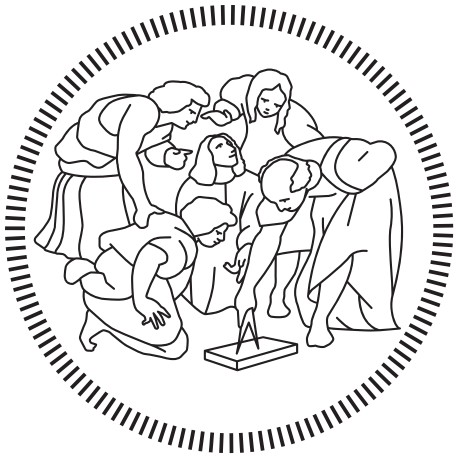

POLITECNICO DI MILANO

Computer Science and Engineering

Acceptance Test Deliverable

DREAM - Data-dRiven PrEdictive FArMing in Telangana

Software Engineering 2 Project

Academic year 2021 - 2022

February 13, 2022

Version 1.0

Authors:
Kinga Marek
Józef Piechaczek
Mariusz Wiśniewski

Professor:
Damian Andrew Tamburri

Contents

Chapter 1

Introduction

The focus of this document is to present the work done during testing of the Data-dRiven PrEdictive FArMing in Telangana (*DREAM*) application's initial prototype, implemented by another group of students. The project was carried out in accordance with the directives and instructions outlined in the *I&T assignment goal, schedule, and rules* [5]. The *Requirement Engineering and Design Project: goal, schedule, and rules* document [6] provides a full background as well as a thorough overview of the problem.

1.1 Analyzed Project

The analyzed code was developed by Marco Domenico Buttiglione, Martina Caffagnini, and Dounia Faouzi and is available in their private repository hosted on GitHub [2].

The most important works referenced when performing the acceptance testing were *Requirements Analysis and Specifications Document* [4], *Design Document* [1], and *Implementation and Test Deliverable* [3] written by the abovementioned people.

1.2 Acronyms

Acronyms	**Expression**
API	Application Programming Interface
DD	Design Document
DREAM	Data-dRiven PrEdictive FArMing in Telangana
G	Goal
ITD	Implementation and Test Deliverable
R	Requirement
RASD	Requirements Analysis and Specifications Document
SQL	Structured Query Language

U	Use case
UI	User Interface

1.3 Revision History

Date	Revision	Notes
13.02.2022	v.1.0	First release.

1.4 Reference Documents

[1] Marco Domenico Buttiglione, Martina Caffagnini, and Dounia Faouzi. *Design Document V2.0, DREAM - Data-dRiven PrEdictive FArMing in Telangana*. Feb. 2022.

[2] Marco Domenico Buttiglione, Martina Caffagnini, and Dounia Faouzi. *DREAM, Telegana farm monitoring platform*. Feb. 2022. URL: https://github.com/marticaffa/CaffagniniFaouziButtiglione (visited on 02/09/2022).

[3] Marco Domenico Buttiglione, Martina Caffagnini, and Dounia Faouzi. *Implementation and Test Deliverable V1.0, DREAM - Data-dRiven PrEdictive FArMing in Telangana*. Feb. 2022.

[4] Marco Domenico Buttiglione, Martina Caffagnini, and Dounia Faouzi. *Requirements Analysis and Specifications Document V2,0, DREAM - Data-dRiven PrEdictive FArMing in Telangana*. Feb. 2022.

[5] Elisabetta Di Nitto, Matteo Rossi, and Damian Tamburri. *A.Y. 2021-2022 Software Engineering 2 I&T assignment goal, schedule, and rules*. Oct. 2021.

[6] Elisabetta Di Nitto, Matteo Rossi, and Damian Tamburri. *A.Y. 2021-2022 Software Engineering 2 Requirement Engineering and Design Project: goal, schedule, and rules*. Oct. 2021.

Chapter 2

Installation Setup

This chapter describes in detail the steps performed to install and successfully run the initial prototype of the *DREAM* application provided. The system consists of two different parts: backend and frontend, which were installed separately.

2.1 Backend

A database and a server constituted together the backend of the system. The database was created using the *MySQL* database management system [7], whereas the server run on the *Java Virtual Machine (JVM)* and was implemented in the *Spring Boot* framework [9].

