Team Members: Yiwei Tu, Mengxuan Wang, Jiaqi Zhan

Project Name: To-do List for Multi-User

Instruction on building project

Database setup:

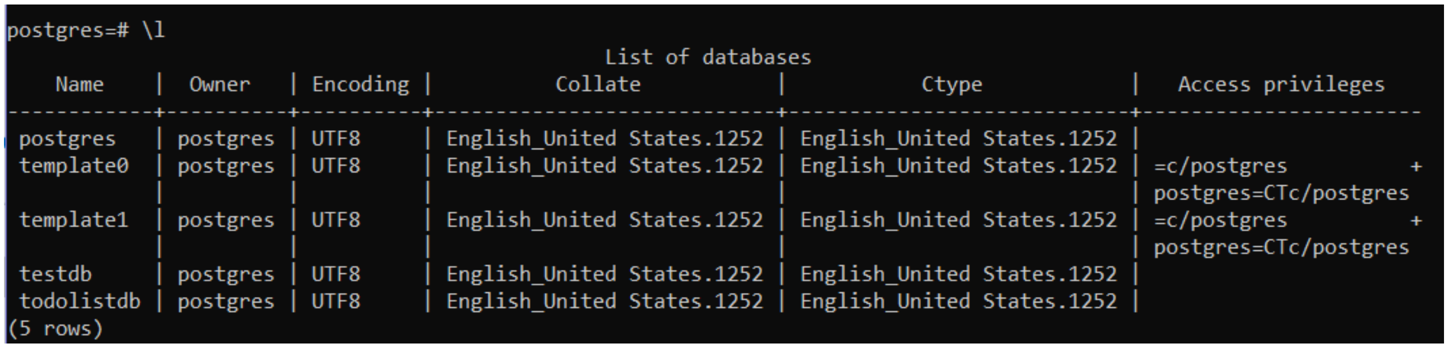
In this project, we are using **Postgresql** as a data storage component. So in order to run and use this application properly, you should have Postgresql installed on your machine. Keep a record of your username and password. These will be used for database connection.

In addition, two databases have been used. So before running the server program, you need to make sure you have these two databases, `todolistdb` and `testdb`.

To check whether you have the two databases in your postgresql server, you can open your terminal and type the following command to run your Postgresql instance.

```
psql -U {your username} -h localhost
```

