System Structure

Specification Index

For

Online Dating

Application

Version 1.0

Prepared by

Navin Gupta

IIHT Pvt. Ltd.

IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,

Bangalore, Karnataka – 560001, India

interest@iiht.com

**ONLINE DATING APPLICATION**

**System Requirements Specification**

**Project Abstract**

**Online Dating** Application is simple Spring boot application with MongoDb/MySQL, where it allows any unregistered users to manage the stocks at any stock exchanges like create, view, modify and delete stock price details and company details.

Every **User** should have the properties like user Id, first name, last name, date of birth, gender, address, login name and password.

Every **Profile** should have the properties like user id, profile Image, contact number, email, qualification, hobbies, food habits and description about the user.

Every **Dating Specifications** should have the properties like user id, receiver Id, dating date, dating time, location, request status, and dating Request.

The relationship between these two entities is that every user would have personal profile and there will be dating aspect only with the registered users.

**Registered Users can perform the follow actions**

* Allows to add a new user details and their respective profile details
* Allows to delete an existing user or existing profile
* Allows to search the user or profile on the basis of user Id
* Allows to display all user information or all dating specifications
* Allows to propose dating among the register users

**Software Requirements** SpringBoot, RESTful Web Services, MySQL or MongoDb, Postman.

**Scope of the project** Create a SpringBoot Server-side application

**Assumptions, Dependencies, Risks / Constraints**

* While adding a user details, if user Id is already existing, it should throw a custom exception
* While deleting the user, ensure that user Id already exists, if not, the operation should throw a custom exception
* 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
* All the business validations must be implemented on DTO object only
* All the database operations must be implemented on model / entity object only
* The conversion from model to DTO can be done in the service layer or at separate package
* The methods required for the conversion is provided in the DatingUtility class in ‘utils’ package in the project.
* The User Details and Profile Details are connected through a field user Id – applying integrity constraint

**URL’s Exposed in the controller**

**UserController**

|  |  |
| --- | --- |
| **URL Exposed** | **Purpose** |
| /user/login | Login to the application |
| /user/addUser | Add a new User Details |
| /user/deleteUser/{userId} | Delete User with given User Id |
| /user/getUserById/{userId} | Fetches the User Details with the given User Id |
| /user/listAllUsers | Fetches all the Users Details |

**UserProfileController**

|  |  |
| --- | --- |
| **URL Exposed** | **Purpose** |
| /profile/addProfile | Add a new Profile Details |
| /profile/deleteProfile/{userId} | Delete Profile with given User Id |
| /profile/getProfileById/{userId} | Fetches the Profile with the given User Id |
| /profile/listAllProfiles | Fetches all the Profile Information |

**DatingController**

|  |  |
| --- | --- |
| **URL Exposed** | **Purpose** |
| /dating/addDatingSpecs | Add a new Dating Specification |
| /dating/deleteDatingSpecs/{userId} | Delete DatingSpecs with given User Id |
| /dating/getDatingSpecsById/{userId} | Fetches the DatingSpecs with the given User Id |
| /dating/listAllDatingSpecs | Fetches all the DatingSpecs Information |

**Business Validations**

* User Id must be not null, unique and type should be Long
* First Name is not null, min 5 and max 100 characters.
* Last Name is not null, min 5 and max 100 characters.
* Date of Birth is not null and never exceed current date
* Gender is not null, min 4 and max 6 characters.
* address is not null, min 5 and max 255 characters.
* Login Name is not null, min 5 and max 255 characters.
* Password is not null, min 5 and max 10 characters.
* Contact Number is not null, should be of type Long
* Email, Qualification, Hobbies, FoodHabits, AboutMe is not null, min 5 and max 255 characters.
* Dating Date is not null and never exceed current date
* Dating time is not null and never exceed current time
* Location, Request Status and Dating Request is not null, min 5 and max 255 characters.

**Resources Available in the platform**

|  |  |  |
| --- | --- | --- |
| **Spring Boot Rest Package Structure** | | |
| **NAME** | **RESOURCE** | **REMARKS** |
| **controller** | UserController,  UserProfileController,  DatingController | * A UserController class to handle all requests regarding User information * A UserProfileController class to handle all requests regarding Profile Details * A DatingController class to handle all requests regarding Dating Specifications |
| **dto** | UserDTO,  ProfileDTO,  DatingSpecsDTO,  InvalidUserExceptionResponse,  InvalidProfileExceptionResponse,  InvalidDatingExceptionResponse | * UserDTO: DTO class,   InvalidUserExceptionResponse: For returning response in case of exception   * ProfileDTO: DTO class,   InvalidProfileExceptionResponse: For returning response in case of exception   * DatingSpecsDTO: DTO class,   InvalidDatingExceptionResponse: For returning response in case of exception |
| **exception** | InvalidUserException,  InvalidProfileException,  InvalidDatingSpecsException,  UserNotFoundException,  ProfileNotFoundException,  DatingSpecsNotFoundException | Custom Exception classes |
| **model** | User,  Profile,  DatingSpecs | Entity / Document classes |
| **repository** | UserRepository,  ProfileRepository,  DatingRepository | Repository interface declarations |
| **Services** | UserService,  UserServiceImpl,  ProfileService,  ProfileServiceImpl,  DartingService,  DatingServiceImpl | Service Layer Implementations |
| **utils** | DatingUtility | Class containing resources for inter conversion between model and dto objects |

**Note**

1. **Need to implement ONLY controller and service APIs in**

* **UserController.java, UserProfileController.java and DatingController.java classes along with**
* **UserServiceImpl.java, ProfileServiceImpl.java and DatingServiceImpl.java classes**

1. **DO NOT ADD ANY method or CHANGE any Rest End-Point pattern.**