IIHT

Time To Complete: 10 to 12 hr

Skill Tracker Application

Contents

[1 Problem Statement 2](#_Toc82421559)

[2 PROPOSED SKILL TRACKER APPLICATON WIREFRAME 2](#_Toc82421560)

[3 Tool Chain 3](#_Toc82421561)

[4 Business Requirements: 4](#_Toc82421562)

[5 Proposed Rest Endpoints to be exposed 6](#_Toc82421563)

[5.1 Rest APIs: 6](#_Toc82421564)

[6 Rubrics/Expected Deliverables 6](#_Toc82421565)

[7 Implementation/Functional Requirements 7](#_Toc82421566)

[7.1 Product and Frameworks: 7](#_Toc82421567)

[7.2 Governance and Tooling: 8](#_Toc82421568)

[7.3 Code Quality/Optimizations 8](#_Toc82421569)

[8 Platform 8](#_Toc82421570)

[8.1 Cloud Specific Design 8](#_Toc82421571)

[8.2 Deployment 8](#_Toc82421572)

[8.3 Microservice Deployment 9](#_Toc82421573)

[8.4 FrontEnd Deployment 9](#_Toc82421574)

[8.5 Management and Governance 9](#_Toc82421575)

[8.6 Migration & Transfer 10](#_Toc82421576)

[9 Methodology 10](#_Toc82421577)

[9.1 Agile 10](#_Toc82421578)

# Problem Statement

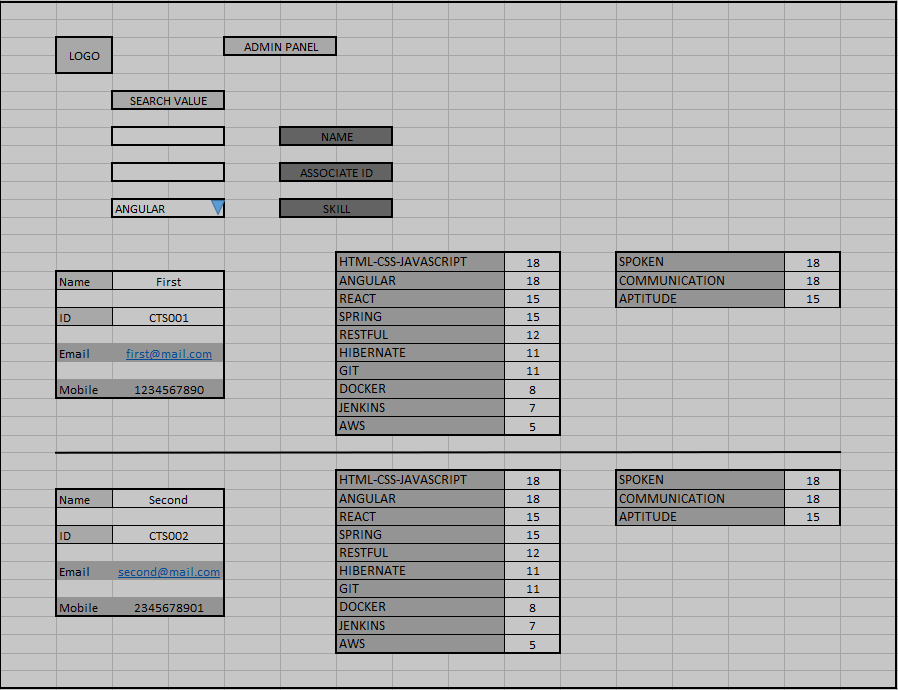
**Skill Tracker Application** is microservice based Cloud Native Application. The Main purpose of **Skill Tracker** is to keeping track of skill of individual Full Stack Engineers. The core modules of Skill Tracker app are:

* Full Stack Engineer would add their profile with skill set (skill set detailed in User stories)
* Full Stack Engineer would be able to update their skill set.
* Admin would be able to search profiles based on certain criteria (criteria are detailed in User Stories)

The scope includes developing the application using tool chain mentioned below.

