# Internship Assignment: Maturity Matrix

## Problem Statement

In our organization, there are five distinct teams, each working on separate projects. Each team is responsible for specific features, and there is some overlap where certain team members can contribute to multiple features. To streamline communication and ensure uniformity across teams, we want to develop a web application that facilitates the enrollment of teams and the tracking of their progress on features and common practices.

Teams have unique sets of features, each accompanied by a fixed set of common questions. Additionally, there are overarching common practices shared across the entire organization, such as agile methodologies, code quality, and continuous integration/continuous deployment (CI/CD) pipelines. Each of these practices may have a variable number of questions.

The questions are standardized and cover various aspects like API integration, workflow, and customer behavior. Each question allows users to provide one of five responses: N/A, 0, 1, 2, or 3. A response of 1 indicates basic knowledge, 2 denotes hands-on experience, and 3 signifies expert-level proficiency.

To address this, we aim to create a Node.js-based backend application to manage teams, features, and common practices. The backend should support adding, updating, and retrieving information about teams, features, and processes. Additionally, we need a frontend application that allows team members to log in, select their team, and answer questions related to the features and common practices assigned to their team.

The frontend should dynamically load questions based on the selected team and feature, providing a smooth and intuitive user experience. Users should be able to submit their responses, and the system should store this information for later review.

The primary objectives of this project are to improve team collaboration, standardize practices across projects, and track the skill levels of team members in different areas. This will contribute to a more efficient and cohesive working environment within the organization.

## Tasks

Crete Front end and Back end

# Back End

## Setup Node.js Project

Initialize a Node.js project using npm init. Install necessary packages such as Express for building the server, and any other dependencies.

## Create Routes and Controllers

Set up routes for handling team, feature, and process-related requests. Create controllers to handle the business logic for each route.

## Data Models

Define data models for teams, features, processes, and questions. You can use an ORM like Mongoose for MongoDB or Sequelize for SQL databases.

## Database Integration

Connect your application to a database to store team, feature, and process information.

## API Endpoints

Create API endpoints for adding teams, features, and processes. Design endpoints for retrieving questions based on the selected team and feature.

## Middleware

Add middleware for handling user sessions and checking user permissions.

# Frontend

## Setup Frontend Project

Set up a frontend project using a framework like React, Angular, or Vue.js.

## User Authentication

Implement a login page where users can enter their credentials and select team

## Feature and Process Pages

Implement pages for features and processes that dynamically load questions based on the selected team and feature.

### Questionnaire Component

Design a reusable component for displaying questions and capturing user responses. Use radio buttons or a similar UI element for users to select their answers.

### Submission

Implement a submission mechanism to send user responses to the backend for storage.

### Integration

Connect Backend and Frontend:

Ensure that the frontend can communicate with the backend through API calls.

### Testing

Perform testing to validate that questions are loaded correctly based on team and feature selections. Verify that user responses are being stored in the database.

## Additional Considerations

### Security

Implement secure coding practices to protect against common web vulnerabilities. Validate and sanitize user inputs to prevent injection attacks.

### Scalability

Design your application to scale by considering potential increases in the number of teams and users.

### Documentation

Provide documentation for both the backend API and the frontend application to assist developers and users. User Interface (UI) and User Experience (UX):

Design a user-friendly interface to enhance the overall user experience.

This is a broad overview, and you may need to adjust the details based on your specific requirements and the technologies you choose for your stack.

# Submission Guidelines

Share the GitHub repository link for evaluation. Include a README file with instructions on how to run the application locally. Ensure clear comments and documentation in your code.

## Evaluation Criteria

- Code structure, readability, and adherence to coding standards.
- Regular commits to the GitHub repository.

## Deadline

Submit your project by 1 week. Regular commits and progress updates are encouraged.

Best of luck with your bilingual dictionary web application!

**Note:** This assignment aims to evaluate your skills in web development, design aesthetics, and attention to detail. Good luck!

© 2023 RecursiveZero, All rights reserved.