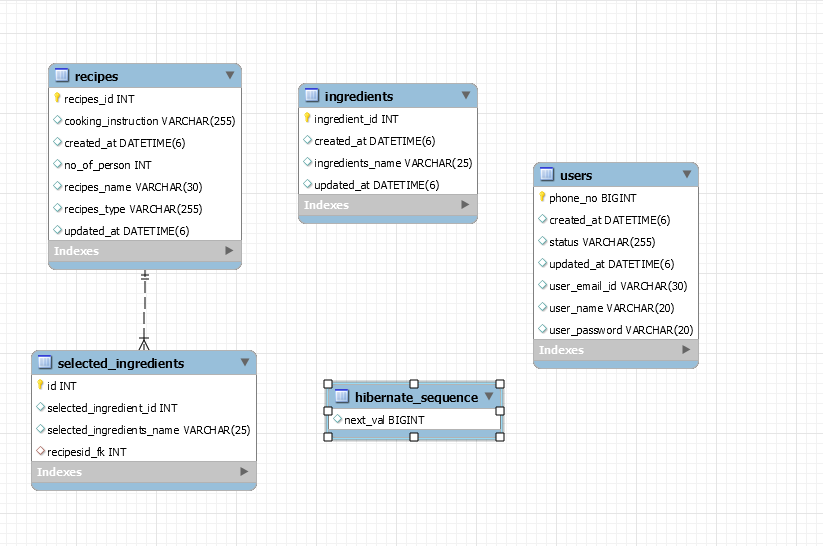
**Favorite Recipes Assignment Documentations**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Created By | Versions | Date |
| 1. | Sumit Vaidya | V1 | 02/06/2021 |
|  |  |  |  |
|  |  |  |  |

**Database Description Document**

1. MySQL server has been used for database creation along with tables.
2. I have created ‘recipes’, ‘selected\_ingredients’, ‘Ingredients’ and ‘users’ tables respectively by Spring boot data JPA specification provided by the spring boot.
3. ‘recipes’ table, it is being used for storing recipes details and has one to many relationships with the selected\_ingredients table where I have persisted selected ingredients from the UI.
4. Below is the E-R diagram of the ‘**favourite\_recipes\_db**’ database.



**Technology Description document**

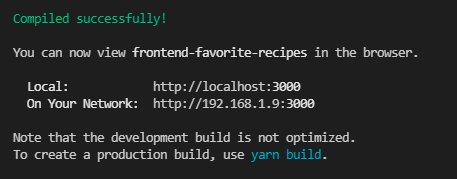
Technologies used to build up this application are as below:

1. I am currently on Java 8 and easy to use.
2. Spring Framework with Spring boot, Spring rest controller and Spring Data JPA along with Swagger 2. Spring boot is easy to configurable along with other module of the spring. Swagger 2 is for API documentation where user can hit to test the API’s and it stores the documentation itself.
3. For Database server, I have used MySQL.
4. For frontend pages, I have used React along with Typescript.

As application is not production ready, therefore need to run on the local system.

Frontend Url: <http://localhost:3000>

1. To run frontend, please follow below steps
2. Run ‘yarn’ => It will download all the dependencies locally that required to run frontend.
3. Run ‘yarn start’ => After completion of above command, now run this command



1. Hit on the browser with above frontend url

Backend Url: <http://localhost:2121>

Swagger 2 : <http://localhost:2121/api/swagger-ui.html#/>

Database Name: **favourite\_recipes\_db**

1. Need to install MySQL locally with port 3306
2. Need to set username and password for the MySQL
3. Need to create Database manually by running below command on the cli

Sql> Create database favourite\_recipes\_db

1. After database setup, need to run backend server of spring boot first time with

spring.jpa.hibernate.ddl-auto=create

It will create all the tables in the database automatically, after tables creation, below are the command to check whether it is created or not.

Sql> use favourite\_recipes\_db

Sql> show tables

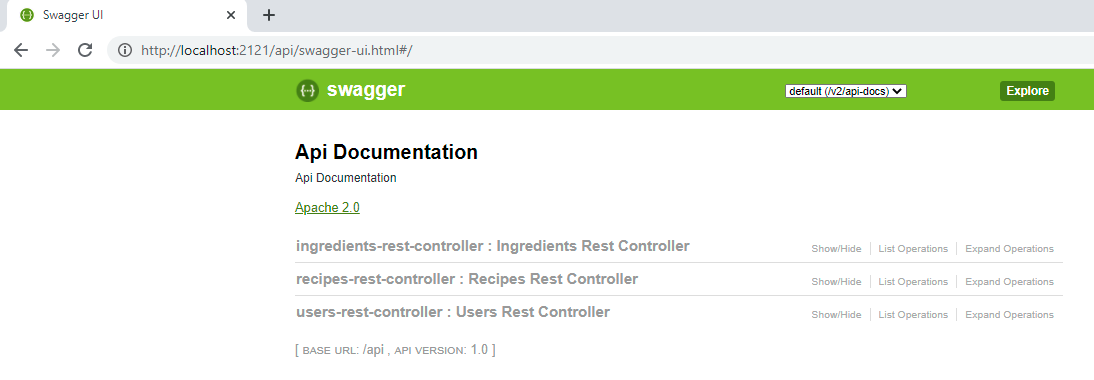
As verification done, need to uncomment spring.jpa.hibernate.ddl-auto=update line in the **application.properties** and comment create one spring.jpa.hibernate.ddl-auto=create so that next time onwards only update will perform on the table otherwise every time new tables will get created.

1. There is integration of UI + Backend + Database therefore to make the application run smoothly, need ensure that all the three are up and run.

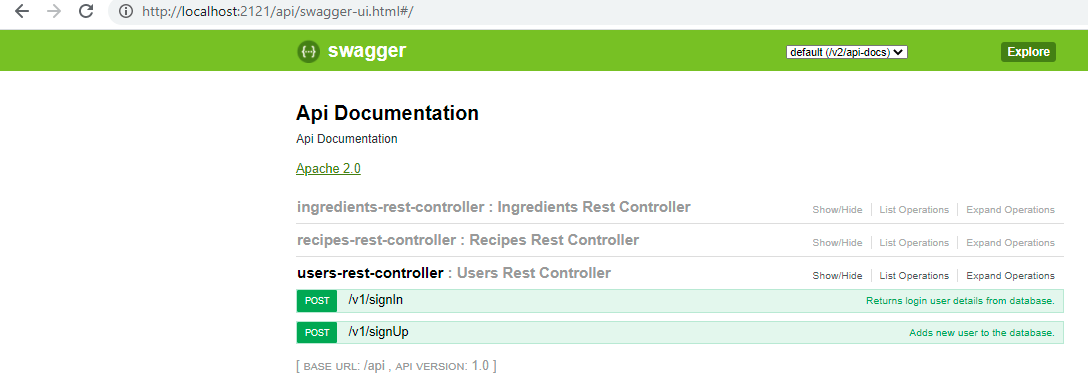
**Swagger for API Documentation**

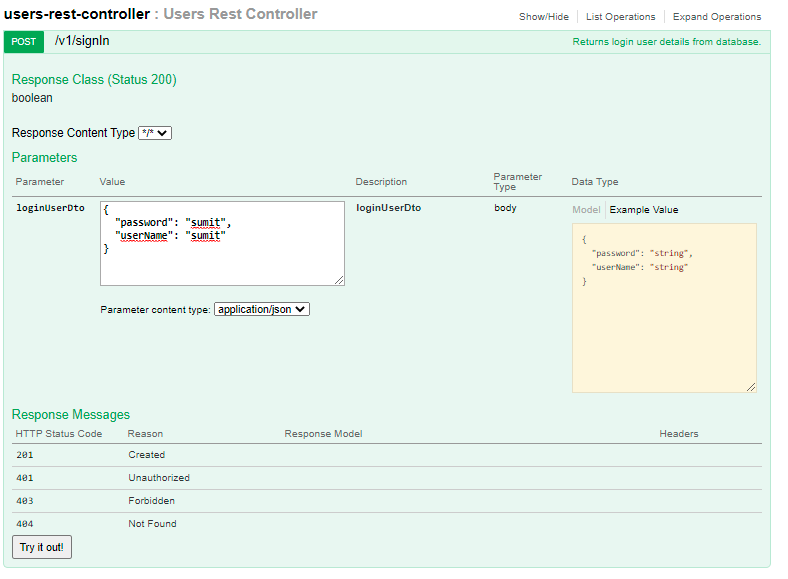
1. **Swagger 2** is used for the API documents for all the three rest calls.
2. **Ingredients Rest Controller** 
   1. Ingredients rest call is used to store ingredients information like id and name along with date.
   2. It has listed all ingredients, add ingredients, update ingredients, and delete ingredients operations.
   3. This ingredients details shows on the UI for Ingredients multiselect dropdown.
3. **Recipes Rest Controller**
   1. Recipes rest call is used to store recipes added from the system. It stores recipe name, recipe type, no. of person, cooking instruction and selected ingredients in the system.
   2. It has listed all recipes, add recipes, update recipes, and delete recipes operation.
   3. Those stored recipes information shows on the UI in table.
4. **Users Rest Controller**
   1. User has two options, Sign in and Sign up.
   2. If it is first time user, then need to sign up with some details like Phone number, username, password, and email id.
   3. Once user added to the system then he can directly login to the system by Sign in.
   4. If user is not register with the system, then it shows Error message like “Username and password is incorrect”.

