AY 2021/2022

Politecnico di Milano

Acceptance Testing Document

Ottavia Belotti Alessio Braccini Riccardo Izzo

Professor
Elisabetta Di Nitto

Version 1.0
February 11, 2022

Contents

1 Tested Project

- **Authors**
  Valerio De Luca
  Giovanni De Novellis
  Sairaghav Venkataraman
- **Repository URL**
  `https://github.com/sairaghav/DeLucaDeNovellisVenkataraman`
- **Documents considered**
  - RASD: Requirements Analysis Specification Document;
  - DD: Design Document;
  - ITD: Implementation and Testing Document;

2 Installation

The installation went very smoothly following the guide present in the ITD. The Docker solution required minimal effort from us to install the whole software. Moreover, the database dump included within the Docker sped up even more the process, since it allowed us to not create another database and make up some new entries to try the WebApp out.

3 Acceptance Test Cases

3.1 Sign up & Log in

i. **Goal** User authentication

ii. **Test Cases**

(a) Create a new user.

(b) Login with existing user.

(c) Login with incorrect password.

iii. **Test Results**

(a) After filling all the required fields, the user is successfully created.

(b) Using an email and password for an existing user, we manage to succesfully log into the application and be redirected toward the user's specific homepage.

(c) When trying to login with an existing user but an incorrect password, the access is rejected as expected, however the application gives an incorrect error message: "User with this email already exists".

iv. **Issues**

– The error messages are not really taken care as one would expect, since they're not always coherent with the error itself. For test *(c)*, the error message is misleading, since it made us doubt that the registration phase of the new user went wrong, but instead we were just typing the password incorrectly.

3.2 Dashboard

i. **Goal** Visualize regional weather conditions' data

ii. **Test Cases**

(a) Retrieved and visualized data from 01/01/2021 and 31/03/2021 (valid date for which we knew there were data in the backend).

(b) Tried to retrieve data for future dates.

iii. **Test Results**

(a) Successfully retrieved all the data.

(b) Nothing to show. It correctly tells that there's no weather data for the specified period.

iv. **Issues** Nothing to highlight. Everything works fine.

3.3 Review Farmer Performance

i. **Goal** View information regarding best and worst performing farmers

ii. **Test Cases**

(a) After validating a farmer's report from the agronomist user, we checked the date range and retrieved the farmer performance from agronomist user.

iii. **Test Results**

(a) Entry about farmer is presented to the policy maker successfully: the efficiency is calculated, the farmer can be awarded as well. The list of best and worst farmers is correctly presented in ascending and descending order respectively.

iv. **Issues**

– Search function doesn't work very well, since a report validated today can only be taken into consideration in computing the efficiency if the search is performed specifying as upperbound the date of tomorrow, which is not very intuitive.

3.4 Review Agronomist Performance

i. **Goal** View information regarding best and worst performing agronomist

ii. **Test Cases**

(a) After validating 2 farmer's reports from agronomist user, retrieve list of best/worst performing agronomists from policy maker user.

iii. **Test Results**

(a) Entry about agronomist is presented to the policy maker successfully: the efficiency is calculated. The list of best and worst agronomist is correctly presented in ascending and descending order respectively.

iv. **Issues**

– Search function doesn't work very well, since a report validated today can only be taken into consideration in computing the efficiency if the search is performed specifying as upperbound the date of tomorrow, which is not very intuitive.

3.5 Visit Schedule Functionalities

i. **Goal** Visualize visit schedule and add/update schedules for agronomist user

ii. **Test Cases**

(a) Create a new daily plan for 10/02/2022 adding farmers from the selection bar.

(b) Confirm an already existing schedule. We marked as completed the daily plan for 10/02/2022 that has been created the day before.

(c) Visualize created schedules in the monthly calendar, searching for 10/02/2022.

(d) Using a new agronomist user created by us, search for a date.

iii. **Test Results**

(a) The plan is successfully created, however it is possible to add the same farmer multiple times.