# PROPOSED SKILL TRACKER APPLICATON WIREFRAME

Below is the wireframe for reference.



# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Networking and Content Delivery |  |
|  | Ways of Working |  |
|  | Consulting Mindset |  |
|  | DevOps |  |
|  | Secure Coding |  |
|  | Code Quality |  |
| Programming Languages | Application Language | Java |
| Products & Frameworks | Presentation | Angular/React |
|  |  | Javascript/Typescript |
|  |  | Bootstrap |
|  | Compute & Integration | Spring Boot |
|  |  | Spring Batch |
|  |  | ELK Stack |
|  |  | Kafka/Rabbit MQ/ Active MQ |
|  |  |  |
|  | Governance & Tooling | Git |
|  |  | Mockito |
|  |  | Jasmine/Jest |
|  |  | Protractor/Cypress |
| Engineering Quality |  |  |
| Platform | Cloud Tools | Amazon Dynamo DB |
|  |  | AWS EC2 |
|  |  | Amazon ElastiCache |
|  |  | AWS Simple Storage Service |
|  |  | AWS CodeDeploy |
|  |  | AWS API Gateway |
|  |  | Amazon Route 53 |
|  |  | AWS Auto Scaling |
|  |  | AWS MQ |
|  |  | AWS Elastic Blob Storage (EBS) |
|  |  | AWS ELB – Load Balancer |
|  |  | Amazon CloudWatch |
|  |  | AWS Cognito |
|  |  | AWS DataSync |

# Development Flow

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **MC** | **Competency** | **Section** | **Indicative**  **Mechanism for Evaluation ( Passing score of 60% in each MC)** | **Points to Note** |
| [**Business Requirement**](#_Business-Requirement:) | | | | |  |
| **1** | **Backend** | Rest API, Database, Messaging, Log/Monitoring, Pros/Cons of adjacent frameworks | [Click here](#_Proposed_Rest_Endpoints) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete this MC with 60% to proceed with the next Frontend MC |
| **2** | **Front End** | Angular/React | [Click here](#_Frontend) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete each MC with 60% to proceed with the next Cloud MC |
| **3** | **Cloud** | Compute, Identity, Compliance, Security and Content Delivery | [Click here](#_Platform) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete this Cloud MC with 60% |

# Business Requirements:

Below are the user stories for the given problem statement

|  |  |  |  |
| --- | --- | --- | --- |
| **User**  **Story #** | **User Story Name** | **User Story** | **Development** |
| US\_01 | Full Stack Engineer Functionalities | As a Full Stack Engineer, I can add my profile  *While adding a profile, following personal information is required.*   * Name * Associate Id * Mobile * Email   It also requires expertise level on following technical skills (Use these static list of skill set):   * HTML-CSS-JAVASCRIPT * ANGULAR * REACT * SPRING * RESTFUL * HIBERNATE * GIT * DOCKER * JENKINS * AWS   It also requires expertise level on following non-technical skills:   * SPOKEN * COMMUNICATION * APTITUDE   Constraints:   1. Expertise level for each skill must range between 0-20, else throw a custom exception 2. Expertise level must not be empty or a non-numeric value, else throw a custom exception   Validations:   1. Name is not null, min 5 and max 30 characters. 2. AssociateID is not null, min 5 and max 30 characters, and must start with “CTS” 3. email is not null, and it should be valid email pattern, containing a single @. 4. mobile is not null, min 10 and max 10 character and all must be numeric.   Post Condition:   1. After successfully adding a new profile in Database a unique userId is to be generated. 2. Maintain date of adding the profile in database | Only API to be developed |
| US\_02 | Full Stack Engineer Functionalities | As a Full Stack Engineer, I am able to edit my profile by providing my userid  *While updating the profile, I am allowed to update only expertise level of my skills. Update on personal information is not allowed*  Constraints:   1. Update of profile must be allowed only after 10 days of adding profile or last change, else throw a custom exception 2. Expertise level for each skill must range between 0-20 3. If Invalid userid is provided, it must throw a custom exception   Post Condition:   1. Maintain date of last update in database | Only API to be developed |
| US\_03 | Admin functionality | As an admin I can search for profile based on following criteria:   1. Name: Initial characters of name can be provided 2. Associate Id 3. Skill: Skill name must be provided from above list (user-story 1), and only those records shall be returned which has expertise level greater than 10 on that skill.   Considerations:   1. While fetching the above details the API must return the personal information along with all skills and their respective expertise level. 2. Skill details must be in descending order of expertise level. 3. If any other criteria is provided for searching other than above mentions, application must throw a custom exception. | API and Frontend to be developed |