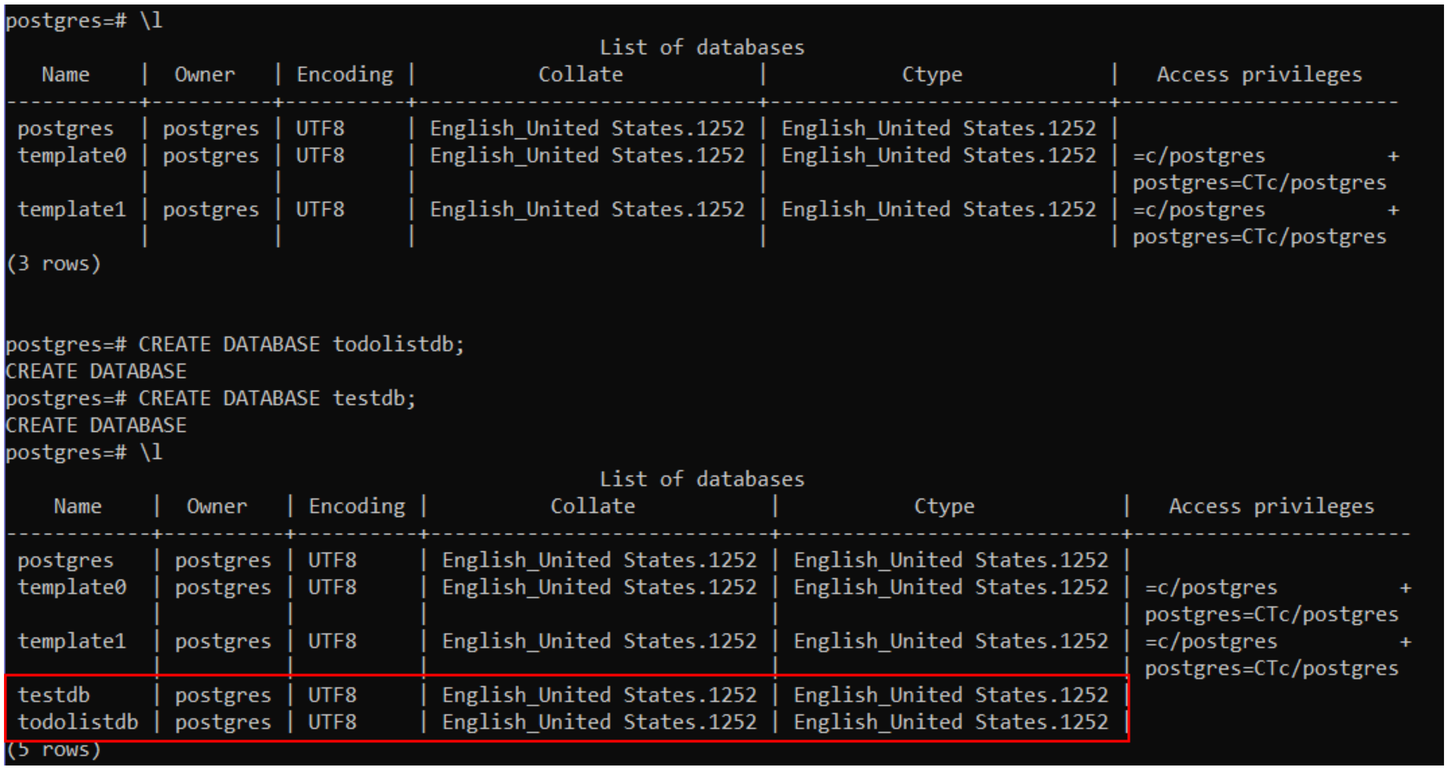
Then you can type in your password. If the password matches, you will enter your Postgresql instance. Then, you can type command `\l` to get all your databases and related information.



```
postgres=# \l
                                          List of databases
    Name    |  Owner   | Encoding |          Collate           |           Ctype            |   Access privileges
------------+----------+----------+----------------------------+----------------------------+-----------------------
 postgres   | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0  | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
            |          |          |                            |                            | postgres=CTc/postgres
 template1  | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
            |          |          |                            |                            | postgres=CTc/postgres
 testdb     | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 todolistdb | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
(5 rows)
```

If you do not have the two databases required. You can use the following SQL commands to create these two databases

```
CREATE DATABASE todolistdb;
CREATE DATABASE testdb;
```

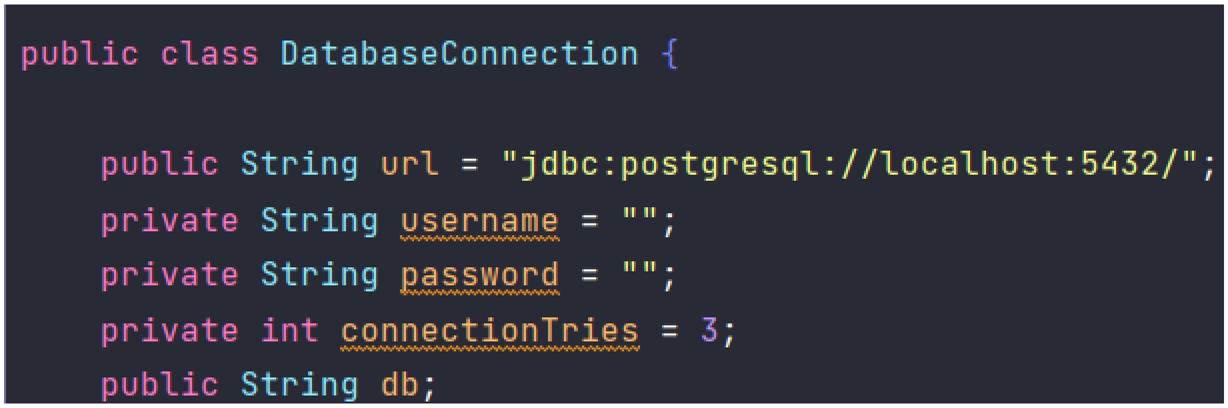


```
postgres=# \l
                                                    List of databases
   Name    |  Owner   | Encoding |          Collate           |           Ctype            |   Access privileges
-----------+----------+----------+----------------------------+----------------------------+-----------------------
 postgres  | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0 | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
           |          |          |                            |                            | postgres=CTc/postgres
 template1 | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
           |          |          |                            |                            | postgres=CTc/postgres
(3 rows)


postgres=# CREATE DATABASE todolistdb;
CREATE DATABASE
postgres=# CREATE DATABASE testdb;
CREATE DATABASE
postgres=# \l
                                                    List of databases
    Name    |  Owner   | Encoding |          Collate           |           Ctype            |   Access privileges
------------+----------+----------+----------------------------+----------------------------+-----------------------
 postgres   | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0  | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
            |          |          |                            |                            | postgres=CTc/postgres
 template1  | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
            |          |          |                            |                            | postgres=CTc/postgres
 testdb     | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 todolistdb | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
(5 rows)
```