(b) We were able to change the schedule status on 10/02/2022, swapping between "completed" and "not completed", as many time as we'd like.

(c) The search results in showing the calendar of February 2022. In cell 10/02/2022 there is the list of farmers planned to be visited for that day. However, the view is a bit buggy (more in the following bullet point).

(d) The WebApp crashes, probably because there are no schedules or farmers associated to that agronomist user yet. So this error has not been managed.

iv. **Issues**

- The same farmer can be added an infinite amount of times in a single schedule and it doesn't make sense.
- The visualization feature via monthly calendar doesn't practically work when an agronomist plans to visit more than 1 farm in a day, since the calendar cell is too small to fit more than that. This results in the list of added farmers to roll down behind the calendar and eventually pop up again where the latter is over. See *fig. 1* for reference.
- It doesn't really make sense that the visualization of reports is filtered by the exact date, but then the whole month is shown.

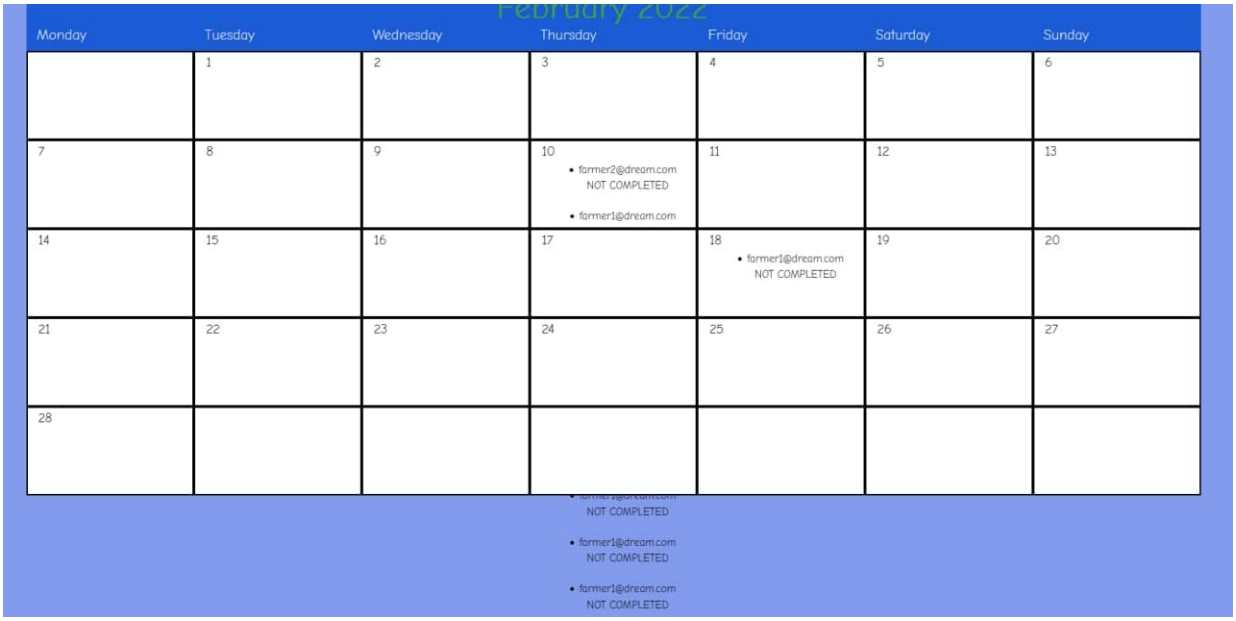

Figure 1: Calendar view bug: many farmers scheduled for a day

3.6 Report Functionalities

i. **Goal** Visualize and review farmers' report for agronomist user

ii. **Test Cases**

(a) Visualize report queue

(b) Report's approval

(c) Report's rejection

iii. **Test Results**

(a) Queue works fine, showing all the reports yet to be evaluated by the agronomist.

