System Requirements Specification Index

For

Django Rest-API
Dating-App

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 Project Abstract

DatingApp is Django Rest API application with SQLite Database, where it allows users to find matches, likes and dislikes the profiles based on the interests of the other users.

Following is the requirement specifications:

	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 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
Likes Module Functionalities	
1	User can get all the Likes by user id
2	User add a like
Dislikes Module Functionalities	
1	User can get all the dislikes by user id
2	User can add a dislike

2 Assumptions, Dependencies, Risks / Constraints

A. 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.

B. 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.

c. Like Constraints

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

d. DislikeConstraints

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

3 Business Validations

- User name is not null, max 100 characters.
- User age is not null, max age is 60.
- User email is not null, max 100 characters and in proper email format.
- User Phone Number is not null, max 10 digits.
- User gender is not null
- User city is not null
- User country is not null
- Interests interestedIn is not null, max 100 characters
- Interests notInterestedIn is not null max 100 characters
- Interests hobbies are a List and is not null
- Interests profileUrl is not null
- Interests about is not null, max 200 characters

4 Rest Endpoints

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

Class Name	Method Name	Purpose Of Method
UserCRUDView	get(self,request,pk=None,format=None)	To get the User by Id and To Get all the Users
	post(self, request,format=None)	To register user Itself
	patch(self,request,pk,format=None)	To Update user Itself
	delete(self,request,pk,format=None)	To Delete user Itself
InterestsCRUDView	get(self,request,pk=None,format=None)	To get the interests by Id
	post(self, request,format=None)	To create Interests
	patch(self,request,pk,format=None)	To update Interests
	delete(self,request,pk,format=None)	To delete the Interests
GetInterestsByUserIdView	get(self,request,pk=None,format=None)	To get interests by user id.
MatchView	get(self,request,pk=None,format=None)	To get all the matches
	post(self, request,format=None)	To create a match
LikesView	get(self,request,pk=None,format=None)	To get all the Likes by user id
	post(self, request,format=None)	To add a like
DisLikesView	get(self,request,pk=None,format=None)	To get all the dislikes by user id
	post(self, request,format=None)	To add a dislike

5 Template Code Structure

Resources (Views)

Class	Description	Status
UserCRUDView	A class for the get, post, patch, delete functionalities on **UserModel** model.	To be implemented

InterestsCRUDView	A class for the get, post, patch, delete functionalities on **InterestsModel** model**.**	To be implemented
GetInterestsByUserIdView	A class for the get functionality on **InterestsModel** model**.**	To be implemented
MatchView	A class for the get, post functionalities on **MatchModel** model**.**	To be implemented
LikesView	A class for the get, post functionalities on **LikesModel** model**.**	To be implemented
DisLikesView	A class for the get, post functionalities on **DisLikesModel** model**.**	To be implemented

Resources (Models)

Class	Description	Status
UserModel	o A model class for User. o It will map to the User Model table.	Already implemented.
InterestsModel	o A model class for Interests. o It will map to the Interests Model table.	Already implemented.
MatchModel	o A model class for Match. o It will map to the Match Model table.	Already implemented.

LikesModel	o A model class for Likes. o It will map to the Likes Model table.	Already implemented.
DisLikesModel	o A model class for DisLikes. o It will map to the DisLikes Model table.	Already implemented.

Resources (Serializers)

Class	Description	Status
UserSerializer	A serializer for UserModel	Already implemented
InterestsSerializer	A serializer for InterestsModel	Already implemented
MatchSerializer	A serializer for MatchModel	Already implemented
LikesSerializer	A serializer for LikesModel	Already implemented
DisLikesSerializer	A serializer for DisLikesModel	Already implemented

Resources (Exceptions)

Class	Description	Status
UserIdNotAvailable	Custom exception **UserIdNotAvailable** is raised in case specified user id is not available.	Already implemented.
InterestsIdNotAvailable	Custom exception **InterestsIdNotAvailable** is raised in case specified Interests Id is not available.	Already implemented.

6 CONSIDERATIONS

A. There is only one Role of possible value
   1. User
B. The user can perform the following 4 possible actions

Interests
Profile Match
Profile Like
Profile Dislike

7 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
   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. The editor Auto Saves the code.
4. If you want to exit (logout) and to 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.
5. 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.
6. 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.
7. 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.

8. Install ‘djangorestframework’ module before running the code. For this use the following command.

   pip install djangorestframework

9. Use the following command to run the server

   python3 manage.py runserver

10. Mandatory: Before final submission run the following commands to execute testcases

    python3 manage.py test dating.test.test_functional

    python3 manage.py test dating.test.test_exceptional

    python3 manage.py test dating.test.test_boundary

11. To test rest endpoints

    Click on ‘Thunder Client’ or use Ctrl+Shift+R ->Click on ‘New Request’ (at left side of IDE)

12. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.

13. 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.

-----*------