System Requirements Specification Index

For

Interview Tracker Application (MS SQL)

Version 4.0

IIHT Pvt. Ltd.

IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

INTERVIEW TRACKER APPLICATION SYSTEM

System Requirements Specification

1. BUSINESS-REQUIREMENT:

1.1 PROBLEM STATEMENT:

The purpose of this application is to allow the official staff to register, Update, delete, show all interviewees by using **user API**. On the other hand, Dashboard **API** allows you to view all Interviews, search Interview by name or interviewer name, Add, Delete, Update interview and Using **Interview API** you can Add New Interview and count total number of interviews.

1.2 FOLLOWING IS THE REQUIREMENT SPECIFICATION:

	Interview Tracker Application
User Controller	1. Register new User
	2. Allows to edit/Delete/GetAll User
Interview Controller	1. Show Total number of interviews.
	2. Add a new interview.
Dashboard Controller	1. Show All Interview
	2. Update/Edit interview.
	3. Remove interview.
	4. Find an interview by Interview/Interviewer Name.

2. RESOURCES AVAILABLE:

2.1 PACKAGE: INTERVIEWTRACKER

Names	Resource	Remarks	Status
Package Structure			
controller	User, Dashboard, Interview Controller	These controllers handle all application Function, update/Edit show information and login existing user.	Partially Implemented
Startup.cs	Startup CS file	Contain all Services settings and Db Configuration.	Already Implemented
Properties	launchSettings.json file	All URL Setting for API	Already Implemented
appsettings.json		Contains connection string forDB	Already Implemented

2.2 PACKAGE: INTERVIEWTRACKER.BUSINESSLAYER

Names	Resource	Remarks	Status
Package Structure			
Interface	IInterviewTracker, IUserInterviewTracker Services interface	Inside all these cs files contains all business logic functions.	Already Implemented
Service	InterviewTracker, UserInterviewTracker Servicesclass file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Partially Implemented
Repository	IInterviewTracker, InterviewTracker, IUserInterviewTracker, UserInterviewTracker Repository CS file and interface.	All these interfaces and class files contain all CRUD operation code for Db.	Partially Implemented
ViewModels	AddInterview, EditInterview, Interview, Register, UserEdit ViewModel Class file	Contain all view Domain entities for show and bind data.	Already Implemented

2.3 Package: InterviewTracker.dataLayer

Names	Resource	Remarks	Status
Package Structure			
DataLayer	InterviewTrackerDb Context cs file	All database setting class	Already Implemented

2.4 Package: InterviewTracker.Entities

Names	Resource	Remarks	Status
Package Structure			
Entities and Enum	ApplicationUser, Interview, InterviewStatus, Status, TechnicalInterviewStatus, UserType CS and Enum file	All Entities/Domain attribute	Already Implemented

2.5 Package: InterviewTracker.Tests

Note: - Under the InterviewTrackerTests.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

3. REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

3.1 UserController

URL Exposed	Purpose
/users Http Method: GET Parameters: - Return: <IEnumerable<ApplicationUser>>	Fetch all User
/user Http Method: POST Parameter 1: RegisterViewModel model Return: HTTP Response StatusCode	Register new User
/user Http Method: PUT Parameter 1: UserEditViewModel model Return: HTTP Response StatusCode	Update/Edit User
/user/{UserId} Http Method: DELETE Parameter 1: Int (UserId) Return: HTTP Response StatusCode	Delete User based on id

3.2 InterviewController

URL Exposed	Purpose
/interview/countall Http Method: GET Parameter 1: - Return: <IEnumerable<Interview>	Fetch total number of interview

/interview	Add new Interview

Http Method	POST
Parameter 1	AddInterviewViewModel model
Return	HTTP Response StatusCode

3.3 DashboardController

URL Exposed	Purpose
/dashboard/allinterviews	Fetch all Interview

Http Method	GET
Parameter 1	-
Return	<IEnumerable<Interview>>

/interview/{interviewId}	Delete interview

Http Method	DELETE
Parameter 1	Int(interviewId)
Return	HTTP Response StatusCode

/interview	Update Existing Interview

Http Method	PUT
Parameter 1	EditInterviewViewModel model
Return	HTTP Response StatusCode

/interview/byname/{name}	Find interview by interviewer and Interview name

Http Method	GET
Parameter 1	String(name)
Return	<IEnumerable<Interviewt>>

4. Business Validations

4.1Common Constraints:

- Following validation constraints are to be added
    a. All the value for user details: must not be null.
    b. All the Interview value: must not be null
    c. Check Enum for all Fixed data under Entities Project
    d. On Update or delete: must not be null of Id and Model
- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- Must not go and touch the test resources, as they will be used for Auto-Evaluation.

5. Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**

2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

3. **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

4. **(InterviewTracker/sqlcmd -S localhost -U sa -P pass@word1)**
    - **To create database from terminal -**
        1. **Create Database InterviewTracker_Db**
        2. **Go**

5. **Steps to Apply Migration(Code first approach):**
    - **Press Ctrl+C to get back to command prompt**
    - **Run following command to apply migration- (InterviewTracker/dotnet-ef database update)**

6. **To check whether migrations are applied from terminal: (InterviewTracker/sqlcmd -S localhost -U sa -P pass@word1)**

```
1> Use InterviewTracker_Db
2> Go
1> Select * From _EFMigrationsHistory
2> Go
```

7. **To build your project use command: (InterviewTracker /dotnet build)**

8. **To launch your application, Run the following command to run the application: (InterviewTracker/dotnet run)**

9. **This editor Auto Save the code.**

10. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**

11. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

12. **To run the test cases in CMD, Run the following command to test the application: (You can run this command multiple times to identify the test case status, and refactor code to make maximum test cases passed before final submission) (InterviewTracker/dotnet test --logger "console;verbosity=detailed").**

13. **If you want to exit (logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B - command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

14. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**

15. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**
