System Requirements Specification

Index

For

Social Networking Application

Version 1.0

TABLE OF CONTENTS

SOCIAL NETWORKING APPLICATION

System Requirements Specification

You need to consume APIs exposed by Backend application in Angular to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Social Networking Application** is a FullStack Application designed to connect people and facilitate social interactions. It encompasses a backend developed using Spring Boot with a MySQL database and a frontend implemented using Angular. The application provides a comprehensive platform to browse and manage users.

Following is the requirement specifications:

	Social Networking Application
Modules	
1	User
User Module Functionalities	
1	Create a User
2	Update the existing User details
3	Get the User by Id
4	Get the User by username
5	Get the User by email
6	Get all User
7	Delete an User

2 Assumptions, Dependencies, Risks / Constraints

2.1 USER CONSTRAINTS:

- When fetching a user by ID, if the user ID does not exist, the operation should throw a custom exception.
- When updating a user, if the user ID does not exist, the operation should throw a custom exception.
- When removing a user, if the user ID does not exist, the operation should throw a custom exception.
- When fetching a user by username, if the user name does not exist, the operation should throw a custom exception.
- When fetching a user by user email, if the user email does not exist, the operation should throw a custom exception.
- When creating a new user, if the user name already exists, 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 business validations must be implemented in dto 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

- Name is not null, min 3 and max 50 characters.
- User Name is not null, min 3 and max 50 characters.
- Email is not null, should be a valid email address, min 0 and max 100 characters.
- Password is not null, min 8 and max 20 characters.
- Profile Picture is optional.
- Bio is optional, min 0 and max 200 characters.
- Location is optional, min 0 and max 100 characters.
- Date Of Birth is optional, have ‘yyyy-mm-dd’ format

- Gender is not null, and select only from MALE, FEMALE, OTHER.
- Interests is optional, min 0 and 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

4.1 USERCONTROLLER

URL Exposed	Purpose
1. /users Http Method: GET Parameter: - Return: List<User>	Fetches all the users
2. /users Http Method: POST Parameter 1: User Return: User	Add a new user
3. /users/{id} Http Method: DELETE Parameter 1: Long (id) Return: -	Delete user with given user id
4. /users/{id} Http Method: GET Parameter 1: Long (id) Return: User	Fetches the user with the given id
5. /users/{id} Http Method: PUT Parameter 1: Long (id) Parameter 2: User Return: User	Updates existing product
6. /users/username/{username} Http Method: GET Parameter 1: String (username) Return: User	Fetches the user with the given username
7. /users/email/{email} Http Method: GET Parameter 1: String (email) Return: User	Fetches the user with the given email

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.SOCIALAPP

Resources

SocialAppApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.SOCIALAPP.REPOSITORY

Resources

Class/Interface	Description	Status
UserRepository (interface)	o Repository interface exposing CRUD functionality for User Entity. o You can go ahead and add any custom methods as per requirements.	Partially implemented.

5.3 PACKAGE: COM.SOCIALAPP.SERVICE

Resources

Class/Interface	Description	Status
UserService (interface)	● Interface to expose method signatures for product related functionality. ● Do not modify, add or delete any method.	Already implemented.

UserServiceImpl (class)	• Implements UserService. • Contains template method implementation. • Need to provide implementation for user related functionalities. • Do not modify, add or delete any method signature	To be implemented.

5.4 PACKAGE: COM.SOCIALAPP.CONTROLLER

Resources

Class/Interface	Description	Status
UserController (Class)	1. Controller class to expose all rest-endpoints for user related activities. 2. May also contain local exception handler methods	To be implemented

5.5 PACKAGE: COM.SOCIALAPP.DTO

Resources

Class/Interface	Description	Status
UserDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

5.6 PACKAGE: COM.SOCIALAPP.ENTITY

Resources

Class/Interface	Description	Status
User (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with userId as primary key. • Map this class with a user table. • Generate the userId using the IDENTITY strategy	Partially implemented.
Gender (Enum)	• This enum is already implemented.	Already implemented.

5.7 PACKAGE: COM.SOCIALAPP.EXCEPTION

Resources

Class/Interface	Description	Status
GlobalExceptionHandler (Class)	• Custom Exception to be thrown when trying to fetch or delete the user info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Partially implemented.
ResourceNotFoundException (Class)	• Custom Exception to be thrown when trying to fetch or delete the user info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Partially implemented.
UserNameAlreadyExistsException (Class)	• This exception needs to be thrown when trying to create user with already registered username	Partially implemented.