2.1.1 Server Prerequisites

1. Downloaded backend code from the backend repository. Nonetheless, it **was not** prepared in the form of a *.zip* file as specified by the assignment document.
2. Installed *IntelliJ Ultimate* from the official website.
3. Allowed the IDE to resolve all the dependencies of the backend code.
4. Built the code.

2.1.2 Database Prerequisites

1. Installed *MySQL Sever & Workbench* from the official website by accepting only community *Server*, *Workbench*, and *Connector/J* packages.
2. Set the default username and password for *MySQL* server to *"root"*. However, this **did not work** since it was not set to *"root"* inside *application.properties* file, which should be the case according to the instruction provided in the group's ITD document. Instead, it was set to *"6EG7DGkcfH8pv2"*. This required the user to manually change the password to *"root"* in the aforementioned file.
3. Created database schema called *db_dream*.

2.1.3 Backend Setup

1. Run *DreamApplication.java* inside *IntelliJ Ultimate*.
2. Executed *db_initialization.sql* inside *MySQL Workbench*.

2.2 Frontend

The frontend of the system was developed using *JavaScript* together with *ReactJS* [8].

2.2.1 Basic Setup

Opened the following links inside a web browser: farmer's app, agronomist's app. This setup was utilized to perform the acceptance testing.

2.2.2 Setup Used for Code Verification Purpose

The following setup (see the listings 2.1 and 2.2) was employed to manually build the source code, and thereby, to verify its correctness. The source code was downloaded from the frontend repository. It is important to notice that there was no instruction provided inside the group's ITD that would help with this process.

Agronomist's WebApplication

```
cd Code/UI/dream-agronomist-webapp/
npm install
npm start
```

Listing 2.1: Install dependencies and build *Agronomist's WebApplication*.

Farmer's WebApplication

```
cd Code/UI/dream-farmer-webapp/
npm install
npm start
```

Listing 2.2: Install dependencies and build *Farmer's WebApplication*.

Chapter 3

Acceptance Testing

The following chapter reminds all project details that are considered to be valuable to design the test cases. Furthermore, it elaborates in detail on how the verification was performed, showing all the scenarios and their results.

3.1 Project Specifics

Goals, use cases, as well as functional requirements that are defined throughout the RASD document are listed below. They are further exploited to define particular test cases in the process of acceptance testing. Since the functionalities were implemented to satisfy only the needs of agronomists and farmers, all the information related to the role of a policy maker is crossed out.

3.1.1 Goals

G1. Allowing farmer to share their knowledge and problems with other farmers and help them with their farm by creating and commenting a topic.

G2. Allowing farmer to keep track of their production.

G3. Showing the farmer personalized advice regarding specific subjects of their need, shared by the agronomist.

G4. Allowing farmers to make a request for help to agronomists.

G5. Allowing agronomists to help farmers in their area by answering to their request.

G6. Allowing agronomists to help farmers in their area by sharing their knowledge and giving them advice.

G7. Allowing agronomists to plan visits with farmers to check their progress and, if needed, help them.

G8. ~~Allowing policy makers to monitor the performance of the farmers to help them or to give them special incentives to encourage them to share their experience.~~ (**Not implemented.**)

G9. Showing customers weather forecasts allowing them to have a better approach with the climate changes.

3.1.2 Use Cases

U1. Show Farmer's Home Page.

U2. Show Agronomist's Home Page.

U3. Show Topic Page.

U4. Show Production.

U5. Show Farm Info.

U6. Send Request for Help.

U7. Sign Up.

U8. Log In.

U9. Create Meeting

U10. Create Knowledge.

U11. Insert Farm Data.

U12. Notification New Farmer.

U13. ~~Show Policy Maker's Home Page.~~ (**Not implemented.**)

U14. Create New Topic.

U15. Update Production Data.

U16. Comment Topic.

U17. Answer Help Request.

3.1.3 Requirements

R1. The system allows the farmer to add data concerning his production.

R2. The system allows the agronomist to view the data concerning their farmers' production.

R3. The system allows the agronomist to book an appointment with the farmer by showing all available slots.

R4. The system allows the agronomist to modify a booked appointment with the farmer.

R5. The system allows the agronomist to cancel a booked appointment with a farmer.

R6. The system check that the agronomist booked at least two appointments per year with each farmer.

R7. The system alerts the farmer before the appointment.

