ISSUE TRACKER TASK (L3)

1.1 DESCRIPTION

Create a RESTful API for an issue tracking website (think JIRA/Kanban/Trello)

1.2 BUSINESS REQUIREMENTS

1.2.1 Users

- We have one type of users in the system
- Users can register on the website using their **email**, **full name**, and **password**
- Users can login to the website using email and password

1.2.2 Projects

- Projects are collections of **issues** worked on by a team of **participants**.
- Users can create projects.
- Each project has:
  - **Name**
  - **Key:** 1 word, 3 or 4 letters, all caps A-Z. E.g.: “TKWZ”. keys are unique in the system and cannot be changed.
  - **Owner** (the user who created the project).
- Project owners can delete any of their projects.

1.2.3 Project Participants

- Participants are users who can work on a project and its issues.
- Project owners are participants by default in their own projects and that cannot be changed.
- Project owners can add/remove **participants** to the project by using existing users’ emails.
- Project participants can list all other participants in a project

1.2.4 Issues

- Projects have multiple types of issues, for the initial requirements we’ll have 3: Story, Bug, and Task.
- We may add new types of issues in the future.
- All issues have the following properties:
  - **Id**: An auto-generated key consisting of the project key, a dash “-”, and an auto-generated sequential number, e.g.: “TKWZ-12”. The sequence for each project should start from 1. Ids cannot be changed.
  - **Title**: Required short description of the issue.
  - **Description**: Optional long description of the issue
  - **Reporter**: The user who created the issue, cannot be changed
  - **Assignee**: The user who is currently assigned to the issue, an issue starts its life as unassigned.
  - **Status**: Todo (Default), InProgress, Done
- A task has no addition properties
- Participants can create an issue of any type.
- Participants can delete an issue (regardless of who created it).
- Participants can update all properties of an issue besides the id.
- Participants can list all the issues in a project. They can filter them by assignee. The properties that should be included in the listing are:
  - Id
  - Type
  - Title
  - Assignee
  - Parent

1.3 Technical Requirements

1.3.1 Technologies

- Use ASP.Net Core 5.0

1.3.2 Data Persistence

- EF Core 5.0 with a relational database (SQL Server or PostgreSQL).
- Use code-first approach.
- Use the fluent API for entity configuration with EF Core, don't use data annotations.

1.3.3 Architecture

- Use clean architecture
- Use CQRS with MediatR

1.3.4 Testing

- At least one "happy scenario" test for each user action in the **projects** scope is required. Testing other scopes is optional

1.3.5 Documentation

- All endpoints should be documented with OpenAPI (swagger), with short description of the endpoint. The body of the request should be clearly documented with an example.
- Enable user login on swagger so that the authenticated endpoints could be tested.
- Write a README.md file with clear instructions (in English) on how to set up the development environment and run the project locally.
- Document the code with comments where necessary.

1.3.6 Version Control

- The project should be hosted on a **public** git repository.
- The commit history should show incremental changes with commits, not just one (or a handful) of commits.
- The latest version should be on the **master** branch.

1.4 Notes

- A lot of the edge cases are not mentioned in the requirements, it's expected from you to catch these edge cases and handle them (how you handle them will be left to you, unlike the real world).
- You're welcome to extend the requirements and add more features if you have the time, just make sure you do that after you implemented the listed requirements first. But adding more unit tests has a higher value for us than adding new features.
- Do not be afraid of over-engineering something, it's a coding challenge that's meant to showcase your abilities.
- Not all candidates have experience with all the requirements above and that's intentional, we'd like to see how quickly you can learn new concepts.

1.5 Bonus Work

- Add capability to have properties that are specific to one type of issues but not the other:
  - A Story has the following extra properties:
    - **Story Points**: optional positive floating point number
  - A Bug has the following extra properties:
    - **Steps to Replicate**: optional text value
- Add a fourth issue type (sub-tasks):
- Subtasks have parents, which are any other type of issue
- Use Identity Server 4
- Use Domain-Driven Design for the issues context. If you do so, add unit tests for the domain layer.