Talent Share Portal

High Level Design Document

Document Version: 1.0

Version	Date	Author	Description
1.0	01/02/2024	Rahul Yadav	Initial version

Table of Content

1. Introduction

1.1 Purpose

The purpose of the document for this website is to outline the features, functionalities, and overall objectives of the Talent Share Portal designed for employees. The document emphasizes the creation of a platform where skilled individuals can act as mentors, offering guidance and mentorship to those seeking technical upliftment or guidance, referred to as mentees.
The project aims to foster a collaborative and learning culture within an organization by facilitating mentorship connections. The mentor can list their skills on the portal, and mentees can search for and request mentorship in specific skills. The overarching goal is to invest in the growth of employees, providing opportunities for them to deepen their understanding of their roles and technology.

1.2 Scope

The high-level design ensures that the Talent Share portal is user-friendly, secure, and capable of efficiently facilitating mentorship relationships within the organization. Each module is designed to address specific functionalities, creating a cohesive and comprehensive mentorship platform. The high-level design of the Talent Share portal will encompass various components and modules to ensure the successful implementation of the mentorship platform. Here is an overview of the key aspects of the high-level design:

User Management Module:

Admin Dashboard: A centralized dashboard for the admin to manage user profiles, create Mentors, Mentees, and Managers and deactivate inactive users.
User Profiles: Detailed profiles for Mentors and Mentees & Managers capturing skills, technical interests, and other relevant information.

Mentee Functionality:

Mentor Search: Mentees can search for Mentors based on skills.
Session Scheduling: Mentees can send session requests to Mentors and schedule mentorship sessions.
Notifications: Mentees will receive notification regarding session request for getting acceptance and rejection.

Manager Functionality:
The Manager in the Talent Share portal serves as a crucial overseer of the mentorship program. Their responsibilities include evaluating the qualifications and skills of individuals seeking to become mentors,

ensuring that they possess the expertise necessary for guiding mentees. The Manager plays a key role in the session approval process, deciding whether to accept or reject mentor requests based on predetermined criteria.

Mentor Functionality:

Session Requests: Mentors can view mentorship session requests from Mentees, along with relevant details about the mentee's profile.
Profile Review: Mentors can review mentee profiles before accepting or rejecting session requests.
Availability: Mentors can set their availability, making it easier for Mentees to schedule sessions.

Notification System:

User Registration:

- Upon user registration, an automated email containing the system-generated password is sent to the user's email address for login credentials.

Password Reset Confirmation:

- When a user initiates a password reset request, a confirmation email is sent to the user's email address to verify the action.

Password Reset:

- In the event of forgetting a password, an automated email containing a newly generated password is sent to the user's email address.

Session Request Notification:

- If a mentee requests a mentorship session, an email notification is sent to the mentor's manager for approval.
- Upon manager approval, a notification is sent to the mentor, confirming the session request.
- In case of manager rejection, a notification is sent to the mentee, informing them of the rejection.

Session Confirmation:

- When a mentor accepts a session request, an email notification is sent to the mentee, confirming the session details.
- Additionally, a notification is sent to the mentee's manager to keep them informed about the scheduled session.

Feedback System:

Post-Session Feedback: Mentees must provide feedback after each mentorship session, contributing to continuous improvement.
Admin Access: The admin can access, and review feedback responses submitted by Mentees.

User Report:

The user report feature could be implemented to store users' historical data. This ensures that interaction of the user like user is active or not active.