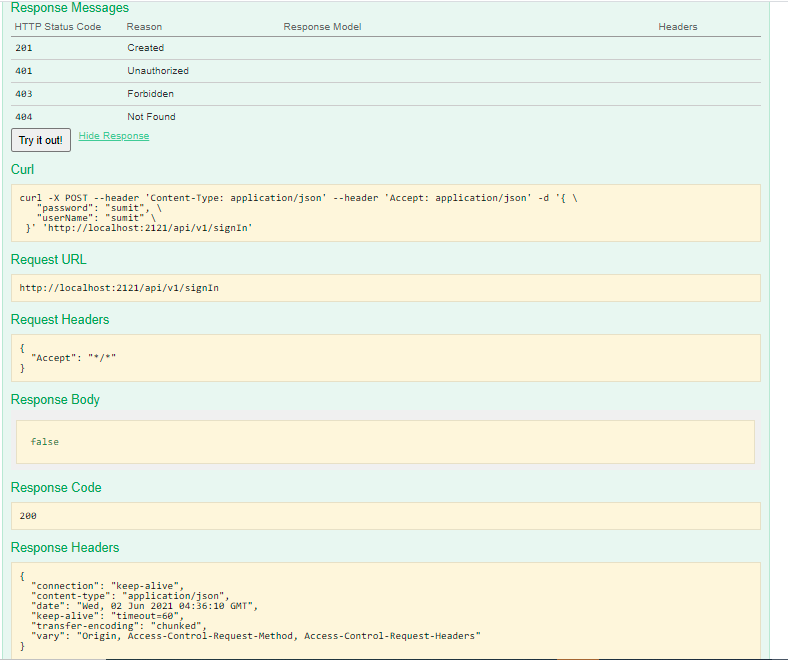
Below are the operations performed on the swagger for available rest end points.

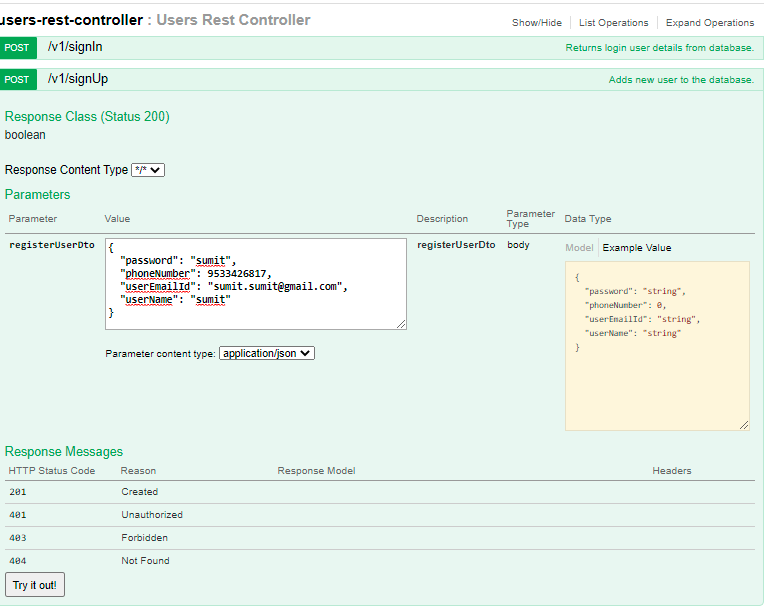


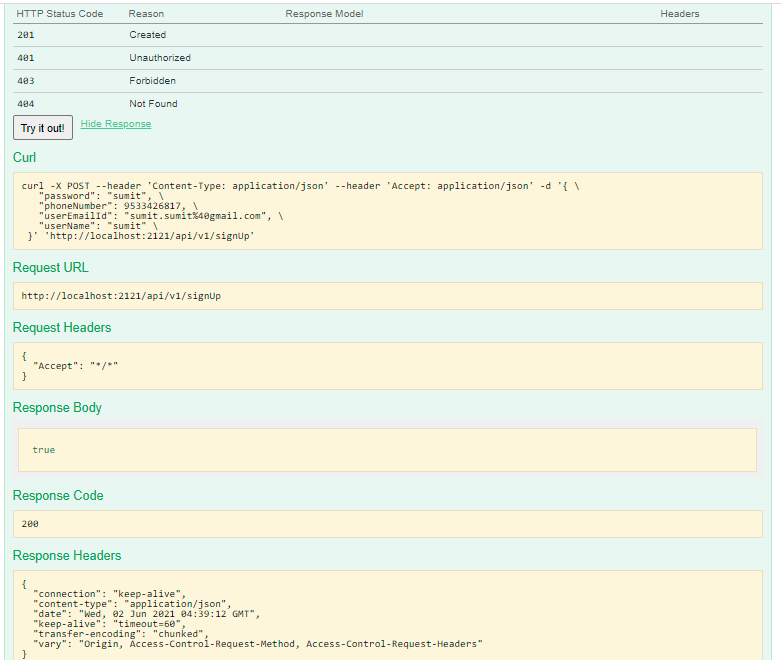
1. **User rest controller with sign in and sign up end points**

