POLITECNICO
MILANO 1863

Software Engineering 2 Project

Design Document (DD)

DREAM

Data-dRiven PrEdictive FArMing in Telangana

Authors:
Marra, Miceli, Mora

Deliverable:	DD
Title:	Design Document
Authors:	Marra Gabriele, Miceli Matteo, Mora Destiny
Version:	1.0
Date:	09-January-2022
Download page:	GitHub
Copyright:	Copyright © 2022, Marra, Miceli, Mora – All rights reserved

Contents

Table of Contents . . . . . **iii**

List of Figures . . . . . **iv**

1 Introduction **1**
1.1 Purpose 1
1.2 Scope 1
1.3 Definitions, Acronyms, Abbreviations 1
1.4 Revision History 2
1.5 Reference Documents 2
1.6 Document Structure 2

2 Architectural Design **3**
2.1 Overview 3
2.2 Component View 3
2.2.1 DREAM Cloud Component 4
2.2.2 Application Server Component 5
2.3 Deployment View 6
2.4 Runtime View 7
2.5 Component Interfaces 9
2.6 Selected Architectural Styles and Patterns 9
2.7 Other Design Decisions 10
2.7.1 Entity–Relationship Model 10
2.7.2 HTTPS connection 11

3 User Interface Design **12**
3.1 Main Landing Pages 12
3.2 Farmer Views 13
3.3 Farmer Request for Help 14
3.4 Agronomist Views 15
3.5 Agronomist Managing Daily Plan 16
3.6 Policy Maker View 20
3.7 Policy Maker Setting New Trigger 20

4 Requirements Traceability **24**

5 Implementation, Integration, And Test Plan **34**
5.1 Overview 34
5.2 Implementation Plan 34
5.2.1 DBMS 34
5.2.2 DREAM Application 34
5.2.3 Web Server 35
5.3 Integration and Testing Plan 35
5.3.1 Continuous Integration 35
5.3.2 System Testing 36

6 Effort Spent **37**

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

List of Figures

1 High-Level 5-tier Architecture. . . . 3
2 High-Level Component View. . . . 4
3 DREAM Cloud Component View. . . . 4
4 Application Server Component View. . . . 5
5 High-Level Deployment View. . . . 6
6 Messages Sequence Diagram 7
7 Agronomist Creating Daily Plan Sequence Diagram 8
8 Policy Maker Ranking Sequence Diagram 8
9 Application Server Component Interfaces Diagram. . . . 9
10 ER Diagram. . . . 10
11 Login. . . . 12
12 Agronomist home. . . . 12
13 Farmer home. . . . 12
14 My Farm mock up. . . . 13
15 Farmer Forum mock up. . . . 13
16 Farmer message inbox. . . . 14
17 Farmer new help request. . . . 14
18 My Area agronomist mock up. . . . 15
19 My Plan agronomist mock up. . . . 15
20 Starting a new plan 16
21 Modify order of visits. . . . 16
22 Start navigation. . . . 17
23 Navigation guidance. . . . 17
24 During a visit. . . . 18
25 New report form. . . . 18
26 First report entered. . . . 19
27 All reports entered. . . . 19
28 Policy maker homepage. . . . 20
29 Launching new trigger form. . . . 20
30 Blank new trigger form. . . . 21
31 Select the parameter for trigger. . . . 21
32 Select crops related to trigger. . . . 22
33 Trigger form complete. . . . 22
34 See new trigger from dashboard. . . . 23
35 Implementation Phases. . . . 35

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

1 Introduction

1.1 Purpose

The purpose of this document is to provide a functional description of the design details of the DREAM product. The design details described in this document are chosen to meet the requirements and specifications outlined in the Requirements and Specification Document (RASD). This document is intended for designers in the implementation phase; the design descriptions provided outline the overall architecture of the system, offer different technical views of the system, provide more detailed user interface mock-ups, and present traceability back to the requirements from the RASD. This document also provides developers a general description of a plan to implement, integrate, and test the system.

1.2 Scope

The scope of this document includes functional descriptions at the architectural level, at the component level, and at the user interface level. Then, this document also includes a section dedicated to the implementation, system integration, and test plan phases of the development process. The main scope of this document is on the design of the DREAM product which is decomposed into its different components and functions. This decomposition of the system is supported by a traceability mapping: the main artery that links the DD and RASD documents. The scope of the DREAM product focuses on three main stakeholders: farmers, agronomists, and policy makers. This includes offering farmers a tool to manage their production and facilitate communication when seeking assistance; offering agronomists a tool to manage their workflows including managing their daily plans, reports, and requests for assistance; and offering policy makers a tool to easily view the data collected from the entire region to observe trends as they relate to their policies. This product is only focused on serving the Telangana region.

1.3 Definitions, Acronyms, Abbreviations

Term	Definition
DREAM	The system described in this document; Data-dRiven PrEdictive FArMing
User	Farmer, agronomist, or policy user; anyone who uses the system.
Policy Maker	Member of the Telangana government who deploys and manages different agriculture-related policies.
Agronomist	Professional who specializes in agriculture sciences.
Farmer	A user who uses DREAM to help manage data relating to their farms and fields.
Field	One enclosed area that corresponds to one crop. Many fields can make up a farm. The locations of the various fields do not need to be co-located.
Farm	A set of one or many fields that are managed by one farmer.
Production yields	The amount of crop harvested compared to the amount of crop planted. Measured comparatively by percentage or numerically by weight.
Flag	A marker on a farmer that signals the system to increase the priority for the farmer to get visited by an agronomist.
TSDPS	Telangana State Development Planning Society which manages the automated weather stations around the state.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

Term	Definition
UML	Unified Modeling Language
DD	Design Document
RASD	Requirements Analysis and Specifications Document
COTS	Commercial Off-the-Shelf
HTTPS	Hypertext Transfer Protocol Secure
CDN	Content Delivery Network
DBMS	Database Management System
API	Application Programming Interface
ER	Entity–Relationship
CI	Continuous Integration

1.4 Revision History

Revision	Date	Description
1.0	09 January 2022	Initial Release.

1.5 Reference Documents

- Assignment RDD A.Y. 2021-2022
- RASD - DREAM - Marra, Miceli, Mora
- ISO/IEC/IEEE 29148 dated 2018, Systems and software engineering - Life cycle processes - Requirements engineerings

1.6 Document Structure

The document is structured with the following sections:

- **Introduction**: This section introduces the document describing the purpose, scope, and other logistical elements.
- **Architectural Design**: This section provides multiple technical views to elaborate on the architecture of the system. This includes the component view, deployment view, runtime view, component interfaces, and other design choices.
- **User Interface Design**: This section provides mockups of the application including specific user flows for each of the three users.
- **Requirements Traceability**: This section explains how the RASD requirements map to the design components described in the Architectural Design section.
- **Implementation, Integration, and Test Plan**: This section details a methodology to implement, integrate, and test the system.
- **Effort Spent**: This section itemizes the time each participant allotted to different phases of the project.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

2 Architectural Design

2.1 Overview

DREAM will be used by thousands of users each day, with different needs, and located all over the Telangana region. To achieve this result and to respect all the requirements stated in the RASD, DREAM will be developed as a distributed application with clients and servers. The client will only show the graphical interface to the end-user, while the back-end will execute and support all the business logic operations.

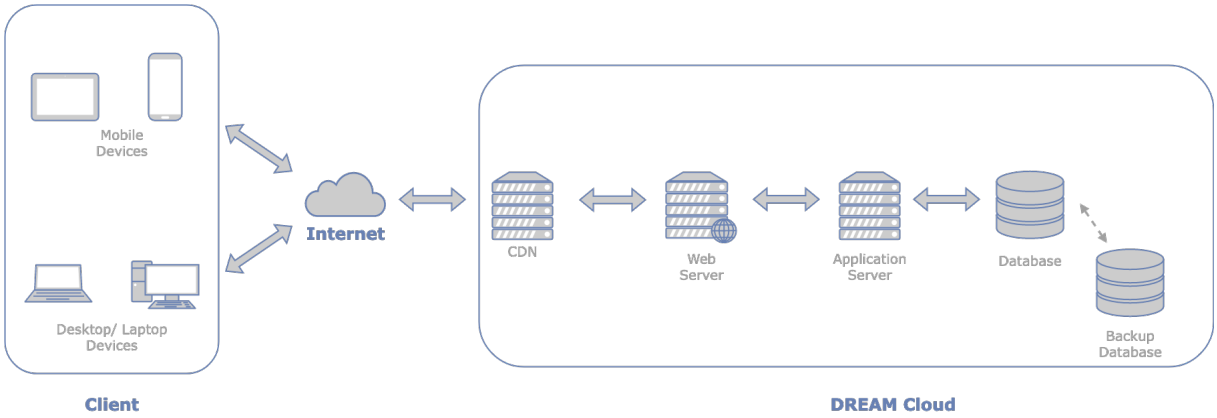

Figure 1: High-Level 5-tier Architecture.

Since the application must be reliable and flexible enough in handling the growth of the user base, a cloud-based architecture is suitable for this task. In particular, DREAM Cloud is a 3-tier architecture with a web server, an application server, and a storage server (DBMS). To enable the system to always work in ideal conditions, a load balancer routes all the requests to different application servers. When the computational load exceeds the available resources, new instances of the servers are created and used to balance the extra requests. An additional tier is then placed between the user and the DREAM Cloud: a content delivery network (CDN). The CDN will relieve some load from the system and improve performances in serving static contents. Overall, the resulting application will have five tiers. Lastly, given the importance of the historical data kept inside the database, a backup system will frequently clone the data on a different server, possibly located in a different availability zone.