Now, you have set up your databases.

Server Setup

First, you need to download the server program, named "**ToDoLists**", to your local machine. After opening in your preferred IDE (I am using Intellij IDEA, and the following image is taken from Intellij IDEA), you will need to go to `ToDoLists/src/main/java/edu/uwb/css533/service/db` folder, `DatabaseConnection.java` file is there. This file is responsible for connecting to databases. In this file, you need to specify the port you your postgresql is connected to, your username, and password to the instance.



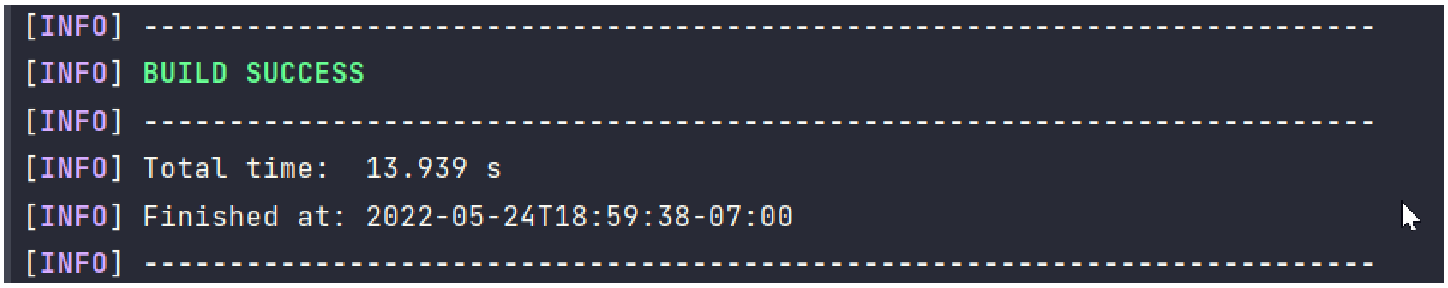
```java
public class DatabaseConnection {

    public String url = "jdbc:postgresql://localhost:5432/";
    private String username = "";
    private String password = "";
    private int connectionTries = 3;
    public String db;
```

Then, you can use the terminal to build and run your server application.

In the terminal, cd to the directory where **ToDoLists** program is located and run the following command to build the program.

```
mvn clean install
```

If you encounter any failure, you can use `mvn clean install -e` to debug. When you see the "BUILD SUCCESS" message, the build is done successfully.