(b) Once we've filled the requested fields on a report and clicked on "Approve" button, it succesfully disappears from the agronomist reports queue and the associated policy maker user can see the approved reports in the Review Agronomist section.

(c) Once we've filled the requested fields on a report and clicked on "Reject" button, it succesfully disappears from the agronomist reports queue and the associated policy maker user can see the rejected reports in the Review Agronomist section.

iv. **Issues**

- Filling all the required fields, mostly the "best practices used" and "practices to be avoided" fields, is difficult because it's not likely that an agronomist can grasp what best and worst practices have been used by only referring to the data included in the report by the farmer.
- After the approval or rejection of a report, the webapp doesn't root back to the Home or even Report Queue, instead it stays on the specific report page as if something didn't work. Only a message at the bottom tells us that the report has been reviewed successfully.

4 Project Inspection

4.1 Document Quality

4.1.1 RASD

The RASD is well written, it is full of meaningful and detailed graphs as state diagrams, use-case diagrams and sequence diagrams.

Moreover, in general it isn't too technical, as this type of document requires since it is meant to be read by different type of stakeholders, even without any IT background.

However, the steering initiatives aspect required by the specification is not even mentioned, neither a functionality for the policy makers to review them is implemented.

Overall, we think the RASD is very well done.

4.1.2 DD

The general Application Server Component diagram presents all the manager components as black boxes, that are further developed later one by one. In

general, this approach causes the interfaces between subcomponents to not be very clear. Also, the specific Performance Manager's diagram is not coherent with the main component diagram, because it is said to communicate with the *Request Router* instead of the DBMS (*AccessDB*).

Finally, there are a lot of useful mockups to visualize the preview of the WebApp.

Testing

The testing plan is detailed since there are specified all the Manager components to be tested and which kind of test to perform on them, however, since they deeply specified them, we can say that they didn't take corner-cases tests into consideration. Comparing the test plan with the actual testing in the development phase, it seems that the group has focused on testing mostly correct behaviours but not assuring that unwanted ones don't occur as well.

4.1.3 ITD

The ITD extensively explains the implemented functionalities, but the users' description reminds us that agronomists and policy makers should be able to send messages to each other directly in-app even though the chat feature is not implemented.

The frameworks choice has been a bit superficially motivated.

The installation section is very well done: the guide is clear and the fact that a Docker container has been provided makes it very easy to install the software and start the server.

4.2 Architecture Quality

The architecture is 3-tier as stated in the DD: client, server (frontend and backend) and database. We believe it's a reasonable choice.

4.3 Code Quality

The software has been written in Python using Django Framework to develop both backend and frontend. These two has been devided into two different Django Apps.

We'd like to point out that this is not a very maintainable choice because, for example, the backend functions providing the API for the frontend are all grouped in one python file (dreamAPI/view.py). In our opinion, it would have been better to divide all the backend functionalities in more Django App, so that every feature can have, at least, its own view.py (for the API functions) and model.py (for the database ORM) files, making the structure more modular.

Testing

The system has been tested (24 test cases present in total) and all the tests are brought out succesfully. However, they don't really test corner cases, since only the regular workflow has been checked, like saving and retrieving data, but without testing any exception that the code might raise. The crashing error found in the schedule functionality is an example that proves this point.

5 Conclusion

Overall, the developed software is functional, however it's not a great user experience because some design choices are not intuitive and at times complex, for instance the fact that everything is filtered by date is a bit of a hassle. Most of the time, the relevant periods would be "today", "last week", "last month" and so on; so we think it would be easier as a user to choose a preset period among those cited ones or consider the entire historical data of the database (in case of review farmer/agronomist performance) by default, while making the custom "filter-by-date" a secondary feature, since it would be likely less used and it's more time consuming.

6 Effort Spent

Student	Time for acceptance testing
Ottavia Belotti	8h
Alessio Braccini	8h
Riccardo Izzo	8h