Case Study: AI Adoption Analyzer

Objective:

The objective of the application, to understand the level of AI adoption in Industries. Relation to job roles, industry types, and company sizes is critical to effectively managing automation risks, aligning skill requirements, and forecasting job growth. The challenge lies in analyzing the adoption levels of AI across multiple roles to identify patterns and make informed decisions on workforce planning, skill development, and risk management.

The application should include a page that displays information from an external source (AIJob API), provides quick access to bookmark details of growth or stable job roles for future analysis

The Information to access the AIJob API can be found at the end of the document.

System users:

- **Registered Users** who can access personalized features like bookmarking
- **Guest Users** who can explore limited features without registration

Key High-Level Functional Features:

Some of the key high-level features of the system include:

- **User Registration:**
  Allows users to create an account to access the application's personalized features.

- **Viewing based on Industry and Location**
  Users need to analyze how location and industry affect AI adoption levels and the associated factors, providing region-specific and industry-specific insights.

  Advanced Search:

  Users should be able to search based on different job roles, company sizes, automation risk, salary and AI adoption level

  Bookmarking Data:

  Users can bookmark the data based on growth projection which could be growth or stable for further analysis

- **Managing Bookmarked Roles:**
  Users can view and delete their bookmarked roles.

- **Updating Personal Details**:
  Users can update their personal information as needed.
  User should be able to reset the password

- **Secure Login/Logout:**
  Data access is secured through registration and login processes. Users must be able to log out/time out after inactivity

Key Non-Functional Features

Architecture and Design

- Microservices Architecture: The application should use a microservices architecture with polyglot persistence to handle various responsibilities.

User Experience:

- User-Friendly Interface: Focus on creating a user-friendly and accessible application, emphasizing usability.
- Responsive Design: Ensure the application works well across different device screens

Security:

- JWT Tokens: Implement JWT for secure authentication and session management.
- Data Protection: Use encryption standards to protect sensitive data

Operational and Maintenance:

- Logging: Implement comprehensive logging to track and record significant events, errors, and transactions. This aids in debugging, monitoring, and maintaining operational transparency.
- Automation Testing: Perform thorough testing to ensure the reliability and quality of the application.

Service Management and Integration:

- Microservices Patterns: Implement key patterns such as API Gateway, Config Server, Service Discovery (Eureka), and Kafka for Messaging to manage service interactions and configurations.

Development and Deployment:

- Containerization (If Applicable) : Use Docker and Docker Compose for containerizing the application, ensuring consistent environments across development, testing, and production.
- CI/CD (If Applicable): Implement Continuous Integration and Continuous Deployment pipelines using tools like Jenkins or GitLab CI to automate building, testing, and deploying processes.

Primary Service Modules

The system's domain-specific microservices include:

- **UserProfile Service:** Manages user profile data and personal information.

  During the registration process, the UserProfile Service communicates with the Authentication Service via a message bus (Kafka/RabbitMQ). This allows the credentials provided during registration to be securely stored by the Authentication service.

- **Authentication Service:** Handles user authentication and session management.

  It is responsible for securely storing credentials received from the UserProfile Service through external *the message bus and for generating and validating JWT tokens for user authentication.*

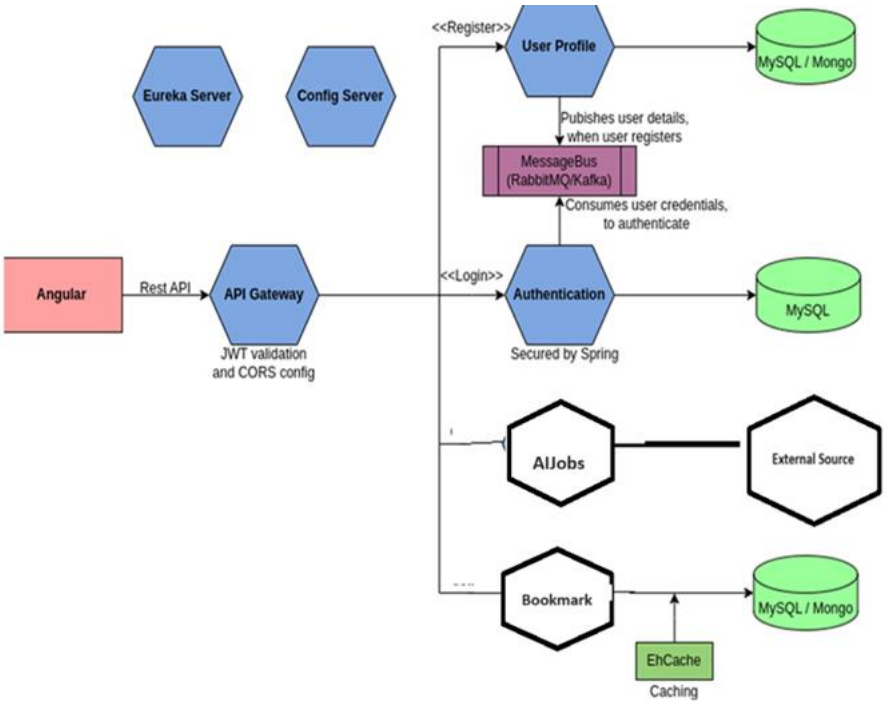
- **AIJobs Service:** Fetches and manages job related from external source.
- **Bookmark Service:** Manages users' bookmarked details

Additional microservices for System Management:

- **API Gateway Service:** Manages routing and load balancing for microservices.
- **Discovery Service:** Enables service registration and discovery using Eureka.
- **Centralized Configuration Service:** Manages configuration settings across microservices.(Optional)

High Level Architecture Diagram

The high-level architecture diagram outlines the microservices and their interactions within the system. This diagram should include components such as API Gateway, Service Discovery, and the various microservices mentioned above.



Note: The system is assumed to be evolvable with addition and removal of services with new requirements

Starting and accessing the AIJobs API:

To start the AIJobs API on your local system or VM, use the following Docker command:

(Use Sudo before all the docker commands if you are using ubuntu/linux)

```
docker run -p3232:3232 --name aicontainer -d stackroutenew/AIJobsapi
```

You can verify running container details using the below command

```
docker ps -a
```

Once the container is running, you can access the API at:

http://localhost:3232/AIJobs

API Endpoints

- **GET** /AIJobs- Retrieve all roles.
- **GET** /AIJobs?propertyname=value - Filter roles by a specific property (where propertyname is any Json key property of the resource like Industry, Location (case sensitive).
- **POST** /AIJobs- Add new job details.

CleanUp

use the following command to stop the container when not in use

```
docker stop aicontainer
```

To remove the container Permanently

```
docker rm aicontainer
```

Errors:

If you are facing any error related to port conflict, kindly stop and remove the container using the above commands

For more details on the available endpoints, visit json-server on npm.