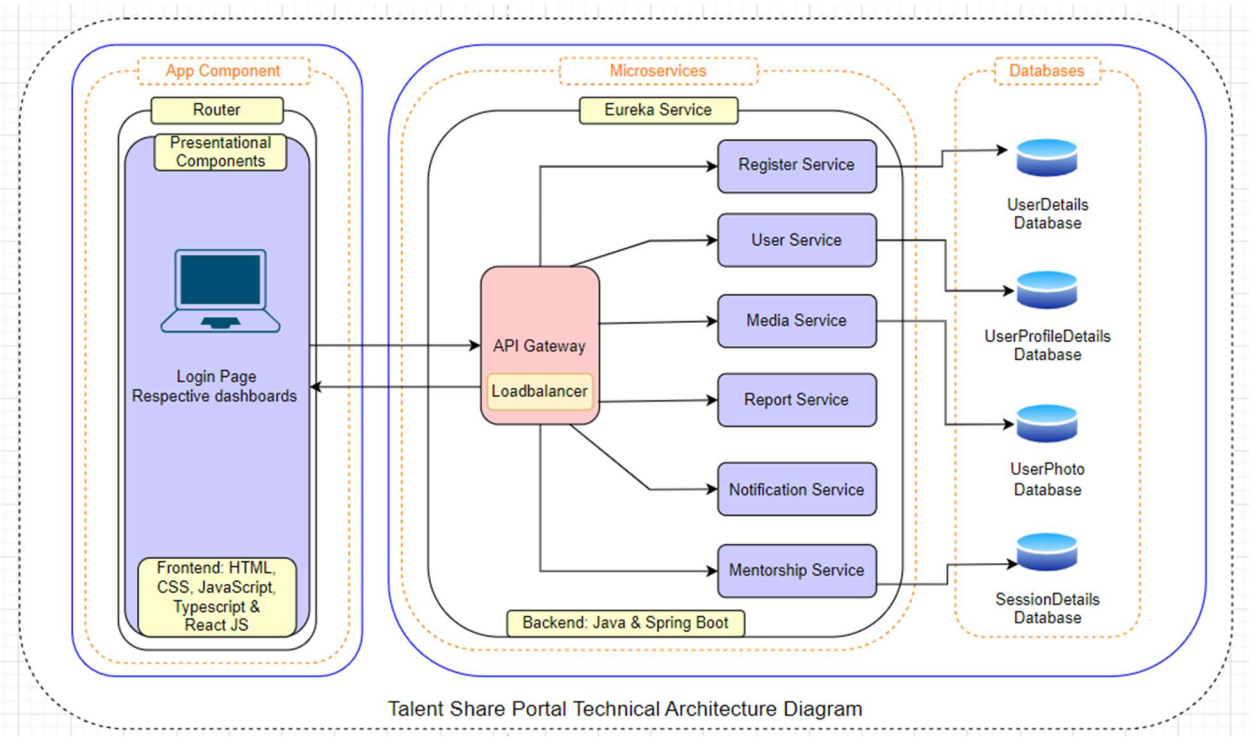
1.3 Document Overview

The document outlines the proposal for a “Talent Share Portal”, a web-based platform designed to facilitate mentorship relationships within the organization. The primary focus is on creating an environment where skilled employees can act as Mentors, sharing their knowledge with Mentees seeking technical upliftment or guidance. The absence of mentorship in the workplace is identified as a challenge, and the project aims to address this by providing a centralized system for mentorship. The document begins by stating the project goals, emphasizing the importance of mentoring as an investment in employee growth and the fostering of a positive company culture. It then delves into the problem statement, highlighting the difficulties faced by new associates when navigating unfamiliar tasks without proper guidance.

2. System Architecture

2.1Overview

The overall architecture of the Talent Share system is designed to be robust, scalable, and user-friendly. It consists of several interconnected modules and layers to facilitate effective mentorship relationships.



The architecture diagram consists of two main parts: Frontend and the backend.

Frontend:

The frontend is built using HTML, CSS, TypeScript, and React JS. The App Component is the root component of the application, and it contains the Router, which is responsible for navigating between different pages of the application. The Presentational Components are the individual components that make up the user interface of the application.

The Login page is the first page that users see when they access the application. It allows users to log in to their accounts using their credentials. Once logged in, users are redirected to their respective Dashboard page, which displays their personalized information and options.

Backend:

The backend is built using Java and Spring Boot. It consists of several microservices, each responsible for a specific functionality of the application.

These microservices are registered with the Eureka Service, which acts as a service registry, allowing the API Gateway to discover and load balance requests to the appropriate microservices.

The API Gateway is the entry point for all external requests to the application. It is responsible for routing requests to the appropriate microservices based on the URL path and HTTP method.

Register Service: Register service handles the creation of new user accounts. It provides an API endpoint for creating new users and validating their credentials.

- Create User: This functionality allows the creation of new user accounts. It takes the user's credentials (username, email, password) as input and stores them in the User Details Database.
- Validate Credentials: This functionality validates the user's credentials (username, email, password) before allowing them to log in. It checks the User Details Database to see if a user with the provided credentials already exists. If a user with the provided credentials exists, the validation is successful, and the user is allowed to log in. Otherwise, the validation fails, and the user is not allowed to log in.
- Delete User: This functionality allows the deletion of a user account. It takes the user's emailed as input and deletes the user's information from the User Details Database.
- Get User Profile: This functionality retrieves the user's profile information from the User Profile Details Database

User Service: User service provides APIs for fetching user profile data. It is responsible for retrieving user information from the User Profile Details Database.

- Update User Profile: This functionality allows the user to update their profile information. It takes the updated profile information as input and updates the User Profile Details Database with the new information.
- Search User: This functionality allows the search for a user based on their username or email. It takes the search query as input and searches the User Details Database for a user with a matching username or email.

- Manager Tagging: This functionality allows a manager to tag a user as a mentor. It takes the user's credentials as input and updates the User Profile Details Database with the manager's tag.
- Fetch Profile Data: This functionality retrieves the user's media files from the Media Storage Database.

Media Service: Media service handles requests related to user media, such as profile pictures. It provides APIs for uploading and retrieving user media.

- Profile Photo Update: This functionality allows the user to update their profile photo. It takes the new profile photo as input and updates the User Photo Database with the new photo.

Mentorship Service: Mentorship service provides APIs for managing mentorship relationships. It is responsible for handling requests related to mentor approvals, mentorship conclusions.

- Mentorship Request: Users can send a request to become a mentor or mentee to another user. The request contains information about the user and the type of mentorship they are seeking. The Mentorship Service handles the creation, retrieval, and management of these requests.
- Manager Approval: Before a mentorship request can be approved, it must go through a manager approval process. The Mentorship Service handles this process by allowing managers to view and approve or reject mentorship requests for mentees.
- Mentor Approval: Once a mentorship request has been approved by a manager, it must then be approved by the potential mentor. The Mentorship Service handles this process by allowing mentors to view and approve or reject mentorship requests for mentees.
- Mentor Conclusion: At the end of a mentorship relationship, the mentor can conclude the relationship in the Mentorship Service. This marks the end of the mentorship and allows for reporting and analysis of the relationship.

Report Service: Report service provides APIs for generating and managing reports. It is responsible for generating various types of reports, such as session reports and User Reports.

