot within Microsoft Teams, the following steps must be taken:

1. **App Registration in Azure AD**:
   a. Register the bot as an **Azure AD App** in the **Azure Portal**.
   b. Assign the required permissions (User.Read, Directory.Read, TeamsAppInstallation.ReadWrite).
   c. Configure authentication endpoints for Single Sign-On (SSO).
2. **Bot Registration in Microsoft Bot Framework**:
   a. Register the bot in the **Azure Bot Service**.
   b. Configure messaging endpoints to connect with the backend API.
   c. Enable Teams Channel and provide App ID & secret.
3. **Manifest File Deployment in Teams**:
   a. Create a **Teams App Manifest file** (manifest.json).
   b. Upload it to the **Teams Admin Center** for organization-wide deployment.
   c. Assign policies to enable the bot for all users, allowing them to manually pin it.
4. **User Role & Access Management**:
   a. Define role-based access control (RBAC) for users interacting with the bot.
   b. Apply **Microsoft Teams policies** to restrict or allow bot interactions.
   c. **Authentication within the app** ensures that only authorized users can access services.

2. Job Description (JD) Creation

1. User selects **"Create a JD"** from the Teams bot menu.
2. The bot prompts the user with predefined questions about the job role.
3. The user's responses are sent to **Azure OpenAI**, which generates a structured JD.
4. The JD is displayed in Teams for review.
5. Upon confirmation, the JD is stored in **Azure MSSQL** via the **AG_HR_Automation API**.
6. The JD is also converted into **DOCX/PDF format** and saved in **Azure Blob Storage**.
7. The user receives a **download link** via Teams, generated using a **SAS (Shared Access Signature) token**.

3. Resume Fetching & Matching

1. User selects **"Fetch Top Resumes"** and inputs:
    a. Job ID
    b. Filters (e.g., education match, skills match)
    c. Number of resumes to fetch
2. The request is sent to the **AG_HR_Automation API**, which queries **Azure MSSQL**.
3. The system retrieves candidate resumes and computes similarity scores based on:
    a. Overall Match Score
    b. Skills Matching
    c. Education Matching
    d. Experience Matching
4. The top-matching resumes are displayed in Teams.
5. Users can download resumes directly via **secure links from Blob Storage**.

4. Security & Compliance Measures

- **Authentication**: Azure AD SSO with multi-factor authentication support.
- **Authorization**: Role-based access control for all bot functions.
- **Data Protection**: Encryption for data both in transit and at rest.
- **Secret Management**: Key Vault for secure storage of sensitive configuration.
- **API Security**: Secured API Gateway with proper authentication enforcement.

5. Performance Considerations

- **Scalability**: Cloud-based architecture allows for dynamic scaling based on user load.
- **Responsiveness**: Azure Bot Service provides low-latency responses for basic interactions.
- **Processing Time**: Complex operations like CV matching may require additional processing time depending on volume.
- **Concurrent Users**: System designed to handle multiple simultaneous users in the organization.