2.2 Component View

The component view of the system depicts the internal structure of the DREAM system, demonstrating the components and interfaces that facilitate communication between them. The architecture consists of three different parts:

1. the Agro-Farmer mobile web application which will be used by farmers and agronomists
2. the Policy desktop web application used by policy makers
3. the DREAM cloud that hosts the backend for all the previously introduced functionalities

An additional element is presented for completeness: the Google Maps API is utilized by the mobile application to provide live turn-by-turn nagivation assistance. For simplicity, these three parts will be presented at different levels of abstraction, starting from the highest level, shown in Figure 2.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

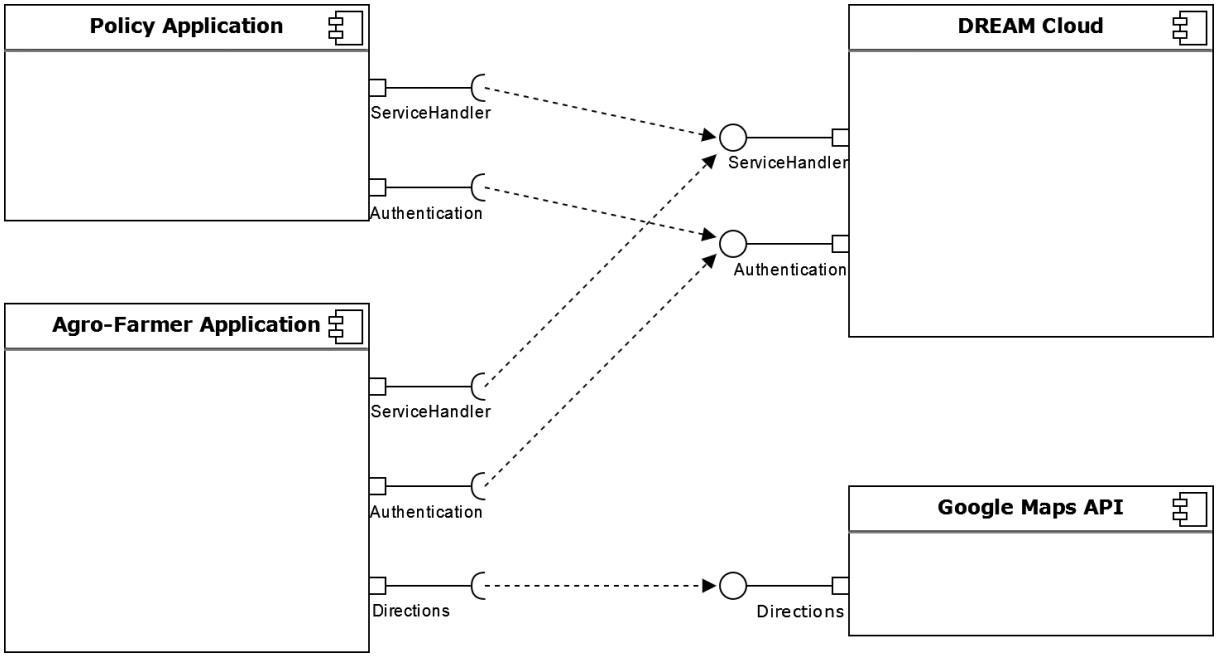

Figure 2: High-Level Component View.

2.2.1 DREAM Cloud Component

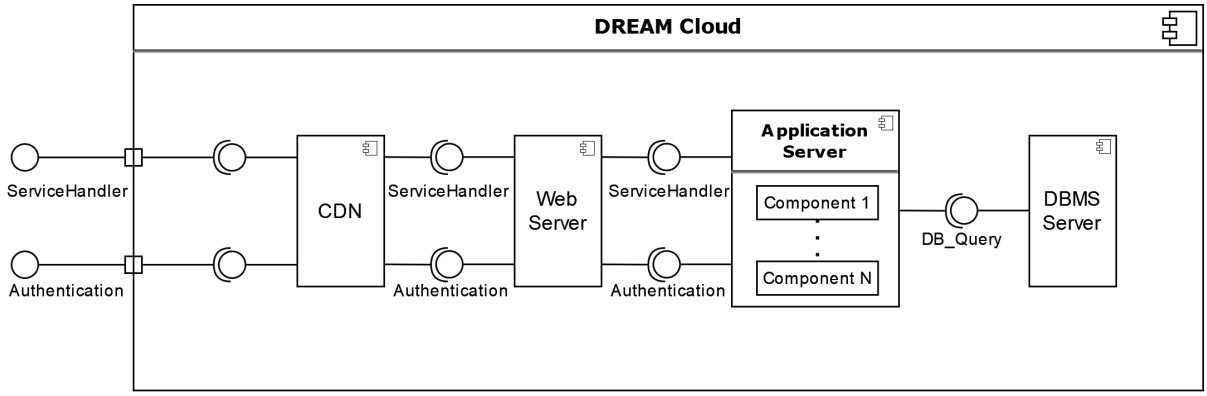

Figure 3: DREAM Cloud Component View.

The cloud architecture illustrated in Figure 3 is based on the following components:

- a CDN is implemented as the first component to improve the availability, performance, and redundancy of the entire system
- the web server is used to accept the HTTPS requests and send HTTPS responses. It provides static data as well as dynamic data obtained by the application server.
- the application server is the main component at this granularity. It contains about 15 smaller components that implement the business logic of the system. The dynamic content of the application is generated here.
- the DBMS server is responsible for the storage of data inside a database

Aside from the application server, all the others can be implemented using commercial off-the-shelf products (COTS).

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

2.2.2 Application Server Component

Figure 4 describes the application server and its components with the highest level of detail, as developers will have to implement the business logic themselves.

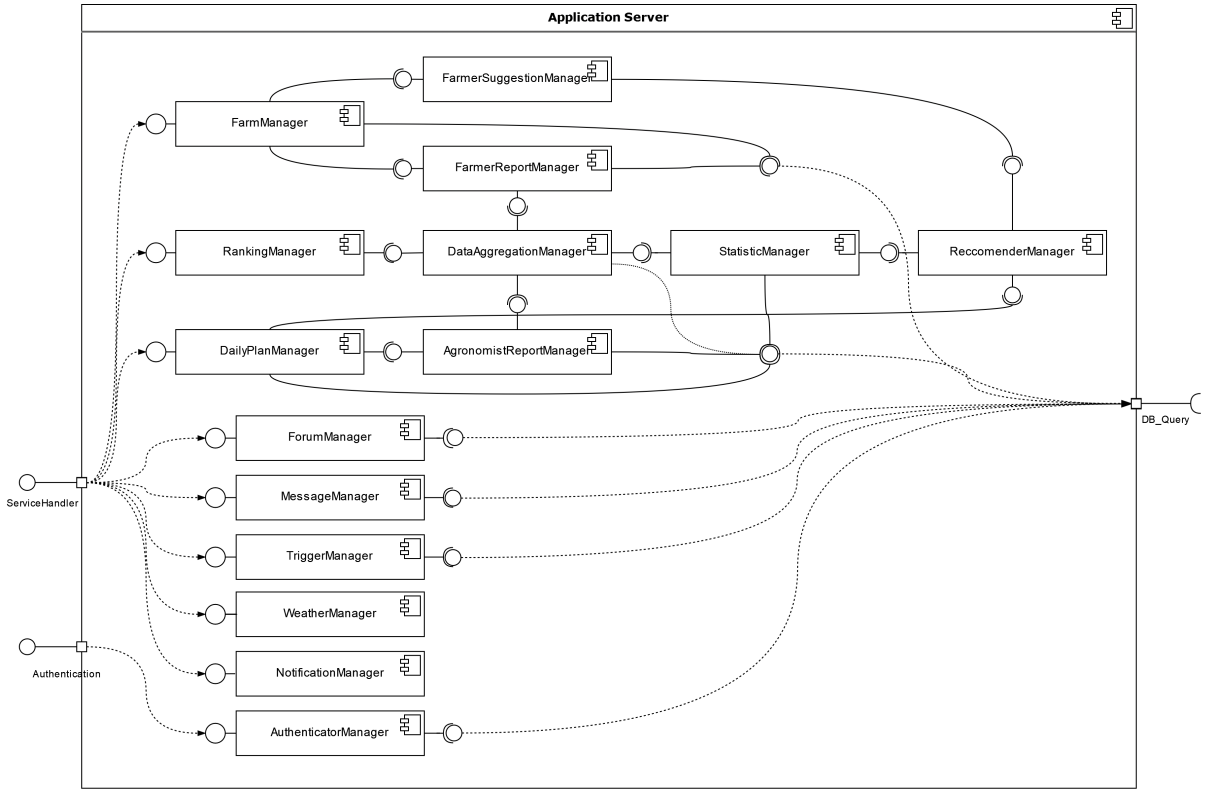

Figure 4: Application Server Component View.