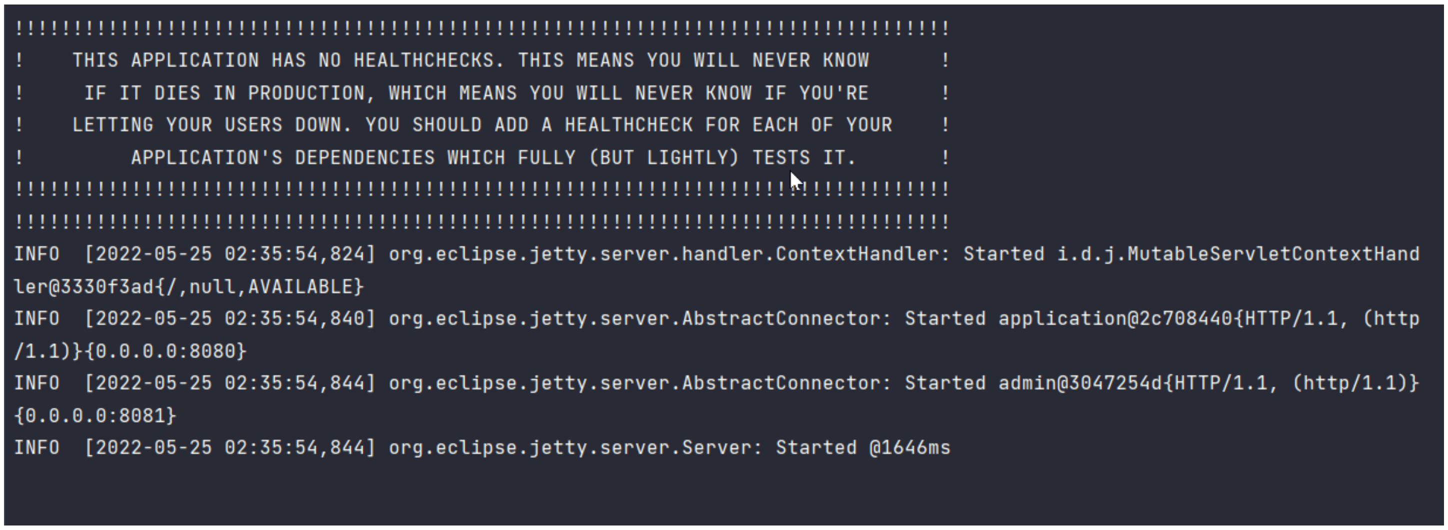


```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  13.939 s
[INFO] Finished at: 2022-05-24T18:59:38-07:00
[INFO] ------------------------------------------------------------------------
```

The next step is to run the program. You can run the following command.

```
java -jar .\target\ToDoLists-1.0-SNAPSHOT.jar server config.yml
```

If you run successfully, you should see the following message.



```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!    THIS APPLICATION HAS NO HEALTHCHECKS. THIS MEANS YOU WILL NEVER KNOW      !
!     IF IT DIES IN PRODUCTION, WHICH MEANS YOU WILL NEVER KNOW IF YOU'RE      !
!    LETTING YOUR USERS DOWN. YOU SHOULD ADD A HEALTHCHECK FOR EACH OF YOUR    !
!         APPLICATION'S DEPENDENCIES WHICH FULLY (BUT LIGHTLY) TESTS IT.       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
INFO  [2022-05-25 02:35:54,824] org.eclipse.jetty.server.handler.ContextHandler: Started i.d.j.MutableServletContextHand
ler@3330f3ad{/,null,AVAILABLE}
INFO  [2022-05-25 02:35:54,840] org.eclipse.jetty.server.AbstractConnector: Started application@2c708440{HTTP/1.1, (http
/1.1)}{0.0.0.0:8080}
INFO  [2022-05-25 02:35:54,844] org.eclipse.jetty.server.AbstractConnector: Started admin@3047254d{HTTP/1.1, (http/1.1)}
{0.0.0.0:8081}
INFO  [2022-05-25 02:35:54,844] org.eclipse.jetty.server.Server: Started @1646ms
```

Client Setup

The client program folder is named "ClientApp_mvn". After you successfully downloaded the folder, change to the directory where your downloaded program is located in the terminal. Then, similar to the server program, you need to build and then run the client program.

