System Requirements

Specification

Index

For

Appointment Scheduling Application

Version 1.0

Appointment Scheduling Application
System Requirements Specification

- PROJECT ABSTRACT

The **Appointment Scheduling Application** is a ASP.NET Core Web API (.Net 6.0) with MS SQL Server database connectivity. It enables users/patients to schedule appointment with doctor.

Following is the requirement specifications:

	Appointment Scheduling Application
Modules	
1	Appointment Scheduling
Appointment Module Functionalities	
1	Schedule/Create Appointment
2	Update the existing appointment details
3	Get the appointment by Id
4	Get the appointment by Range (start datetime and end datetime)
5	Get the appointment by PatientId
6	Delete an appointment

- ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

  - EVENT CONSTRAINTS

    - When fetching an Appointment by ID, if the Appointment ID does not exist, the operation should throw a custom exception.
    - When updating an Appointment, if the Appointment ID does not exist, the

operation should throw a custom exception.

- When removing an Appointment, if the Appointment ID does not exist, the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

• BUSINESS VALIDATIONS

- AppointmentID (Int) Key, Not Null (system generated)
- PatientID(int), not null and must not be more than 50 characters in length.
- PatientName (String), not null and must not be more than 50 characters in length.
- PatientPhone (String), not null and must not be more than 10 characters in length.
- StartDate (DateTime), not null.
- EndDate (DateTime), not null.

• REST ENDPOINTS

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

• EVENTCONTROLLER

<table>
<tr><th colspan="2">URL
Exposed</th><th>Purpose</th></tr>
<tr><td colspan="2">1. /api/Appointments/GetAppointments</td><td rowspan="4">Fetches all the Appointments</td></tr>
<tr><td>Http Method</td><td>GET</td></tr>
<tr><td>Parameter</td><td>-</td></tr>
<tr><td>Return</td><td><IEnumerable<Appointment>></td></tr>
<tr><td colspan="2">2. /api/Appointments/AddAppointment</td><td rowspan="2">Add a new</td></tr>
<tr><td>Http Method</td><td>POST</td></tr>
</table>

Parameter 1	Appointment	Appointment
Return	Appointment	
3. /api/Appointments/DeleteAppointment		Delete Appointment with given Apppintment id
Http Method	DELETE	
Parameter 1	Int (id)	
Return	-	
4./api/Appointments/GetAppointment		Fetches the Appointment with the given id
Http Method	GET	
Parameter 1	Int (id)	
Return	Appointment	
5. /api/Appointments/UpateAppointment		Updates existing Appointment
Http Method	PUT	
Parameter 1	Int (id)	
Parameter 2	Appointment	
Return	Appointment	

6./api/Appointments/GetAppointmentByPatientId				Fetches the Appointment with the given Patient Id
	Http Method	GET		
	Parameter 1	Int (patientid)		
	Return	IEnumerable<Appointment>		

7. /api/Appointments/GetAppointmentByRange				Fetches Appointments by startdate and enddate
	Http Method	GET		
	Parameter 1	DateTime (startdate)		
	Parameter 2	DateTime (enddate)		
	Return	IEnumerable<Appointment>		

5. Template Code Structure

5.1 Package: AppointScheduling

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	Appointments Controller	Controller class to expose all rest-endpoints for auction related activities.	Already implemented
appsettings.json	appsettings.json file	Contain all Services settings and SQL server Configuration.	Already Implemented

Interface	IAppointmentService, interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	AppointmentService CS file file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Already Implemented
Repository	IAppointmentRepository AppointmentRepository CS file and interface.	All these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Already Implemented
Models	Appointment cs file	All Entities/Domain attribute are used for pass the data in controller.	Already Implemented

5.2 Package: AppointmentScheduling.Tests

Resources

The AppointmentScheduling.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

6. Execution Steps to Follow

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

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

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

- **To create database from terminal -**
    - **Create Database AppointmentSchedulingDB**
    - **Go**

- **To build your project use command:**
  (AppointmentScheduling /dotnet build)

- **To launch your application, Run the following command to run the application:**
  (AppointmentScheduling /dotnet run)

- **This editor Auto Saves the code.**

- **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.**

- **To test web-based applications on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

  Note: The application will not run in the local browser

- **To run the test cases in CMD, Run the following command to test the application: (AppointmentScheduling.Tests/dotnet test --logger "console;verbosity=detailed") (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)**

- **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.**

- **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.**

- **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.**