**Bench Management tool high level architecture**

**Project Overview**

This project is expected to handle the bench process from end to end. Once the employee is onboarded to organization till the project allocated or deboarding from one project and making sure that he will be assigned to new client project. During this time employee may need to undergo different training program, Skill review, identify the technology gap and assigned them with the right training program and make sure employee will present organization standard Infront of client. This tool will help to track this process and build a report about the status.

**Technical Specification of this project**

As part of this project initial design is to develop the 5 micro services (2 Front end and 3 back end) and deployed in Azure environment with containerization.

First microservice will be used to provide the functionality where user will upload the excel spread sheet downloaded from Radar tool. Same will be processed by the back end microservice and stored in Postgres DB.

Note: Azure active directory will be used for authorization and authentication

Once data is stored in Postgres DB Kafka connector (Confluence) will copy the data from current DB and post in to other Postgres DB.

Second UI microservice basically provides the user interface where users can go over the employee list and assign the skill review and select the interviewer, date and send a meeting invite and other functionality.

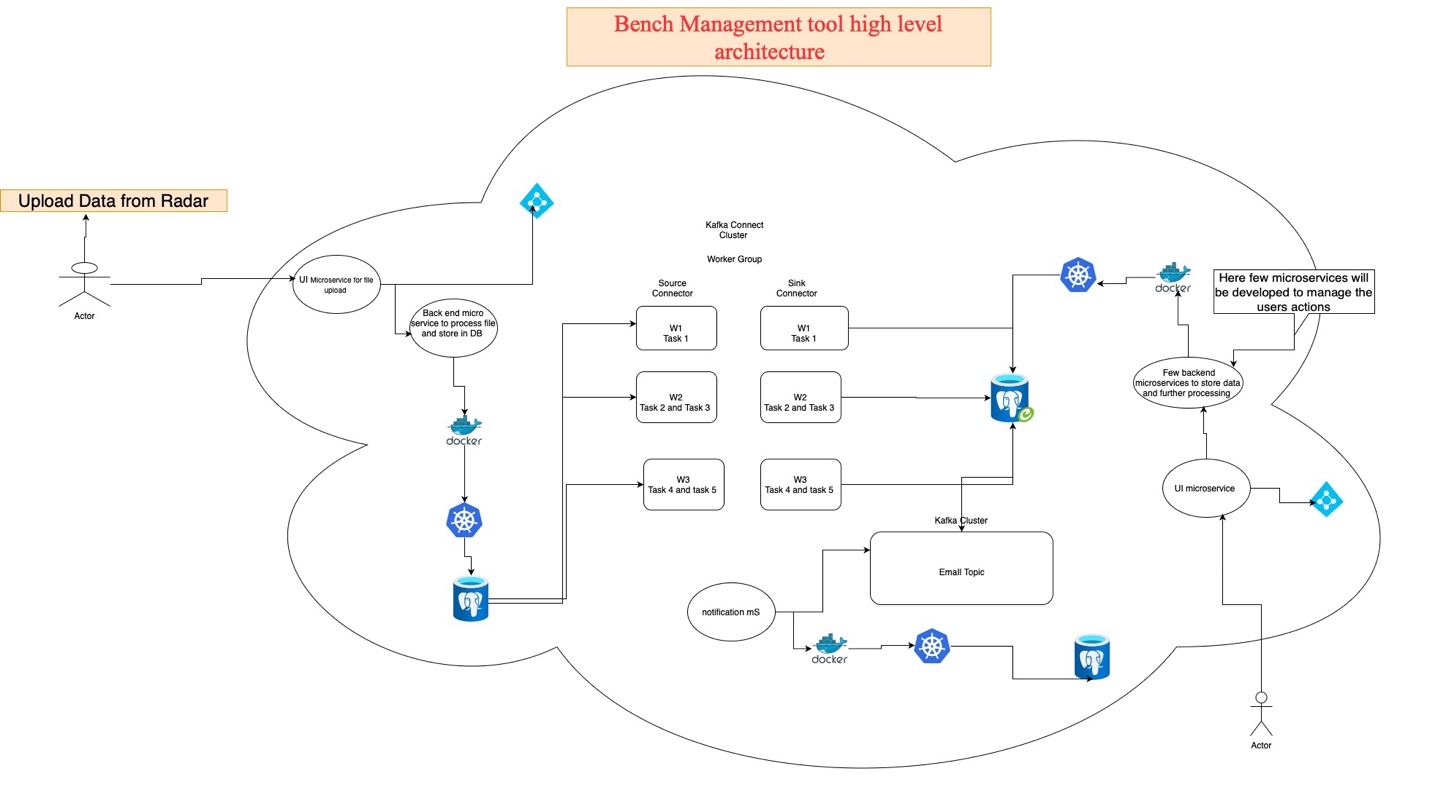
All these transactions will be sent to respective back end microservice and same will updated in Postgres database.