R8. The system alerts the agronomist before the appointment.

R9. The system allows the agronomist to view the calendar.

R10. The system allows the farmer to view the calendar.

R11. The system allows the farmer to send a help request to the agronomist.

R12. The system allows the agronomist to answer to the help requests that are sent to him.

R13. The system allows the agronomist to view all the help requests sent to him.

R14. The system allows the farmer to view all the help requests sent by him.

R15. The system allows the farmer to publish a topic on the forum.

R16. The system allows all farmers to view the forum.

R17. The system allows all farmers to publish an answer to a topic on the forum.

R18. The system allows the farmers to check the weather forecasts for their area.

R19. The system allows the agronomists to share knowledge with the farmer.

R20. The system allows the farmers to visualize the *knowledges* shared by their agronomist.

R21. The system allows all agronomists to check the weather forecasts for their area.

R22. The system allows the agronomists to publish a grade for each farmer after each visit.

R23. ~~The system allows the policy makers to view the grades of all farmers of Telangana.~~ (**Not implemented.**)

R24. The system requires a sign up and a login to access to the data.

R25. The system shows to each agronomist the list of all farmers they are responsible for.

R26. The system notifies the agronomists when a new farmer has registered under their area.

R27. The system allows the farmer to insert their position when they first register.

3.2 Test Cases

The following test cases are used in the process of acceptance testing of the initial prototype of the DREAM system. They are derived from project's use cases, which are additionally mapped to the functional requirements so that the fulfillment of both is verified. All the testing was conducted manually.

Before starting the acceptance tests, the application was seeded with data required for specified test cases. Two farmer's accounts, help requests together with responses, forum topics and comments, products and their production data, as well as meetings and knowledge items were created.

3.2.1 Show Farmer's Home Page

Use case	**U1.** Show Farmer's Home Page
Goals	G1, G2, G3, G4, G9
Requirements	R7, R18, R20
Actor	Farmer
Scenario	1. Open *Farmer's WebApplication*. 2. Click the *Sign In* button. 3. Fill in *E-Mail* and *Password* fields with credentials used in the *Sign Up* process. 4. Press the *Sign In* button.
Expected Output	The *Home Page* with weather forecast, knowledge, and planned meetings is displayed.
Test Result	Success

Table 3.1: Test case verifying U1.

3.2.2 Show Agronomist's Home Page

Use case	**U2.** Show Agronomist's Home Page
Goals	G5, G6, G7, G9
Requirements	R8, R13, R21
Actor	Agronomist

Scenario	1. Open *Agronomist's WebApplication*. 2. Click the *Sign In* button. 3. Fill in *E-Mail* and *Password* fields with credentials used in the backend setup process. 4. Press the *Sign In* button.
Expected Output	The *Home Page* with weather forecast, help requests, and planned meetings is displayed.
Test Result	Success

Table 3.2: Test case verifying U2.

3.2.3 Show Topic Pane

Use case	**U3.** Show Topic Pane
Goals	G1
Requirements	R17
Actor	Farmer
Scenario	1. Sign in to the *Farmer's WebApplication*. 2. Click the *Forum* tab.
Expected Output	The *Topic Page* containing a list of forum topics is displayed.
Test Result	Success

Table 3.3: Test case verifying U3.

3.2.4 Show Production

Use case	**U4.** Show Production
Goals	G2
Requirements	*No valid requirement specified.*
Actor	Farmer
Scenario	1. Sign in to the *Farmer's WebApplication*. 2. Click the *Production* tab.
Expected Output	The *Production Page* with farmer's production data is displayed.

Test Result	Success

Table 3.4: Test case verifying U4.

3.2.5 Show Farm Info

Use case	**U5.** Show Farm Info
Goals	G6
Requirements	*No valid requirement specified.*
Actor	Farmer
Scenario	1. Sign in to the *Farmer's WebApplication*. 2. Click the *Farm Info* tab.
Expected Output	The *Farm Info Page* containing soil, production, and sensor data is displayed.
Test Result	**Failure**

Table 3.5: Test case verifying U5.

Failure Description

There is no tab called *Farm Info* available. Such data can be seen inside *Production* tab, nonetheless, it does not follow the use case description and no information regarding that change could be found in the provided documents.

