Job Candidate Hub API

1. Introduction and Overview

- **Project Overview**: This project is created with the purpose of making a WebAPI interface to create or update candidate details as necessary. It uses only one endpoint, i.e. POST to complete both update and create requests.
- **Technologies Used**: The whole project is created on .NET v8 with implementation of Entity Framework Core, Xunit, and for the purpose of testing, SQL Server database for storage. For testing, InMemory is used.
- **Goals**: The objective of this project is to make sure it works with multiple database connections, performance-oriented, and maintaining quality code.

2. Execution

- As requirement, the project should work out of the box with no package dependencies as .NET Core is used.

3. Architecture and Design

- **Architecture Overview**: It follows a typical ASP.NET Core Web API structure leveraging ASP.NET Core MVC for handling requests, EF Core for data access and dependency injection for managing dependencies. It follows RESTful principles of API design and uses asynchronous programming for performance.
- **Database Design**: The database is created on code-first basis with implementation of migration.

4. Code Documentation

- **Code Structure**: The code is a typical ASP.NET Core controller based for handling HTTP requests and implements a single API endpoint. Model class represents the database context used to interact with the underlying database and provides abstraction for querying and saving data.
- **API Used**: Following endpoint is used.

  (“…./api/Candidate/CreateOrUpdate”)

- **Database Access**: EF Core abstracts the database access through DbContext and DbSet. Dependency Injection manages DbContext Lifecycle and controller does the create/update operation using the injected DbContext.
- **Testing Strategy**: xUnit test covers both individual and interactive units. In-memory database is used for isolated testing with false data. Various validation methods are used in test controller to simulate Empty fields, bad requests, null arguments and so on.

7. Feedback and Reviews

- **Feedback Mechanism**: Some ways the properties defined on Model class can be rather validated on controller class such as trailing whitespace characters, duplicate personnel entries with same details but different Emails and so on.
- **Review Process**: Some considerations that can be taken in the testing process includes using multiple databases to simulate performance.

8. Project Metrics

- **Total Time Spent**: The project took nearly 4 hours to complete. The initial code was written in an hour, excluding xUnit test code. The testing of API endpoint was done for an hour. XUnit test code was written in an hour. Testing the whole project took the remainder of the time.

9. Appendix

- **References**: Some online references were taken to develop the project. Here are some examples.
  https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?
  view=aspnetcore-8.0&tabs=visual-studio
  https://dotnettutorials.net/course/asp-net-web-api/