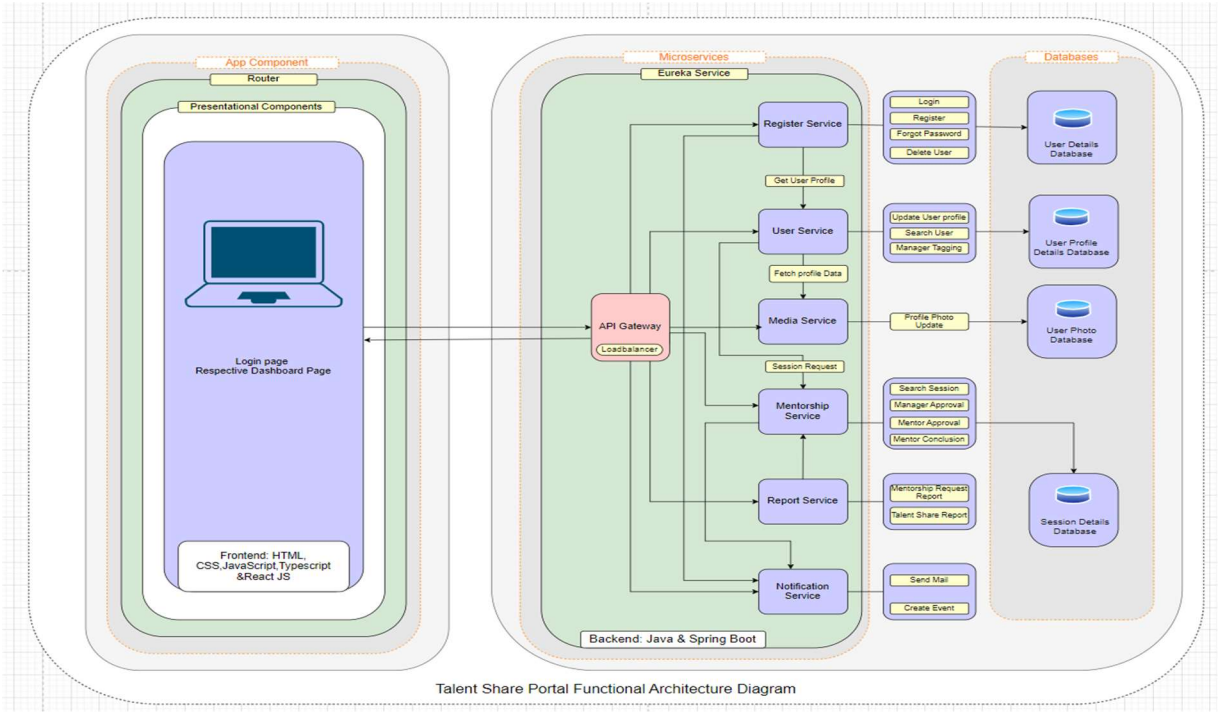
- Mentorship request Report: This functionality generates a report for a mentorship session. It takes the session ID as input and generates a report based on the session details from the Session Details Database.
- User Report: This functionality generates a report of all the user's talent share activities. Like whom is active and inactive.

Notification Service: Notification service provides APIs for sending mails to users. It is responsible for handling requests related to sending emails and other types of notifications.

- Send Mail: This functionality sends a notification to a user. It takes the user's ID and the notification message as input and sends a notification to the user's email address or other communication channel.

2.2 Key Components

The Talent Share Portal comprises several major key components, each playing a crucial role in facilitating mentorship relationships with Talent Share Portal. Here are the major Key components of the system:



User Management Session: This component allows the admin to manage user accounts. It includes features like creating Mentors, Mentees, and Managers, resetting passwords, controlling profile pictures, and deactivating inactive users.

Mentee Functionality: This component enables Mentees to search for Mentors based on skills. Once a suitable Mentor is found, Mentees can schedule sessions, automatically blocking their calendars. Notifications are sent to Mentors, who can accept or reject the session. After the session, Mentees can provide feedback.

Manager Functionality: Managers play a key role in the session approval process. They decide whether to accept or reject mentee requests based on predetermined criteria. This component ensures that mentorship aligns with organizational goals and standards.

Mentors Functionality: This component is designed for users with Mentor privileges. Mentors can view mentorship session requests from Mentees, review Mentee profiles, and choose to accept or reject session requests based on their availability.

Notification Session: This component handles email notifications for various system activities. Notifications are sent when a mentorship session request is received, when a new Mentor registers, and post-mentorship for feedback. It keeps all stakeholders informed and engaged throughout the mentorship process.

User Report: The User Report component is responsible for storing historical data related to completed mentorship relationships. It ensures that past interactions, mentorships, or shared resources are accessible for reference, reporting, or analysis purposes.

2.3 Interactions

The interaction among key components is crucial for the seamless functioning of the Talent Share Portal. Here's an overview of how these components interact with each other:

User Management Session: The User Management Session, handled by the admin, serves as the foundational component. The admin creates Mentors, Mentees, and Managers, setting up the user base for the mentorship platform. This information is essential for the other components to function effectively.

Mentee Functionality: The Mentee Functionality component interacts with the User Management Session to allow Mentees to search for Mentors. Mentees can then send mentorship session requests, and upon acceptance by Mentors, scheduling details are communicated. This component relies on both the Mentors' and Mentees' user data.

Manager Functionality: Managers, established through User Management, play a role in the approval process. When a Mentee requests a mentor, the Manager Functionality interacts with User Management to determine if the Manager approves or rejects the request based on criteria.

Mentors Functionality: Once Mentors are created through User Management, they can log in and access the Mentors Functionality component. Mentors can review profiles of Mentees, receive mentorship session requests, and either accept or reject them. This component relies on the user data established during the User Management Session.

Notification Session: When a mentee requests a mentorship session, an email notification is sent to the mentor's manager for approval. Upon manager approval, a notification is sent to the mentor, confirming the session request. In case of manager rejection, a notification is sent to the mentee, informing them of the rejection.

User Report: The user report feature is implemented to store users' data. This ensures tracking of user interactions, such as whether the user is active or inactive.

2.4 Technology Stack

Category	Technology
Backend	
Server-side scripting	Java
Server-side framework	Spring Boot
Tools	SonarQube and Kibana
Frontend	
Client-side scripting	TypeScript
Markup and Styling	HTML and CSS
Web Development	React.js & Spring Boot
Database	MySQL
Authentication	Base64
Communication	RESTful API
UI Design	Material-UI (React) & Bootstrap
Version Control	GitLab
Deployment	Docker & Jenkins
Testing	Junit (backend), Mockito (backend), Jest (frontend)
IDE	IntelliJ & Visual Studio

3. Major Components

3.1 Component

3.1.1 Overview

Major components work cohesively to establish, facilitate, and enhance mentorship relationships. The integration of these functionalities creates a robust Talent Share Portal, fostering a culture of continuous learning and support within the organization.

User Management Session: The User Management Session is the administrative hub of the system. Admins utilize this component to create and manage user accounts, including Mentors, Mentees, and Managers. It provides functionalities for password reset, control over profile pictures, and deactivation of inactive users. The accurate setup and maintenance of user profiles through this component are fundamental to the effective operation of other functionalities.

Mentee Functionality: The Mentee Functionality empowers Mentees to actively search for Mentors based on skills or technical interests. Once a suitable Mentor is found, Mentees can schedule sessions and automatically block their calendars. This component enhances the ease with which Mentees can connect with Mentors, fostering a collaborative learning environment.