3.2.6 Send Request for Help

Use case	**U6.** Send Request for Help
Goals	G4
Requirements	R11, R14
Actor	Farmer
Scenario	1. Sign in to the *Farmer's WebApplication*. 2. Click the *Help* tab. 3. Press the *Create a new help request* button. 4. Fill in the *Subject* and *Body* fields.

Expected Output	A request is created and sent to the agronomist responsible for the given area.
Test Result	Success

Table 3.6: Test case verifying U6.

Unexpected Behavior

The case when valid data is introduced works as expected. However, the system allows creating empty help requests and no input validation is in place.

3.2.7 Sign Up

Use case	**U7.** Sign Up
Goals	G1 - G9
Requirements	R24, R26, R27
Actor	Farmer
Scenario	1. Open *Farmer's WebApplication* 2. Click the *Sign In* button. 3. Click *Don't have an account? Sign Up* button. 4. Provide *First Name, Last Name, Aadhaar, Telephone number, E-Mail, Password*. 5. Click *Sign Up* button.
Expected Output	An account is created.
Test Result	Success

Table 3.7: Test case verifying U7.

Unexpected Behavior

The system correctly checks for missing fields and existing accounts. Agronomist receives a notification about a new farmer in the area.

However, the system does not ask user for the password confirmation, as described in the use case description in the RASD. Moreover, the system does not validate the format of the inserted data.

3.2.8 Login

Use case	**U8.** Login Page
Goals	G1 - G9
Requirements	R24
Actor	Farmer, Agronomist
Scenario	1. Open *Farmer's WebApplication*/*Agronomist's WebApplication*. 2. Click the *Sign In* button. 3. Provide *E-Mail, Password*. 4. Click the *Sign In* button.
Expected Output	User is logged in and redirected to the home page.
Test Result	Success

Table 3.8: Test case verifying U8.

Additional Notes

The case was tested and resulted in *Success* on both clients' panels.

3.2.9 Create Meeting

Use case	**U9.** Create Meeting
Goals	G7
Requirements	R3, R4, R5, R6
Actor	Agronomist
Scenario	1. Sign in to the *Agronomist's WebApplication*. 2. Click on *Farmers* in the navigation top bar. 3. Click the button with three dots. 4. Choose *Plan a meeting* option. 5. Insert *Date, Start time, End time*. 6. Click the *OK* button.
Expected Output	A meeting is correctly created and displayed on the meeting list.
Test Result	Success

Table 3.9: Test case verifying U9.

Unexpected Behavior

The system correctly checks for overlapping time periods during meeting creation, however allows meeting with the same start and end time. The system does not allow creating a meeting during *working hours*, nonetheless their definition is not provided neither in the RASD nor in the DD.

3.2.10 Create Knowledge

Use case	**U10.** Create Meeting
Goals	G6
Requirements	R19
Actor	Agronomist
Scenario	1. Sign in to the *Agronomist's WebApplication*. 2. Click on *Knowledge* in the navigation top bar. 3. Provide *Title, Category, Description, Article*. 4. Click the *OK* button.
Expected Output	A Knowledge instance is correctly created.
Test Result	Success

Table 3.10: Test case verifying U10.

3.2.11 Insert Farm Data

Use case	**U11.** Insert Farm Data
Goals	G7, G8
Requirements	*No valid requirement specified.*
Actor	Farmer
Scenario	1. Open *Farmer's WebApplication* 2. Click the *Sign In* button. 3. Click *Don't have an account? Sign Up* button. 4. Provide *First Name, Last Name, Aadhaar, Telephone number, E-Mail, Password*. 5. Click *Sign Up* button. 6. Provide *Area, Address, Square kilometer*. 7. Click *Sign Up* button.

Expected Output	Farm data is updated, and the user is redirected to the home page.
Test Result	Success

Table 3.11: Test case verifying U11.

Unexpected Behavior

Input format is not validated, which makes it possible to input negative number of farm's square kilometers.

3.2.12 New Farmer Notification

Use case	**U12.** Notification New Farmer
Goals	G7
Requirements	R26
Actor	Farmer
Scenario	1. Open *Farmer's WebApplication* 2. Click the *Sign In* button. 3. Click *Don't have an account? Sign Up* button. 4. Provide *First Name, Last Name, Aadhaar, Telephone number, E-Mail, Password*. 5. Click *Sign Up* button. 6. Provide *Area, Address, Square kilometer*. 7. Click *Sign Up* button. 8. Sign in to the *Agronomist's WebApplication*.
Expected Output	There is a notification regarding a new farmer in the area in Agronomist's home page.
Test Result	Success

Table 3.12: Test case verifying U12.