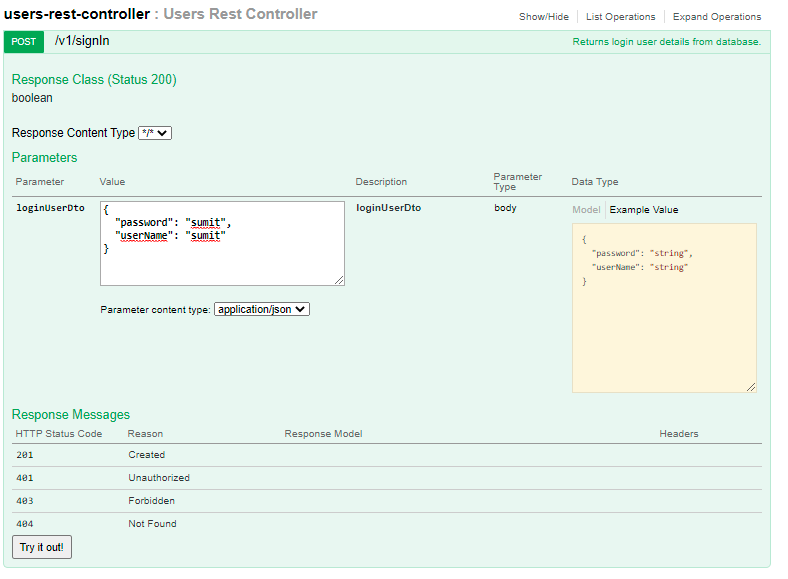


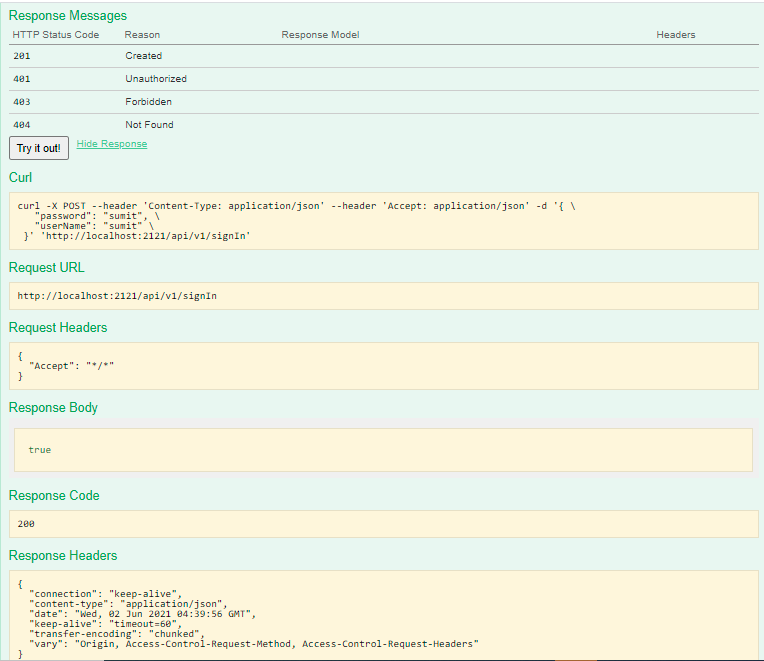




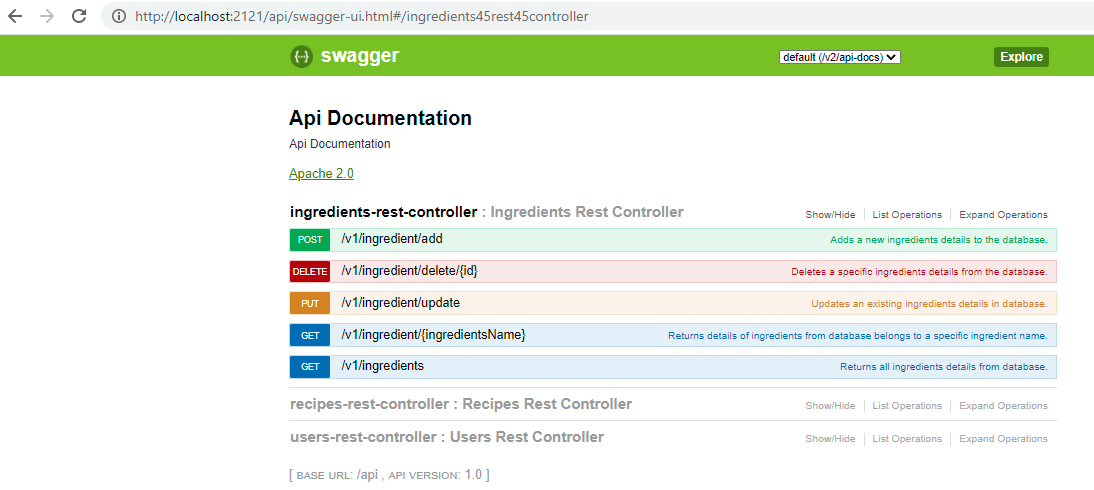


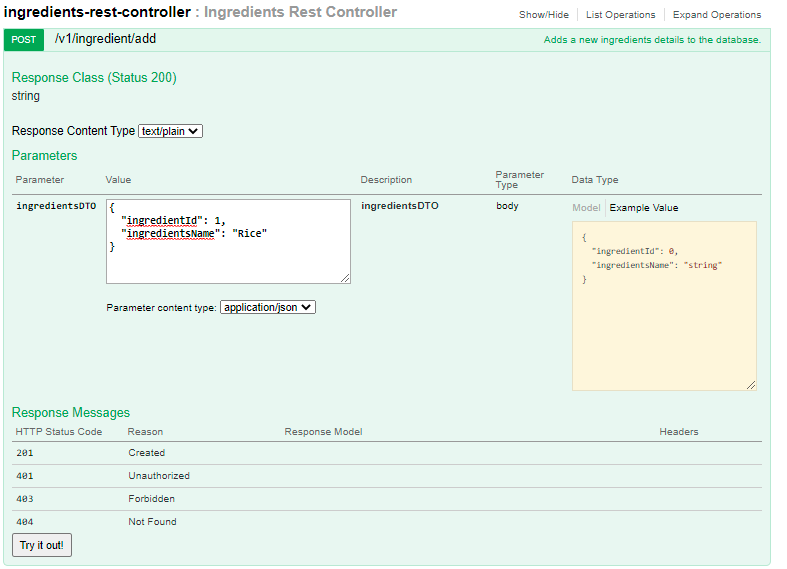


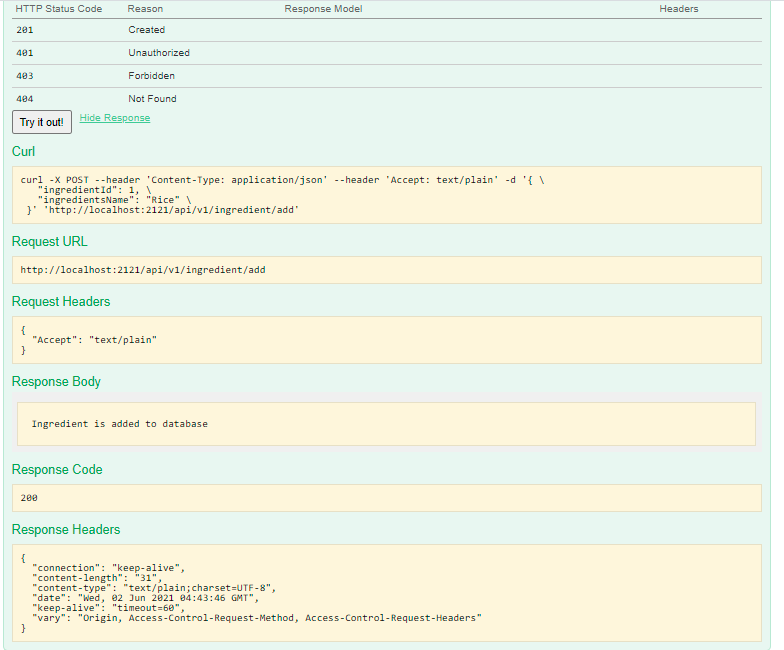


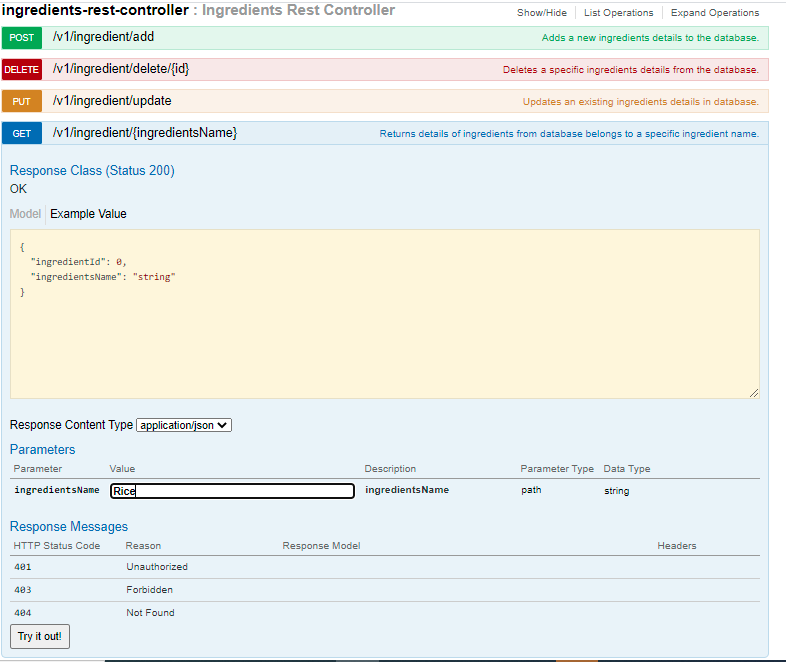


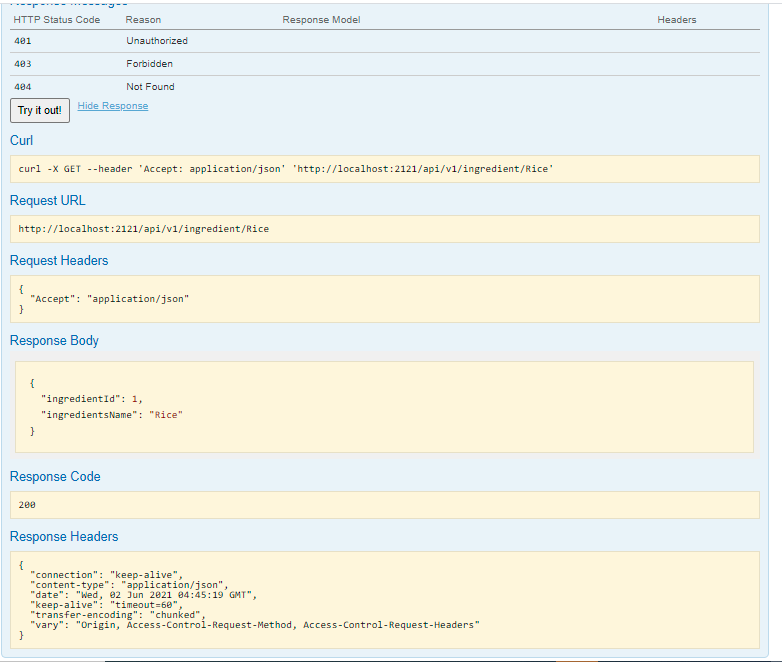
1. **Ingredients rest controller with its all operations**

