

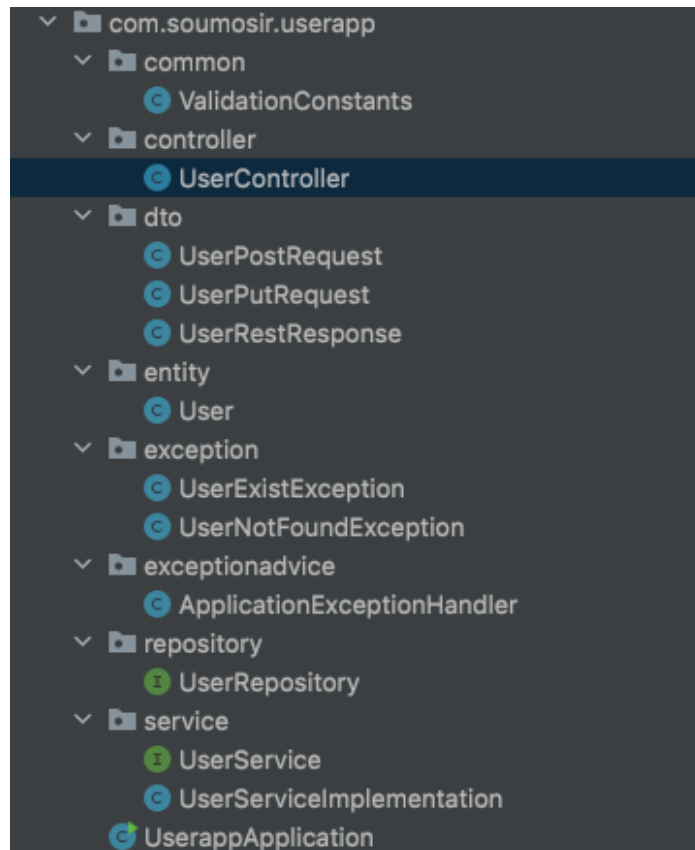
Question:- Backend user app

Answer:-

(Installation instructions at the end of pdf)

I have developed the application in Java Spring (Java version - openjdk 11.0.12 2021-07-20)

This is what the project structure looks like :-



I have segregated the Application in

1. **Entity** - which is the User object
2. Also, there are **Data Transfer Objects** for REST API call for PUT, POST, and GET
3. There is a **Controller** which is the REST API Controller for the User and it maps to UserService
4. For the **Service** layer , there is UserService interface has a concrete implementation called UserServiceImplementation
5. There is **ExceptionHandler** - ApplicationExceptionHandler which handles the exceptions and displayed appropriate messages to the user in the REST API response.
6. Lastly, the Service layer communicates with the **Repository** layer((JPA) for persisting the User's Data.

API examples according to requirements:-

1. POST /userapp/users should create a new user based on the given values.

- name - should be more than 2 characters and less than 20 characters
- address - should be more than 4 characters and less than 200 characters
- dateOfBirth - should be of format MM/dd/yyyy and a past date
- username - should be alphanumeric A-Z, a-z, 0-9 and can have _ and . characters except for beginning and end

The curl request for that is

```
curl --location --request POST 'http://localhost:9292/userapp/users' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "name": "Soumosir Dutta",  
  "username": "soumosir_dutta",  
  "dateOfBirth": "11/12/1993",  
  "address": "3412 Tulane Drive, Maryland"  
}'
```

Example

The screenshot displays a REST client interface with a sidebar on the left containing a collection named 'userapp_cisco' with several endpoints. The main panel shows a POST request to 'http://localhost:9292/userapp/users'. The request body is a JSON object with the following fields: 'name' (Soumosir Dutta), 'username' (soumosir_dutta), 'dateOfBirth' (11/12/1993), and 'address' (3412 Tulane Drive, Maryland). The interface includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The 'Body' tab is active, showing the raw JSON data. Below the request, there are tabs for Body, Cookies, Headers (5), and Test Results. The 'Body' tab is also active here, showing the 'Pretty' view of the JSON response.

```
POST http://localhost:9292/userapp/users
```

```
{  
  "name": "Soumosir Dutta",  
  "username": "soumosir_dutta",  
  "dateOfBirth": "11/12/1993",  
  "address": "3412 Tulane Drive, Maryland"  
}
```

```
{  
  "name": "Soumosir Dutta",  
  "username": "soumosir_dutta",  
  "dateOfBirth": "11/12/1993",  
  "address": "3412 Tulane Drive, Maryland"  
}
```

