ITD

POLITECNICO DI MILANO 1863

Title: DREAM - Data-dRiven PrEdictive FArMing in Telangana

Description: Based on the project document from teachers

Document version: 1.0.2

Authors: Luca Bertelli, Matteo Savino, Giacomo Vinati

Release Date: 06/02/2022

Table of contents

1. INTRODUCTION

1.1 Description

The Implementation and Testing document purpose is to provide a thorough description of both the coding steps, the code structure and the testing activities, executed accordingly with the DD document.

Only part of the project functions have been fully developed. Mainly, we focused on policy makers and farmers functionalities by completing their most critical requirements.

In addition, this document aims to show the structure of the code and the setup used to run the project along with the frameworks used and a simple guide to run it.

1.2 Definitions, Acronyms, Abbreviations

1.2.1 Definitions

Terms	Definition
Farmer	A person who owns or manages a farm.
Policy maker	A person responsible for or involved in formulating policies.
Agronomist	An expert in the science of soil management and crop production.
Weather Forecast	An analysis of the state of the weather in an area with an assessment of likely developments.
Information	Facts provided or learned about something or someone.
Best practices	Professional procedures that are accepted or prescribed as being correct or most effective.
Problem	A matter or situation regarded as unwelcome or harmful and needing to be dealt with and overcome.
Discussion Forum	A web page where users can post comments about a particular issue or topic and reply to other users' postings.

1.2.2 Acronyms

Acronyms	Description
DREAM	**D**ata-d**R**iven Pr**E**dictive F**A**r**M**ing in Telangana

RASD	Requirements Analysis and Specification Document
DD	Design Document
IEEE	Institute of Electrical and Electronics Engineers
API	Application Programming Interface
UAT	User Acceptance Testing
JEE	Java Enterprise Edition
JPA	Java Persistence API
ORM	Object Relational Mapping
EJB	Enterprise Java Beans

1.2.3 Abbreviations

Abbreviations	Description
PC	Computer
IT	Information technologies
e.g.	Exempli gratia
w.r.t.	With reference to
NB	Note well
Webapp	Web Application
Enum	Enumeration

1.3 Revision history

Version	Date	Description
0.0	12/01/2022	Project opening.
1.0	04/02/2022	First release.
1.1	06/02/2022	Fixed typos.

1.4 Document Structure

The document is divided into eight parts.

- **INTRODUCTION**: It gives an overview on the purpose and scope of the document along with the definitions, acronyms, and abbreviations of the most used terms. It also contains the revision history and the reference documents to better underline how it has been developed.
- **IMPLEMENTATION:** It describes the developed functions implemented in the software and gives an overview on the most critical choices.
- **FRAMEWORK AND TECHNOLOGIES:** It gives an outline of the adopted frameworks and technologies along with the reasons that brought us to use them.
- **CODE STRUCTURE:** It presents a description of the code organization and how files have been arranged in the project.
- **TESTING:** It describes the testing approaches used to test the project and the input values inserted to evaluate the correctness of the web application.
- **INSTALLATION INSTRUCTIONS:** It shows the necessary steps to setup and run the web application along with the corresponding tests.
- **EFFORT SPENT:** It includes information about the number of hours each group member has worked for the development of the document.
- **REFERENCES:** It contains the references used for coding the web application, testing it and writing the document.

2. IMPLEMENTATION

We implemented part of the main functionalities to fulfill the most meaningful requirements and create a working deliverable. Mainly, we focused on the development of the Policy Maker and Farmer functionalities.

All the Policy Maker and Farmer main functionalities have been developed with exception of the Request page. In particular, the following main functionalities were implemented: forum management, production data management and aggregate statistics management for Farmers and then, aggregate statistics and advanced statistics for Policy Makers. We decided to not implement the Request section for farmers since it's very similar to the Forum page and thus such implementation would not bring any great additions for the demonstration.

3. FRAMEWORKS AND TECHNOLOGIES

The project is developed in the Java programming language and it adopts the Java Enterprise Edition framework. Moreover, we adopted the JPA specification of JEE, which uses an ORM approach to bridge the gap between an object oriented view of the data layer and a relational database. The web interfaces offered to the final users were created through Java servlets.

We used the Tomee server which is an evolution of the classic Tomcat application server along with the EclipseLink JPA provider which is a comprehensive open source Java persistence solution. The database which communicates with JPA is a MySQL database.