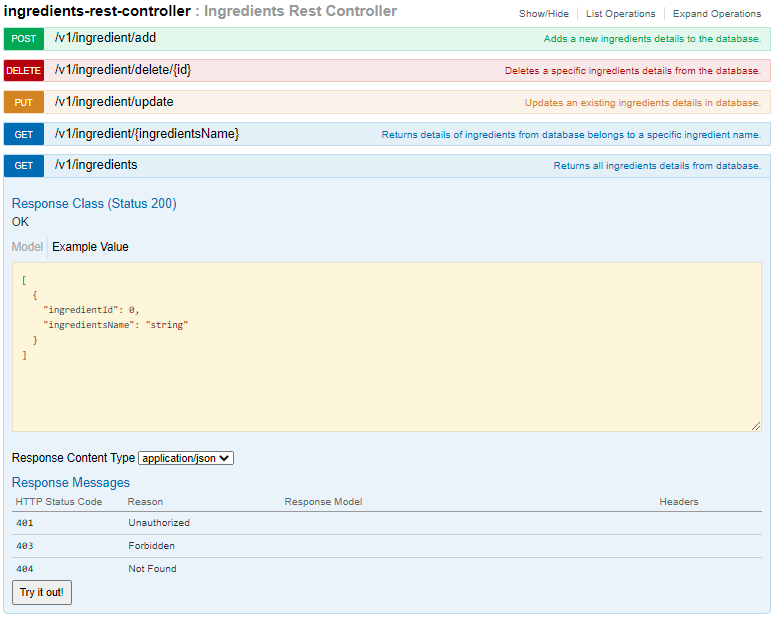


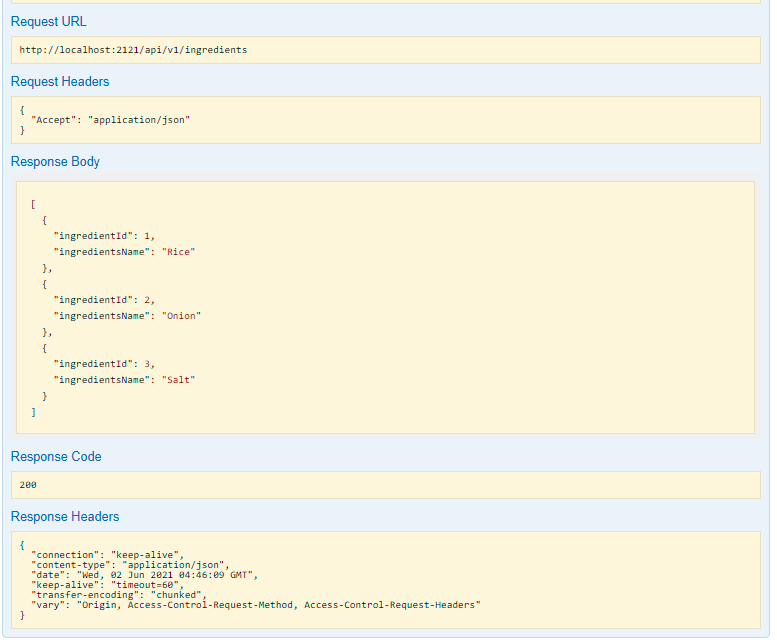


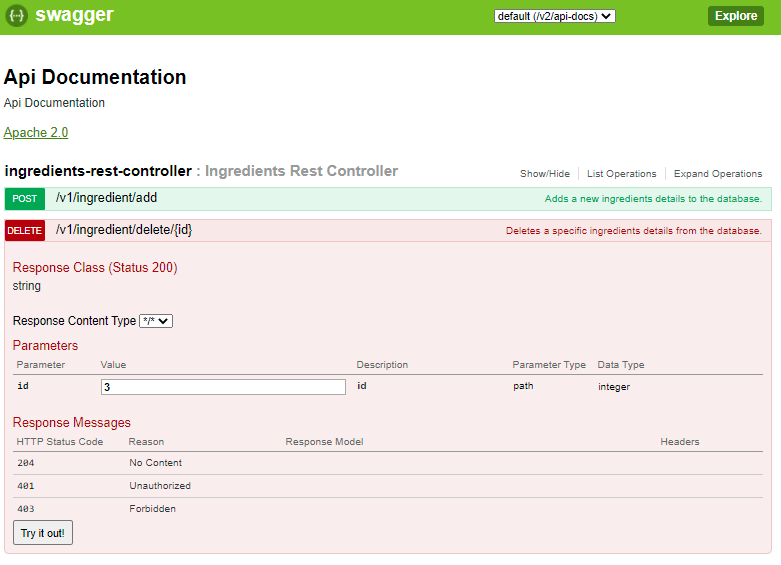


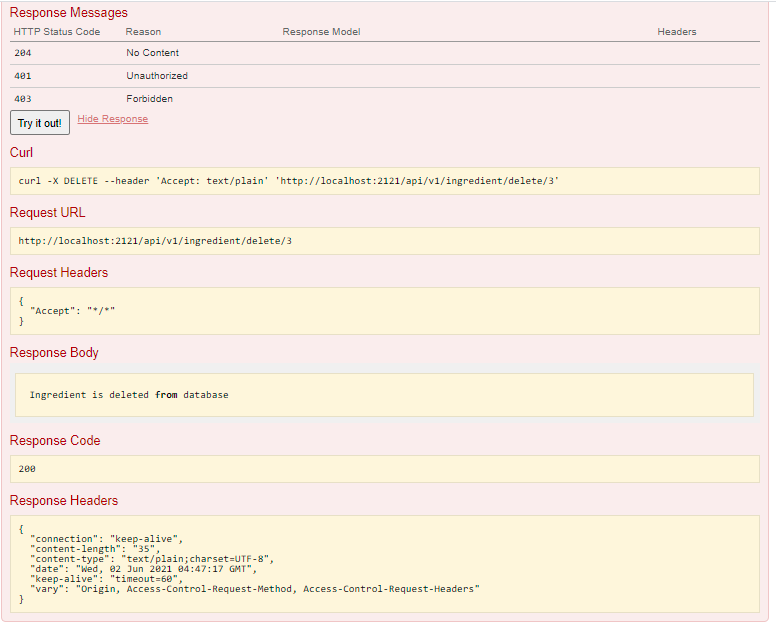


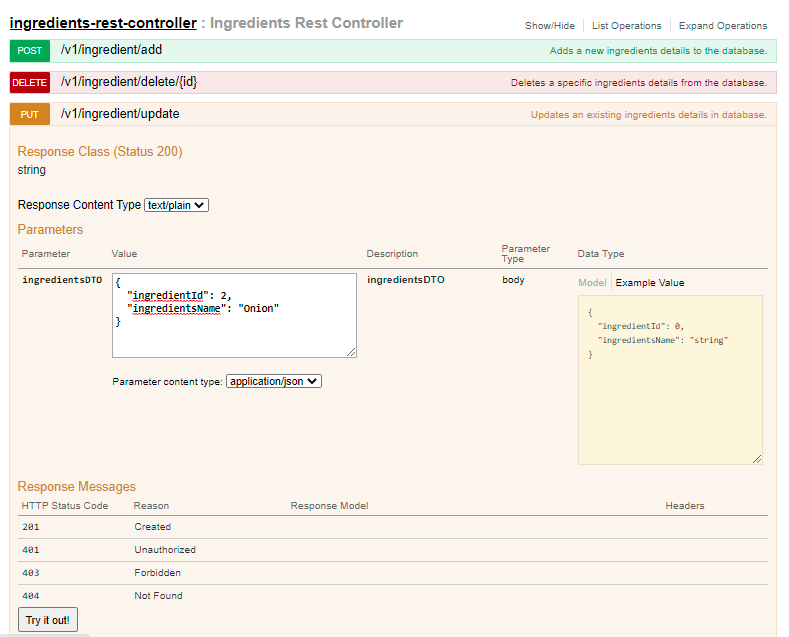


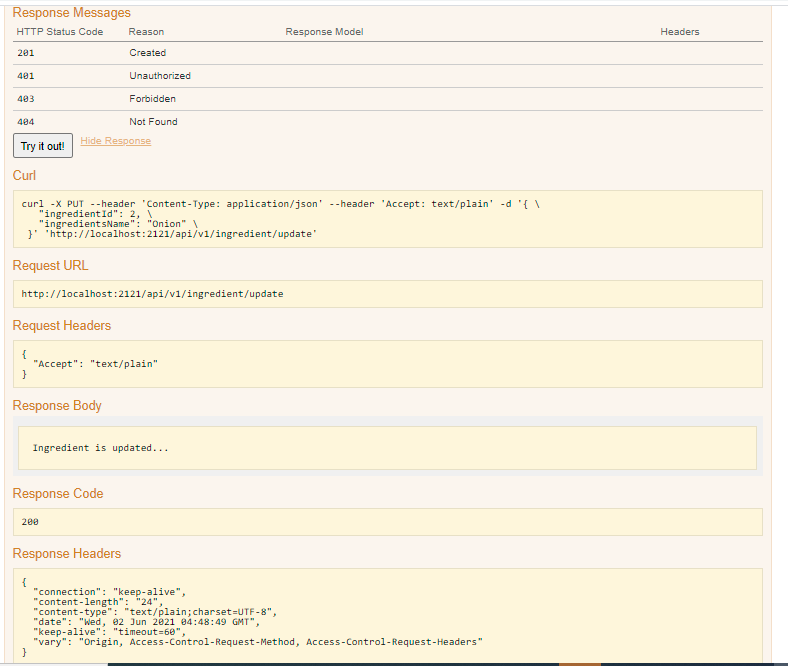


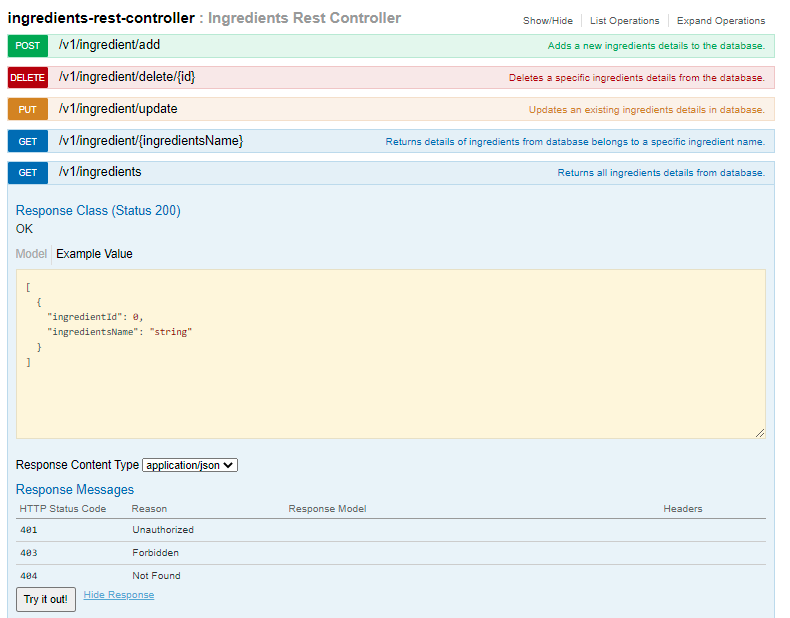