Validations for the fields which is given by a json with 400 status code

```
{  
  errorMessage : { field1 : <reason> }  
}
```

Example:-

POST http://localhost:9292/userapp/users

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "name" : "Pe",  
3   "username" : "pdu",  
4   "dateOfBirth" : "11/12/2050",  
5   "address" : "1"  
6 }  
7
```

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 78 ms Size: 45;

Pretty Raw Preview Visualize JSON

```
1 {  
2   "errorMessage": {  
3     "address": "Address should more than 4 charectors.",  
4     "name": "Name should more than 2 charectors.",  
5     "dateOfBirth": "Date of birth should not be a future date.",  
6     "username": "Username should be between 8 to 20 alphanumeric charectors (a-zA-Z0-9) with . and _ allowed except for beginning and end."  
7   }  
8 }
```

Some more validation on error in date format

POST http://localhost:9292/userapp/users

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

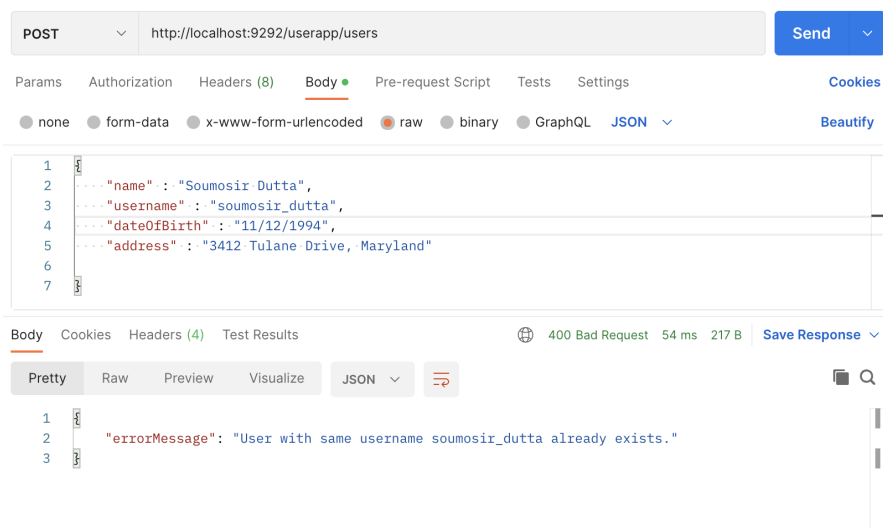
```
1 {  
2   "name" : "Paul Doe",  
3   "username" : "pdutta123",  
4   "dateOfBirth" : "11/12/19950",  
5   "address" : "123 Bris Drive, Maryland"  
6 }  
7
```

Body Cookies Headers (4) Test Results Status:

Pretty Raw Preview Visualize JSON

```
1 {  
2   "errorMessage": {  
3     "dateParseError": "Text '11/12/19950' could not be parsed at index 6. Expected format MM/dd/yyyy."  
4   }  
5 }
```

Validation error if username already exists



POST `http://localhost:9292/userapp/users` Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "name": "Soumosir Dutta",
3   "username": "soumosir_dutta",
4   "dateOfBirth": "11/12/1994",
5   "address": "3412 Tulane Drive, Maryland"
6 }
```