The main components in the application server, providing all the business logic in DREAM, are:

- The **AuthenticationManager** is responsible for the login, registration, and logout operations. It allows the users to use their credentials to get access to the application. It is also in charge of the password reset procedure.
- The **ForumManager** manages the Farmer Forum with the creation, deletion, and edits of posts and threads.
- The **FarmManager** handles the My Farm section in the mobile application, providing the farmer with all the information about their farm and fields.
- The **FarmerReportManager** is in charge of collecting the production data. Other information such as production yields is collected from the farmer and stored inside the database.
- The **AgronomistReportManager** is in charge of collecting data from the agronomist. Data such as production data, a numerical score, and other information about the yields is stored inside the database.
- The **DataAggregationManager** is responsible for aggregating data and information obtained from the reports.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

- The **StatisticsManager** manages the calculation of various statistics based on historical data.
- The **RecommenderManager** generates recommendations for farmers and agronomists, based on the previously calculated statistics.
- The **FarmerSuggestionManager** relays suggestions about the production, such as which plant species to plant or the fertilizers to use, to the farmers.
- The **DailyPlanManager** is responsible for choosing the area the agronomist is responsible of and the creation, modification, and confirmation of the daily plans.
- The **MessageManager** handles the communication between farmers and agronomists. It collects the message from the sender and forwards it to the recipient.
- The **NotificationManager** module is in charge of sending notifications to users. It is used on different occasions, such as when an agronomist receives a new message from a farmer or when there is a new thread in the Farmer Forum.
- The **TriggerManager** handles the creation, modification, and deletion of triggers by the policy makers. It creates a trigger at the database level, whenever possible, and executes a check for each trigger each time new data is provided through the reports.
- The **RankingManager** is responsible for the creation and update of farmer rankings based on the provided data and calculated statistics.
- The **WeatherManager** collects and provides data about the weather to farmers and agronomists.

2.3 Deployment View

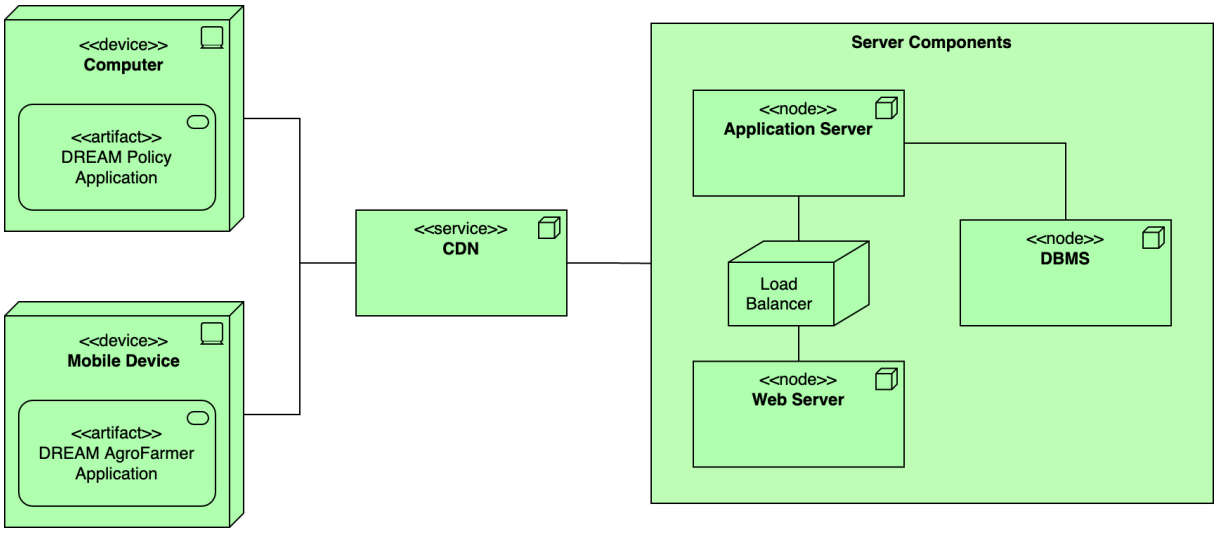

Figure 5: High-Level Deployment View.

The main components in the deployment view in Figure 5 are the client devices and the cloud platform. Policy makers will access the DREAM Policy Application via a computer device whereas agronomists and farmers will access the DREAM AgroFarmer Application via a mobile device. The CDN is represented as an external service. Then, the server components of the DREAM Cloud include the application server, the web server, the database server, and the load balancer.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

2.4 Runtime View

As shown in the Figure 6, when the farmer user accesses the Ask Experts area in the application, the farmer will send a request for help to their assigned agronomist in the form of a message. The agronomist and the farmer can continue to exchange messages until either user chooses to end the conversation. The sequence diagram only represents one message exchanged between them for clarity. The MessageManager has a key role in this interaction; it is the component that communicates with the DBMS for storing data and with the NotificationManager for notifying the other user of the message received.

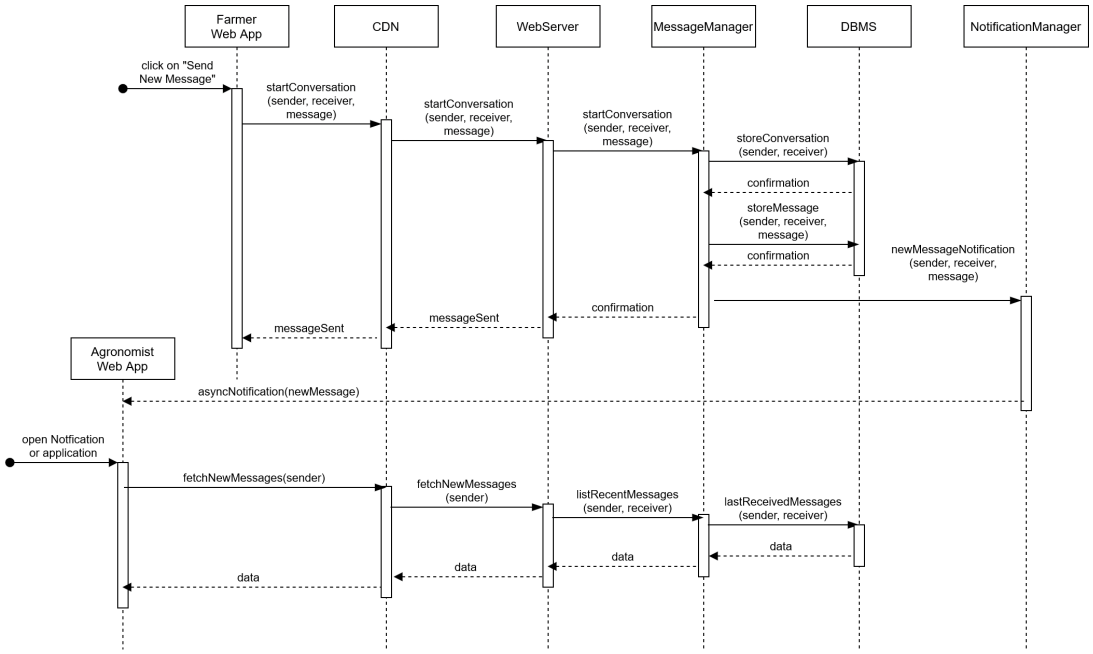

Figure 6: Messages Sequence Diagram

Figure 7 represents the daily plan creation performed by an agronomist. The key component this time is the DailyPlanManager. It interfaces with the RecommenderManager that, after retrieving data and statistics by the other components, generates a suggested list of farmers to return to the DailyPlanManager. Then, when it receives the list of suggested farmers, the DailyPlanManager also has to communicate with the Google API to get the path connecting them all. It then stores the daily plan by communicating with the DBMS server.

Figure 8 represents the actions of viewing the ranking of the farmers and applying a filter to the ranking. The RankingManager is the protagonist of this interaction; it instigates the DataAggregationManager and StatisticManager to retrieve data from all the other components shown in the figure. Then, when it has all the information, it can generate the ranking and returns it to the web application. Once it has calculated the ranking, it is available for applying filters to it and showing them to the user.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