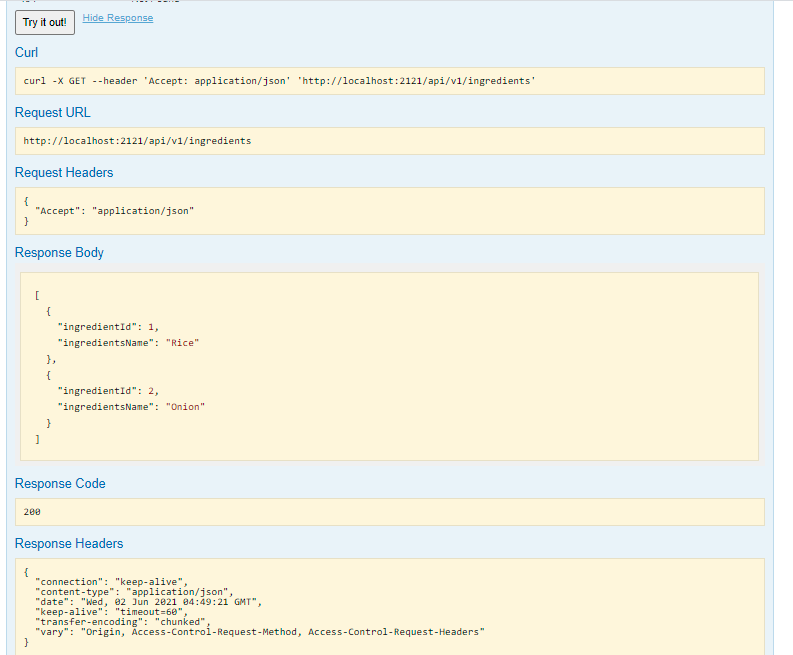




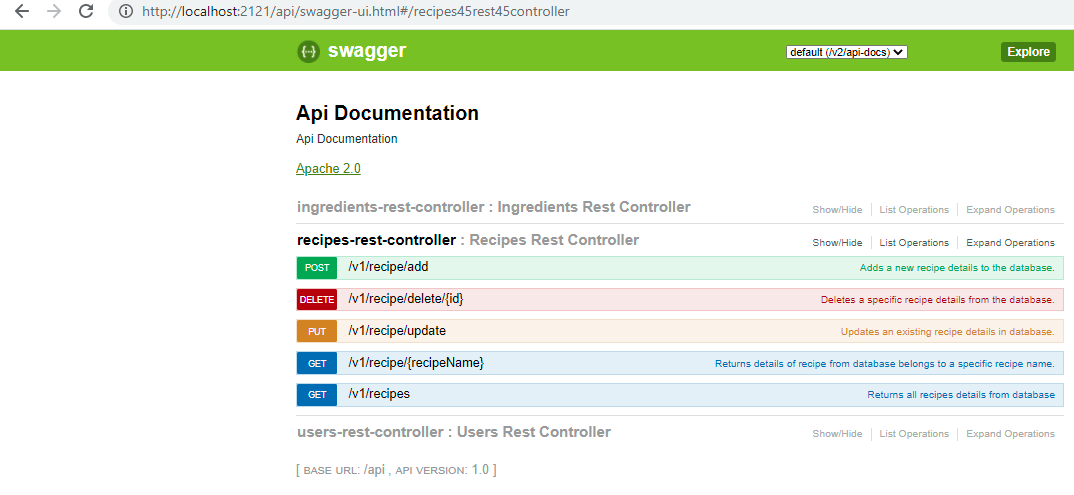


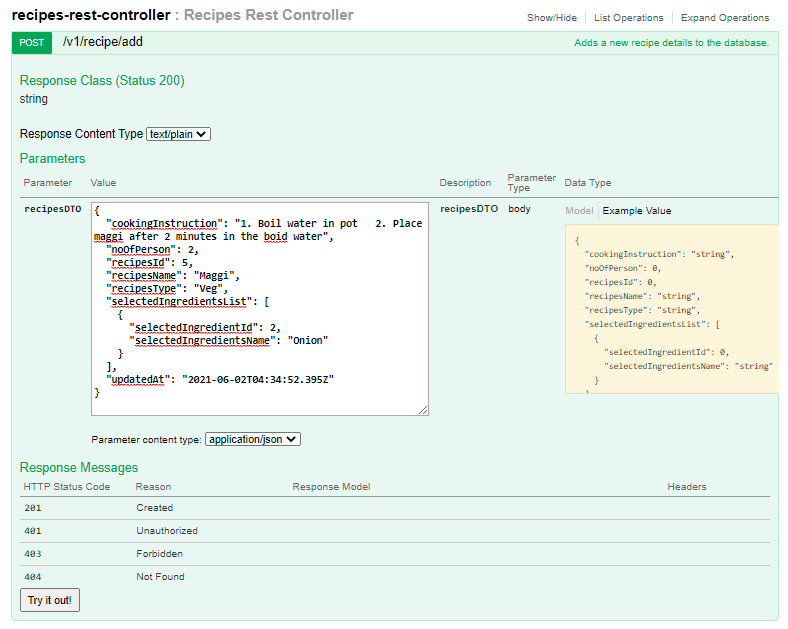


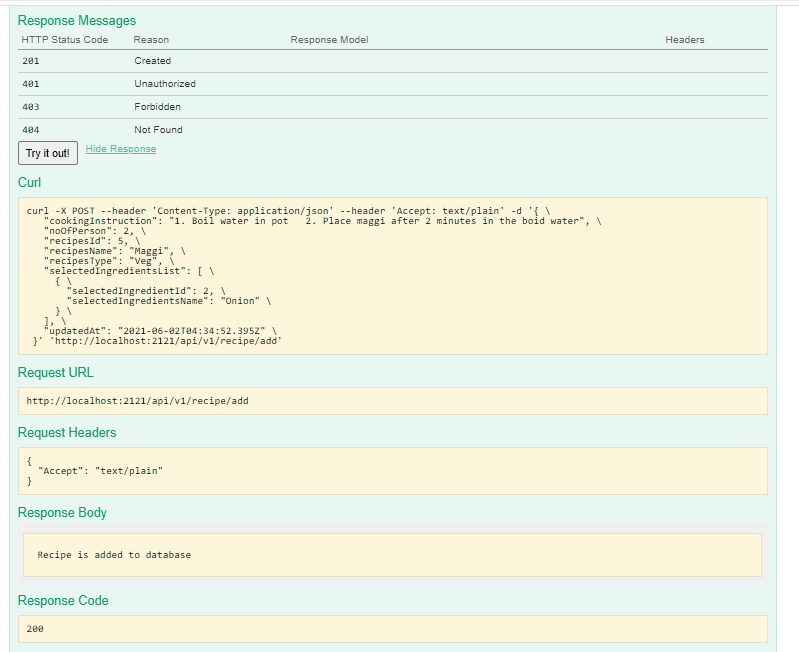


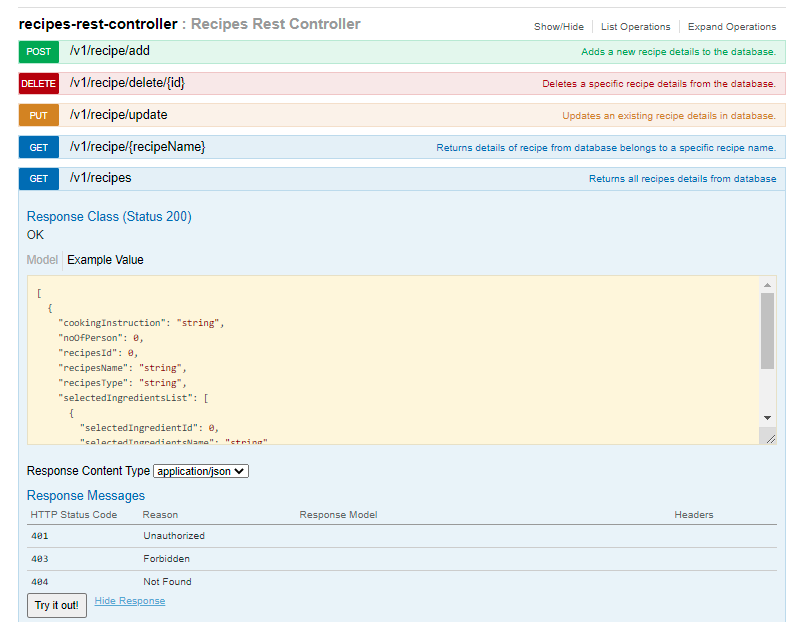


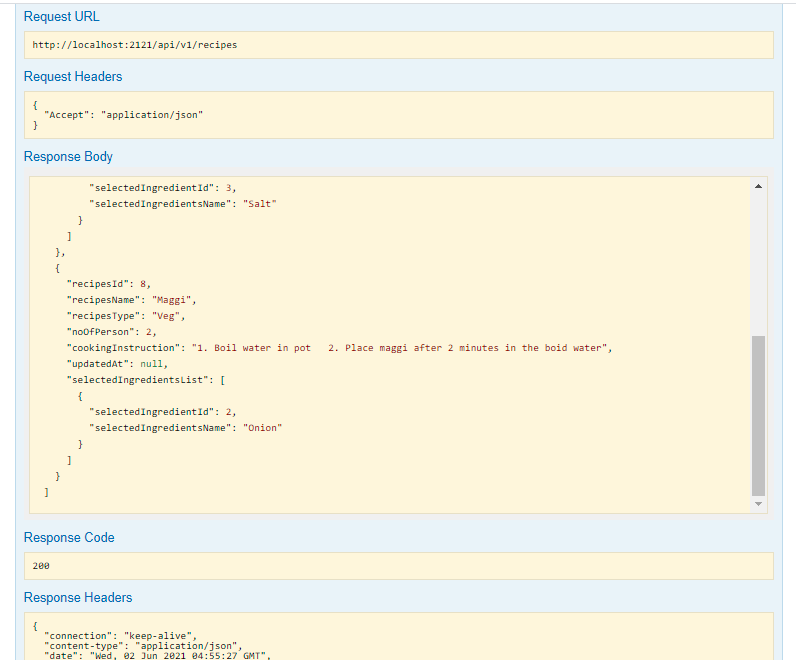
1. **Recipes rest controller with its all endpoint detail**