Body Cookies Headers (4) Test Results 400 Bad Request 54 ms 217 B Save Response

Pretty Raw Preview Visualize JSON

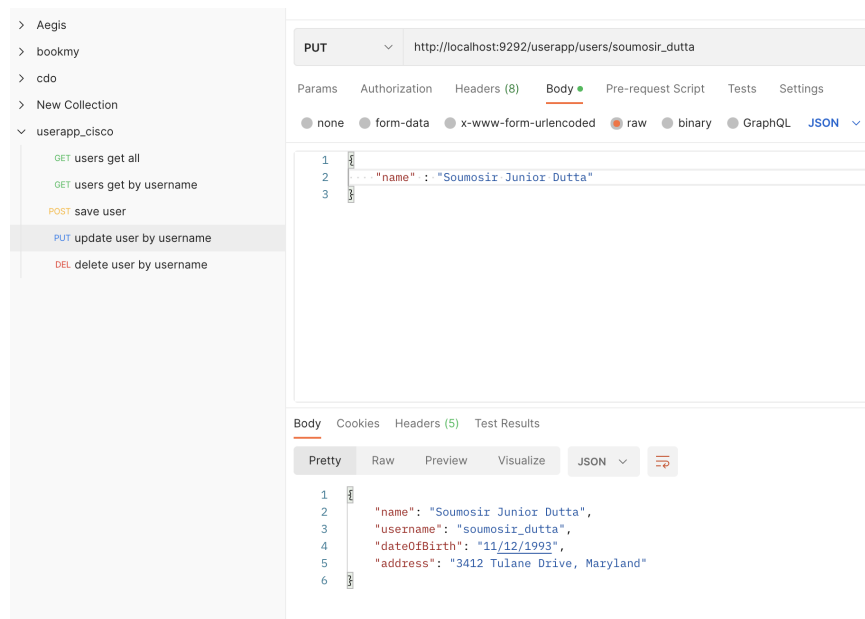
```
1 {
2   "errorMessage": "User with same username soumosir_dutta already exists."
3 }
```

2. PUT /userapp/users/<username> should update the user with the given information.

The curl request for that is

```
curl --location --request PUT 'http://localhost:9292/userapp/users/soumosir_dutta' \
--header 'Content-Type: application/json' \
--data-raw '{
  "name": "Soumosir Junior Dutta"
}'
```

Here we are updating only the name field



PUT `http://localhost:9292/userapp/users/soumosir_dutta`

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "Soumosir Junior Dutta"
3 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "name": "Soumosir Junior Dutta",
3   "username": "soumosir_dutta",
4   "dateOfBirth": "11/12/1993",
5   "address": "3412 Tulane Drive, Maryland"
6 }
```

3. GET /userapp/users should return a list of users.

curl --location --request GET 'http://localhost:9292/userapp/users'

The screenshot shows a REST client interface with a sidebar on the left containing a tree view of collections. The main panel displays a GET request to `http://localhost:9292/userapp/users/`. The 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON array of two user objects in 'Pretty' format.

KEY	VALUE
Key	Value

```
1 [
2   {
3     "name": "Soumosir Junior Dutta",
4     "username": "soumosir_dutta",
5     "dateOfBirth": "11/12/1993",
6     "address": "3412 Tulane Drive, Maryland"
7   },
8   {
9     "name": "Paul Doe",
10    "username": "pdutta123",
11    "dateOfBirth": "11/12/1995",
12    "address": "123 Brisbane Drive, Maryland"
13  }
14 ]
```

4. GET /userapp/users/<username> should return details of a specific user.

curl --location --request GET 'http://localhost:9292/userapp/users/soumosir_dutta'

The screenshot shows a REST client interface with a sidebar on the left. The main panel displays a GET request to `http://localhost:9292/userapp/users/soumosir_dutta`. The 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON object representing a single user in 'Pretty' format.

KEY	VALUE
Key	Value