Manager Functionality: Manager Functionality introduces an additional layer of approval in the mentorship process. Managers, equipped with decision-making authority, review and decide whether to accept or reject mentor requests. This component ensures that mentorship aligns with organizational goals and standards, adding a strategic dimension to the mentorship program.

Mentors Functionality: Mentors Functionality is designed for users with Mentor privileges. Mentors access this component to view and respond to mentorship session requests received from Mentees. They can review Mentee profiles before deciding to accept or reject session requests. This component ensures that experienced individuals are efficiently matched with those seeking mentorship.

Notification Session: The Notification Session is a communication lifeline for all stakeholders. It sends email notifications at critical junctures, including when mentorship session requests are received, post-mentorship for feedback, upon new Mentor registrations (subject to Admin approval), and when a new user is created by the Admin. This component keeps users informed and engaged throughout the mentorship lifecycle.

User Report: The user report feature is implemented to store users' data. This ensures tracking of user interactions, such as whether the user is active or inactive.

3.1.2 Functionality

Component's functionality is designed to fulfil specific roles in the mentorship process. Together, these components create a comprehensive Talent Share Portal that promotes a culture of continuous learning and support.

User Management Session:

Create Users: The admin can create user accounts for Mentors, Mentees, and Managers.
Password Management: Ability to reset passwords for users.
Profile Control: Complete control over profiles, allowing the admin to manage user profiles.
Deactivation: Admin can deactivate inactive user accounts.

Mentee Functionality:

Mentor Search: Mentees can search for Mentors based on skills or technical interests.
Session Scheduling: Once a suitable Mentor is found, Mentees can schedule sessions and block calendars.
Notifications: Notifications are sent to Mentors, who can then accept or reject the session request.

Manager Functionality:

Approval Process: Managers play a key role in the approval process, deciding whether to accept or reject mentor requests based on predefined criteria.
Align with Goals: Ensures that mentorship aligns with organizational goals and standards.

Mentors Functionality:

Session Requests: Mentors can view mentorship session requests from Mentees.
Profile Review: Mentors can review Mentee profiles before accepting or rejecting session requests.
Accept/Reject: Mentors can accept or reject session requests based on availability.

Notification Session:

Session Requests: Sends email notifications to Mentors when mentorship session requests are received.
Feedback: Notifies Mentees with an MS-Form link for post-mentorship feedback.
New Registrations: Admin receives notification emails for new Mentor registrations and approvals.
User Creation: Users receive notification emails when created by the Admin.

User Report:

Data Retention: The user report feature is implemented to store users' data. This ensures tracking of user interactions, such as whether the user is active or inactive.

3.1.3 Interfaces

Admin (User) Dashboard: A centralized dashboard where the admin can access all user management functionalities.

Interfaces:

- User Management
- Feedback
- User Report
- Manager Tagging
- Mentor Ship Record
- BU Session

Mentee Dashboard: A user-friendly dashboard providing access to mentor search, session scheduling, and notifications.

Interfaces:

- Profile
- Mentor Search Page
- Request for Mentorship Session
- Mentorship Session History

Manager Dashboard: A dedicated dashboard for Managers, showing pending mentor requests and decision-making tools.

Interfaces:

- Profile
- Approval/Rejection Panel
- Mentorship Session History
- Feedback

Mentors Dashboard: A personalized dashboard displaying mentorship session requests, profile reviews, and availability settings.

Interfaces:

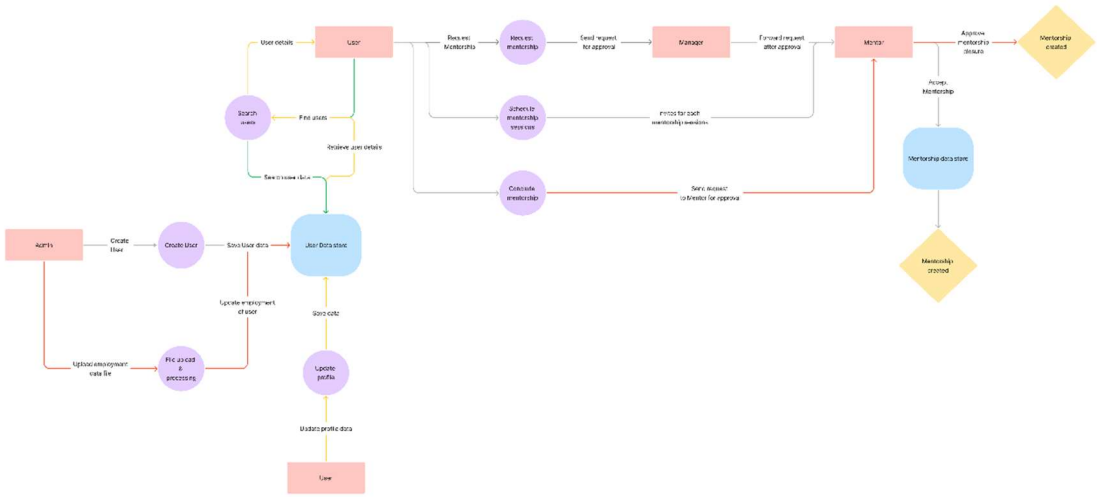
- Profile
- Accept/Reject Buttons
- Mentorship Session History

Notification Session: Predefined templates for different email notifications (e.g., session requests, feedback).

4. Data Flow Diagram

4.1 Overview

The high-level data flow in the Talent Share Portal involves the movement of information between different components to facilitate mentor-mentee relationships. Here's an overview of the high-level data flow:



User Management Session:

Data Flow:

- The admin creates and manages user accounts for Mentors, Mentees, and Managers.
- User data, including profiles and roles, is stored in the database.

Mentee Functionality:

Data Flow:

- Mentees search for Mentors based on skills.
- Mentees send mentorship session requests, and their profiles are updated accordingly.
- Scheduled session information is stored in the system.

Manager Functionality:

Data Flow:

- Managers review mentorship session requests and decide to approve or reject.
- Approval/rejection status is updated in the system.
- Manager criteria and decisions are stored for reference.

Mentors Functionality:

Data Flow:

- Mentors access the system and review mentorship session requests.
- Upon accepting or rejecting a request, the status is updated in the system.
- Mentor availability and profile information are retrieved from the database.