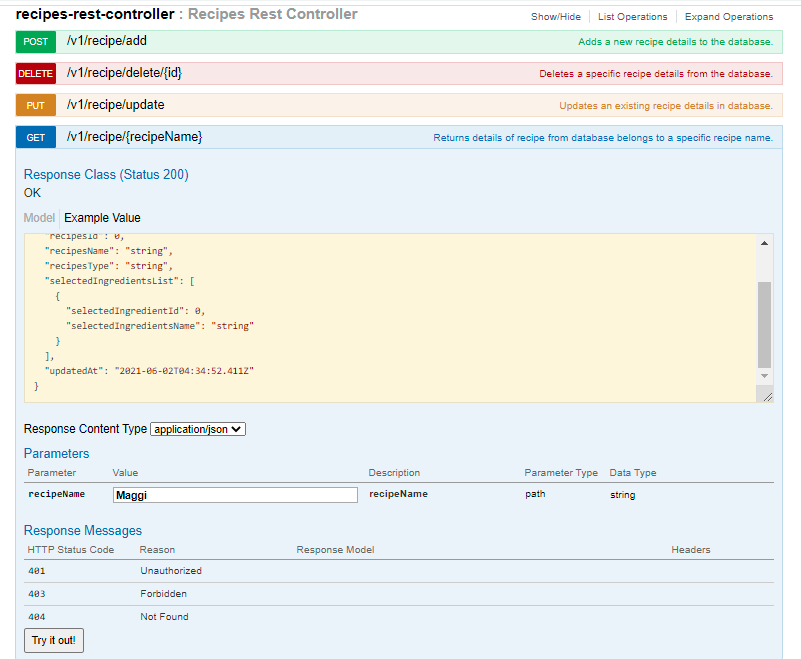


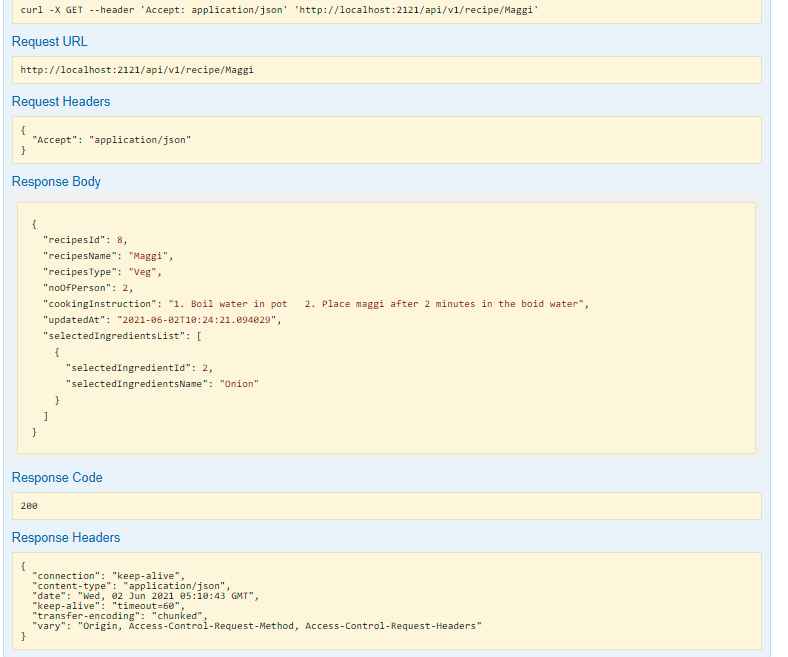


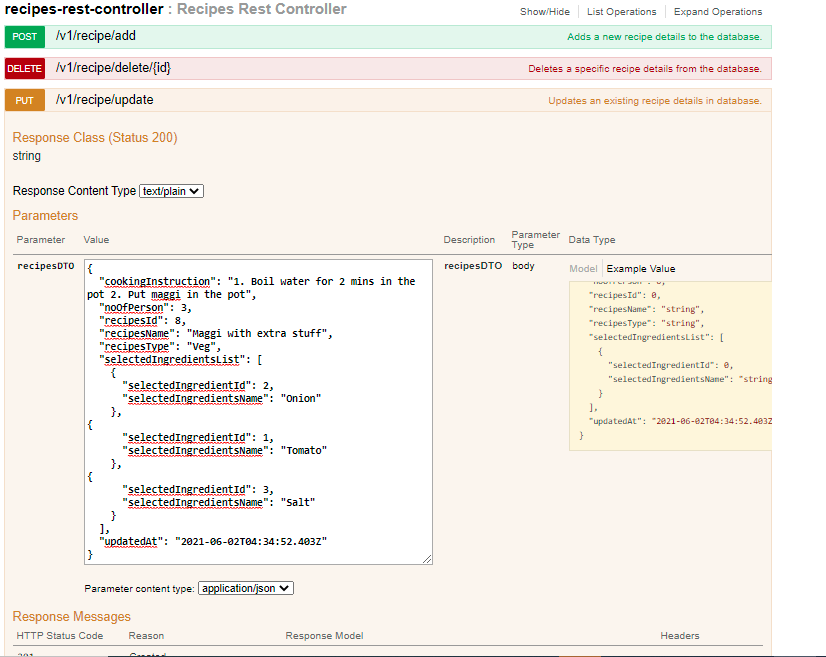


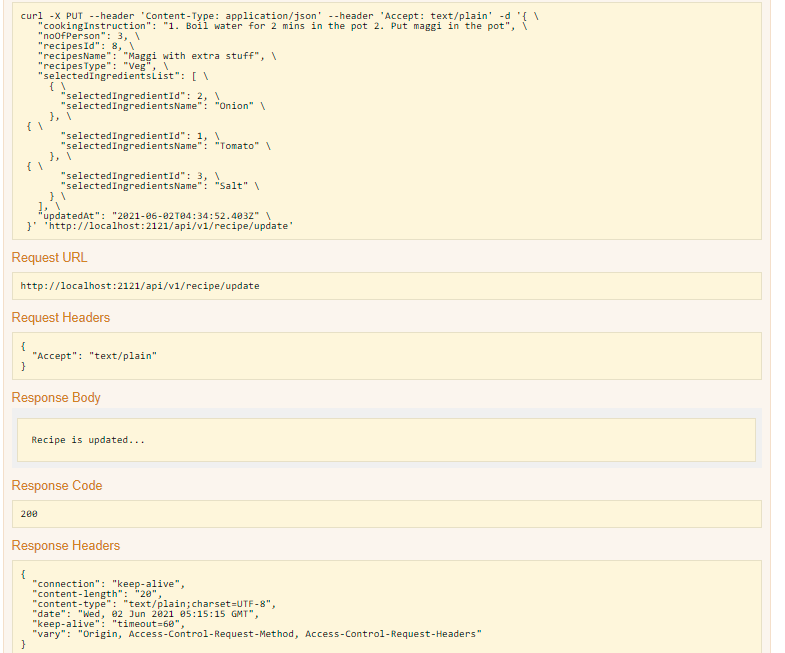


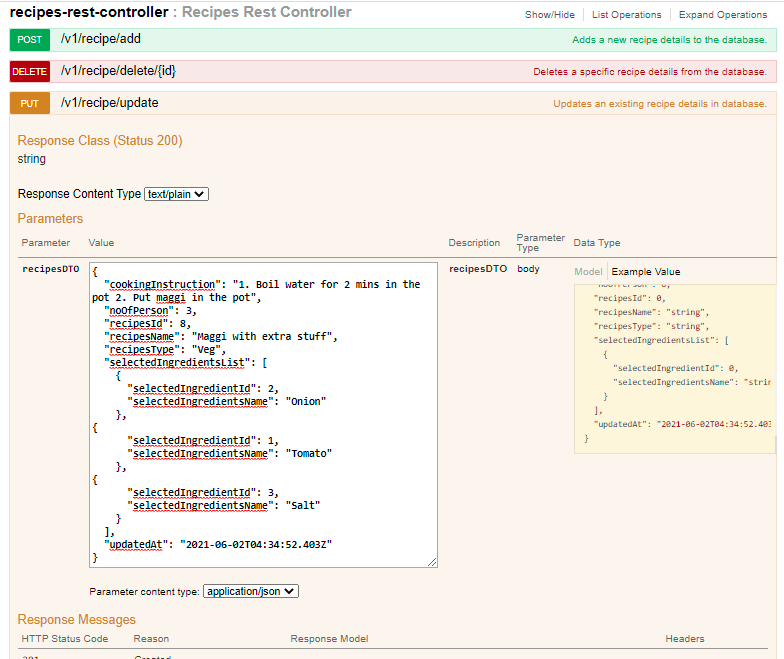


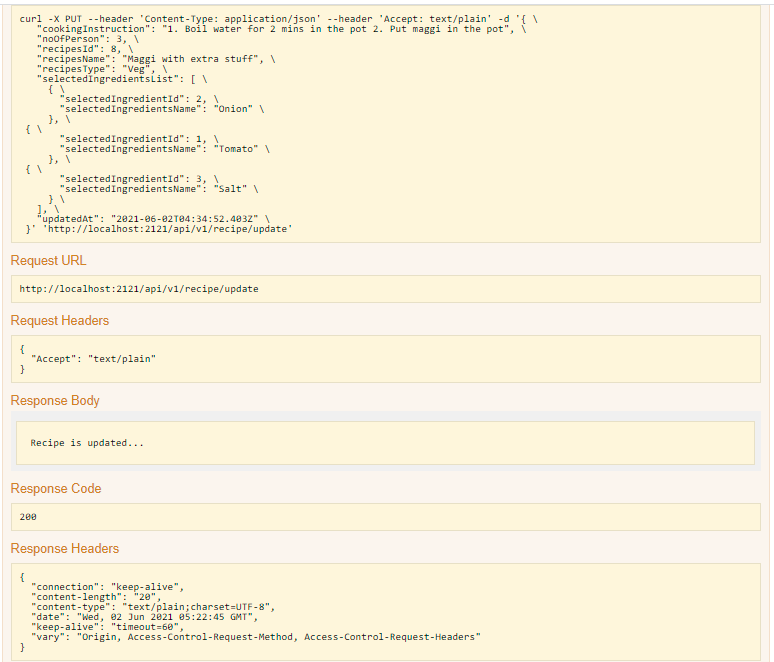


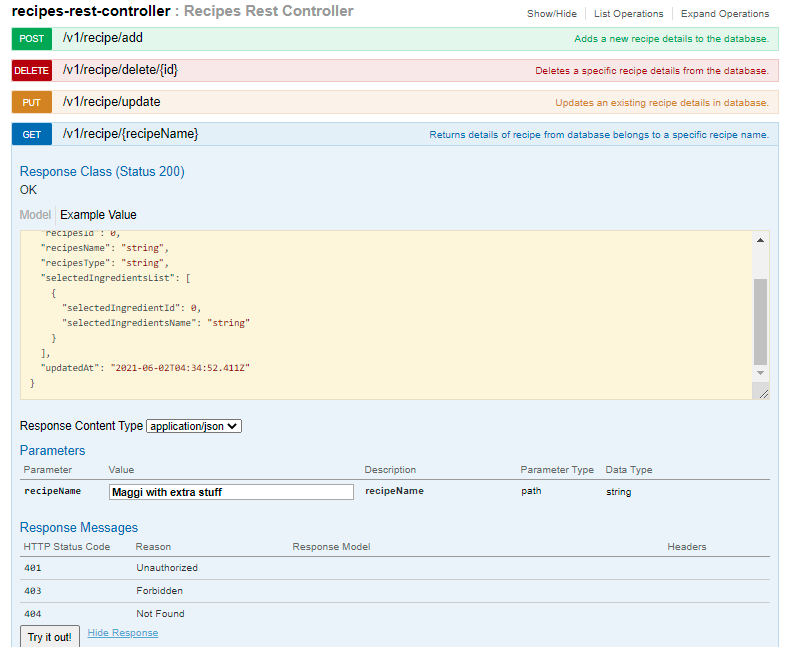


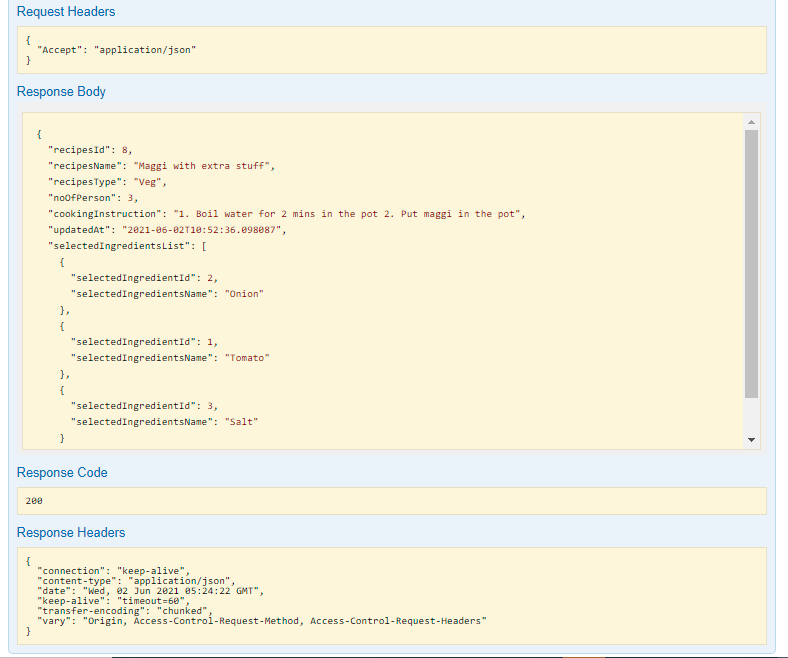