There is one more microservice is designed only to handle the notification that needs to send to users for respective action. Basically, when user takes some action on the UI screen example notify the user for upcoming skill review then message will be published to Kafka topic from where it will be consumed by microservice and send automated email from tool.

Azure log Analytics will be used to capture the log.

API will be secured with JWT token authorization.

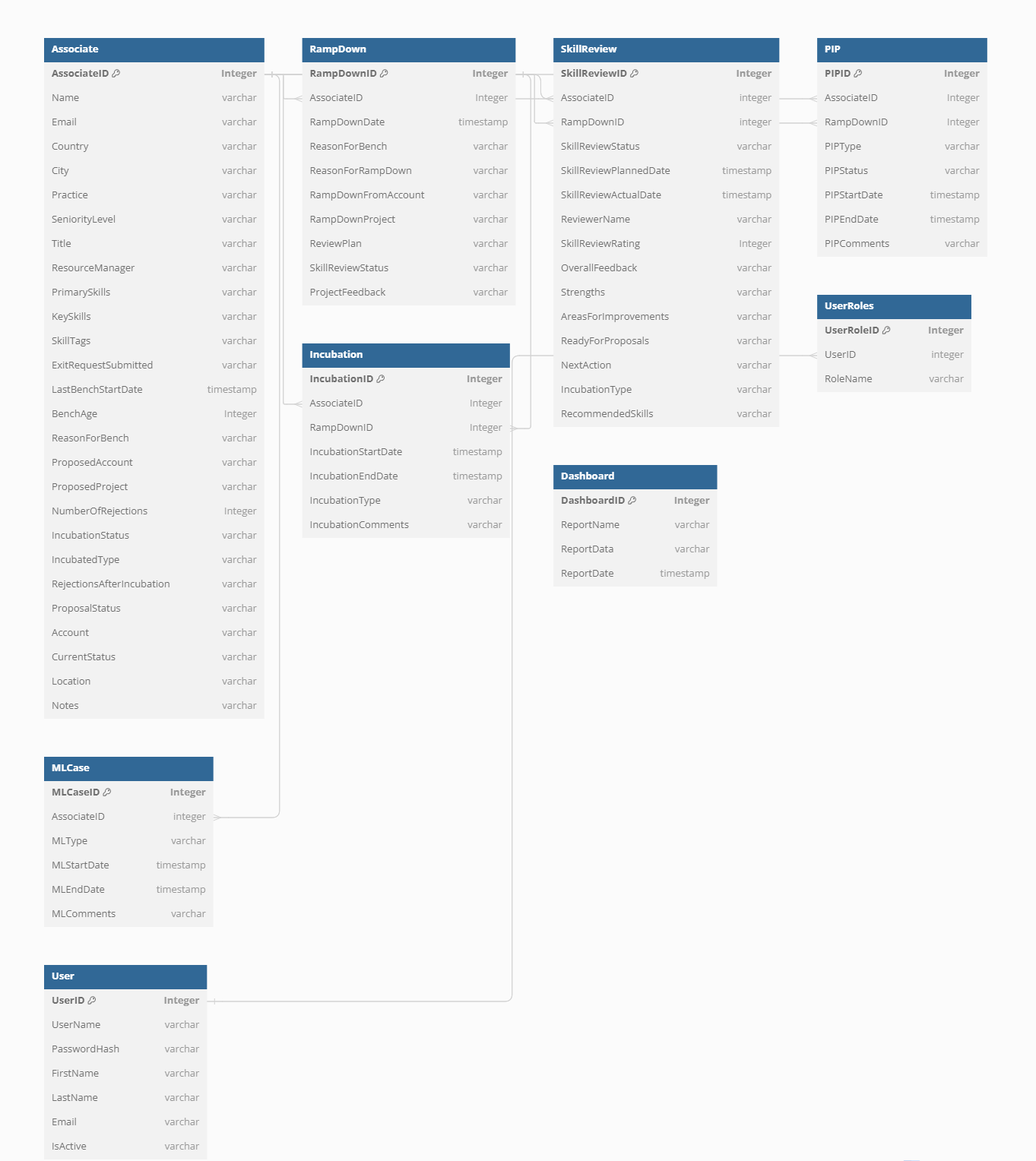
Attached the Design of the application



**Technical Stocks used in assignments**

|  |  |
| --- | --- |
| Software Used | Version |
| Spring boot | 3.0.X |
| Java | 17 |
| Azure Kubernetes Service | 1.29 |
| Docker | 24.0. |
| Log Analytics |  |
| Azure container service |  |
| Azure active directory |  |
| Postgres | 13 |
| React |  |
| GitHub |  |
| Azure DevOps |  |