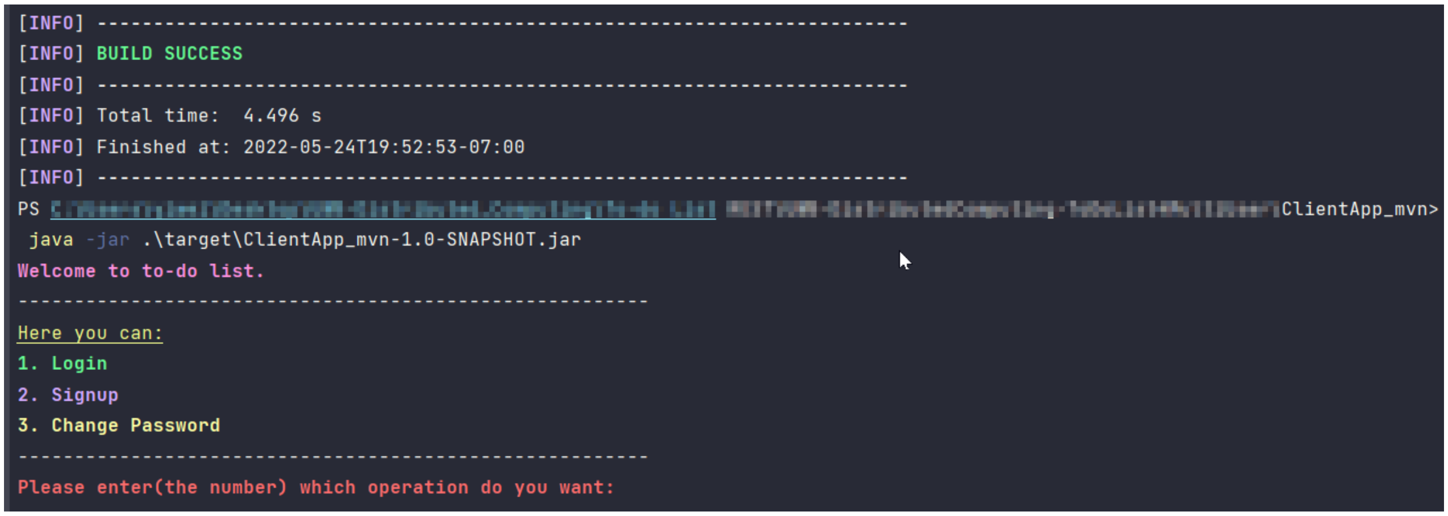
In the terminal, type

```
mvn clean install.
```

And on build success, type

```
java -jar .\target\ClientApp_mvn-1.0-SNAPSHOT.jar
```

On successfully running, now you can play with your to-do lists.



```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.496 s
[INFO] Finished at: 2022-05-24T19:52:53-07:00
[INFO] ------------------------------------------------------------------------
PS [redacted]ClientApp_mvn>
 java -jar .\target\ClientApp_mvn-1.0-SNAPSHOT.jar
Welcome to to-do list.
---------------------------------------------------------
Here you can:
1. Login
2. Signup
3. Change Password
---------------------------------------------------------
Please enter(the number) which operation do you want:
```

Instruction on executing steps in demo

Client program has clear instructions on how to do your action. In the demo, we simulate two users, user1 and user2. The following is the executing steps in the demo sequentially.

User 1

- Sign up as “user1”, type your password
- Get all your lists - it should show empty
- Log out

User 2

- Sign up as “user2”

User 1

- Change password
- Log in with new password
- Add a list - demo1 (id = 1)
- Add a list - demo2 (id = 2)
- Add a list - demo3 (id = 3)
- Enter demo list (id = 1)
- Get all user - [user1]
- Grant access to “user3” - no such user
- Check “user2” access to listid = 1 - no access
- Grant access to “user2” - succeed
- Check “user2” access to listid = 1 - can access
- Get all users - [user1, user2]
- Get all task - empty
- Add a task - demo user service, (taskid = 1)
- Add a task - demo lists service, (taskid = 2)
- Add a task - demo task service, (taskid = 3)

- Enter into task with taskid = 1
- Update task content
- Update task status
- Display details of this task

User 2

- Get all lists
- Enter listid = 1
- Get all tasks of listid = 1
- Delete a taskid = 1
- Get all tasks
- Delete all tasks

User 1

- Display details of this task - error no such task
- Back to ToDoList main menu
- Enter into demo2 (listid = 2)
- Grant “user2” access to demo2 (listid = 2)
- Back to ToDoList main menu
- Enter into demo3 (listid = 3)
- Grant “user2” access to demo3 (listid = 3)
- Get all lists