3.2.13 Create New Topic

Use case	**U14.** Create New Topic
Goals	G1
Requirements	R15, R16

Actor	Farmer
Scenario	1. Log in as a farmer to the *Farmer's WebApplication*. 2. Click on *Forum* in the navigation top bar. 3. Click on the *Create new topic* button in the left bottom corner. 4. Fill the body, tag and the topic. 5. Click on the *Create* button.
Expected Output	A topic is successfully created and shown in the forum view.
Test Result	Success

Table 3.13: Test case verifying U14.

Unexpected Behavior

It is possible to create a new topic with empty title, body, and without tags.

3.2.14 Update Production Data

Use case	**U15.** Update Production Data
Goals	G2
Requirements	*No valid requirement specified.*
Actor	Farmer
Scenario	1. Log in as a farmer to the *Farmer's WebApplication*. 2. Click on the *Production* in the navigation top bar. 3. Click on the *See more* button at the bottom of *Production* box. 4. Click on the pencil icon button present in the item of the production entries list that is to be edited. 5. Change amount and note in the pop-up. 6. Click on the *OK* button.
Expected Output	The production data is updated and new, updated values are presented in the production entries list.
Test Result	Success

Table 3.14: Test case verifying U15.

Unexpected Behavior

Input format for the *Amount* in the *Edit the production* pop-up allows for inserting alphanumerical characters. Additionally, the inputs can be cleared and successfully saved. In both cases, the production data entry contains the *Q.ta* field equal to zero and empty *Note* field.

3.2.15 Comment Topic

Use case	**U16.** Comment Topic
Goals	G1
Requirements	R17
Actor	Farmer
Scenario	1. Log in as a farmer to the *Farmer's WebApplication*. 2. Click on *Forum* in the navigation top bar. 3. Click on the comment icon button in the left bottom corner of a topic. 4. Write a comment in the text box. 5. Click on the send icon button.
Expected Output	The comment is saved in a topic and the number of topic's comment has been increased.
Test Result	Success

Table 3.15: Test case verifying U16.

Unexpected Behavior

It is possible to add comment with empty content.

3.2.16 Answer Help Request

Use case	**U17.** Answer Help Request
Goals	G1
Requirements	R12
Actor	Farmer
Scenario	1. Log in as an agronomist to the *Agronomist's WebApplication*. 2. Click on *Help requests* in the navigation top bar. 3. Click on the comment icon button in the bottom right corner of a help request to be answered. 4. Fill a text box with response. 5. Click on the send icon button.

Expected Output	The response is saved and visible in the farmer that is the help request's author.
Test Result	Success

Table 3.16: Test case verifying U17.

Unexpected Behavior

- It is possible to add a response with empty content.
- When the farmer tries to see the response, he selects the *Forum* tab in the top navigation bar. The response is available in the *No feed* tab, what is very counter-intuitive. Then, if the farmer answers again to the agronomist's response, the help request is considered as *Closed* what is even more misleading. The names of help request's states does not reflect the states for described in the RASD, in the Figure 3.

Chapter 4

Comments and Remarks

The following sections are intended to communicate any issues or remarks that arose throughout the acceptance testing process. They are concerned with the user interface, the documentation and code provided, and the implemented functionalities in general.

4.1 Documentation Concerns

4.1.1 General Concerns

- Requirements and use cases look like they were not revised in further stages of the project, which leads to inconsistencies between them and the final product.
- Some use cases are missing. An example of such use case can be a farmer responding to a meeting request, which is mentioned in scenarios and use case diagrams, but not in use cases and requirements. This functionality is partially implemented, and always results in an error due to forbidden access, when trying to confirm/reject a meeting. This restricts possibility to test further functionalities, like meeting evaluation.
- Some of the terms are not explained clearly enough. An example of such a term can be *Farm data*, which is only briefly mentioned in one paragraph of the DD.
- The Sign Up functionality, described in **U8** is available only for farmers. There is no possibility to add a new agronomist inside the *Agronomist's WebApplication* is available only for farmers as stated in the provided documents. During testing, there was only one agronomist utilized, that was added using the provided SQL script.
- The requirements do not cover all the specified use cases. Such examples occurring in the section 3.2 are marked as *"No valid requirement specified"*.
- There are requirements, which do not correspond to the state of the final system and were not implemented. An example can be **R27**, which mentions insertion of farmer's location during sign up.
- There is no information on how to specify what area is an agronomist responsible for throughout the documents provided by the development team.