We adopted these frameworks or technologies for multiple reasons. First of all, they are all open source and widely spread, thus they are provided with a ton of documentation and support. Secondly, the JEE framework allows an easy implementation of multi-tiered architectures as the DREAM project required. Besides, JEE allows the use of EJB containers which automatically handle the management of transactions in the application.

Finally, we used the thymeleaf framework to inject the data into the web pages together with Chart.js framework to plot the diagrams in the home pages. We adopted these frameworks because they are easy to use, well documented and plenty of custom setups.

4. CODE STRUCTURE

The code is divided into 3 main parts: 2 java projects, as shown in the images below and a testing file which is a simple python program that is used for testing the servlets. In this section we focus more on the java projects, more information on the python file can be found in Section 5.
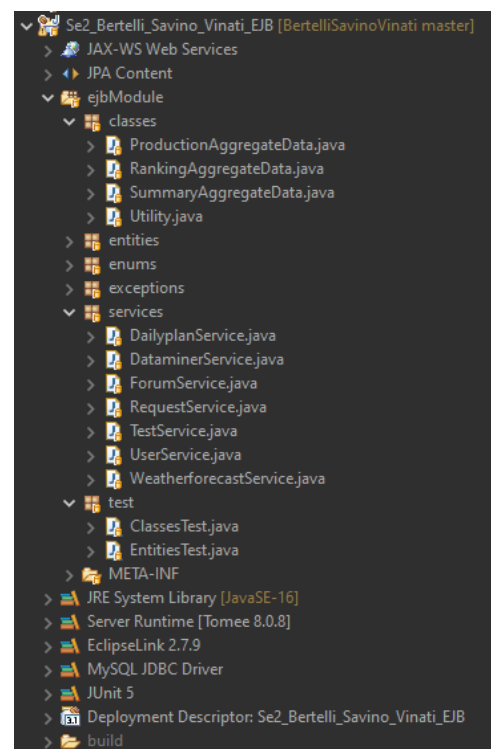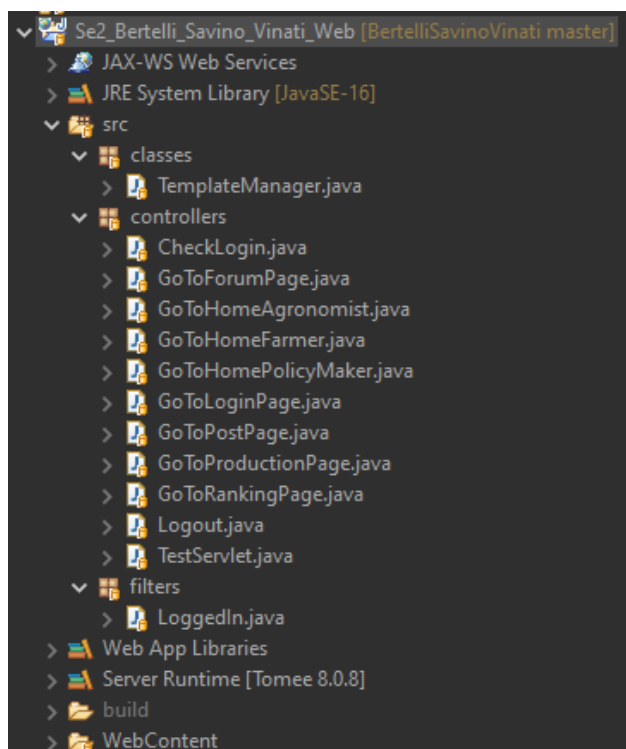
The web project contains:
- **The controllers**: which are the endpoints of the web server and provide the functionalities to the end user (implemented using servlets).

- **The TemplateManager**: which is simply a utility class that helps manage the usage of the Thymeleaf framework.
- **The LoggedIn filter**: which is a filter class that intercept the user calls to the endpoints and check whether the user is currently logged in or not. If the user is logged in it passes the call along the chain to the endpoint otherwise it redirects him to the login page.
- **The WebContent folder**: which contains all the html and css files.

The EJB project contains:
- **The entities**: which are the classes that represent the ORM of the database.
- **The services**: which are the classes that contain the business logic of the project. In the DD they were called "Managers".
- **The exceptions**: which are the exceptions used in the services.
- **The enums**: which are enum classes that specify a categorization of some values in the database.
- **The AggregateData classes**: which are classes that are used to pre digest some data before injecting it into web pages through thymeleaf.
- **The Utility class**: which contains some utility methods useful in many situations and called multiple times in the services classes.