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following possible action

User

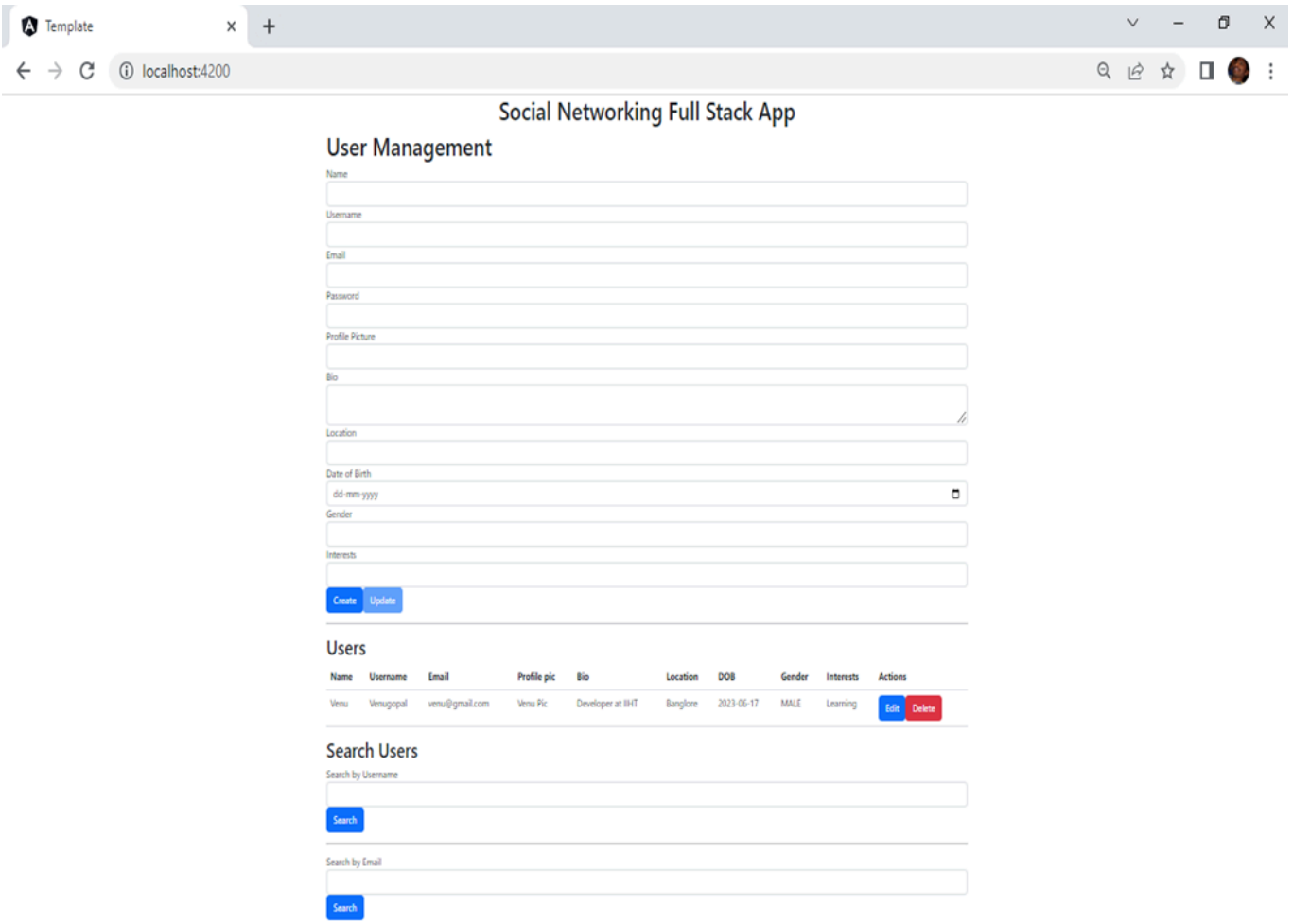
1 PROBLEM STATEMENT

Social Networking is SPA (Single Page Application), it allows to add user, update user, delete user, get single user, get all users, search user by user name, and search user by email.

2 PROPOSED SOCIAL NETWORKING MANAGEMENT WIREFRAME

UI needs improvisation and modification as per given use case and to make test cases passed.

2.1 WELCOME PAGE



3 BUSINESS-REQUIREMENT:

As an application developer, develop the Social Networking App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the Home page as the default page.
US_01	Home Page	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. As a user I should be able to furnish the following details at the time of creating a user. 1.1 Name 1.2 User Name 1.3 E-mail 1.4 Password 1.5 Profile picture 1.6 Bio 1.7 Location 1.8 Date of birth 1.9 Gender 1.10 Interests 2. Update button should be disabled by default, and should be enabled when you click on the Edit button. 3. Name field min length is 3 and max length 50. 4. User name field min length is 3 and max length 20. 5. E-mail should be valid email. 6. Password min length is 8 and max length 20. 7. Bio max length is 200. 8. Location max length is 100.

		9. Name, username, email, password fields are mandatory. If any constraint is not satisfied, a validation message must be shown.

4 EXECUTION STEPS TO FOLLOW FOR BACKEND

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. **cd into your backend project folder**
4. **To build your project use command:**
   mvn clean package -Dmaven.test.skip
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**
   java -jar <your application jar file name>
6. **This editor Auto Saves the code.**
7. **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.**
8. **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.**
9. **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.**
10. **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.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

12. To login to mysql instance: Open new terminal and use following command:
    a. sudo systemctl enable mysql
    b. sudo systemctl start mysql

        NOTE: After typing any of the above commands you might encounter any warnings.

        >> Please note that this warning is expected and can be disregarded. Proceed to the next step.

    c. mysql -u root -p

        The last command will ask for password which is 'pass@word1'

13. Mandatory: Before final submission run the following command:

        mvn test

14. 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.

8 EXECUTION STEPS TO FOLLOW FOR FRONTEND

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. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min**
   b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min**
   c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. **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.**