Notification Session:

Data Flow:

- Email notifications are triggered for various events (e.g., session requests, feedback links, new registrations).
- Data related to notifications (e.g., recipient, content) is processed and sent.

User Report:

Data Flow:

- The user report feature could be implemented to store users' historical data. This ensures that interaction of the user like user is active or not active.

Frontend (UI) Interactions:

Data Flow:

- User interactions on the frontend trigger requests to the backend API.
- Data is sent and received through RESTful API & GraphQL queries.

Database Interactions:

Data Flow:

- User data, mentorship session details, and historical data are stored in the database.

- Database queries and updates are performed based on user interactions and system events.

4.2 Data Sources

Talent Share Portal, various sources contribute to the data flow and functionality of the system. Here are the key sources of data:

User Input and Registration: Admin provide information during the registration process. Mentors, Mentees, and Managers provide details such as names, email addresses, roles, and other profile-related information.
Role in Data Flow: Initiates the creation of user accounts and profiles.

User Profiles: User profiles store information about each user, including their role (Mentor, Mentee, Manager), skills, expertise, availability, and other relevant details.
Role in Data Flow: Used by various components to display user information, match mentors with mentees, and make decisions based on user roles.

Mentorship Session Requests: Mentees send requests for mentorship sessions, providing details about the skills they want to learn and their preferences.
Role in Data Flow: Triggers the notification system, informs Mentors, and initiates the approval process by Managers.

Manager: Managers set criteria for approving or rejecting mentorship session requests. They make decisions based on predefined standards and organizational goals.
Role in Data Flow: Guides the manager functionality in the approval or rejection of mentor requests.

Mentor and Mentee Availability: Mentors and Mentees specify their availability for mentorship sessions, including time slots and preferred hours.
Role in Data Flow: Ensures that mentorship sessions are scheduled at mutually convenient times.

Notification Emails: The system sends notification emails to alert users about various events, such as new mentorship session requests, approvals, rejections, and post-session feedback links.
Role in Data Flow: Keeps users informed and engaged throughout the mentorship process.

Feedback Forms: Mentees provide feedback after mentorship sessions through forms, sharing their experiences and suggestions.
Role in Data Flow: Offers insights into the effectiveness of mentorship relationships and helps in evaluating mentor performance.

4.3 Data Processing

In the Talent Share Portal, data is processed through a series of steps and interactions among various components. The processing involves user inputs, system-generated events, and the manipulation of data to facilitate mentorship relationships and ensure the smooth functioning of the platform. Below is an explanation of how data is processed within the system:

User Registration and Profile Creation:

Input: Users (Admin, Mentors, Mentees, Managers) provide registration details.

Processing:

- User accounts are created based on the provided information.
- User profiles are generated, storing relevant details such as skills, expertise, and roles.

Mentorship Session Requests:

Input: Mentees send mentorship session requests with details about the skills they want to learn.

Processing:

- Notifications are triggered to inform Mentors about session requests.
- Managers may review and approve or reject requests based on predefined criteria.

Notification System:

User Registration:

- Upon user registration, an automated email containing the system-generated password is sent to the user's email address for login credentials.

Password Reset Confirmation:

- When a user initiates a password reset request, a confirmation email is sent to the user's email address to verify the action.

Password Reset:

- In the event of forgetting a password, an automated email containing a newly generated password is sent to the user's email address.

Session Request Notification:

- If a mentee requests a mentorship session, an email notification is sent to the mentor's manager for approval.
- Upon manager approval, a notification is sent to the mentor, confirming the session request.
- In case of manager rejection, a notification is sent to the mentee, informing them of the rejection.

Session Confirmation:

- When a mentor accepts a session request, an email notification is sent to the mentee, confirming the session details.

- Additionally, a notification is sent to the mentee's manager to keep them informed about the scheduled session.

4.4 Data Destinations

In the Talent Share Portal, processed data is directed to various destinations to fulfil different purposes within the system. Here are the destinations for processed data:

1. **User Profiles:** Processed user registration data and profile information are stored in the user profiles.
    - **Purpose:** User profiles serve as a centralized repository of user information, including roles, skills, and availability.
2. **Notification Emails:** Processed notification data triggers the sending of email notifications to relevant users.
    - **Purpose:** Email notifications keep users informed about mentorship session requests, approvals, rejections, feedback, and other significant events.
3. **Mentorship Session Details:** Data related to mentorship sessions, including session requests, approvals, and feedback, is stored in the system.
    - **Purpose:** This data is crucial for tracking ongoing mentorship relationships, evaluating mentor performance, and generating historical reports.
4. **Manager Approval/Rejection Decisions:** Data reflecting Manager decisions on mentorship requests is stored in the system.
    - **Purpose:** This information guides the mentorship approval process, ensuring alignment with organizational goals and predefined criteria.
5. **Feedback Forms and Data:** Processed feedback data from Mentees is stored for performance evaluation.
    - **Purpose:** Feedback data provides insights into the effectiveness of mentorship relationships, helps in evaluating mentor performance, and contributes to continuous improvement.
6. **User Report:**
    - **Purpose:** The user report feature is implemented to store users' data. This ensures tracking of user interactions, such as whether the user is active or inactive.
7. **User Interfaces (Frontend):** Processed data is presented to users through the frontend interfaces.
    - **Purpose:** The user interfaces display relevant information, notifications, and interactive elements for seamless user interactions.
8. **Reporting and Analysis Tools:**
    - **Description:** Processed and User Report data is used by reporting and analysis tools within the system.
    - **Purpose:** Reporting tools generate insights into mentorship program effectiveness, user engagement, and other key metrics.

5. Security Considerations

5.1Authentication

User Registration: During user registration, the user provides necessary information, including a unique username and a strong, unique password.

Password Hashing: The user's password is securely hashed using a strong base64 as encoder.

Database Storage: The hashed password, along with other user details, is stored in the database. The original password is not stored.

Login Process: When a user attempts to log in, they provide their username and password.

Password Verification: The system retrieves the hashed password associated with the provided username from the database.