4.1.2 Vague farmer's evaluation

- At first on page 12 of the RASD document it is stated that:

  "Each farmer is graded every six months to evaluate his performance and his resilience. The grade the Agronomist gives to the Farmer is based on the farmer's production report, the amount of water they have used, how well they demonstrate to be resilient to meteorological adverse events and how well they show to be transparent towards other farmers by sharing their acquired knowledge through the forum."

- Then, *Scenario 9* on the page 13 says:

  "Dounia is a policy maker that after logging in sees the farmers performance, that is showed by the application after making some computations."

- Next, the R22 describes that: *"The system allows the agronomists to publish a grade for each farmer after each visit."*

- Last, on page 18, a *Grading System* is introduced. It is responsible for evaluating the farmer and then sharing the grade with the policy makers.

Taking all the above into consideration, it is impossible to clearly answer the following questions:

- Who or what is responsible for farmer's evaluation?
- How often does the evaluation take place? Is it every six months, or after each farm visit, or maybe even each time a policy maker logs in to the system?

4.2 Code Remarks

- The back-end part of the system is very well commented.
- During building of the frontend application, a great number of warnings is produced about *declared but not used* variables.
- POST HTTP methods are used for both creating and updating API resources. For updates, PUT, or PATCH methods would be more appropriate.
- POST HTTP request is sent every time a user wants to save changes to some data (for example, edit of production or meeting data), even when no data was actually changed.
- Commit messages do not carry any meaningful information.

4.3 Additional functionalities

The application has some small additional features that compose supplementary value to the users. Specifically:

- Farmers are able to see the graphs of selected production type in the *Production* view.
- Farmers are able to like comments inside specific forum topics.
- Users receive notifications regarding important events that occurred in the system.

4.4 User Interface Remarks

The design of the application is very smart and modern. Nonetheless, there are numerous issues that arose during the testing process.

- If a user logged in as a farmer goes to *Production* view and tries to type something in the *Choose a product* dropdown when it is empty, the website turns to display a blank page. A user can go back to the application only after a refresh of the web page fixes the issue.
- Poor data validation mechanism.
  - Users can create empty objects.
  - Users can create a farm with negative *Square Kilometer* number.
  - Users can enter letters in the telephone number field.
- After signing out of the application and then signing in again, the *Logout* dropdown is displayed (see the figure 4.1).