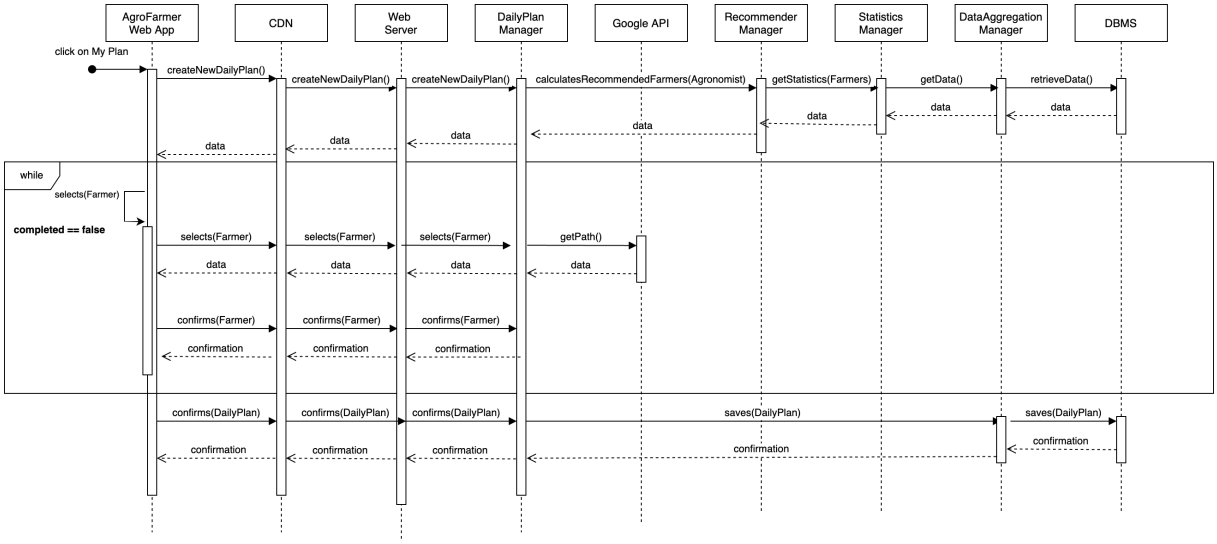

Figure 7: Agronomist Creating Daily Plan Sequence Diagram

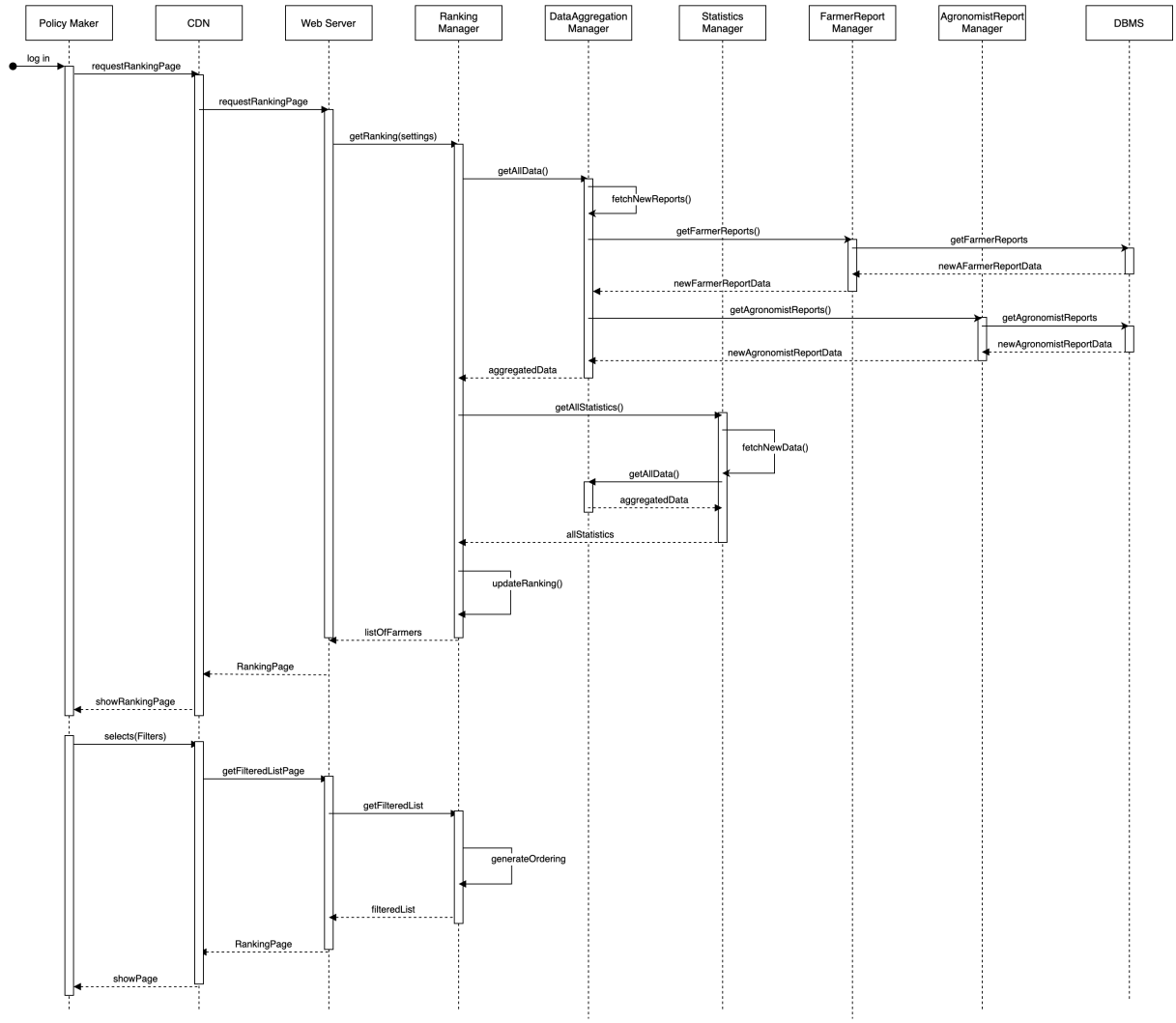

Figure 8: Policy Maker Ranking Sequence Diagram

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

2.5 Component Interfaces

This diagram shows the components of the application server with the main accessible methods.

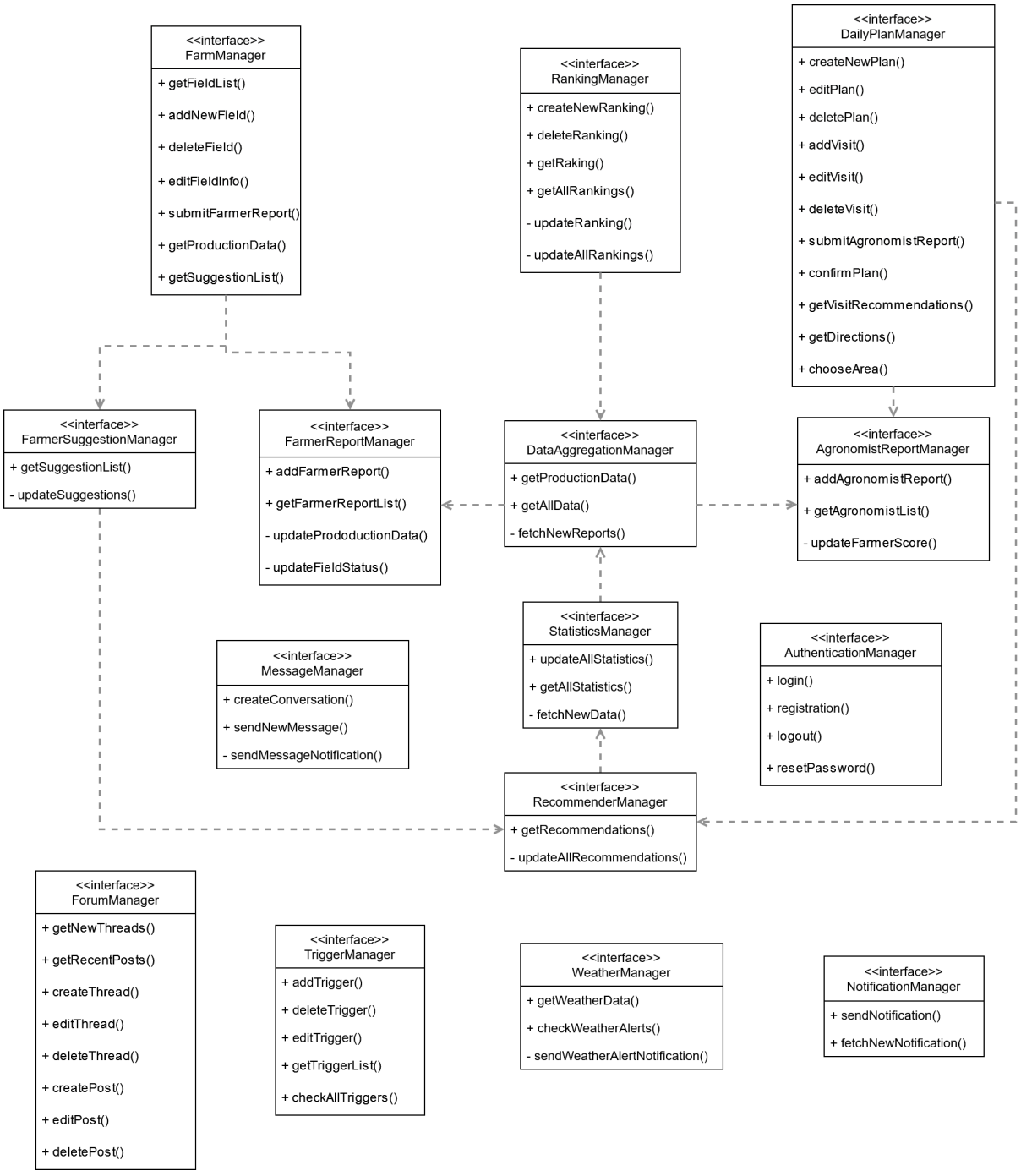

Figure 9: Application Server Component Interfaces Diagram.

2.6 Selected Architectural Styles and Patterns

As stated in Section 2.1, the system is designed as a **client-server architecture**, adapted in a **multi-tiered cloud configuration**. This choice favors, first of all, the decoupling of the system, which maximizes reusability, scalability, and flexibility.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

The presence of only web applications, thus implementing only the presentation layer, is typical of a **thin client**. The main advantage of such a configuration is the simplified management both for the user, not forced to download any software and for the developers, who can adopt a "write once, run everywhere" approach and push updates to the clients very easily.