# Proposed Rest Endpoints to be exposed

## Rest APIs:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /skill-tracker/api/v1/engineer/add-profile | Adds a new profile | | / skill-tracker/api/v1/engineer/update-profile/{userId} | Update the skill details of engineer with provided userId | |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /skill-tracker/api/v1/admin/{criteria}/{criteriaValue} | Searches for profile based on provided criteria | |

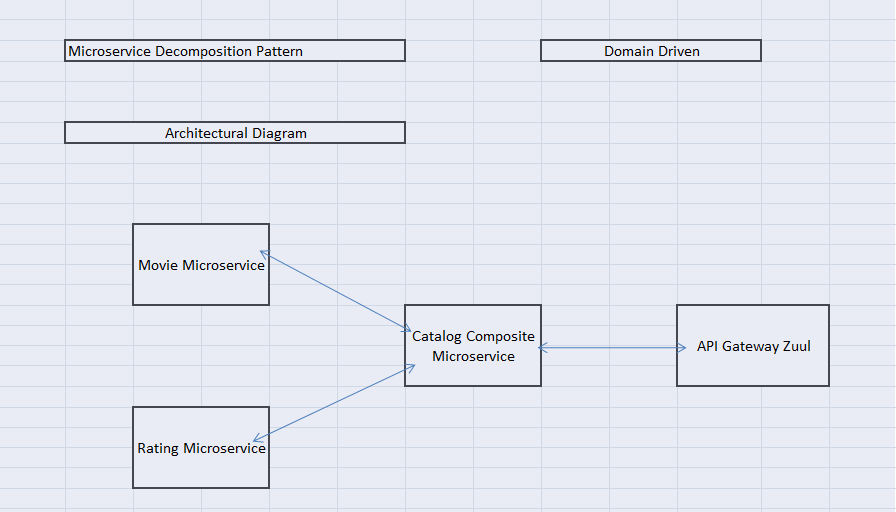
# Rubrics/Expected Deliverables

**Architecture definition**

You are expected to share a design document describing the below. It may contain diagram, flowcharts etc. You may use a presentation tool like PowerPoint or Word Document or simple text file.

* The Design approach should follow SOLID principles.
* Identify the design patterns and application of the same should be clearly stated in the document.
* Architectural assumptions, dependencies, and risks should be clearly stated with mitigation for the same.
* Capture how the Non-Functional requirements like Security, Performance, Availability, Scalability and Resilient are achieved.

**A Reference representation can be as follows:**

****

# Implementation/Functional Requirements

## Product and Frameworks:

1. **Compute and Integration**

Develop the backend application as a microservice architecture. (Implementation as follows)

* + 1. Identify the best decomposition pattern and create microservice based on that (mention the architecture of same in design document)
    2. Identify the best Database Deployment pattern for use case and implement it (mention the same in design document)
    3. Integrate a message broker in your microservice (Kafka, RabbitMQ or ActiveMQ) to implement CQRS pattern. Implement it for adding a profile (User-Story 1: Command Part) and searching the profile (User-Story-3: Query part)
    4. Use any one of the Creational Design patterns for composing the model object to be sent back as response on following endpoint:

/skill-tracker/api/v1/admin/{criteria}/{criteriaValue}

(Searches for profile based on provided criteria. Mention the patterns used in design document and also specify the reason for selecting the one)

* + 1. Identify and implement the best possible use of Java collections and stream APIs
    2. Optimize you REST endpoints to allow filtering, sorting, and pagination.
    3. Document REST endpoints with OpenAPI or Swagger
    4. Expose all rest Endpoints using a common API Gateway.
    5. If any inter microservice call is required, use feign client.
    6. Implement service discovery and circuit breaker pattern in microservice architecture.
    7. Implement at least 2 Security OWASP recommendations in your spring boot applications.
    8. Use Spring batch to load initial data to the DB

## Governance and Tooling:

* + - 1. Follow the practise of Creating Testable Component
      2. Configure your frontend application to implement End-To-End Testing using either Protractor or Cypress.
      3. Test suites must contain exception situation testing.
      4. Generate the Code Coverage report of the same.
      5. Implement logging in your end-to-end testing and report the same using either ELK stack.
      6. Perform unit and integration testing of your application and do proper CI/CD

## Code Quality/Optimizations

1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

# Frontend

Develop the frontend for User Story 3. (Implementation as follows)

1. Implement using either Angular or React.
2. Implement any one of the Gang of four Patterns to compose data using typescript before presenting the same on UI.
3. Implement at least one approach for UI performance consideration.
4. Identify and Implement client-side Optimization Techniques for Bootstrap.
5. Implement the prevention of XSS cross-site security threats for frontend application.
6. Perform unit and integration testing for the front end application

# Platform

## Cloud Specific Design

* + - 1. Update the messaging service (Kafka/RabbitMQ/ActiveMq/KubeMQ) with Cloud based service viz. AWS MQ.
      2. The backend service for the API should be implemented using any of the NoSQL service supported on AWS.
      3. As per the below specification create an elaborated architecture diagram of Cloud Deployment in cloud section of design document.

## Deployment

* + - 1. Use AWS CodeDeploy to implement all deployment activities
      2. Follow security best practices for CodePipeline resources
      3. Configure server-side encryption for artifacts stored in Amazon S3 for CodePipeline (for Frontend Deployment: refer to pointer 8.4)

## Microservice Deployment

* + - 1. Deploy microservice responsible for searching profile on separate instances of EC2
  1. Configure the microservice responsible for searching profile as follows:
     1. Configure EC2 instance to connect and add support of memchache using AWS ElastiCache
     2. Whenever a new profile is added, queue it to MemChache
     3. Consume the profile details from this Cache
     4. Ensure that Elastic Block Store (EBS) attached volumes are encrypted to meet security requirements.

1. Configure at least one microservice as serverless
2. Use AWS Cognito to secure the functionality of updating the profile
3. Configure the Amazon API Gateway to expose all rest endpoints exposed by different microservices on public internet.

## FrontEnd Deployment

Deploy the Frontend solution as follows:

Maintain the production build of Frontend application on S3 bucket

Configure an EC2 instance to access Frontend artifacts from S3 bucket and expose it as dynamic web application

Configure the S3 to cache the Javascript build files

Configure the Route 53 to register domain name to expose the Frontend solution.

Ensure that Privacy Protection feature is enabled for Amazon Route 53 domain.

## Management and Governance

* + - 1. Configure each EC2 instance to register all logs to CloudWatch
      2. Configure CloudWatch to filter API calls for updating the profile.
      3. For at least one of the microservice, configure the AutoScaling to scale the Amazon EC2 instance metrics with a 1-minute frequency. Test the same using either JMeter for local testing or use AWS SageMaker for the same

## Migration & Transfer

* + - 1. Use AWS DataSync to transfer the Frontend Deployment files(production build) from on-premise(your local machine) to S3

# Methodology

## Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
   1. Q/A
   2. New Ideas, New feature implementations and estimation.
   3. Any development related challenges
   4. Skill Gaps
   5. Any other pointers key to UI/UX and Middleware Development