# Bench Process Management System

## Proposed Architecture Design

Contents

[Bench Process Management System 1](#_Toc173764793)

[Proposed Architecture Design 1](#_Toc173764794)

[Technology Stack 1](#_Toc173764795)

[System Architecture 1](#_Toc173764796)

[System Components 2](#_Toc173764797)

[Benefits of the Proposed Architecture: 2](#_Toc173764798)

[Architecture Diagram 2](#_Toc173764799)

### Technology Stack

**Backend:**

|  |  |
| --- | --- |
| Programming Language | Java 17, React |
| Framework | Spring Boot 3.0.X |
| Database | PostgreSQL 13 |
| Containerization | NA |
| Deployment | VM or OnPrem Environment |
| Logging | Log back |
| Authentication & Authorization | OIDC (Local DB authorization process) |
| CI/CD Pipeline | Jenkins |
| Version Control | GitHub |
| Code Quality tool | SonarQube |
| Api security | JWT token |

### System Architecture

**Monolithic Architecture**

### System Components

1. **Skill Management Service:** This service will manage skill gap reviews, track feedback, and maintain a skill matrix.
2. **Incubation Management Service:** This service will oversee the administration of incubation programs, including tracking progress, planning & tracking skill reviews, feedback, and other related aspects.
3. **ML Leave management service**: This service will support for tracking the female employee duration of leave and other details.
4. **Authorization & authentication service**: It will take care of making sure that right person is accessing the application and JWT token generation for the respective API.
5. **File upload service**: It will read the data from user CSV file upload and load in to respective tables for the further processing.
6. **People Management service**: It will allow user to perform the CRUD operation like initiation skill review, update skill review, delete records etc.
7. **Ramp down service**: It will take care of CRUD operation like initiate the skill review for the candidate, validate the review rating and send notification.
8. **Skill review service** : It takes care about employee sill review status and feedback and next path.
9. **Incubation service :** It will take care about the employes who are in incubation and status of Mid and final sill review.
10. **Notification service** : It will make sure required emails are sent to respective persons
11. **~~Assessors and Mentors Management Service~~**~~: This service will automate the management of assessors and mentors’ availability, ensuring optimal utilization of these resources.~~