Finally, the architecture uses an **external CDN** to distribute static content with the highest efficiency and performance. The CDN caches the static contents implemented by the web applications, while it acts as a proxy for the dynamic contents and API requests to the back-end.

2.7 Other Design Decisions

2.7.1 Entity–Relationship Model

Figure 10 represents an entity-relationship diagram of the database to help the reader identify the key entities and the main relationships and attributes.

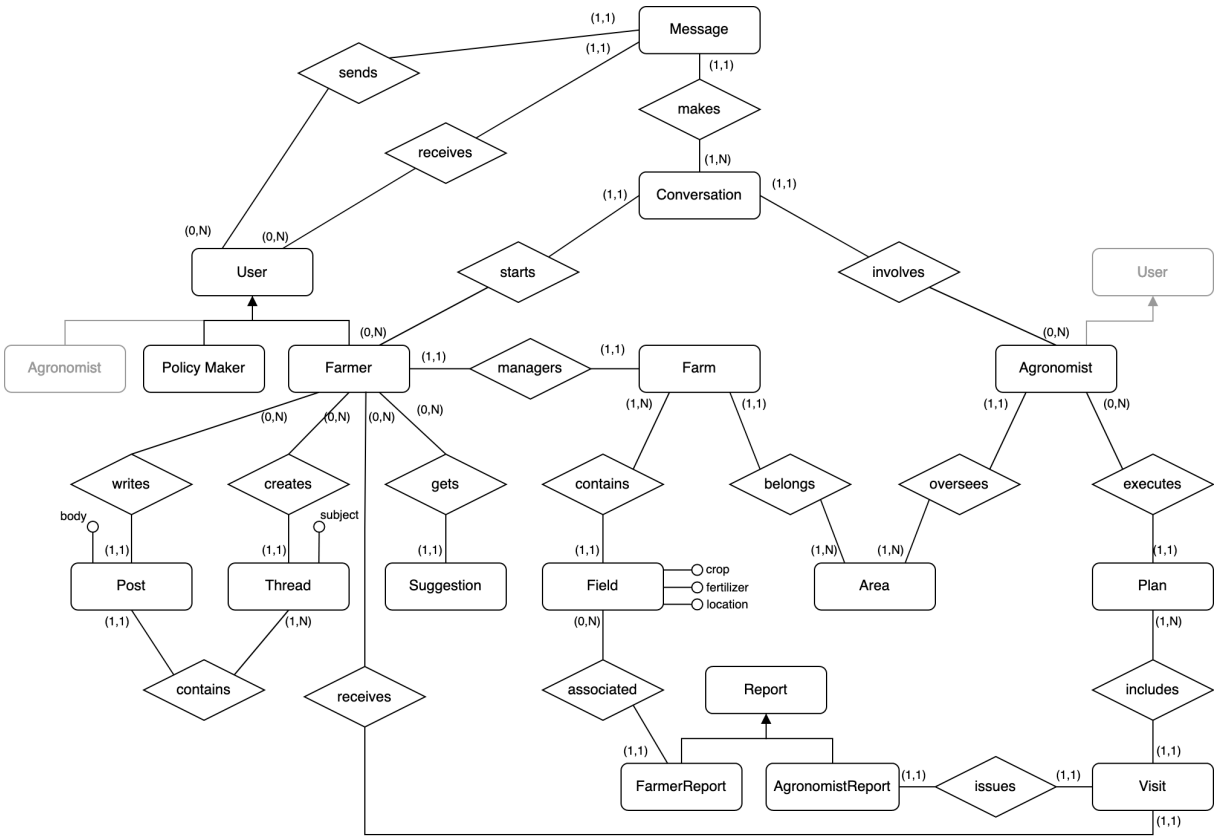

Figure 10: ER Diagram.

For more graphical clarity, the Agronomist entity is represented twice:

- on the left to show the exclusive hierarchy relation between the User, Agronomist, Policy maker, and Farmer entities
- on the right with all the other relations.

The hierarchy relation between the Report, FarmerReport, and AgronomistReport entities is exclusive. The ER diagram closely follows the relationships as they are described in the Alloy model in the RASD. For graphical clarity, not all the attributes are included in this diagram. Only some interesting attributes, such as crop, fertilizer, and location for the Field entity, are

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

included to specify that such descriptions are attributes instead of entities themselves. Other features of the system such as ranking are not included in this diagram because those are calculated at runtime; only entities related to the database storage are relevant in this diagram.

2.7.2 HTTPS connection

The communication between the front-end client and the back-end servers is implemented with an encrypted standard such as HTTPS in order to provide a high level of security.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

3 User Interface Design

3.1 Main Landing Pages

When a farmer user or agronomist user logs into their account from the mobile-friendly web application as shown in Figure 11, they will land on their associated homepages as shown in Figures 13 and 12, respectively.

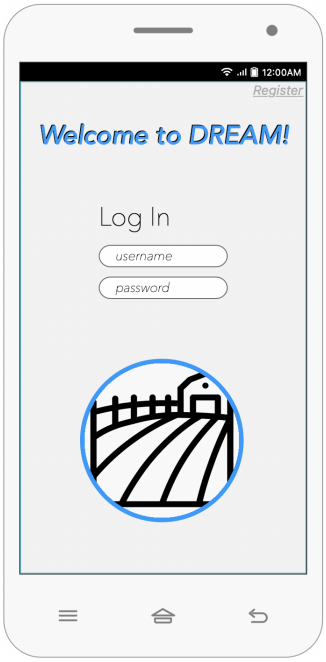

Figure 11: Login.

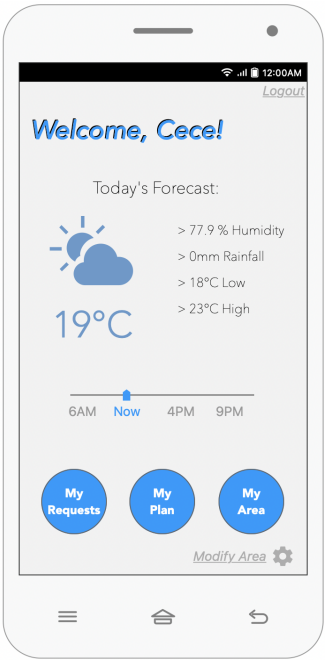

Figure 12: Agronomist home.

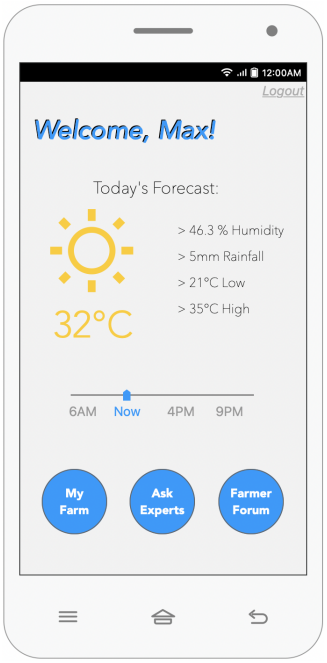

Figure 13: Farmer home.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

3.2 Farmer Views

The following mock ups show some of the main interfaces for a farmer user. From a farmer's homepage as shown in Figure 13, the farmer user can either navigate to the My Farm interface, Farmer Forum interface, or the Ask Experts interface using the three blue buttons at the bottom of the screen. Figures 14 and 15 show the My Farm and the Farmer Forum pages, respectively.

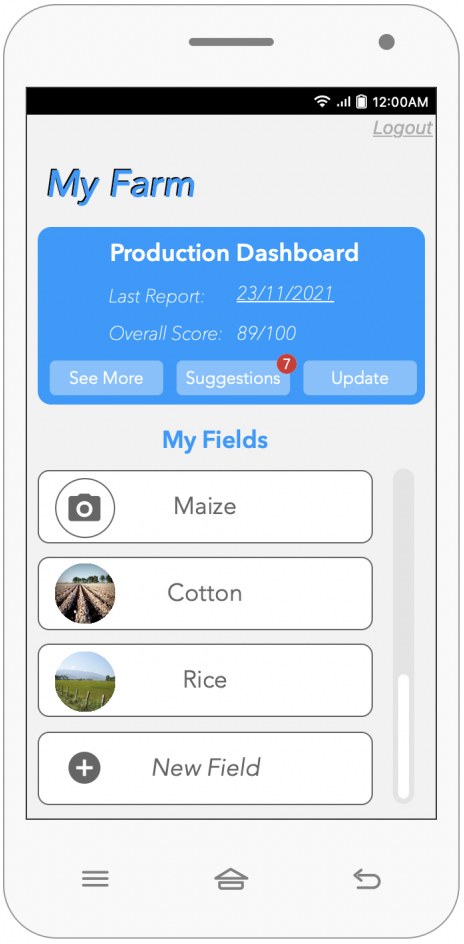

Figure 14: My Farm mock up.

