System Requirements Specification Index

For

Dating Application

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

DATING APPLICATION SYSTEM

System Requirements Specification

1.Business-Requirement:

1.1 Problem Statement:

Dating Application is .Net Core 3.1 RESTful application with Ms SQL Server, where it allows Users to find matches, likes and dislikes the profiles based on the interests of the other users.

1.2 Following is the requirement specification:

	Dating Application
Modules	
1	User
2	Interests
3	Match
4	Like
5	Dislike
User Module Functionalities	
1	Can register Itself
2	Can Update Itself
3	Can Delete Itself
4	Can get the User by Id
5	Can Get all the Users
Interests Module Functionalities	
1	User can create Interests
2	User can update Interests
3	User can delete the Interests
4	User can get the interests-by-Interests Id
5	User can get the interests by User id

Match Module Functionalities	
1	User can get all the matches
2	User can create a match
3	User can get Potential matches based on Age, Gender, City and Country
Likes Module Functionalities	
1	User can get all the Likes
2	User add a like
Dislikes Module Functionalities	
1	User can get all the dislikes
2	User can add a dislike

2.Assumptions, Dependencies, Risks / Constraints

2.1 User Constraints:

- While deleting the user, if the user id does not exist then the operation should throw a custom exception.
- While getting the user by id, if user id does not exist then the operation should throw a custom exception.

2.2 Interests Constraints

- While deleting the interests by user, if interest id does not exist then operation should throw custom exception.
- While getting the interests by user, if interest does not exist then operation should throw custom exception.

2.3 Match Constraints

- While matching the profile, if the match id does not exist then the operation should throw a custom exception.

2.4 Like Constraints

- While getting all the likes by user, if user id does not exist then operation should throw custom exception.

2.5 Dislike Constraints

- While getting all the dislikes by user, if user id does not exist then operation should throw custom exception

2.6 Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in model classes only.
- 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.**

3.Business Validations

3.1 User Entity:

- User Id (long) is not null, Key Attribute.
- User age (int) is not null, min age is 18 and max age is 60.
- User name (string) is not null, min 3 and max 100 characters.
- User email (string) is not null, min 3 and max 100 characters and in proper email format.
- User Phone Number(string) is not null, min 10 and max 10 digits.
- User gender (string) is not null
- User city (string) is not null
- User country (string) is not null
- IsDeleted (bool) is not null

3.2 Interest Entity:

- Interests Id (long) is not null, Key attribute
- Interests interestedIn (string) is not null, min 3 and max 100 characters
- Interests notInterestedIn (string) is not null, min 3 and max 100 characters
- Interests hobbies (string) are a List and is not null
- Interests profileUrl (string) is not null
- Interests about (string) is not null, min 3 and max 100 characters
- IsDeleted(bool) is not null

3.3 Match Entity:

- Match Id (long) is not null, Key attribute
- User Id (long) is not null, Key Attribute.
- IsDeleted (bool) is not null

3.4 Like Entity:

- Like Id (long) is not null, Key attribute
- User Id (long) is not null, Key Attribute.
- IsDeleted (bool) is not null

3.5 Dislike Entity:

- Dislike Id (long) is not null, Key attribute
- User Id (long) is not null, Key Attribute.
- IsDeleted (bool) is not null

4. Rest Endpoints

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

4.1 UserRestController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/users<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>UserViewModel model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Register a User</td></tr>
<tr><td>/users<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>UserViewModel model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Update a User</td></tr>
<tr><td>/users<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<User>></td></tr></table></td><td>Fetches the list of all registered Users</td></tr>
<tr><td>/users/{userId}<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long (userId)</td></tr><tr><td>Return</td><td><User></td></tr></table></td><td>Fetches the details of a user</td></tr>
<tr><td>/users /{userId}<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Long (userId)</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Delete a User</td></tr>
</table>