```
1 {
2   "name": "Soumosir Junior Dutta",
3   "username": "soumosir_dutta",
4   "dateOfBirth": "11/12/1993",
5   "address": "3412 Tulane Drive, Maryland"
6 }
```

5. DELETE /userapp/users/<username> should delete the specified user.

curl --location --request DELETE 'http://localhost:9292/userapp/users/soumosir_dutta'

The screenshot shows a REST client interface with a sidebar on the left containing a collection named 'userapp_cisco' with several endpoints. The main panel is configured for a DELETE request to 'http://localhost:9292/userapp/users/soumosir_dutta'. The 'Params' tab is active, showing a 'Query Params' table with one entry: 'Key' with 'Value'. Below the table, the 'Body' tab is selected, displaying '1 true' in a 'Pretty' JSON format.

KEY	VALUE
Key	Value

Body: 1 true

Get call to verify it is actually deleted

The screenshot shows a REST client interface with a collection named 'userapp_cisco' and an endpoint 'users get by username'. The main panel is configured for a GET request to 'http://localhost:9292/userapp/users/soumosir_dutta'. The 'Params' tab is active, showing a 'Query Params' table with one entry: 'Key' with 'Value'. Below the table, the 'Body' tab is selected, displaying the response in a 'Pretty' JSON format: '1 { "errorMessage": "User with username soumosir_dutta does not exists." }'. The status bar at the bottom indicates a 404 Not Found response with a 74 ms response time and 240 B of data.

KEY	VALUE	DESCRIPTION
Key	Value	Description

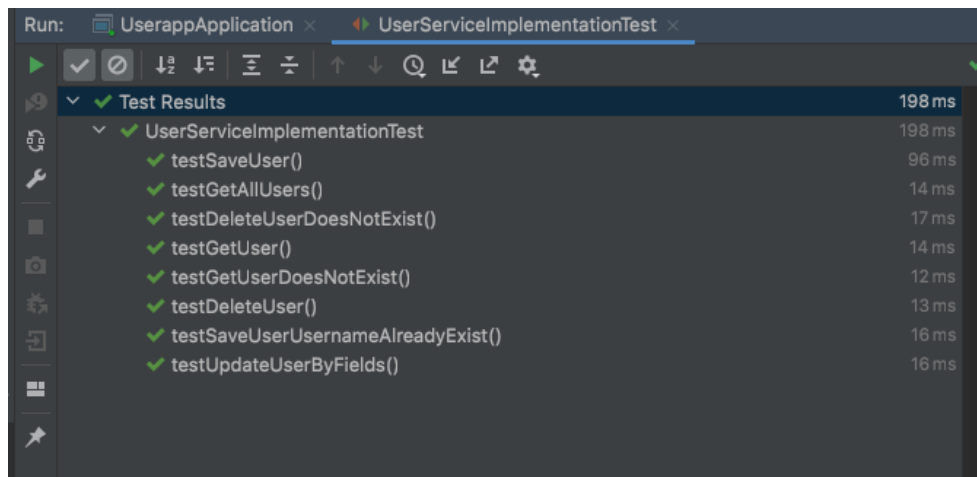
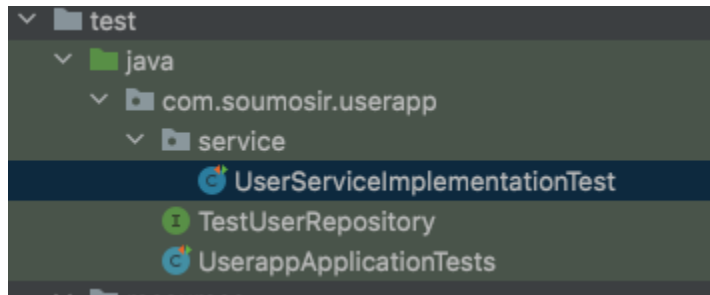
Body: 1 { "errorMessage": "User with username soumosir_dutta does not exists." }

404 Not Found 74 ms 240 B

Also, there are validations in place if there is no user with a username.

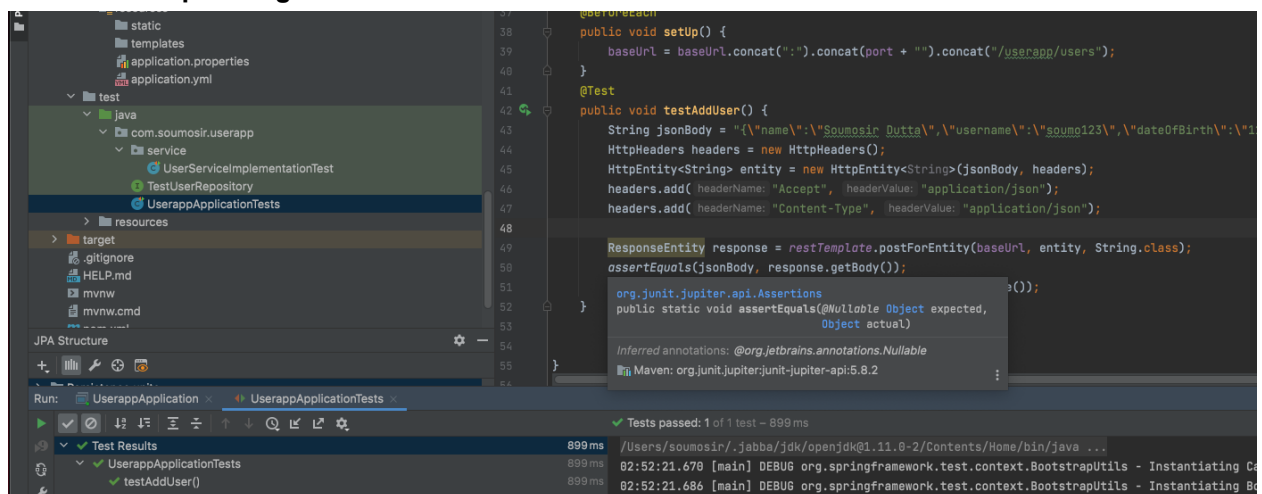
UNIT TEST CASES

I have written some unit test cases for UserServiceImplementation



src/test/java/com/soumosir/userapp/service/UserServiceImplementationTest.java

There is a simple integration test written for add user.



src/test/java/com/soumosir/userapp/UserappApplicationTests.java

Logging

There is some logging functionality added with `@Slf4j`