User 2

Currently in ToDoList 1

- Get all tasks
- Get all users
- Remove “user1” access
- Get all users
- Back to ToDoList main menu
- Delete list = 1
- Get all lists

User 1

- Get all lists - list 1 is missing

User 2

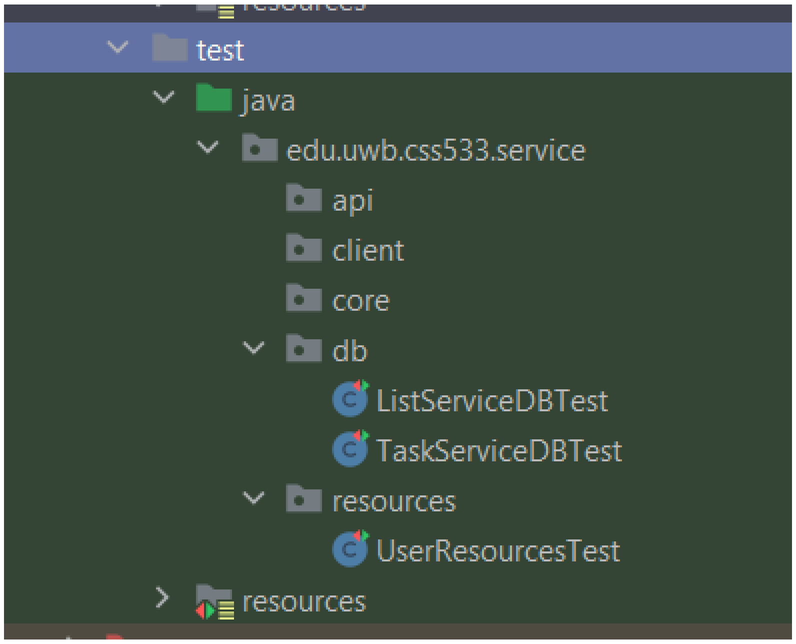
- Delete all lists
- Get all lists
- Log out

User 1

- Get all lists - no lists
- Log out

Integration test instruction

The integration tests for this application are stored in `ToDoLists/src/test/java/edu/uwb/css533/service/db` and `ToDoLists/src/test/java/edu/uwb/css533/service/resources` folders.

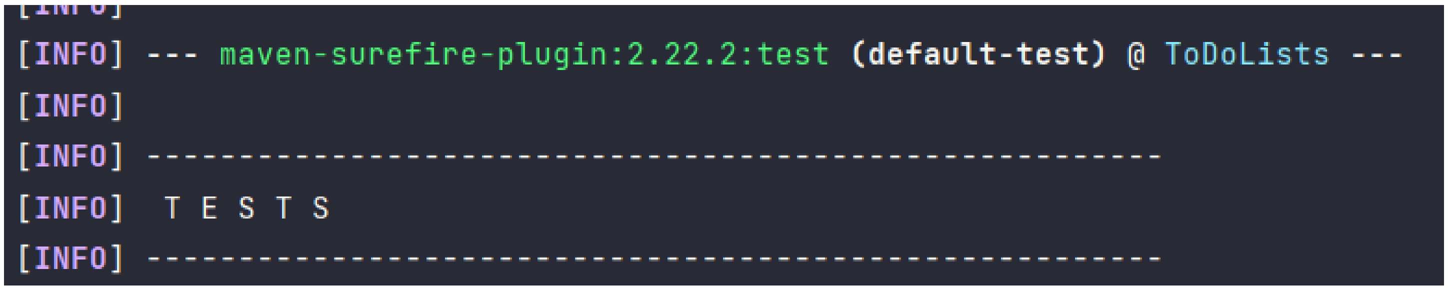


```
test
  java
    edu.uwb.css533.service
      api
      client
      core
      db
        ListServiceDBTest
        TaskServiceDBTest
      resources
        UserResourcesTest
  resources
```

To run the test, open the terminal and change the directory to where the ToDoLists folder is located, and run the following command.

```
mvn test
```

Then you will see the following message, indicating the test has started.



```
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ ToDoLists ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
```