4.2 InterestsRestController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/interests
<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>Interests interests</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>User adds an Interest</td></tr>
<tr><td>/interests
<table><tr><td>Http Method</td><td>PUT</td></tr><tr><td>Parameter 1</td><td>InterestViewModel model</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>User updates an interest</td></tr>
<tr><td>/interests/{interestId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long (interestId)</td></tr><tr><td>Return</td><td><Interests></td></tr></table></td><td>Fetches details of Interests based on interestId</td></tr>
<tr><td>/interests/by-user-id/{userId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long (userId)</td></tr><tr><td>Return</td><td><IEnumerable<Interests>></td></tr></table></td><td>Fetches details of all interests based on the userId</td></tr>
<tr><td>/interests/{interestId}
<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>Long (interestId)</td></tr><tr><td>Return</td><td>HTTP Response StatusCode</td></tr></table></td><td>Deletes the interest based on interestId</td></tr>
</table>

4.3 MatchRestController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/match/{userId}
<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>Long(userId),User user</td></tr><tr><td>Return</td><td><IEnumerable<Match>></td></tr></table></td><td>Fetches all matches based on userId or matchedUserId</td></tr>
</table>

/match/{userId} Http Method: POST Parameter 1: Long(userId) Parameter 2: User user Return: <IEnumerable<Match>>	Fetches all the users based on the potential matches like based on Age, Gender, City or Country

4.4 LikesrestController

URL Exposed	Purpose
/likes/{userId} Http Method: GET Parameter 1: Long (userId) Return: <IEnumerable<Like>>	Fetches list of likes based on userId
/likes Http Method: POST Parameter 1: DislikeViewModel model Return: HTTP Response StatusCode	Adds a like to the user profile

4.5 DislikesrestController

URL Exposed	Purpose
/dislikes/{userId} Http Method: GET Parameter 1: Long (userId) Return: <IEnumerable<Dislike>>	Fetches list of dislikes based on userId
/dislikes Http Method: POST Parameter 1: DislikeViewModel model Return: HTTP Response StatusCode	Adds a dislike to the user profile

5. Template Code Structure

5.1 Package: DatingApplication

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	User Controller Interests Controller Match Controller Like Controller Dislike Controller	Controller class to expose all rest-endpoints for auction related activities.	Partially Implemented
Startup.cs	Startup CS file	Contain all Services settings and MS SQL Configuration.	Already Implemented
Properties	launchSettings.json file	All URL Setting for API	Already Implemented
	appsettings.json file	Contain connection string for database	Already Implemented

5.2 Package: DatingApplication.BusinessLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
Interface	IUserServices interface IInterestServices interface IMatchServices interface ILikeServices interface IDislikeServices interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	User Services CS file Interest Services CS file Match Services CS file Like Services CS file DislikeServices CS file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Partially Implemented
Repository	IUserRepository UserRepository IInterestRepository InterestRepository IMatchRepository	All these interfaces and class files contain all CRUD operation code for the database.	

	MatchRepository ILikeRepository LikeRepository IDislikeRepository DislikeRepository CS file and interface.	Need to provide implementation for service related functionalities	Partially Implemented
ViewModels	UserViewModel, InterestViewModel, MatchViewModel, LikeViewModel, DislikeViewModel	Contain all view Domain entities for show and bind data. All the business validations must be implemented.	Partially Implemented

5.3 Package: DatingApplication.DataLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
DataLayer	DatingAppDbContext cs file	All database Connection and collection setting class	Already Implemented

5.4 Package: DatingApplication.Entities

Resources

Names	Resource	Remarks	Status
Package Structure			
Entities	User, Interests,Match,Like,Dislike CS file	All Entities/Domain attribute are used for pass the data in controller Annotate this class with proper annotation to declare it as an entity class with **Id** as primary key.	Partially Implemented

		Generate the **Id** using the **IDENTITY** strategy	

5.5 Package: DatingApplication.Tests

Resources

The DatingApplication.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

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. **To connect SQL server from terminal:**
   (DatingApplication /sqlcmd -S localhost -U sa -P pass@word1)
   - **To create database from terminal -**

     ```
     1>  Create Database DatingApp_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-**
     (DatingApplication /dotnet-ef database update)

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

   ```
   1> Use DatingApp_Db
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go
   ```

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

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

9. **This editor Auto Saves 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 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
12. **To run the test cases in CMD, Run the following command to test the application:**
    (DatingApplication /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)

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.**