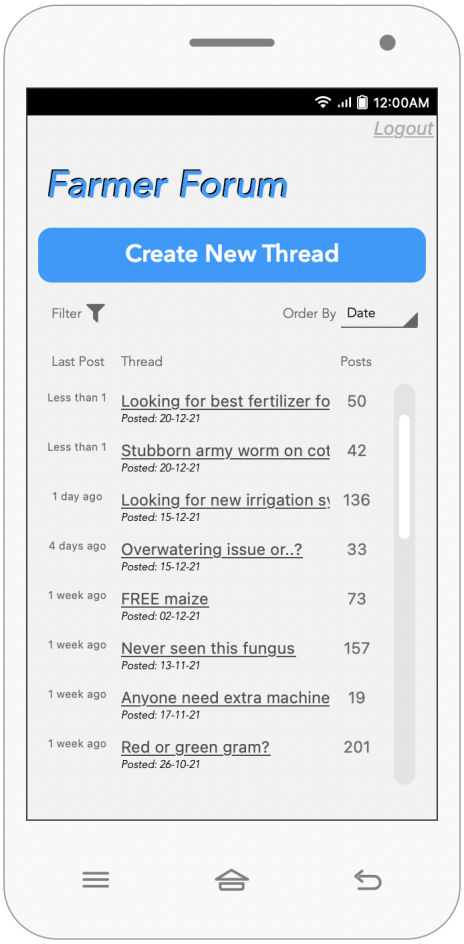

Figure 15: Farmer Forum mock up.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

3.3 Farmer Request for Help

In order to issue a new request for help, a farmer can click on the "Ask Experts" button as shown in the bottom of Figure 13. This will navigate the user to their main inbox as shown in Figure 16. The user can see their requests as messages to their agronomist. New responses from their agronomist are highlighted and shown at the top of the list of their other messages or help requests. To issue a new help request, the farmer user can click on the "New Request" button from Figure 16. The help request form is shown in Figure 17.

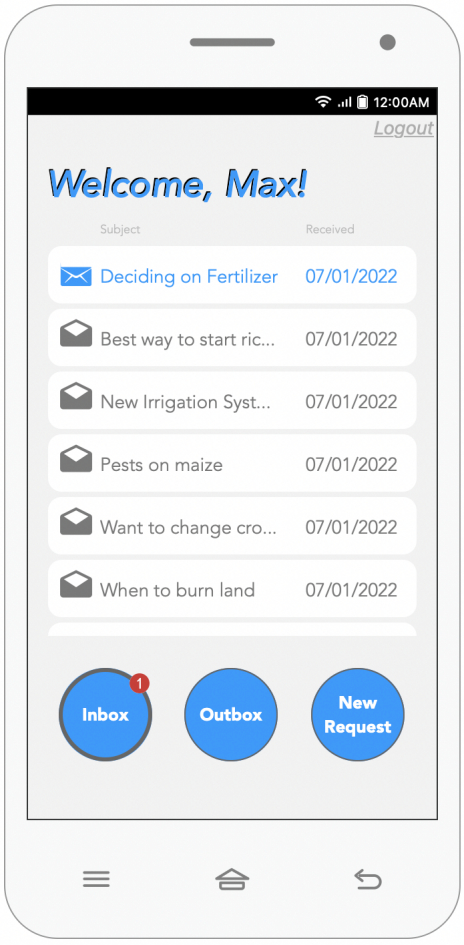

Figure 16: Farmer message inbox.

Figure 17: Farmer new help request.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

3.4 Agronomist Views

The following mock ups show some of the main interfaces for an agronomist user. From an agronomist's homepage as shown in Figure 12, the agronomist user can either navigate to the My Requests interface, My Plan interface, or the My Area interface using the three blue buttons at the bottom of the screen. Figures 18 and 19 show the My Area and the My Plan pages, respectively. The My Requests interface is very similar to that of the farmers' Help Request interface as shown in Figure 16.

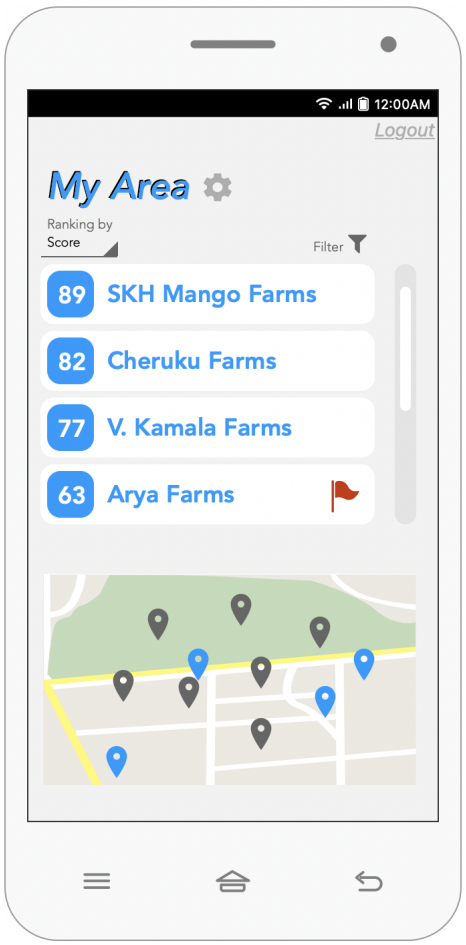

Figure 18: My Area agronomist mock up.

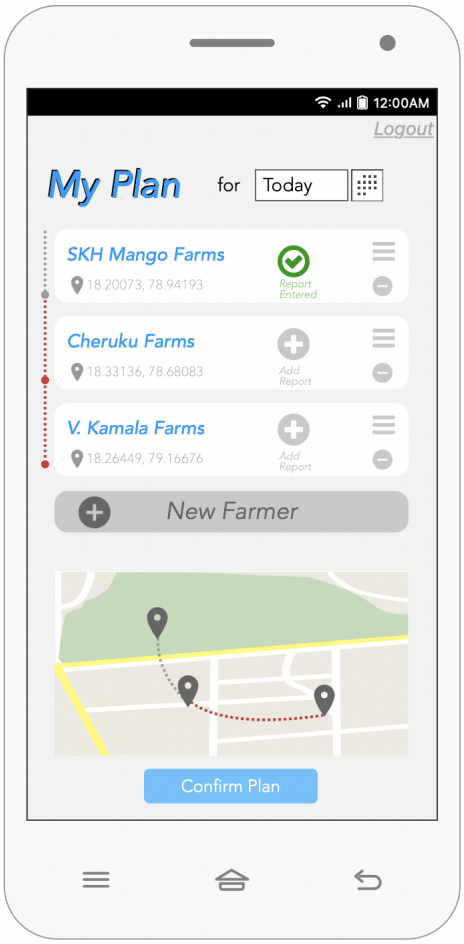

Figure 19: My Plan agronomist mock up.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

3.5 Agronomist Managing Daily Plan

When an agronomist user navigates to their My Plan interface using the blue button labeled "My Plan" from their homepage shown in Figure 12, the user is navigated to a page that looks like Figure 20. The farmers listed have been automatically chosen based on the DREAM algorithm that queues farmers based on different factors such as ranking, date of last visit, help requests, and more. The user can use the three-bars button on the left-hand side of the farmers' names in order to modify the order in which the farmer's are scheduled. When the order is modified as in Figure 21, the map with the tentative itinerary view is also updated.

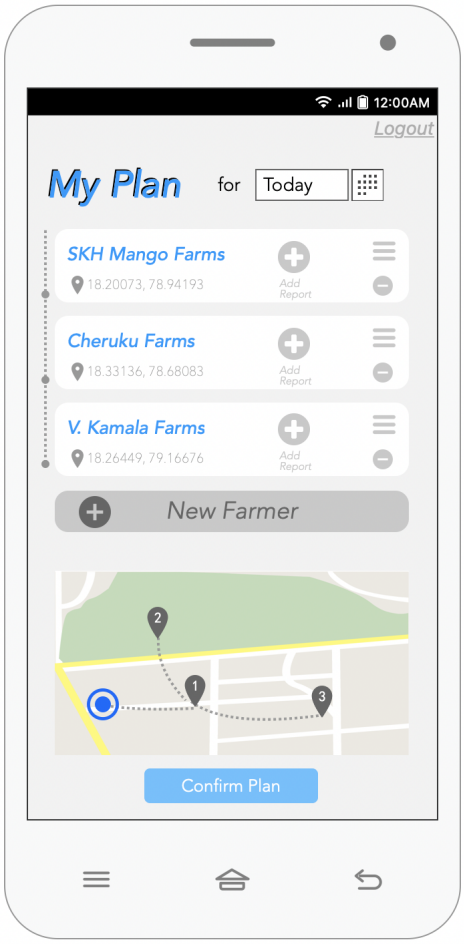

Figure 20: Starting a new plan

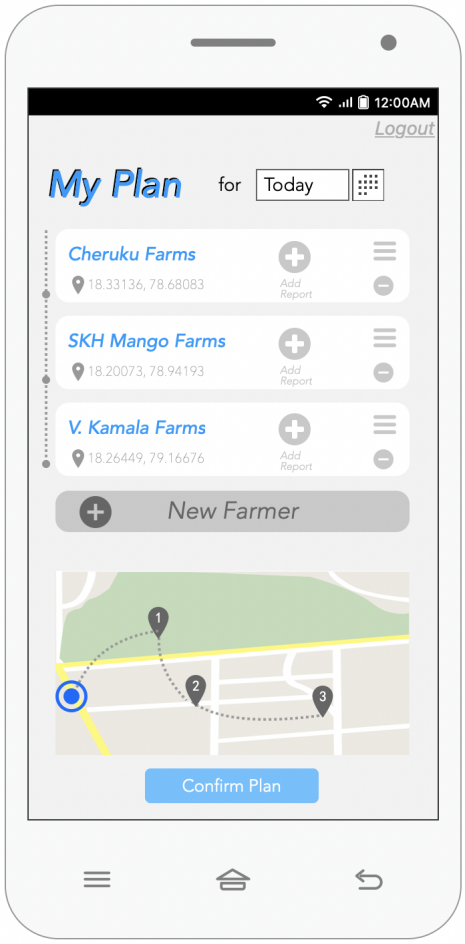

Figure 21: Modify order of visits.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

When the agronomist user clicks on the address or the GPS coordinates below the corresponding farm name, the user is prompted to start the navigation guidance as shown in Figure 22. If the user consented to GPS tracking, the user is shown turn-by-turn navigation directions as shown in Figure 23.