Authentication Decision: If the base64 encoded value matches, the entered password is correct, and the user is authenticated. Otherwise, authentication fails.

5.2 Authorization

Role-Based Authorization:

User Roles: Users in the Talent Share Portal are assigned specific roles such as Admin, Mentor, Mentee, or Manager during the registration process.

- Role in Authorization: User roles determine the level of access and permissions a user has within the system.

Role-Based Access Control (RBAC): RBAC is implemented to define and manage user roles, along with the corresponding permissions associated with each role.

- Role in Authorization: Determines what actions a user is allowed to perform based on their role, such as creating mentorship sessions, approving requests, or managing user accounts.

5.3 Data Encryption

Encrypting sensitive data is crucial to ensure that even if unauthorized access occurs, the data remains protected and unreadable. In the context of the Talent Share Portal, where user information, mentorship details, and potentially other sensitive data are involved, encryption plays a vital role in securing this information.

Example: **Password Encryption:** Protecting user passwords from unauthorized access.

Implementation:

- Using a strong base64 encoder.
- Ensure that only the hashed representation of passwords is stored in the database, not the actual passwords.

6. Scalability and Performance

6.1 Scalability Measures

Handling scalability is crucial to ensure that the Talent Share Portal can efficiently accommodate a growing number of users, mentorship sessions, and data without compromising performance. Scalability can be achieved through Load balancing scaling.

Load Balancing: Load balancing distributes incoming network traffic across multiple servers to ensure that no single server becomes a bottleneck.

Implementation:

- Employ a load balancer to distribute requests evenly among multiple server instances.
- This helps distribute the processing load and prevents overload on any individual server.

6.2 Performance Optimization

Optimizing system performance is essential for ensuring that the Talent Share Portal operates efficiently and provides a responsive user experience. Here are strategies for optimizing system performance that we are using:

Code Optimization:

Minimize Code Execution Time:

- Identifying and optimizing code segments that contribute significantly to execution time.
- Using profiling tools to identify performance bottlenecks and optimize critical code paths.

Reduce Redundant Operations**:**

- Eliminate redundant or unnecessary operations in code.
- Implement efficient algorithms and data structures to improve overall code efficiency.

Database Optimization:

Indexing:

- Ensure that relevant database fields are indexed to speed up data retrieval.
- Regularly analyse and optimize database queries to utilize indexes effectively.

Query Optimization:

- Optimize complex queries and avoid unnecessary joins.
- Use database query optimization tools to analyse and enhance query performance.

Load Balancing: Use load balancers to evenly distribute incoming traffic across multiple servers. Ensure that the load is balanced based on server capacity and performance.

7. Error Handling and Logging

7.1 Error Handling Strategies

Error Logging: Implement a robust logging mechanism to record details of errors, warnings, and exceptions.

Clear and Informative Messages: Provide user-friendly error messages that convey the nature of the issue without revealing sensitive information.

Using Appropriate HTTP Status Codes: Use standard HTTP status codes to communicate the outcome of HTTP requests. For example, use 404 Not Found for missing resources and 500 Internal Server Error for generic server errors.

7.2 Logging Mechanisms

Logging mechanisms are crucial for monitoring and troubleshooting the behaviour of a system. In the context of the user registration, authentication, and password management system described above, several logging mechanisms can be employed at various stages to ensure smooth operation and security. Here's how logging can be incorporated into the system:

User Registration Logging:

When an admin registers a new user, log the following information:

- Date and time of registration.
- Admin user who performed the registration.
- User's details such as name & email.

Email Notification Logging:

When the system sends an email containing the generated password to a user, log the following information:

- Date and time the email was sent.
- User's email address.
- Status of the email delivery.

User Login Logging:

Whenever a user logs in, log the following information:

- Date and time of login.
- User's email address.
- Success or failure of the login attempt.

Password Change Logging:

When a user changes their password, log the following information:

- Date and time of the password change.
- User's email address.
- Success or failure of the password change attempt.

Password Reset Logging:

When a user initiates a password reset request, log the following information:

- Date and time of the password reset request.
- User's email address.
- Status of the password reset request (success or failure).

Error Logging:

- Log any errors or exceptions encountered during the operation of the system.
- Include relevant details such as the error message, stack trace, and context information.
- This log helps in debugging and resolving issues with the system's functionality.

Security Logging:

- Log security-related events such as failed login attempts, account lockouts, and password policy violations.
- Monitor for suspicious activities and potential security threats.

This log helps in detecting and responding to security incidents.

8. Dependencies

8.1 External Dependencies

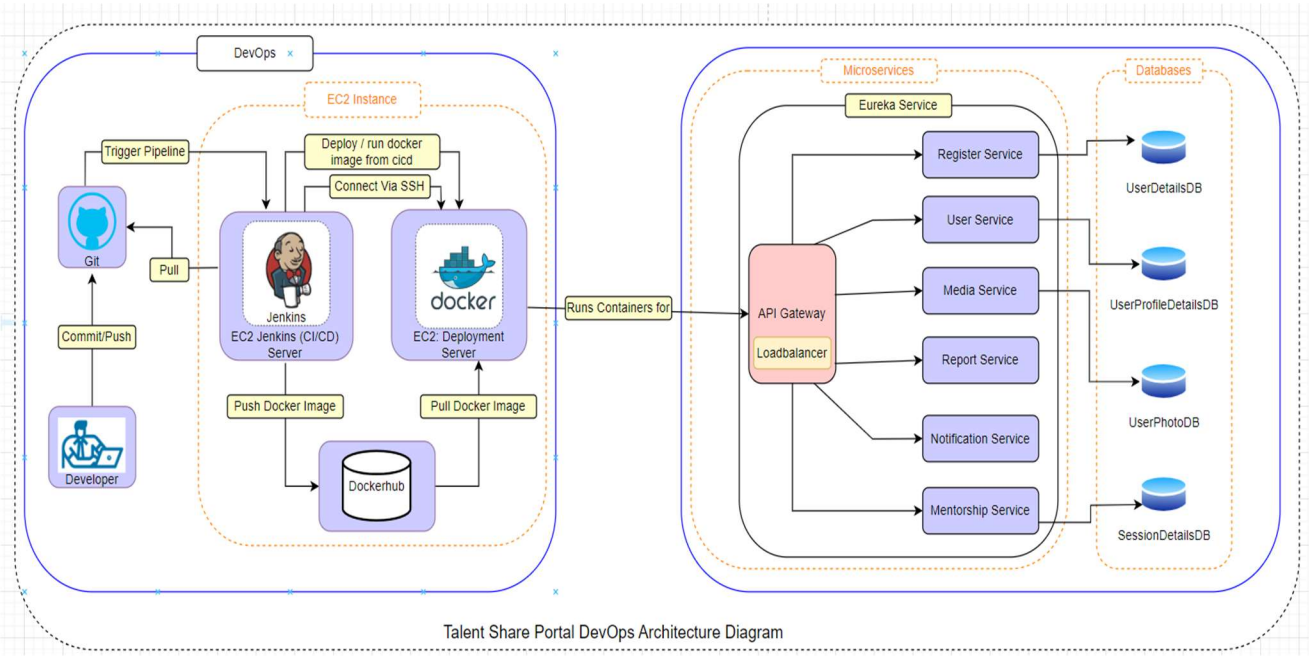
- Spring Boot
- Java
- React
- MySQL
- GitLab
- Rest API
- Junit (Mockito)