5. TESTING

5.1 Test Plan

5.1.1 Unit Tests

Concurrently with the implementation phase, white box Unit Testing was performed in order to test individual components using JUnit version 5. Through a white box approach, the tester had a complete overview of the code. This testing phase was partially automated. The next portion of the document shows the test plan. We have chosen to carry out unit tests, costly in terms of resources, only for the entities and for the support classes in the EJB project. Below it's described the test plan for the two class groups:

5.1.1.1 Entities

Answer
1. *AnswerCommentTest* - test for setting and getting a Comment in an Answer object;
2. *AnswerIdTest* - test for setting and getting a Id in an Answer object;
3. *AnswerDateHourTest* - test for setting and getting a date in an Answer object;
4. *AnswerRequestTest* - test for setting and getting a request in an Answer object;
5. *AnswerUserTest* - test for setting and getting a User in an Answer object;

Area
6. *AreaIdTest* - test for setting and getting an Id in an Area object;
7. *AreaForecastTest* - test for setting and getting a Forecast list object in an Area object;

8. ***AreaFarmTest*** - test for setting and getting a Farm Object in Area object;
9. ***AreaNameTest*** - test for setting and getting a name in an Area object;
10. ***AreaUserTest*** - test for setting and getting a User object in an Area object;

Discussion
11. ***DiscussionIdTest*** - test for setting and getting an Id in a Discussion object;
12. ***DiscussionDatehourTest*** - test for setting and getting a Date in a Discussion object;
13. ***DiscussionTitleTest*** - test for setting and getting a Title in a Discussion object;
14. ***DiscussionPostTest*** - test for setting and getting a list of Post objects in a Discussion object;
15. ***DiscussionUserTest*** - test for setting and getting a user object in a Discussion object;
16. ***DiscussionAddPostTest*** - test for add a post to an existing list of Post objects in a Discussion object;

Farm
17. ***FarmAreaTest*** - test for setting and getting an Area object in a Farm object;
18. ***FarmIdTest*** - test for setting and getting an Id in a Farm object;
19. ***FarmAddressTest*** - test for setting and getting an address in a Farm object;
20. ***FarmDimensionTest*** - test for setting and getting a dimension value not equal to 0 in a Farm object;
21. ***FarmDimensionTest2*** - test for setting and getting a dimension value equal to 0 in a Farm object;
22. ***FarmHumidityTest*** - test for setting and getting a list of HumidityOfSoil objects in a Farm object;
23. ***FarmProductionTest*** - test for setting and getting a list of Production objects in a Farm object;
24. ***farmProductionAmountM2Test*** - test for calculate the Production amount normalized for farm dimension in a Farm object from a date;
25. ***FarmWaterConsumptionTest*** - test for setting and getting a list of WaterConsuption objects in a Farm object;
26. ***FarmWaterConsumptionM2Test*** - test for calculate the cumulative water usage in a Farm object from a date;
27. ***FarmWaterConsumptionM2NormTest*** - test for calculate the cumulative water usage, normalized for farm dimension, in a Farm object from a date;
28. ***FarmWaterConsumptionM2NormTest2*** - test for calculate the cumulative water usage, normalized for farm dimension, in a Farm object from a date;

Production
29. ***ProductionAmountTest*** - test for setting and getting the amount of Production in a Production object;
30. ***ProductionCommentTest*** - test for setting and getting the comment of a Production in a Production object;
31. ***ProductionFarmTest*** - test for setting and getting the Farm of a Production in a Production object;
32. ***ProductionIdTest*** - test for setting and getting the Id of a Production in a Production object;

33. ***ProductionTypeOfProductTest*** - test for setting and getting a TypeOfProduct enum of a Production in a Production object;
34. ***ProductionDateTest*** - test for setting and getting the Production date in a Production object;

Post

35. ***PostIdTest*** - test for setting and getting the Id in a Post object;
36. ***PostCommentTest*** - test for setting and getting an Id in a Post object;
37. ***PostDiscussionTest*** - test for setting and getting a Discussion object in a Post object;
38. ***PostUserTest*** - test for setting and getting a User object in a Post object;
39. ***PostDatehourTest*** - test for setting and getting a date in a Post object;