```
2022-12-19 04:06:11.260 INFO 62436 --- [nio-9292-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-12-19 04:06:11.264 INFO 62436 --- [nio-9292-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
2022-12-19 04:06:12.005 INFO 62436 --- [nio-9292-exec-1] c.s.u.service.UserServiceImplementation : Saving user with username soumosir_dutta.
2022-12-19 04:06:19.463 INFO 62436 --- [nio-9292-exec-2] c.s.u.service.UserServiceImplementation : Updating user with username soumosir_dutta.
2022-12-19 04:06:26.807 INFO 62436 --- [nio-9292-exec-3] c.s.u.service.UserServiceImplementation : Deleting user with username soumosir_dutta.
2022-12-19 04:06:40.498 ERROR 62436 --- [nio-9292-exec-6] c.s.u.e.ApplicationExceptionHandler : User with username soumosir_dutta does not exists.
```

The logs are stored in **customlog.txt** in the root directory.

The logs for tests are stored in customlogTest.txt in root

PTO

Test Case Scenarios:-

Note: Separate Test cases can be created for all edge conditions:-

INVALID USER POST/PUT FIELDS

1. Name < 3 characters
2. Name > 200 characters
3. Address < 5 characters
4. Address > 5 characters
5. Date of birth - future date
6. Date of birth - wrong format other than MM/dd/yyyy
7. Username - less than 8 characters
8. Username - more than 20 characters
9. Username - non alphanumeric character with . or _ beginning
Example of invalid username "abc 123445", "__ab2222c", ".qwerty123"

Consolidated

Sl_no	Description
1	Save user with invalid fields - name < 3 and name > 20 - address < 5 and address > 200 - date of birth - future date - username - not alphanumeric number and < 8 or > 20

Input

```
POST 'http://localhost:9292/userapp/users' \n{\n  "name": "Pa",\n  "username": "pdu w",\n  "dateOfBirth": "11/12/2050",\n  "address": "12"\n}
```

Expected Output

```
{\n  "errorMessage": {\n    "address": "Address should more than 4 charectors.",\n    "name": "Name should more than 2 charectors.",\n    "dateOfBirth": "Date of birth should not be a future date.",\n    "username": "Username should be between 8 to 20 alphanumeric charectors (a-zA-Z0-9) with . and _ allowed except for beginning and end." \n  }\n}
```

Pre-Condition: Create a user with username - soumosir_dutta

Sl_no	Description
2	Save another user with username - soumosir_dutta

Input