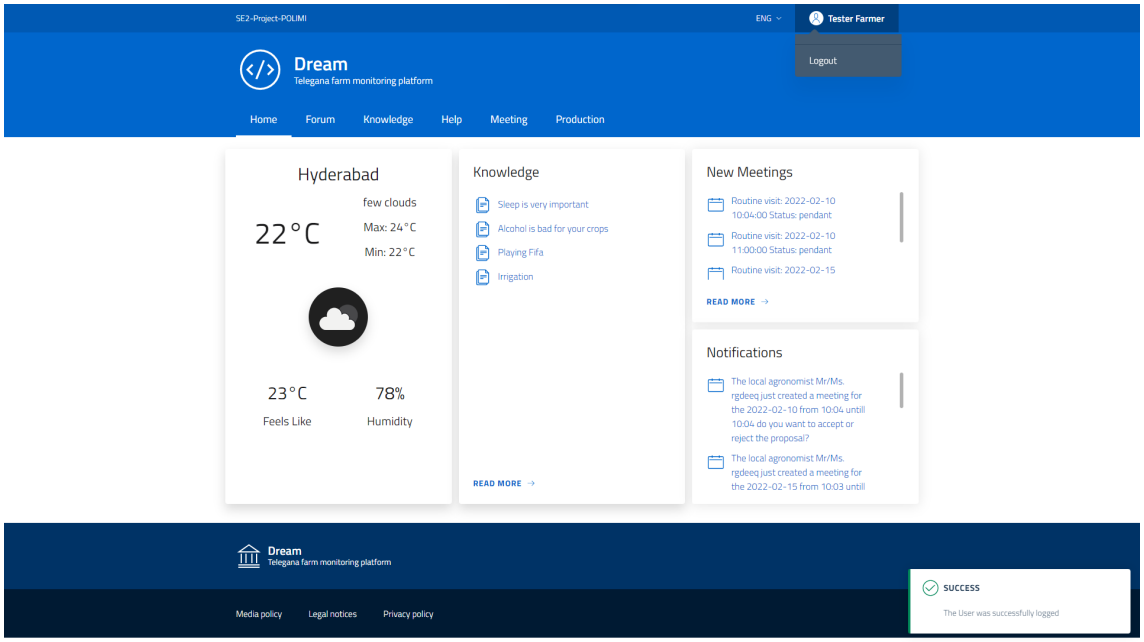

Figure 4.1: Unnecessary *Logout* dropdown shown after a successful log in operation.

- Some buttons display in inconvenient areas of the screen, which make them difficult to notice. An example of this issue is shown in the figure 4.2. Such unresponsiveness is also visible in the placement of the pop-up notifications when the page is zoomed out (example in the figure 4.4).

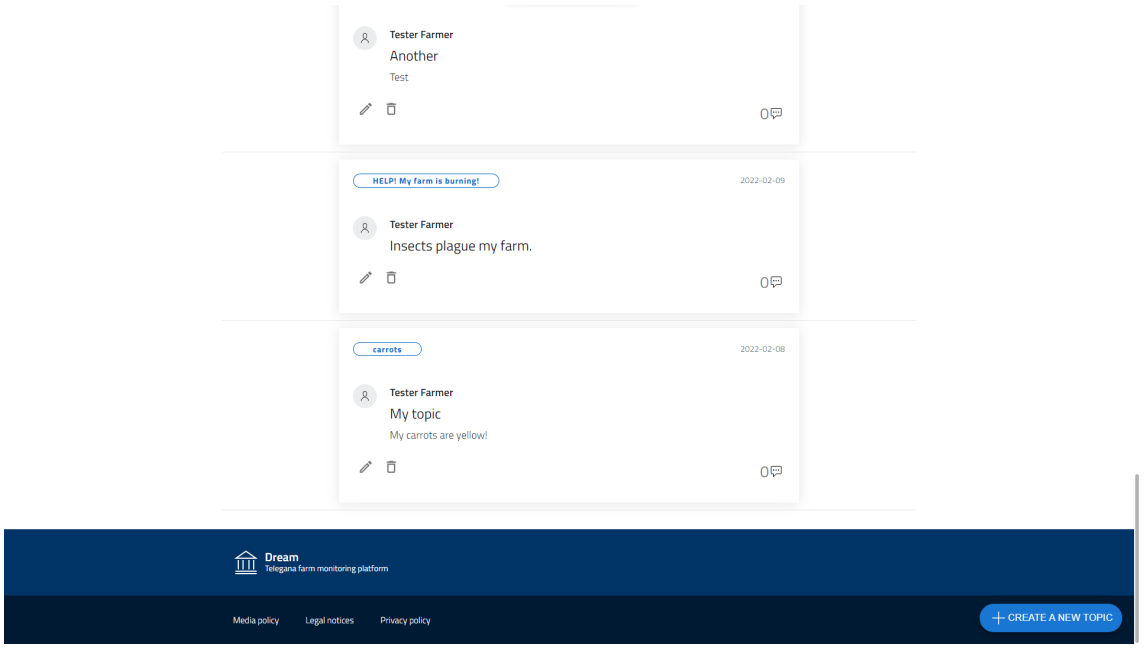

Figure 4.2: Inconvenient button placement example.