Forecast

40. ***ForecastAreaTest*** - test for setting and getting an area in a Forecast object;
41. ***ForecastValueTest*** - test for setting and getting rain value in a Forecast object;
42. ***ForecastCreationDateTest*** - test for setting and getting a creation date in a Forecast object;
43. ***ForecastTest*** - test for setting and getting a date in a Forecast object;
44. ***ForecastClassificationTest*** - test for setting and getting a classification enun in a Forecast object;

WaterConsumption

45. ***WaterConsumptionIdTest*** - test for setting and getting an Id in a WaterConsumption object;
46. ***WaterConsumptionDateTest*** - test for setting and getting a date in a WaterConsumption object;
47. ***WaterConsumptionAmountTest*** - test for setting and getting the amount of water used in a WaterConsumption object;
48. ***WaterConsumptionFarmTest*** - test for setting and getting a Farm object in a WaterConsumption object;

User

49. ***UserIdTest*** - test for setting and getting an Id in a User object;
50. ***UserMailTest*** - test for setting and getting a Mail in a User object;
51. ***UserFarmTest*** - test for setting and getting a Farm Object in a User object;
52. ***UserNameTest*** - test for setting and getting a name in a User object;
53. ***UserPasswordTest*** - test for setting and getting a Password in a User object;
54. ***UserSurnameTest*** - test for setting and getting a Surname in a User object;
55. ***UsertypeTest*** - test for setting and getting a User type enum in a User object;
56. ***UserCompareToTest*** - test for compareTo method with mock data between two User objects;

HumidityOfSoil

57. ***HumidityOfSoilIdTest*** - test for setting and getting the Id in a HumidityOfSoil object;
58. ***HumidityOfSoilDateTest*** - test for setting and getting the Date in a HumidityOfSoil object;
59. ***HumidityOfSoilFarmTest*** - test for setting and getting a Farm Object in a HumidityOfSoil object;
60. ***HumidityOfSoilClassificationTest*** - test for setting and getting a ClassificationH enum in a HumidityOfSoil object;

Utility

61. ***UtilityRoundTestStd*** - rounding test of a non-zero double variable to the second digit;
62. ***UtilityRoundTestZero*** - rounding test of a zero double variable to the second digit
63. ***UtilityCalculateEntropyTest1Std*** - entropy computation test for a finite set of double values;
64. ***UtilityCalculateEntropyTest2AllZero*** - entropy computation test for a finite set of zero double values;
65. ***UtilityCalculateEntropyTest3Paired*** - entropy computation test for a finite set of two repeated double values;
66. ***UtilityCalculateEntropyTest4MinEntropy*** - entropy computation test for a finite set of identical double values;
67. ***UtilityCompareTest1LessThan*** - comparison test of two double values in ascending order;
68. ***UtilityCompareTest2MoreThan*** - comparison test of two double values in descending order;
69. ***UtilityCompareTest3Equal*** - comparison test of two identical double values;
70. ***UtilityCompareTest4Zero*** - comparison test of two zero double values.

5.1.2 Integration and System Test

The integration test for the system was performed with a bottom-up approach from the entities to the web pages and the controllers of the application. Hence, the test on the controllers made it possible to evaluate the cascade of calls to services and to the database, using JPA calls. Below it's described the test plan for the controller:

1. ***testConnectionCheckLogin*** - Connection test Login Page
2. ***testLoginNoPassword*** - login failed No password test
3. ***testLoginNoMail*** - login failed No mail test
4. ***testLoginFailedWrongPassword*** - login failed wrong password test
5. ***testLoginFailedWrongMail*** - login failed wrong mail test
6. ***testLoginFarmer*** - login for farmer test
7. ***testLoginPolicyMaker*** - login for Policy Maker test
8. ***testConnectionGoToLoginPage*** - Connection test GoToLoginPage
9. ***testRedirectToLoginPage*** - redirect to Login Page test
10. ***testGoToRankingPageNoDirectConnection*** - no direct connection to GoToRankingPage test
11. ***testGoToRankingPageConnection*** - connection to GoToRankingPage with session cookie test
12. ***testRankingFilterSelection1*** - change page after filter of GoToRankingPage with session cookie test