**DB Design**



**DB Schema details**

CREATE TABLE "Associate" (

"AssociateID" Integer PRIMARY KEY,

"Name" varchar,

"Email" varchar,

"Country" varchar,

"City" varchar,

"Practice" varchar,

"SeniorityLevel" varchar,

"Title" varchar,

"ResourceManager" varchar,

"PrimarySkills" varchar,

"KeySkills" varchar,

"SkillTags" varchar,

"ExitRequestSubmitted" varchar,

"LastBenchStartDate" timestamp,

"BenchAge" Integer,

"ReasonForBench" varchar,

"ProposedAccount" varchar,

"ProposedProject" varchar,

"NumberOfRejections" Integer,

"IncubationStatus" varchar,

"IncubatedType" varchar,

"RejectionsAfterIncubation" varchar,

"ProposalStatus" varchar,

"Account" varchar,

"CurrentStatus" varchar,

"Location" varchar,

"Notes" varchar

);

CREATE TABLE "RampDown" (

"RampDownID" Integer PRIMARY KEY,

"AssociateID" Integer,

"RampDownDate" timestamp,

"ReasonForBench" varchar,

"ReasonForRampDown" varchar,

"RampDownFromAccount" varchar,

"RampDownProject" varchar,

"ReviewPlan" varchar,

"SkillReviewStatus" varchar,

"ProjectFeedback" varchar

);

CREATE TABLE "SkillReview" (

"SkillReviewID" Integer PRIMARY KEY,

"AssociateID" integer,

"RampDownID" integer,

"SkillReviewStatus" varchar,

"SkillReviewPlannedDate" timestamp,

"SkillReviewActualDate" timestamp,

"ReviewerName" varchar,

"SkillReviewRating" Integer,

"OverallFeedback" varchar,

"Strengths" varchar,

"AreasForImprovements" varchar,

"ReadyForProposals" varchar,

"NextAction" varchar,

"IncubationType" varchar,

"RecommendedSkills" varchar

);

CREATE TABLE "PIP" (

"PIPID" Integer PRIMARY KEY,

"AssociateID" Integer,

"RampDownID" Integer,

"PIPType" varchar,

"PIPStatus" varchar,

"PIPStartDate" timestamp,

"PIPEndDate" timestamp,

"PIPComments" varchar

);

CREATE TABLE "MLCase" (

"MLCaseID" Integer PRIMARY KEY,

"AssociateID" integer,

"MLType" varchar,

"MLStartDate" timestamp,

"MLEndDate" timestamp,

"MLComments" varchar

);

CREATE TABLE "Incubation" (

"IncubationID" Integer PRIMARY KEY,

"AssociateID" Integer,

"RampDownID" Integer,

"IncubationStartDate" timestamp,

"IncubationEndDate" timestamp,

"IncubationType" varchar,

"IncubationComments" varchar

);

CREATE TABLE "Dashboard" (

"DashboardID" Integer PRIMARY KEY,

"ReportName" varchar,

"ReportData" varchar,

"ReportDate" timestamp

);

CREATE TABLE "UserRoles" (

"UserRoleID" Integer PRIMARY KEY,

"UserID" integer,

"RoleName" varchar

);

CREATE TABLE "User" (

"UserID" Integer PRIMARY KEY,

"UserName" varchar,

"PasswordHash" varchar,

"FirstName" varchar,

"LastName" varchar,

"Email" varchar,

"IsActive" varchar

);

ALTER TABLE "SkillReview" ADD FOREIGN KEY ("AssociateID") REFERENCES "Associate" ("AssociateID");

ALTER TABLE "SkillReview" ADD FOREIGN KEY ("RampDownID") REFERENCES "RampDown" ("RampDownID");

ALTER TABLE "RampDown" ADD FOREIGN KEY ("AssociateID") REFERENCES "Associate" ("AssociateID");

ALTER TABLE "Incubation" ADD FOREIGN KEY ("AssociateID") REFERENCES "Associate" ("AssociateID");