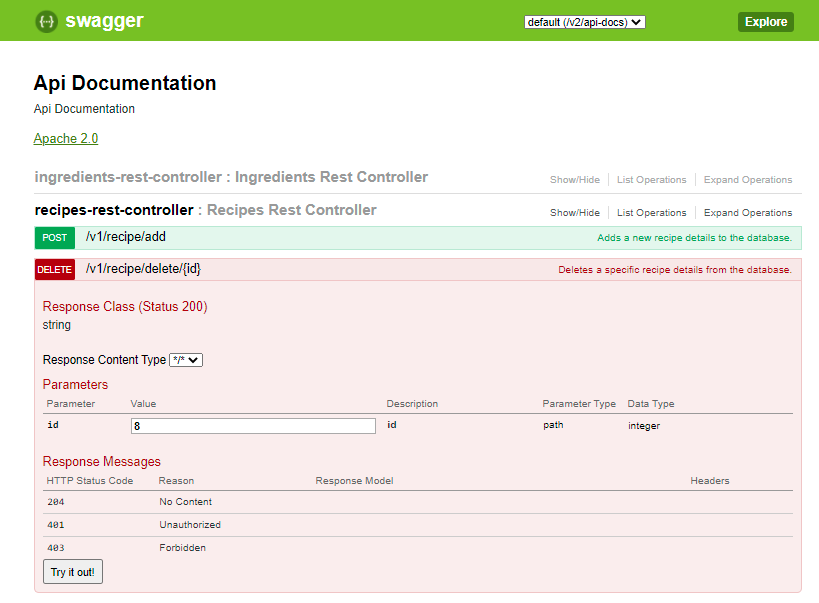


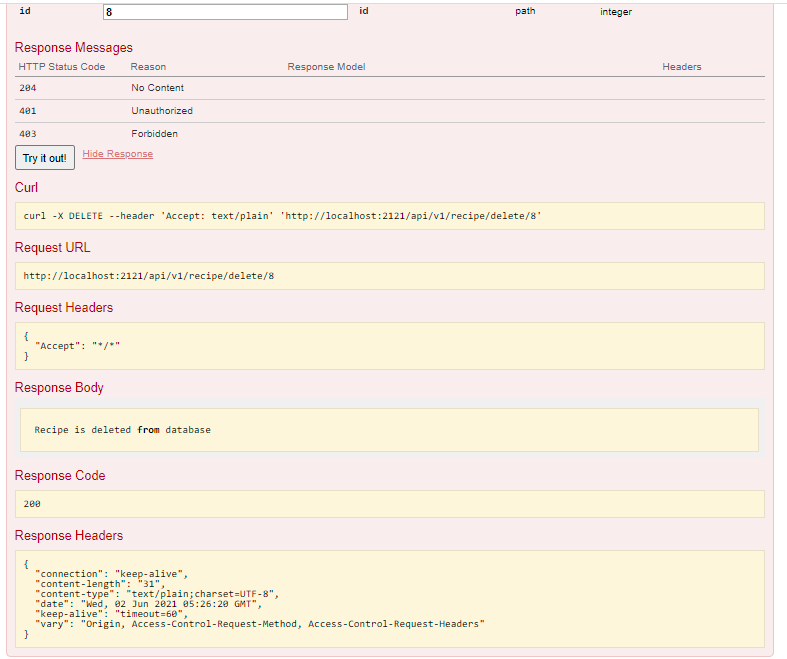


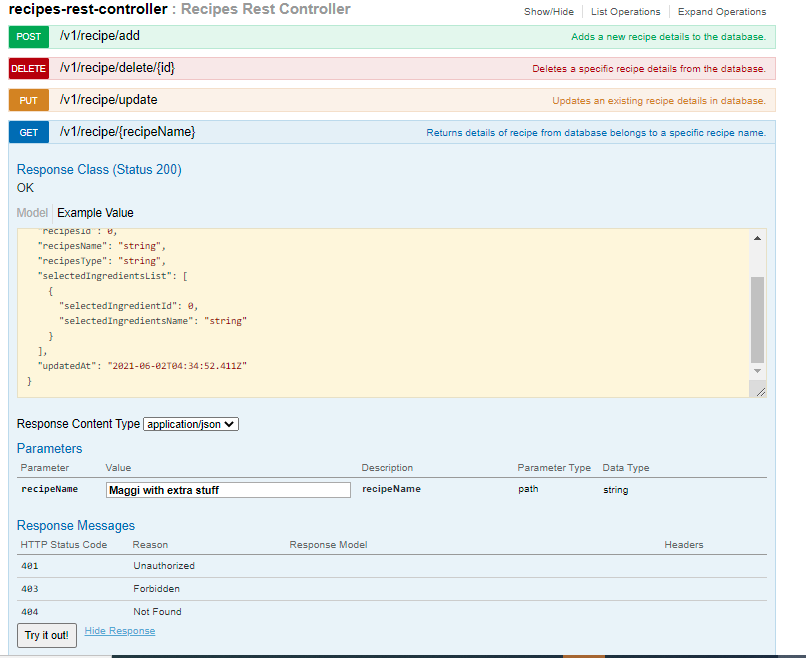


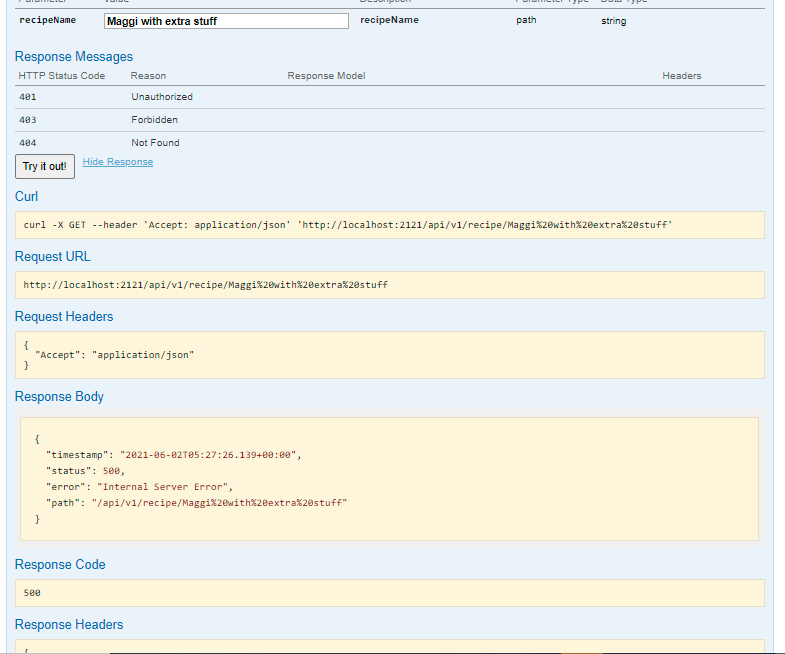






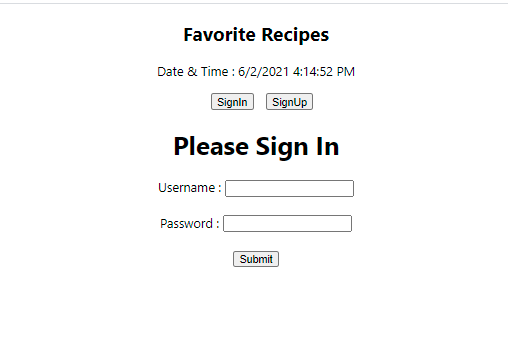




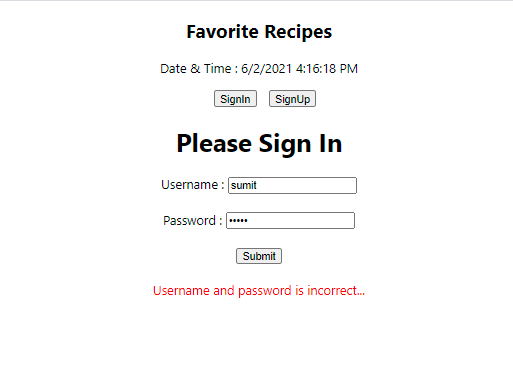


**UI Screens for Favorite Recipes**

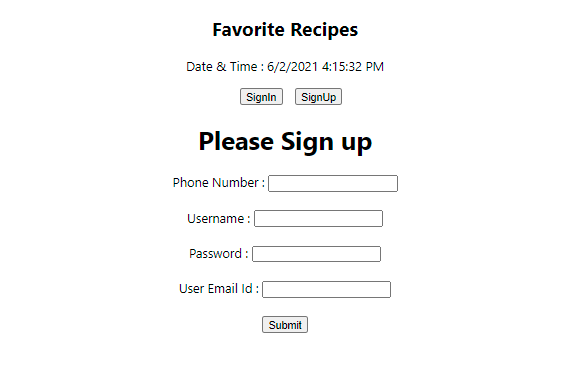
1. **User can Sign in from below screen**