13. ***testRankingFilterSelection2*** - change area of interest of GoToRankingPage with session cookie test
14. ***testRankingFilterSelection3*** - change area of interest (1) of GoToRankingPage with session cookie test
15. ***testRankingFilterSelection4*** - change area of interest (2) of GoToRankingPage with session cookie test
16. ***testRankingFilterSelection5*** - change order of records of GoToRankingPage with session cookie test
17. ***testRankingFilterSelection6*** - change number of records of GoToRankingPage with session cookie test
18. ***testRankingFilterSelection7*** - change period of interest of GoToRankingPage with session cookie test
19. ***testGoToHomePolicyMakerNoDirectConnection*** - no direct connection to GoToHomePolicyMaker test
20. ***testGoToHomePolicyMakerConnection*** - connection GoToHomePolicyMaker with session cookie test
21. ***testBestAreaGoToHomePolicyMaker*** - selection of best area GoToHomePolicyMaker with session cookie test
22. ***testWorstAreaGoToHomePolicyMaker*** - selection of worst area GoToHomePolicyMaker with session cookie test
23. ***testBestFarmerGoToHomePolicyMaker*** - selection of best farmer GoToHomePolicyMaker with session cookie test
24. ***testWorstFarmerGoToHomePolicyMaker*** - selection of worst farmer GoToHomePolicyMaker with session cookie test
25. ***testGoToHomeFarmerNoConnection*** - no direct connection to HomeFarmer page test
26. ***testGoToHomeFarmerConnection*** - GoToHomeFarmerPage connection with session cookie test
27. ***testGoToHomeFarmerGetRightMonthProductionAmount*** - GoToHomeFarmer get Right Month Production Amount session cookie test
28. ***testGoToHomeFarmerGetRightWaterConsumptionAmount*** - GoToHomeFarmer get Right Month Water Consumption Amount session cookie test
29. ***testGoToHomeFarmerGetRightHumidityOfSoil*** - GoToHomeFarmer get Right HumidityOfSoil session cookie test
30. ***testGoToHomeFarmerGetWeatherForecast*** - GoToHomeFarmer get Right Weather Forecast session cookie test
31. ***testGoToProductionPageNoDirectConnection*** - no direct connection to Production page test
32. ***testGoToProductionPageConnection*** - GoToProductionPage connection with session cookie test
33. ***testGoToProductionPageInsertProduction*** - GoToProductionPage insert data session cookie test
34. ***testGoToProductionPageInsertProductionWrongVegetable*** - GoToProductionPage not insert data with wrong vegetable session cookie test
35. ***testGoToProductionPageInsertProductionNoDate*** - GoToProductionPage insert data without date session cookie test
36. ***testGoToProductionPageInsertProductionNoVoidAmount*** - GoToProductionPage insert data only with amount >0 session cookie test

37. ***testGoToProductionPageInsertProductionNoVoidComment*** - GoToProductionPage insert data only with no void comment session cookie test
38. ***testGoToForumPageNoDirectConnection*** - no direct connection to Forum page test
39. ***testGoForumConnection*** - ForumPage connection with session cookie test
40. ***testGoToForumPageCreateDiscussion*** - GoToForumPage create discussion session cookie test
41. ***testGoToPostPageNoDirectConnection*** - no direct connection to Forum page test
42. ***testGoToPostConnection*** - PostPage connection to last discussion with session cookie test
43. ***testGoToPostConnection*** - PostPage creation of a Post into an existing discussion with session cookie test
44. ***testGoToPostNoVoidPostCreation*** - No creation of a void Post into an existing discussion with session cookie test
45. ***testGoToPostNoVoidPostCreation*** - get Error for non existing discussion with session cookie test

5.3 User acceptance testing

The user acceptance test will be carried out by an external group of users to verify the software system before moving the software application to the production environment.