```
curl --location --request POST 'http://localhost:9292/userapp/users' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "name" : "Soumosir Dutta",  
  "username" : "soumosir_dutta",  
  "dateOfBirth" : "11/12/1993",  
  "address" : "3412 Tulane Drive, Maryland"  
}'  
curl --location --request POST 'http://localhost:9292/userapp/users' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "name" : "Soumosir Dutta",  
  "username" : "soumosir_dutta",  
  "dateOfBirth" : "11/12/1993",  
  "address" : "3412 Tulane Drive, Maryland"  
}'
```

Expected Output

```
{  
  "errorMessage": "User with same username soumosir_dutta already exists." }
```

Pre-Condition: Create a user with username - soumosir_dutta

Sl_no	Description
2	Update user with username - soumosir_dutta , name to be "Soumosir Junior Dutta"

Input

```
curl --location --request POST 'http://localhost:9292/userapp/users' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "name" : "Soumosir Dutta",  
  "username" : "soumosir_dutta",  
  "dateOfBirth" : "11/12/1993",  
  "address" : "3412 Tulane Drive, Maryland"  
}'  
  
curl --location --request PUT 'http://localhost:9292/userapp/users/soumosir_dutta' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "name" : "Soumosir Junior Dutta"  
}'
```

Expected Output (Name is successfully updated)

```
{  
  "name": "Soumosir Junior Dutta",  
  "username": "soumosir_dutta",  
  "dateOfBirth": "11/12/1993",  
  "address": "3412 Tulane Drive, Maryland"  
}
```

uri = /userapp/users/

sl_no	description	input	Expected output
3	Pre-Condition : create user with username soumosir_dutta Delete an user which exists	DELETE uri/<username> curl --location --request DELETE 'http://localhost:9292/userapp/users/soumosir_dutta'	true
4	Delete an user which doesnot exist	Delete uri/<username> curl --location --request DELETE 'http://localhost:9292/userapp/users/soumosir_dutta3'	404 { "errorMessage": "User with username soumosir_dutta3 does not exists." }
5	Get all users Should return list of users	GET uri curl --location --request GET 'http://localhost:9292/userapp/users'	[{ "name": "Paul Doe", "username": "pdutta123", "dateOfBirth": "11/12/1995", "address": "123 Brisbane Drive, Maryland" }, { "name": "Soumosir Junior Dutta", "username": "soumosir_dutta", "dateOfBirth": "11/12/1993", "address": "3412 Tulane Drive, Maryland" }]
6	Pre-condition :create User with pdutta123 Get user by username Should return a user in json	GET URI/<username> curl --location --request GET 'http://localhost:9292/userapp/users/pdutta123'	{ "name": "Paul Doe", "username": "pdutta123", "dateOfBirth": "11/12/1995", "address": "123 Brisbane Drive, Maryland" }
7	Get user by username which doesnt exists	GET URI/<username> curl --location --request GET 'http://localhost:9292/userapp/users/pdutta12377'	404 { "errorMessage": "User with username pdutta12377 does not exists." }

Additional work to be done:-

1. Adding more unit test cases for userController and userServiceImplementation class
2. Adding unit test cases for Exception handler
3. Adding integration test for the remaining flow

Steps to run the application: -

1. Download java version 11

```
java --version
openjdk 11.0.12 2021-07-20
OpenJDK Runtime Environment Microsoft-25199 (build 11.0.12+7)
OpenJDK 64-Bit Server VM Microsoft-25199 (build 11.0.12+7, mixed mode)
```

2. Unzip userapp.zip

3. Run the jar file directly

```
java -jar target/userapp-0.0.1-SNAPSHOT.jar
```

Or

Run in development mode

```
mvn --version
Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63)
Maven home: /opt/homebrew/Cellar/maven/3.8.6/libexec
Java version: 18.0.2, vendor: Homebrew, runtime: /opt/homebrew/Cellar/openjdk/18.0.2/libexec/openjdk.jdk/Contents/Home
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "13.0.1", arch: "aarch64", family: "mac"
```

```
cd userapp
mvn install
mvn spring-boot:run
```

Or

Open the Spring application in an ide
Install dependencies - mvn install

And run the
[src/main/java/com/soumosir/userapp/UserappApplication.java](#) file

Please contact soumosirdutta@gmail.com for any setup help!