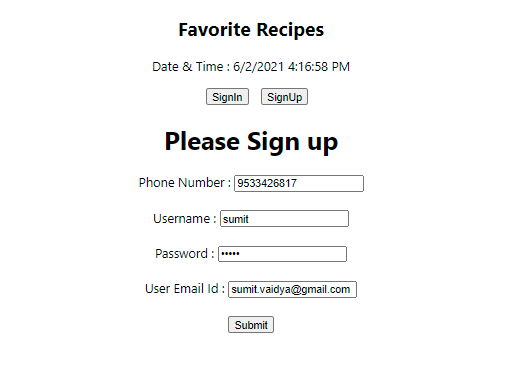
1. **Validation check if user is already available or not**



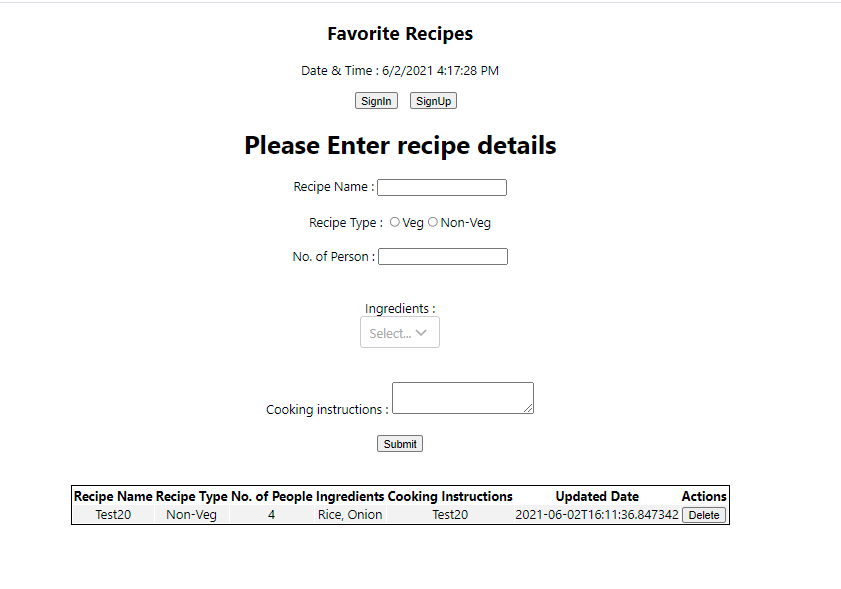
1. **Register user by Sign up screen**



1. **Registering user from the UI**



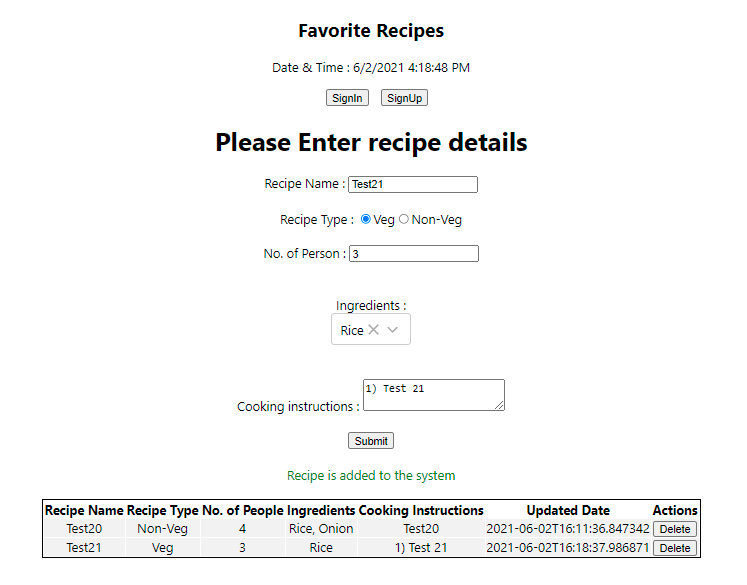
1. **UI for entering recipe details**



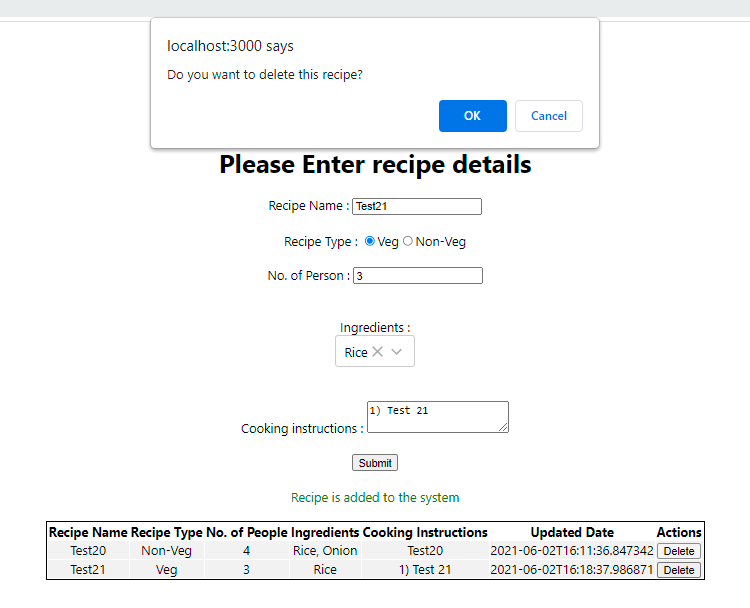
1. **Entered details on the Recipe UI screen**



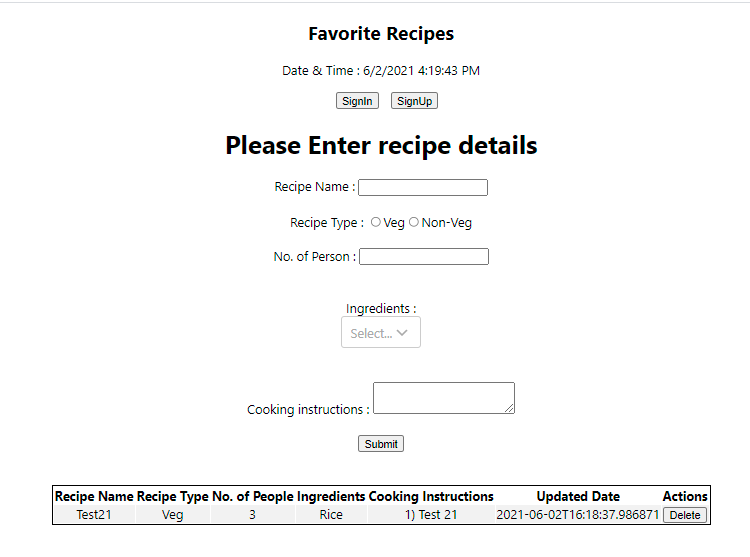
1. **As soon as recipe enter to the system, it displays in the table below with ‘Recipe is added to the system’**



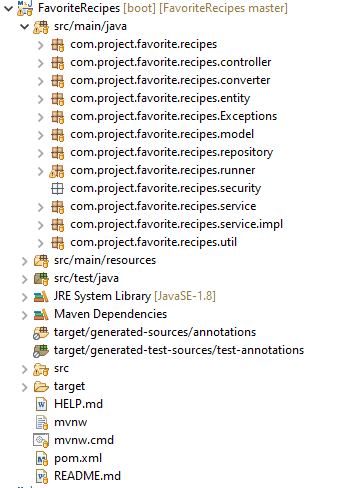
1. **User can delete the same added recipes from the table**



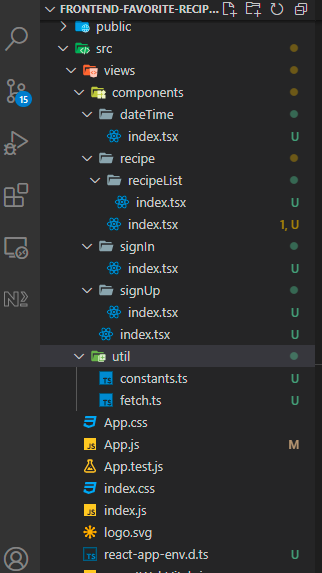
1. **After user delete the entry deleted from the table.**



**Spring boot project**



**React frontend project**



**Remaining actions items are below:**

1. Applications need to be production ready.
2. Test case with all test scenarios.
3. API needs to be wrapped and secured from the security attacks.
4. Need to work on UI look and fill.
5. Update functionality is pending from UI.
6. Exception handling when something goes wrong.