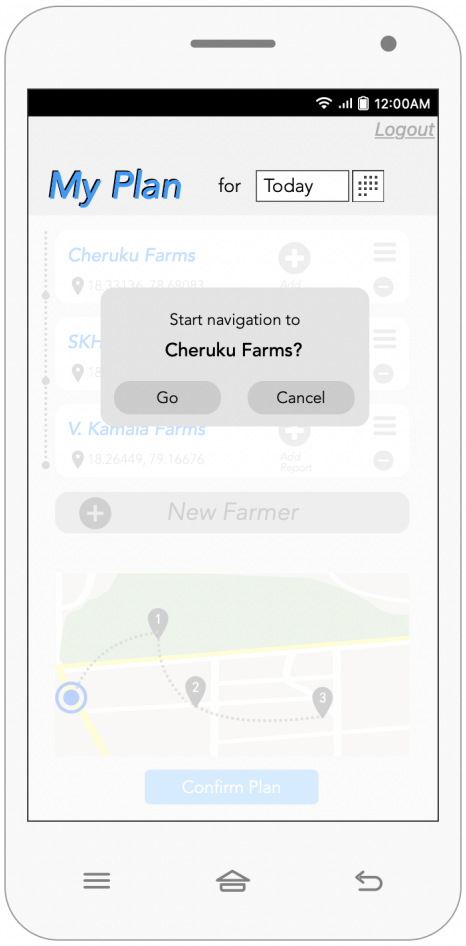

Figure 22: Start navigation.

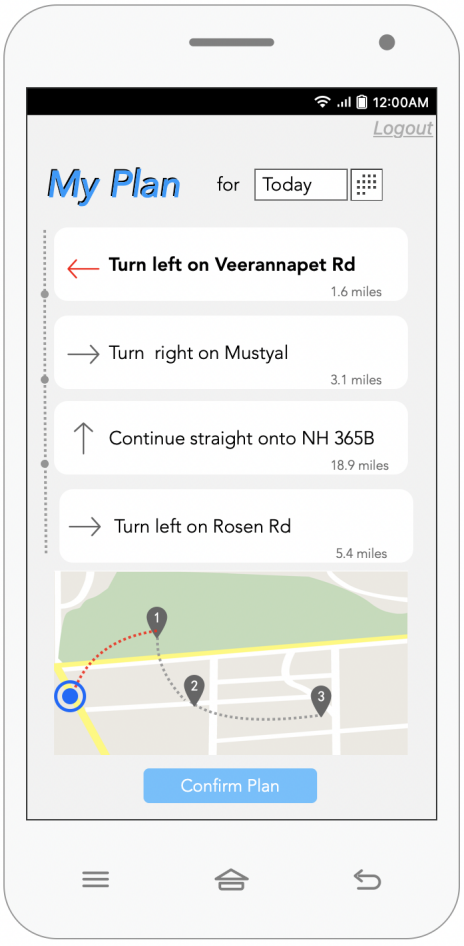

Figure 23: Navigation guidance.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

After the agronomist has arrived to one of the farm destinations, they will see the screen on Figure 24. Here, the agronomist can see the time left for the scheduled visit and choose one of three different operations: enter the report for that visit, start navigation for the next farmer, or add a new farmer to the plan. If the agronomist chooses to enter the information for the report, they will be navigated for a form as shown in Figure 25. Otherwise, if the agronomist enters a new farmer to the plan, they will return back to the plan homepage as in Figure 20. Again, the agronomist can reorder and modify the plan as they wish but the agronomist is responsible for entering the data for the reports, so it is helpful to complete the report forms directly during the farm visits.

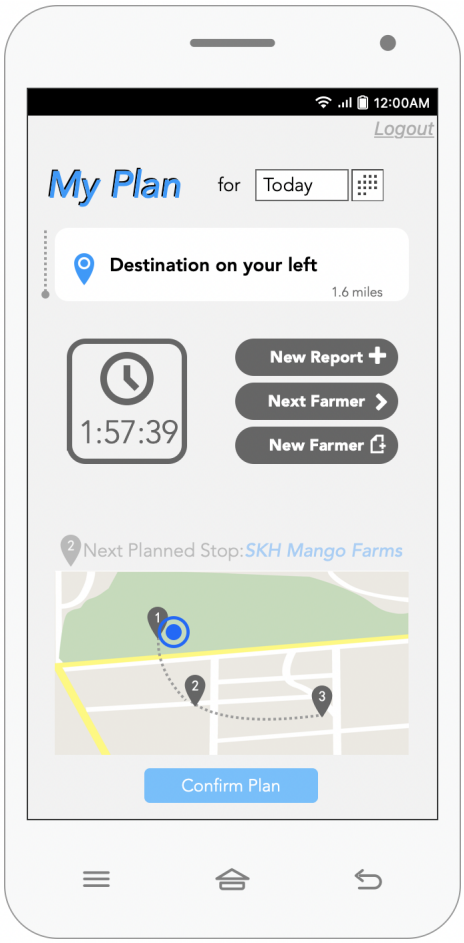

Figure 24: During a visit.

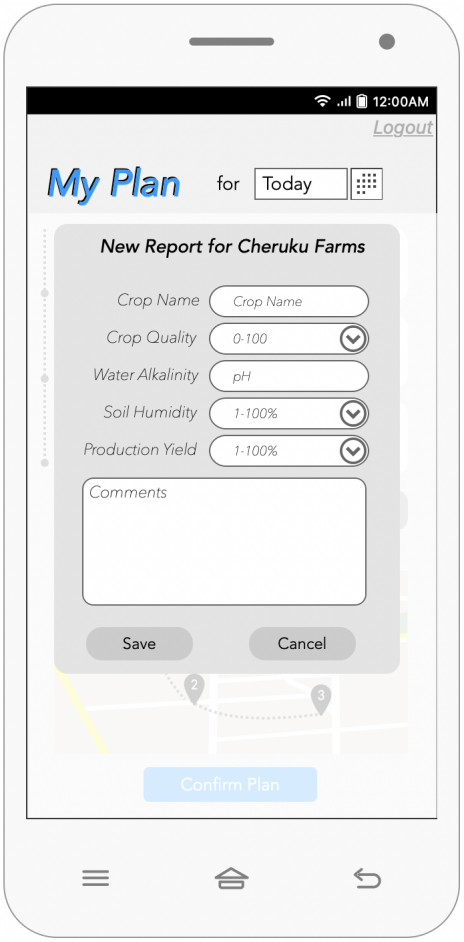

Figure 25: New report form.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

Once the report is entered, the agronomist user is navigated back to the main My Plan homepage as shown in Figure 26. Here, the interface indicates that one report has been entered. Using the application, the agronomist can continue executing their plan by going to their scheduled farm visits and entering the associated reports. After all the reports have been entered, the agronomist will see a screen as shown in Figure 27. Again, the agronomist is encouraged to check that all the information about the plan, including the order of the visits and the content of the reports, is accurate. When all the reports are entered, the "Confirm Plan" button becomes active as indicated by the color change.

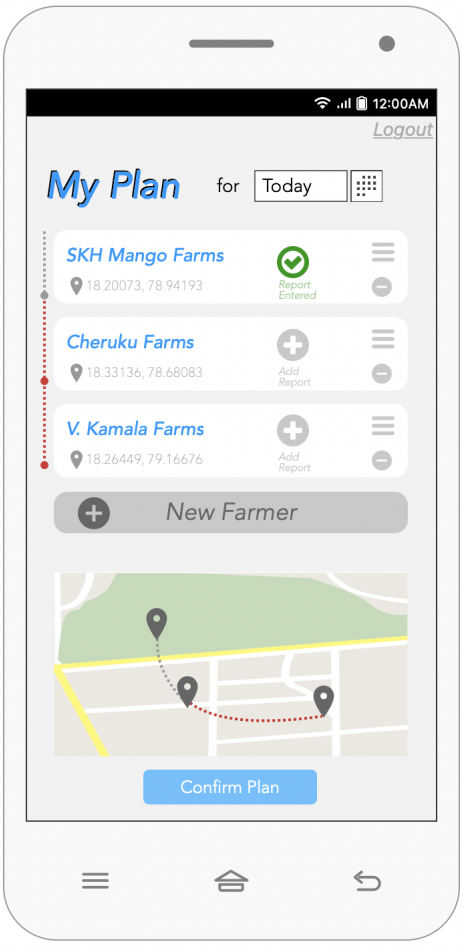

Figure 26: First report entered.