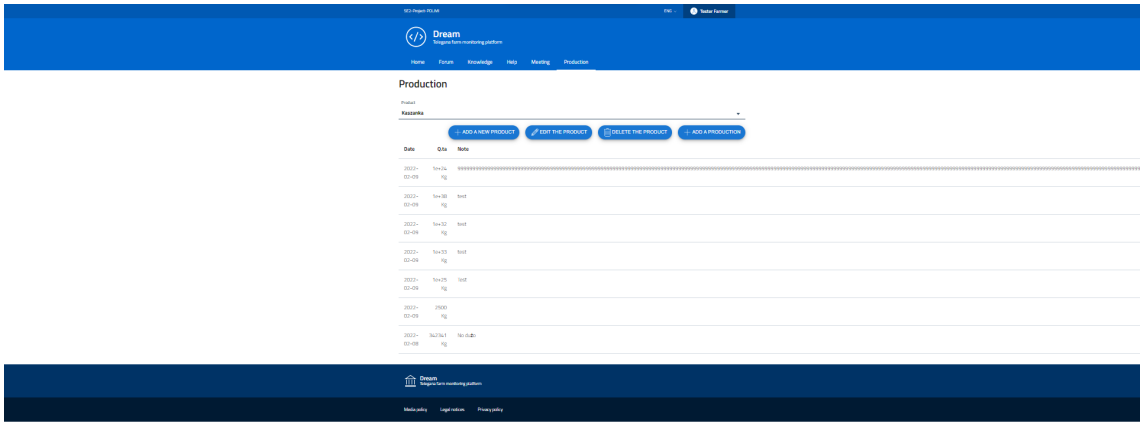

Figure 4.3: Very long input breaks the page.

- Some components do not scale well when given a long input string of characters. The text, instead of being wrapped, causes the page to become wider. This issue is depicted in the figure 4.3.

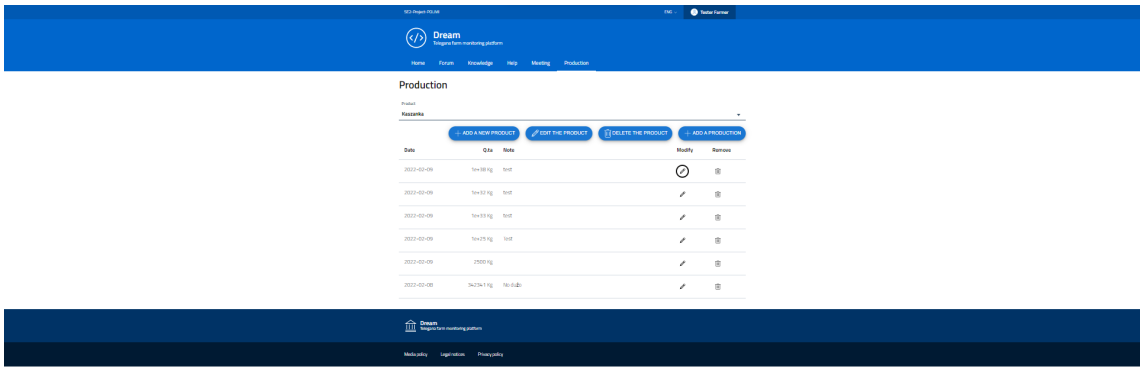

Figure 4.4: An example of the design being unresponsive.

Chapter 5

Effort Spent

Mariusz Wiśniewski

Topic	Hours	Date
Document structure	1	07.01.2022
Project analysis	2	08.02.2022
Introduction: purpose and analyzed project description	1	08.02.2022
Installation setup description	1	08.02.2022
Acceptance testing: project specifics and test cases defined	4	08.02.2022
Acceptance testing: test cases	3	09.02.2022
Comments and remarks	2	09.02.2022
General document adjustments	1	09.02.2022
Total	15	

Józef Piechaczek

Topic	Hours	Date
Project analysis	2	12.02.2022
Project installation	1	12.02.2022
Acceptance testing: authentication, meetings, notifications	3	12.02.2022
Comments and remarks	1.5	13.02.2022
Document adjustments	0.5	13.02.2022
Total	8	

Kinga Marek

Topic	Hours	Date
Project analysis	2	08.02.2022
Project installation	0.5	08.02.2022
Acceptance testing: project specifics and test cases defined	4	09.02.2022
Comments and remarks in chapter 4	1	09.02.2022
Minor improvements in the chapter 3	1	13.02.2022
Total	8.5	

References

[7] MySQL AB. *MySQL*. May 1995. URL: https://dev.mysql.com/doc/ (visited on 02/09/2022).

[8] Meta Platforms, Inc. *React*. May 2013. URL: https://reactjs.org (visited on 02/09/2022).

[9] Pivotal Software. *Spring Boot*. Apr. 2014. URL: https://spring.io/projects/spring-boot (visited on 02/09/2022).