5.2 Test Results

5.2.1 Unit Test results

The unit test was performed through the Junit version 5 package. For this purpose, two special test classes have been created: EntietiesTest.java and ClassesTest.java. The summary of a test session is collected in the following figure.
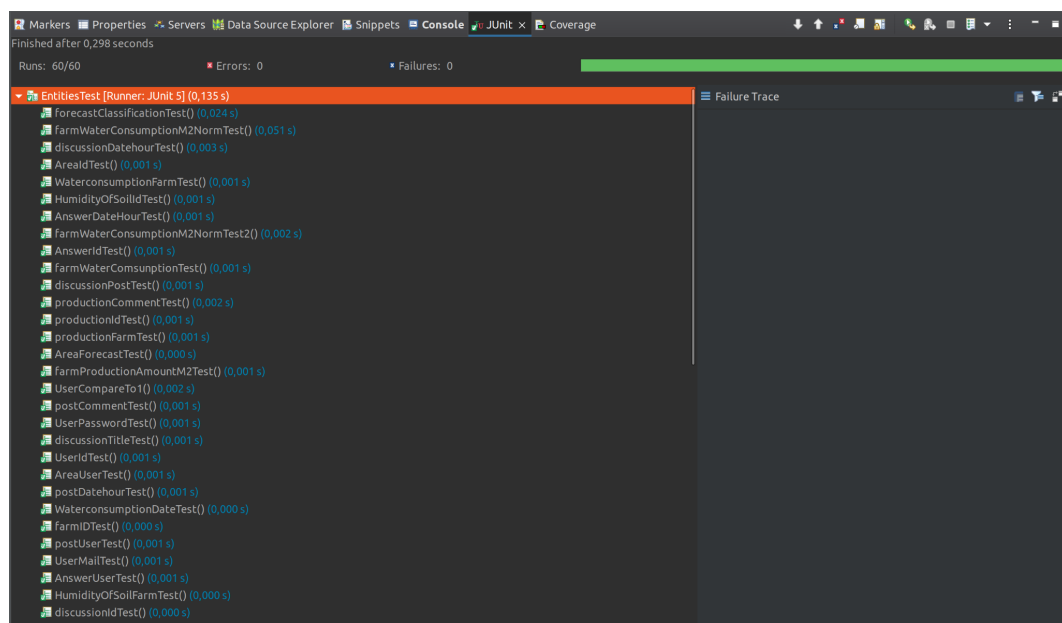
5.2.2 Integration and system Test results

The Integration test was performed through a script created with Python language, version 3.8.10. For this purpose, two main packages were used: Mysql.connector and PyTest. Moreover, a Mysql mock database was created and used. The script retrieved mock data directly from the database to check the correctness of the data shown from the main application running on a Tomcat server. The summary of a test session is collected in the following figure.

```
testController.py::testConnectionCheckLogin PASSED                                      [  2%]
testController.py::testLoginNoPassword PASSED                                           [  4%]
testController.py::testLoginNoMail PASSED                                               [  6%]
testController.py::testLoginFailedWrongPassword PASSED                                  [  8%]
testController.py::testLoginFailedWrongMail PASSED                                      [ 11%]
testController.py::testLoginFarmer PASSED                                               [ 13%]
testController.py::testLoginPolicyMaker PASSED                                          [ 15%]
testController.py::testConnectionGoToLoginPage PASSED                                   [ 17%]
testController.py::testRedirectToLoginPage PASSED                                       [ 20%]
testController.py::testGoToRankingPageNoDirectConnection PASSED                         [ 22%]
testController.py::testGoToRankingPageConnection PASSED                                 [ 24%]
testController.py::testRankingFilterSelection1 PASSED                                   [ 26%]
testController.py::testRankingFilterSelection2 PASSED                                   [ 28%]
testController.py::testRankingFilterSelection3 PASSED                                   [ 31%]
testController.py::testRankingFilterSelection4 PASSED                                   [ 33%]
testController.py::testRankingFilterSelection5 PASSED                                   [ 35%]
testController.py::testRankingFilterSelection6 PASSED                                   [ 37%]
testController.py::testRankingFilterSelection7 PASSED                                   [ 40%]
testController.py::testGoToHomePolicyMakerNoDirectConnection PASSED                     [ 42%]
testController.py::testGoToHomePolicyMakerConnection PASSED                             [ 44%]
testController.py::testBestAreaGoToHomePolicyMaker PASSED                               [ 46%]
testController.py::testWorstAreaGoToHomePolicyMaker PASSED                              [ 48%]
testController.py::testBestFarmerGoToHomePolicyMaker PASSED                             [ 51%]
testController.py::testWorstFarmerGoToHomePolicyMaker PASSED                            [ 53%]
testController.py::testGoToHomeFarmerNoConnection PASSED                                [ 55%]
testController.py::testGoToHomeFarmerConnection PASSED                                  [ 57%]
testController.py::testGoToHomeFarmerGetRightMonthProductionAmount PASSED               [ 60%]
testController.py::testGoToHomeFarmerGetRightWaterConsumptionAmount PASSED              [ 62%]
testController.py::testGoToHomeFarmerGetRightHumidityOfSoil PASSED                      [ 64%]
testController.py::testGoToHomeFarmerGetWeatherForecast PASSED                          [ 66%]
testController.py::testGoToProductionPageNoDirectConnection PASSED                      [ 68%]
```

```
testController.py::testGoToProductionPageConnection PASSED                              [ 71%]
testController.py::testGoToProductionPageInsertProduction PASSED                        [ 73%]
testController.py::testGoToProductionPageInsertProductionWrongVegetable PASSED          [ 75%]
testController.py::testGoToProductionPageInsertProductionNoDate PASSED                  [ 77%]
testController.py::testGoToProductionPageInsertProductionNoVoidAmount PASSED            [ 80%]
testController.py::testGoToProductionPageInsertProductionNoVoidComment PASSED           [ 82%]
testController.py::testGoToForumPageNoDirectConnection PASSED                           [ 84%]
testController.py::testGoForumConnection PASSED                                         [ 86%]
testController.py::testGoToForumPageCreateDiscussion PASSED                             [ 88%]
testController.py::testGoToPostPageNoDirectConnection PASSED                            [ 91%]
testController.py::testGoToPostConnection PASSED                                        [ 93%]
testController.py::testGoToPostNewPostCreation PASSED                                   [ 95%]
testController.py::testGoToPostNoVoidPostCreation PASSED                                [ 97%]
testController.py::testNoConnectionToNonExistingDiscussion PASSED                       [100%]

=========================== 45 passed in 5.78s ===========================
```

