DREAM

Data-dRiven PrEdictive FArMing in Telangana

Acceptance Testing Document - ATD

POLITECNICO
MILANO 1863

Christian Grasso - Filippo Lazzati - Chiara Magri

Year: 2021/2022

Contents

1 Introduction

In every software engineering project, the process of `verification` and `validation` is essential in order to detect defects and errors. As a matter of fact, there are many different kinds of "errors", ranging from human errors, to system errors, to system failures. This is why every software has to undergo a complete and exhaustive process of `verification` and `validation` not only at the end of the project, but along all the steps of the development process.
In this document the main focus is the validation performed through the acceptance testing of the implemented functionalities.

2 Project to analyze

The project we have to analyze is that of **Federica Tommasini** (federica.tommasini@mail.polimi.it) and **Alessandro Verosimile** (alessandro.verosimile@mail.polimi.it). Their repository can be found on `Github` at federicatommasini/TommasiniVerosimile.git.
The project provides a server implemented in `Java` through the Spring framework and an `iOS` application as client for the farmers.

3 Installation setup

Backend We have been able to follow the instructions provided correctly; they are clear and correct. However, according to the installation instructions, a IDE should be used to start the backend server; this is not a good way to distribute the application, a `JAR` file or at least a shell command would have been better for this task.

Mobile app We have been able to build the application following the instructions and run it in an emulator. We were not able to test it in a real device because none of us own an `iOS` device.

4 Acceptance test cases

The group has implemented the following functionalities (see *RASD - Product functions* as well as *ITD - Implemented functionalities*):

- **Registration**: the process of registration is clear and intuitive, and is coherent with what inserted in the documents. However, we have to warn about some things:

  1. if we try to insert a password of only numbers, the application throws an exception without notifying the user about anything; this possibility was considered because the code to manage it is present, however a line of code to show an error message to the end user is missing, thus the exception is thrown;
  2. email verification is not implemented even though it is a requirement in *RASD* (see *R*4 and *R*5). Inside the *ITD* they have not mentioned this fact;
  3. some checks on the data inserted by the user are missing. For instance: if we try to insert a `string` in the form requiring the size of the farm (a number), an exception is thrown without providing any error message to the end user.

  Nevertheless, these are minor bugs, and can, of course, be neglected considering that the given software is a prototype.

- **Login**: clear and coherent with the documents;
- **Visualize meteorological forecasts**: they have decided to use a table to show the forecasts, which is simple and effective, and works as expected;
- **Exchange suggestions**: this functionality is clear and coherent with what present in the documents;
- **Visualize an evaluation**: this feature works coherently with what present in the documents. It shows a value saved in the database;
- **Update production quantities and used products**: the insertion of the products works coherently with what present in the documents. The user can insert both the tools he used for farming and the products that he has grown. However, both the tools and the products are referenced through the same word "product" (and this is not really clear). Furthermore, we found the specification a little bit lacking, as it doesn't allow to keep a track record of production over time; this makes computing aggregate data more difficult;
- **Request for help**: the basic functionality works fine, however we have noticed that every time we insert an help request in the app, it is always sent to the first user in the database. This is due to a bug in the implementation, detailed below;

- **Answer to help requests**: help requests can be answered by the receiver via the Messages function, which is implemented according to the specification;
- **Visualize messages from a policy maker**: the feature is implemented correctly from the farmer's point of view (the policy maker side was not implemented);
- **Discuss a topic**: we have noticed that:
    1. when writing a comment, saving it, then going back to the topic list and re-entering the topic page, the new comment disappears; if we leave the forum section altogether it appears again (the comment is saved in the database, however the topic entity is not reloaded when saving);
    2. the user can not visualize all the existing topics but has to enter the first characters of the topic name, which is not really convenient as other users don't know topic titles beforehand.

5 Technical observations

- Looking at the code, we see that every time we try to insert an help request in the app, it is sent always to the first user created. The server is supposed to choose a receiver randomly, however there is a bug in the implementation caused by the following expression:

  ```
  (int) Math.random()
  ```

  which is always rounded down to 0, thus the first user in the database is selected;
- the "title" and the "description" inserted into the help requests' forms are inserted into the database as a single column, and then retrieved and shown always together. This is not a good design choice, because it does not allow to show only the title without parsing;
- the code for both the backend and the app does not contain any documentation nor comments;
- there are some minor issues with code quality in the backend implementation, such as long chains of if-else statements, hardcoded constants and duplicated code (for example, SuggestionService and HelpRequestService include the same logic for farmer selection);
- user passwords are saved in plaintext in the database, which should be always avoided;
- input validation in general is lacking.

6 Conclusions

To sum up, we can list the following thoughts about the project:

- the implementation document is detailed and well-written and does a good job of explaining the adopted technical choices;
- the specification has been mostly respected, with the exception of a few non-major specific requirements (e.g. email verification, GPS location);
- most of the features are implemented correctly; there are some small bugs in the code, which are not important given that this is a prototype.

Overall, with the exception of a few minor issues, we found the implementation to be of good quality and compliant with the documents.

7 Effort spent

Table 1: Effort spent on this document.

Effort spent	
Filippo Lazzati	6h
Chiara Magri	6h
Christian Grasso	6h

8 References

8.1 Used tools

- Overleaf: Latex editor;
- Github: to share the files.