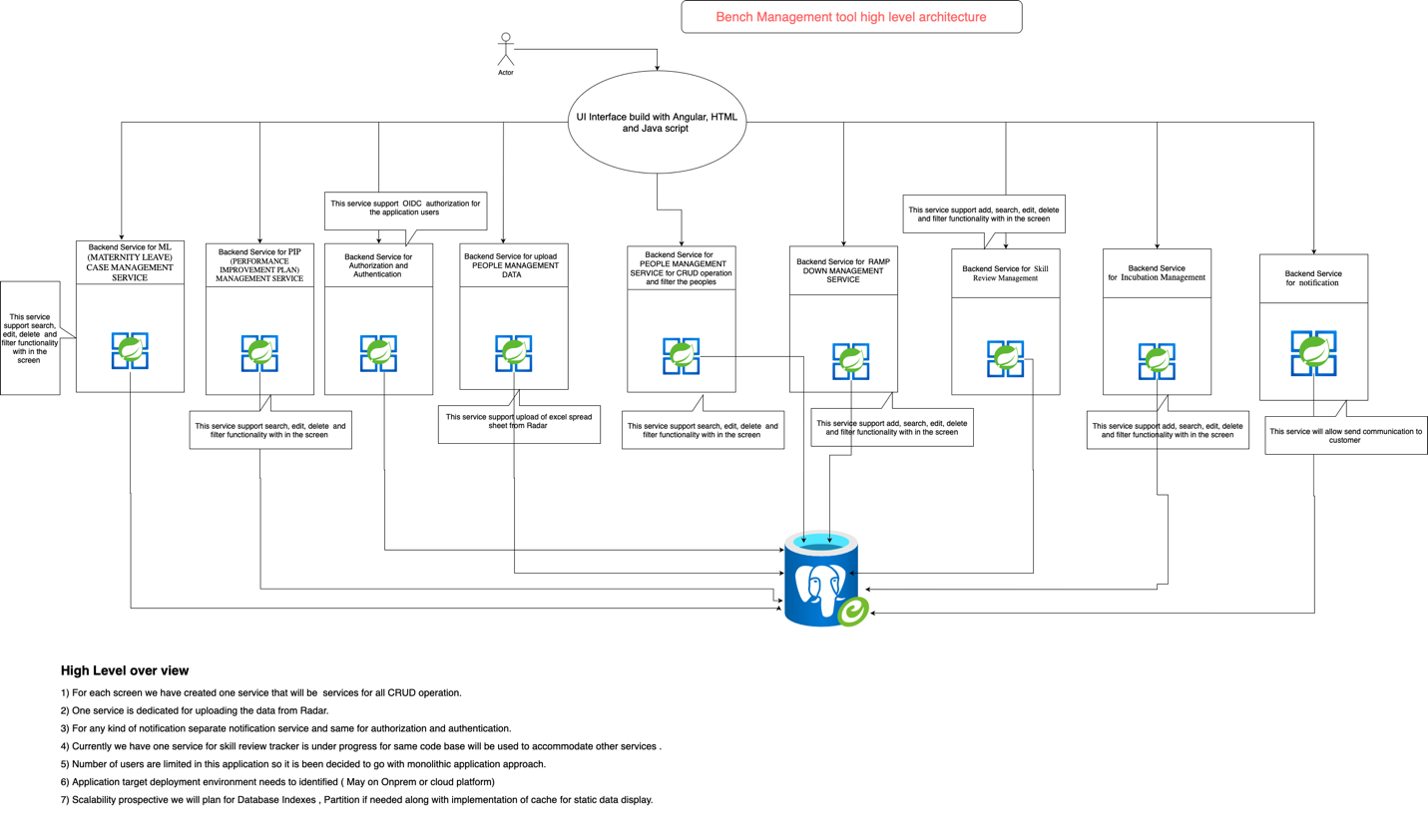
**~~d).~~**

1. ~~PIP Management service :~~
2. **~~Development and Improvement Plans Service~~**~~: This service will handle the administration of development and improvement plans, ensuring effective implementation and tracking.~~
3. **~~RTBs Management Service:~~** ~~This service will handle and track RTBs, improving efficiency and reducing manual labor.~~
4. **~~Reporting Service:~~** ~~This service will generate interactive reports with customizable charts, graphs, and visualizations, allowing researchers to filter and analyze data based on specific criteria.~~
5. **User Interface (UI):** The React-based UI will provide a user-friendly interface for accessing all functionalities and operations.
6. **~~Dashboard:~~**

### Benefits of the monolith architecture:

1. By considering the application numbers of users and functionality. All components are tightly integrated, which can simplify the development process and reduce the complexity of managing multiple services or modules.
2. **Easier Deployment**: Deploying a monolithic application is often simpler because you only need to manage and deploy a single executable or set of files. This can be less cumbersome compared to managing multiple microservices or distributed components
3. **Reduced Overhead**: Monolithic applications avoid the overhead associated with inter-service communication, which is common in microservices. This can lead to better performance and reduced latency in applications where all functionalities are tightly coupled.
4. **Consistent Development Environment**: With all components residing in a single codebase, developers can work within a consistent environment, reducing the need for complex configuration and integration between different services.
5. **Easier Debugging**: Debugging a monolithic application can be more manageable because you don’t need to track down issues across multiple services. Everything is in one place, making it easier to trace and fix bugs.
6. **Unified Codebase**: Having a single codebase means that all the functionality is in one place, which can make it easier to understand and maintain, particularly when the application is small, and the team is familiar with the entire codebase.

### Architecture Diagram

**Attached the initial draft monolith architecture **

**Components:**

|  |  |
| --- | --- |
| User Interface | The React-based user interface provides access to all functionalities and operations. |
| API Gateway | The API gateway acts as a single-entry point for all API requests. |
| Microservices | The system consists of various microservices, each responsible for a specific functionality. |
| Service Discovery | Service discovery enables microservices to locate and communicate with each other dynamically. |
| Azure Kubernetes Service (AKS) | AKS provides a managed Kubernetes environment for container orchestration and scaling. |
| Docker Containers | Each microservice is containerized using Docker for portability and consistency. |
| PostgreSQL | PostgreSQL serves as the primary database for storing application data. |

**Data Flow:**

* Users interact with the user interface to access various functionalities.
* API requests are routed through the API gateway to the appropriate microservice.
* Microservices communicate with each other through service discovery to perform their tasks.
* Data is stored and retrieved from the PostgreSQL database.
* The reporting service generates reports based on data from various microservices.