ALTER TABLE "Incubation" ADD FOREIGN KEY ("RampDownID") REFERENCES "RampDown" ("RampDownID");

ALTER TABLE "MLCase" ADD FOREIGN KEY ("AssociateID") REFERENCES "Associate" ("AssociateID");

ALTER TABLE "PIP" ADD FOREIGN KEY ("AssociateID") REFERENCES "Associate" ("AssociateID");

ALTER TABLE "PIP" ADD FOREIGN KEY ("RampDownID") REFERENCES "RampDown" ("RampDownID");

ALTER TABLE "UserRoles" ADD FOREIGN KEY ("UserID") REFERENCES "User" ("UserID");

**Micro Services**

**People Management Service**

* Responsible for managing information about associates on the bench.
* Handles CRUD operations for associate details, including new joiners.
* Interfaces with the frontend for data presentation and user interaction.

URL: http://localhost:8080/api/people.

* GET /api/people - Retrieve all people.
* GET /api/people/{id} - Retrieve an people by ID
* POST /api/people - Create a new people.
* PUT /api/people/{id} - Update an existing people.
* DELETE /api/people/{id} - Delete an people.

 **Ramp Down Management Service**

* Manages associates who are ramped down or will be ramped down.
* Tracks reasons for ramp down, associated projects, and feedback.
* Integrates with skill review and incubation services for further actions.

URL: http://localhost:8080/api/rampdown.

* GET /api/rampdown - Retrieve all ramp down records.
* GET /api/rampdown/{id} - Retrieve a ramp down record by ID.
* POST /api/rampdown - Create a new ramp down record.
* PUT /api/rampdown/{id} - Update an existing ramp down record.
* DELETE /api/rampdown/{id} - Delete a ramp down record.

 **Skill Review Management Service**

* Plans, tracks, and manages skill reviews for bench associates.
* Stores skill review statuses, ratings, and feedback.
* Integrates with ramp down and incubation services based on review outcomes.

URL: http://localhost:8080/api/skillreviews.

* GET /api/skillreview - Retrieve all skill review records.
* GET /api/skillreview/{id} - Retrieve a skill review record by ID.
* POST /api/skillreview - Create a new skill review record.
* PUT /api/skillreview/{id} - Update an existing skill review record.
* DELETE /api/skillreview/{id} - Delete a skill review record.

 **Incubation Management Service**

* Assigns and tracks incubation periods for associates based on skill reviews or client feedback.
* Manages incubation types, progress, and outcomes.
* Coordinates with skill review and ramp down services for initiating and concluding incubation periods.

URL: http://localhost:8080/api/incubations.

* GET /api/incubations - Retrieve all incubation records.
* GET /api/incubations/{id} - Retrieve an incubation record by ID.
* POST /api/incubations - Create a new incubation record.
* PUT /api/incubations/{id} - Update an existing incubation record.
* DELETE /api/incubations/{id} - Delete an incubation record.

 **PIP (Performance Improvement Plan) Management Service**

* Tracks performance improvement plans and return-to-bench cases.
* Monitors progress, outcomes, and associated actions.
* Interfaces with other services for data integration and reporting.

URL: http://localhost:8080/api/pips.

* GET /api/pips - Retrieve all PIP records.
* GET /api/pips/{id} - Retrieve a PIP record by ID.
* POST /api/pips - Create a new PIP record.
* PUT /api/pips/{id} - Update an existing PIP record.
* DELETE /api/pips/{id} - Delete a PIP record.

 **ML (Maternity Leave) Case Management Service**

* Manages maternity leave cases for associates.
* Tracks leave periods, planning, and return dates.
* Integrates with other services for associate status updates and reporting.

URL: http://localhost:8080/api/mlcases.

* GET /api/mlcases - Retrieve all ML Case records.
* GET /api/mlcases/{id} - Retrieve an ML Case record by ID.
* POST /api/mlcases - Create a new ML Case record.
* PUT /api/mlcases/{id} - Update an existing ML Case record.
* DELETE /api/mlcases/{id} - Delete an ML Case record.

**Assumptions**

* Necessary Git account will be provided to maintain the code repository.
* Access to azure subscription will be provided to above mentioned services.
* Required developer (back end and front end) will be provided from the current bench.
* Three different environments will be maintained (Dev, QA & Prod)

**Risks**

* Frequently changing the resources from bench will lead in to delay in target completion date
* If we need to deploy the project outside the cloud environment, then need to change the authorization architecture.

**Estimation**

Total Duration of this project: 4-6 Months, depends on the resource availability

Initially we will start with 4 developers and 2 testers along with one Architect