8.2 Internal Dependencies

- User Management Module
- Mentor and Mentee Functionality
- Manager Approval Process
- Notification System
- Session Management
- Search Functionality
- User Report

9. Deployment Architecture

The architecture provides a scalable and flexible way to build and deploy microservices-based applications, leveraging the power of DevOps practices and tools.



9.1 Server Configuration

- Register Service
- User Service
- Media Service
- Report Service
- Notification Service
- Mentorship Service

9.2 Deployment Topology

Manual Deployment Using JAR File

Web Servers: Deploy web servers (e.g., Apache Tomcat, Nginx) to serve the application. Configure the web servers to handle HTTP requests and proxy them to the application server.

Application Server: Package the application as a JAR file. Deploy the JAR file to an application server (e.g., Apache Tomcat, Jetty, or a Java application server).

Database Layer: Set up a database server (e.g., MySQL) to store application data. Configure the application server to connect to the database using appropriate connection settings.

Deployment Process: Manually deploy the JAR file to the application server. Configure environment-specific properties, such as database connection details, through configuration files or environment variables. Start the application server to make the application accessible.

Automated Deployment Using Docker and Jenkins: To set up an automated deployment topology using Docker and Jenkins for the Talent Share Portal, we'll design a CI/CD pipeline that builds Docker images, pushes them to a registry, and deploys them to a Docker environment. Here's an outline of the topology:

Source Control Repository:

- The source code of the Talent Share Portal is stored in a version control system (VCS) such as GitLab.

Jenkins Server:

- Jenkins is installed on a dedicated server or virtual machine.
- Jenkins is configured with necessary plugins for Docker integration and pipeline execution.

Docker Registry:

- A Docker registry (e.g., Docker Hub, AWS ECR, or a private registry) is used to store Docker images built by Jenkins.

Docker Host:

- A Docker host environment (e.g., Docker Swarm, & Docker server) where the Talent Share Portal containers will be deployed.

Automated Deployment Pipeline:

- Jenkins Pipeline is configured to automate the deployment process.

The pipeline consists of several stages:

- Checkout: Jenkins pulls the latest source code from the VCS repository.
- Build: Jenkins builds the Docker image for the Talent Share Portal using the Dockerfile.
- Test: Automated tests are run to ensure the integrity of the application.
- Push Image: Jenkins pushes the built Docker image to the Docker registry.

- Deploy: Jenkins deploys the Docker image to the Docker host environment using Docker commands or orchestration tools like Docker Swarm or Docker Server.
- Post-deployment Tests: Additional tests may be performed after deployment to verify that the application is functioning correctly in the production environment.
- Notifications: Email notifications or other alerts are sent to relevant stakeholders to inform them of the deployment status.

Security Considerations:

- Access controls and permissions are implemented to restrict access to sensitive resources such as the Jenkins server and Docker registry.
- Secrets management solutions are utilized to securely store credentials and API tokens needed for authentication with the Docker registry and other services.

Monitoring and Logging:

- Monitoring tools such as Prometheus and Grafana are set up to monitor the health and performance of the Docker environment.
- Logging solutions like ELK Stack (Elasticsearch) are used to aggregate and analyse logs generated by the Talent Share Portal containers.

Scaling and High Availability:

- The Docker environment is designed for scalability and high availability using container orchestration platforms like Kubernetes or Docker Swarm.
- Load balancing and auto-scaling configurations are implemented to handle increased traffic and ensure uptime.

9.3 Deployment Steps

Manual Deployment Using JAR File:

Prerequisites:

- Java Runtime Environment (JRE)
- Application server (e.g., Tomcat)
- Database server (e.g., MySQLL)
- Web server (e.g., Apache)

Deployment Steps:

Build the Application: Compile and build the application into a JAR file. Use the build tool specified in the project documentation (e.g., Maven).

Database Configuration: Set up a database instance and create a database for the Talent Share Portal. Configure the application to connect to the database by providing connection details in the application's configuration files.

Web Server Configuration**:** Install and configure a web server (e.g., Apache) to act as a reverse proxy for the application server. Configure the web server to forward requests to the application server.

Automated Deployment Using Docker and Jenkins:

Prerequisites:

- Docker installed on the deployment machine.
- Jenkins server set up and running.
- Access to a container registry (e.g., Docker Hub)

Deployment Steps: Automating the deployment of the Talent Share Portal using Docker and Jenkins involves several steps. Here's a high-level overview of the process:

Prepare Dockerized Application:

- Dockerize the Talent Share Portal application by creating a Docker file that specifies the environment and dependencies needed to run the application.
- Include any necessary configurations, such as environment variables or volume mounts, in the Docker file.
- Build a Docker image for the application using the Docker file.