6. INSTALLATION INSTRUCTIONS

6.1 Windows User

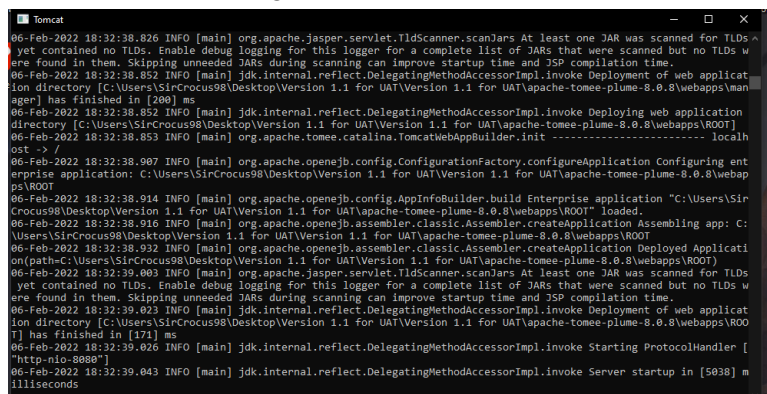Follow these steps to run the DREAM project:
- ❖ DATABASE:
  - ● Download the MySql installer version 8.0.26 (important) from here.
  - ● Start the downloaded installer and accept all default configurations and steps to install missing prerequisite packages. Remember to store the root

password, it'll be required in following steps. By default the MySQL service is launched at startup.
- (Optional: Instead of using the root account, another one can be created.)
- Open MySQL Workbench and check that everything works fine.

❖ PROJECT
- Download the zip folder "Version 1.1 for UAT.zip" from the I&T folder.
- Unzip it and open it.
- Right click the DB settings link and open it with a text editor of your choice (Notepad is ok).
- Modify username and password with your own credentials from the DATABASE setup. Then, save and close.
- Open MySQL Workbench, access the account used in the previous step, open a new Query Tab and drag into it the file "dumpForUAT.sql".
- Execute the query by clicking on the lightning icon. If everything has been completed successfully you can close MySQL Workbench.
- Return to the unzipped folder.
- Double click on the "RUN for Windows" link (a prompt window should appear like the one in the image below).

- Open a browser of your choice between Chrome, Edge, Firefox or Brave.
- Write in the URL bar http://localhost:8080/Se2_Bertelli_Savino_Vinati_Project
- Enjoy the DREAM application.