Set Up Jenkins:

- Install Jenkins on a server or virtual machine.
- Configure Jenkins with necessary plugins for Docker integration, such as Docker Pipeline plugin.
- Set up authentication and access controls in Jenkins as per your organization's requirements.

Configure Jenkins Pipeline:

- Create a Jenkins Pipeline script (Jenkins file) that defines the stages and steps for deploying the Talent Share Portal.
- Define stages for building the Docker image, pushing the image to a Docker registry, and deploying the application.
- Use Jenkins Pipeline syntax to specify the commands and parameters for each stage.

Integrate with Version Control System (VCS):

- Connect Jenkins to your preferred version control system (e.g., Git) where the source code of the Talent Share Portal is hosted.
- Configure Jenkins to trigger the deployment pipeline automatically whenever changes are pushed to the VCS repository.

Deploy to Docker Environment:

- Within the Jenkins Pipeline script, use Docker commands or Docker Pipeline syntax to interact with Docker.
- Pull the latest changes from the VCS repository.
- Build the Docker image using the Docker file.
- Push the Docker image to a Docker registry (e.g., Docker Hub).
- Deploy the Docker image to the Docker environment (e.g., Docker Swarm, Kubernetes).

Test Deployment:

- Include automated tests as part of the deployment pipeline to ensure the deployed application is functioning correctly.
- Using tools like Jest for UI testing, JUnit with Mockito for unit testing, Swagger or Postman for API testing.

Monitor Deployment:

- Implemented monitoring and logging solutions to track the deployment process and monitor the health of the deployed application.
- Set up alerts and notifications for any deployment failures or performance issues.

Continuous Integration/Continuous Deployment (CI/CD):

- Once the Jenkins Pipeline is set up, Jenkins will automatically trigger the deployment process whenever changes are pushed to the VCS repository.
- This establishes a CI/CD workflow, where code changes are automatically built, tested, and deployed to production.

10. Maintenance and Support

10.1 Monitoring

Monitoring the Talent Share Portal system involves observing its performance, availability, security, and user interactions. Here's how the system can be monitored based on the provided information:

Performance Monitoring:

- Tracking system response times, throughput, and resource utilization (CPU, memory, disk usage) to ensure optimal performance.
- Monitoring database performance, including query execution times and database server metrics.
- Utilizing performance monitoring tools and frameworks to identify bottlenecks and optimize system performance.

Availability Monitoring:

- Monitoring server uptime and availability to ensure the Talent Share Portal is accessible to users.
- Setting up automated alerts for server downtime or service disruptions.
- Implementing redundancy and failover mechanisms to minimize downtime and ensure high availability.

Security Monitoring:

- Monitoring system logs for any suspicious activities or security breaches.
- Utilizing intrusion detection and prevention systems (IDPS) to detect and respond to security threats.
- Regularly review access logs, authentication logs, and audit logs to identify unauthorized access attempts or unusual behaviour.

User Interaction Monitoring:

- Tracking user activity within the Talent Share Portal, including login attempts, session requests, and feedback submissions.
- Monitoring user engagement metrics, such as the number of mentorship sessions scheduled, feedback provided, and mentorship relationships formed.
- Analysing user behaviour patterns to identify areas for improvement and enhance user experience.

Notification Monitoring:

- Monitor the delivery of email notifications to ensure timely and accurate communication with users.
- Set up alerts for failed email deliveries or bounced emails.
- Regularly review email delivery logs to identify any issues with the notification system.

Feedback Monitoring:

- Monitor feedback submissions from mentees after each mentorship session.
- Analize feedback data to assess the effectiveness of mentorship relationships and identify areas for improvement.
- Set up alerts for negative feedback or unresolved issues reported by mentees.

User Report Monitoring:

- Monitor the generation and storage of User Reports to ensure data accuracy and completeness.
- Regularly review report generation logs and storage utilization.
- Implement backup and data retention policies to safeguard user's data.

10.2 Support Mechanisms

When Talent Share Portal is UNRESPONSIVE:

Domain	Email-Id
Backend & DevOps Developer	rahulyadavv2011421@gmail.com
Frontend Developer	maheshkmnk95@gmail.com

11. Appendix

11.1Acronyms and Abbreviations

- **MS-Excel-** Microsoft Excel
- **JRE-** Java Runtime Environment
- **API-** Application Programming Interface
- **JAR-** Java Archive
- **HTTP-** Hypertext Transfer Protocol
- **HTTPS-** Hypertext Transfer Protocol Secure
- **IP-** Internet Protocol
- **URL-** Uniform Resource Locator
- **URI-** Uniform Resource Identifier

11.2 Glossary

Admin: An employee designated as an administrator with the authority to create and manage user accounts, including mentors, mentees, and managers.

Mentee: An individual seeking guidance and support in specific skills or technologies from a more experienced mentor.

Manager: A role responsible for overseeing and managing the mentorship relationships, including approving mentor requests and ensuring alignment with organizational goals.

Mentor: An experienced individual who offers guidance, knowledge, and support to less experienced individuals (mentees) in specific skills or technologies.

Talent Share Portal: The web-based platform designed to facilitate mentorship relationships within organization, connecting mentors and mentees based on their skills and interests.

User Management Session: A session within the portal where the admin manages user accounts, including creating mentors, mentees, and managers, resetting passwords, and handling profile pictures.

Feedback Form: A form provided to mentees after a mentorship session to collect feedback on the mentor's performance and the overall mentoring experience.

Notification Session: The system-generated email notifications sent to users to inform them of mentorship session requests, session acceptances or rejections, and other relevant events.

User Report: The user report feature is implemented to store users' data. This ensures tracking of user interactions, such as whether the user is active or inactive.

Docker: A platform for containerization, allowing the creation and deployment of containers to run applications consistently in different environments.

Jenkins: An automation server used for continuous integration and continuous deployment (CI/CD) processes, automating the building, testing, and deployment of the Talent Share Portal.

Base64: A binary-to-text encoding scheme used for encoding binary data, such as user authentication credentials, for safe transmission in the Talent Share Portal.

Load Balancer: A component that distributes incoming network traffic across multiple web servers to ensure optimal resource utilization, high availability, and reliability.

----------END----------