6.2 Linux

Follow these steps to run the DREAM project:
❖ DATABASE (mySql + MySQL Workbench installation):
- Open a terminal*.
- Execute the following commands:
  sudo apt update
  sudo apt install mysql-server
  sudo mysql_secure_installation
  sudo apt install mysql-workbench-community
  mysql-workbench

14

- During the installation you will be prompted to provide a password for the root user of the sql server, remember the password and store it in a safe and protected place.
- Select server>data import.
- Select Import from Self-Contained File.
- Select the file dumpForUAT.sql from the unzipped folder.
- Click Start Import.
- Enjoy the database.

*Alternatively Install MySQL and MySQL Workbench using Deb packages in Debian-like distros

❖ PROJECT
  - Download the zip folder "Version 1.1 for UAT.zip" from the I&T folder.
  - Unzip the file.
  - Open a terminal.
  - Navigate to the folder: /apache-tomee-plume-8.0.8/conf.
  - Open the tomee.xml file (e.g. *nano tomee.xml*).
  - Change username and the password with the ones of mysql server.
  - Navigate to the folder: /apache-tomee-plume-8.0.8/bin.
  - Execute the following commands from terminal:
    chmod +x catalina.sh
    chmod +x startup.sh
    chmod +x shutdown.sh
  - Start the server with the command:
    ./startup.sh
  - Open a browser of your choice between Chrome, Edge, Firefox and Brave.
  - Write in the URL bar http://localhost:8080/Se2_Bertelli_Savino_Vinati_Project
  - Enjoy the DREAM application.

6.3 MacOS

Follow these steps to run the DREAM project:
❖ DATABASE:
  - Download the MySQL Community server 8.0.26 (important) from [here](#):
    Choose the right version (ARM or x86).
  - Download the MySQL Workbench 8.0.26 (important) from [here](#).

  MySQL Community:
  - Install MySQL Community Server with all default configurations.
  - When a password is required insert one. Be sure to store the root password in a safe and protected place.

  MySQL Workbench:
  - Open the .dmg and move the MySQL workbench to the Applications Folder (depending on your OS X version you might need to install an older version).

- After installing MySQL Community and MySQL Workbench, open a terminal and type:
  ```
  sudo ln -s
  /Applications/MySQLWorkbench.app/Contents/MacOS/mysqldum
  p /usr/local/bin/mysqldump
  ```
- If when you open MySQL workbench for the first time you encounter a "security issue", go to Settings-> Security Open anyway.
- Open System Preferences and select MySQL. Clicking on the Start MySQL Server button will cause the MySQL server to start. (If it doesn't appear automatically after MySQL installation try by restarting the computer).

❖ PROJECT
- Download the zip folder "Version 1.1 for UAT.zip" from the I&T folder.
- Unzip the file.
- Open a terminal.
- Navigate to the folder: /apache-tomee-plume-8.0.8/conf.
- Open the tomee.xml file (e.g. *nano tomee.xml*).
- Change username and the password with the ones of mysql server.
- Navigate to the folder: /apache-tomee-plume-8.0.8/bin.
- Execute the following commands from terminal:
  chmod +x catalina.sh
  chmod +x startup.sh
  chmod +x shutdown.sh
- Start the server with the command:
  ./startup.sh
- Open a browser of your choice between Chrome, Edge, Firefox and Brave.
- Write in the URL bar http://localhost:8080/Se2_Bertelli_Savino_Vinati_Project
- Enjoy the DREAM application.

7. EFFORT SPENT

5.1 Teamwork

Task	Hours
Initial briefing	1
Brainstorming	2
Implementation (Include coding)	10
Frameworks	0.5
Code Structure	0.5
Testing (Include coding)	5

Task	
Installation Instructions	2

5.2 Luca Bertelli

Task	Hours
Implementation (Include coding)	10
Frameworks	1
Code Structure	0.5
Testing (Include coding)	23

5.3 Matteo Savino

Task	Hours
Introduction	1
Implementation (Include coding)	30
Frameworks	1
Code Structure	1
Testing (Include coding)	5

5.4 Giacomo Vinati

Task	Hours
Implementation (Include coding)	28
Frameworks	0.5
Code Structure	0.5
Testing (Include coding)	7

8. REFERENCES

- Slides of the lectures of SE2
- Slides of the lectures of DB2

- Slides of the lectures of TIW
- Telangana website: [Telangana State Development Planning Society](#)
- [EclipseLink - JPA](#)
- [Chart.js](#)
- [Apache TomEE](#)
- [Thymeleaf](#)