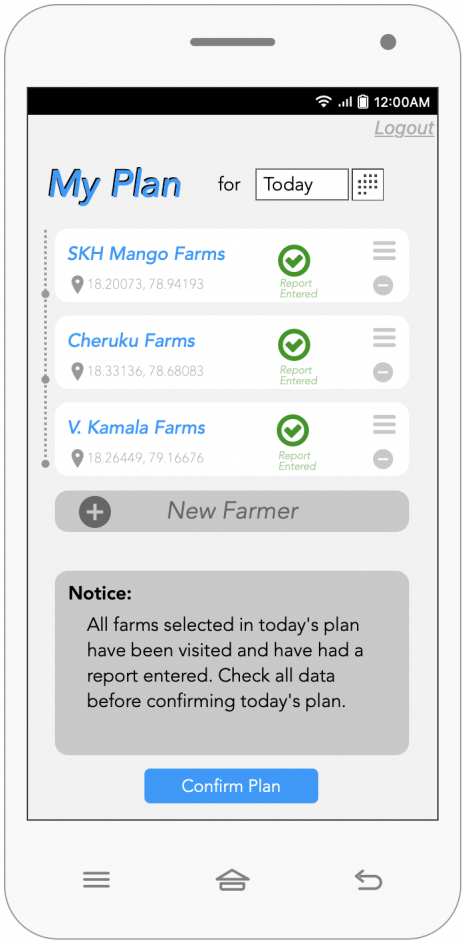

Figure 27: All reports entered.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

3.6 Policy Maker View

When the policy maker user logs in with valid credentials, they are navigated to their homepage view as shown in Figure 28. Policy makers can see a summary of the data in the top left-hand tile, filter their ranking on the bottom left-hand tile, see the farmers from the ranking in a map-view on the bottom right-hand tile, and access their Trigger Dashboard in the top right-hand tile.

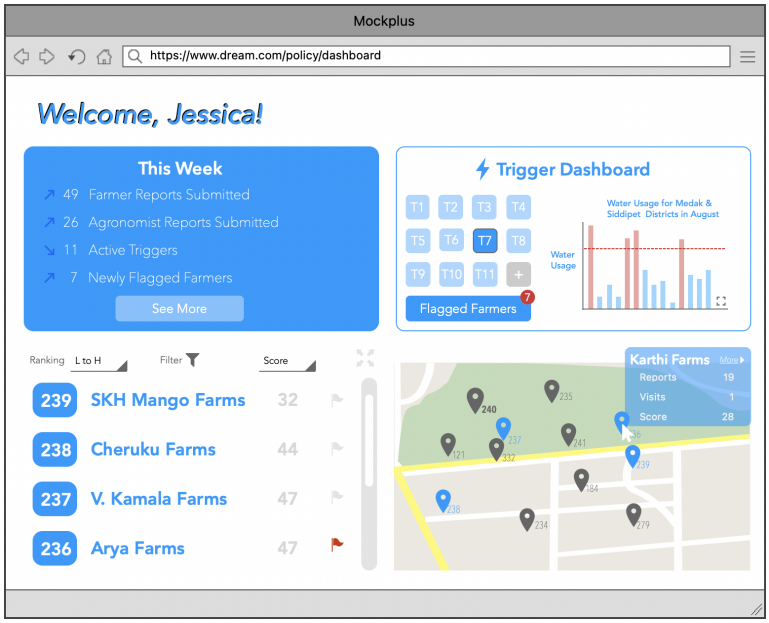

Figure 28: Policy maker homepage.

3.7 Policy Maker Setting New Trigger

To add a new trigger, the policy maker simply clicks on the grey plus button in the Trigger Dashboard as indicated by the cursor in Figure 29.

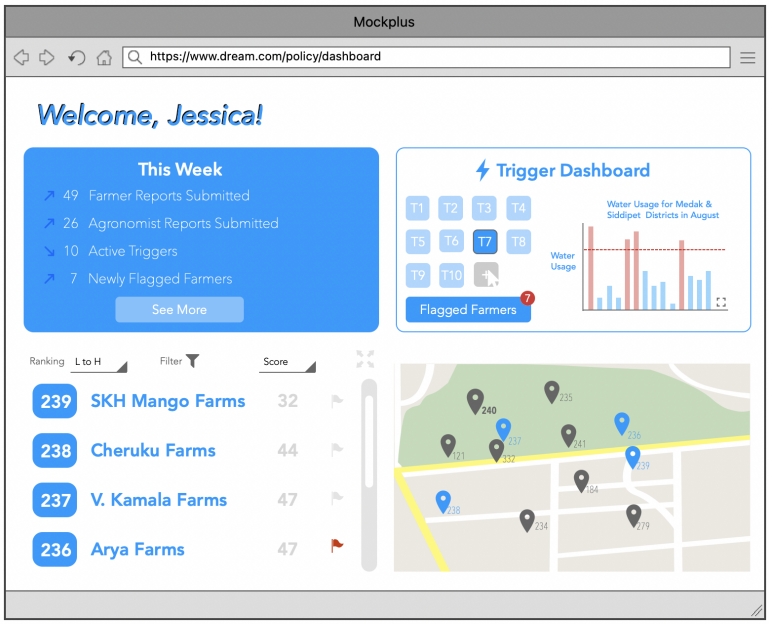

Figure 29: Launching new trigger form.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

The policy maker can configure a new trigger with the following options: the districts or areas to include; a specific time duration or limit to expire the trigger, optional; the crops to include in the trigger; the parameter to define the trigger over; and the maximum and minimum limits to define the trigger over. These options are also shown in Figures 30, 31, and 32.

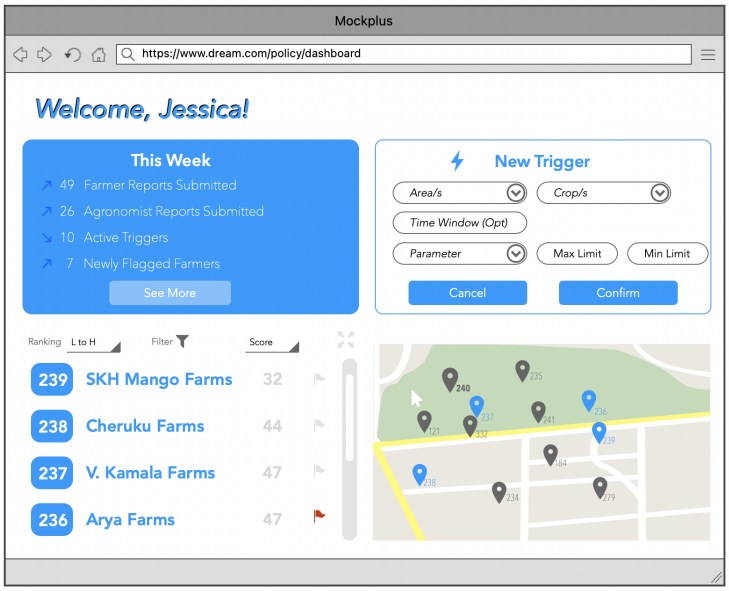

Figure 30: Blank new trigger form.

As shown in Figure 31, triggers can be configured around numerical data such as overall score as determined by the DREAM algorithms, crop quality as rated by agronomists in their reports, production yield as calculated by the data provided by farmers, soil humidity as measured by the sensors deployed by TSDPS, water usage as determined by weather data compared with data provided by farmers, or water alkalinity as measured by agronomists during their farm visits. If a farm becomes associated with data that falls within the range of the minimum and maximum of the parameter defined by the trigger (and the other conditions defined by the trigger are met), the flag is set for that farmer.

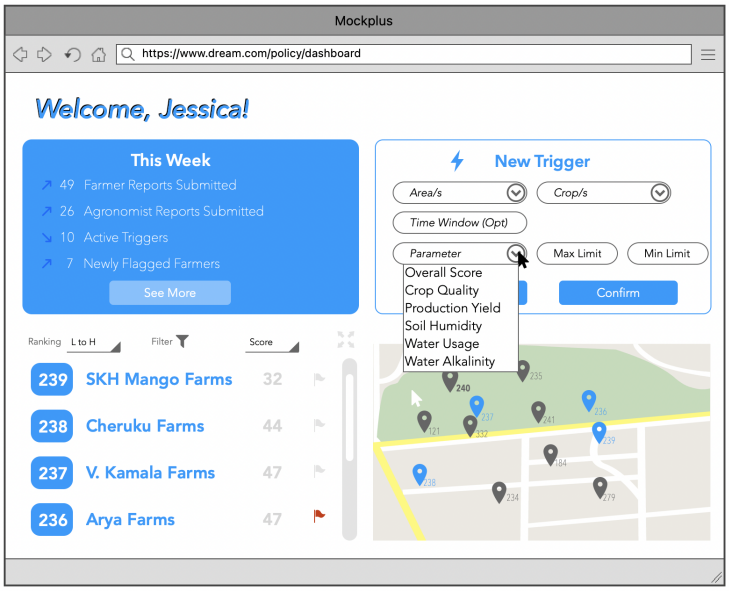

Figure 31: Select the parameter for trigger.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

As shown in Figure 32, triggers can be set to only be relevant for farms involved in farming specific crops. Alternatively, all crops can be selected in order to ensure farmers of all crops are included in the trigger.

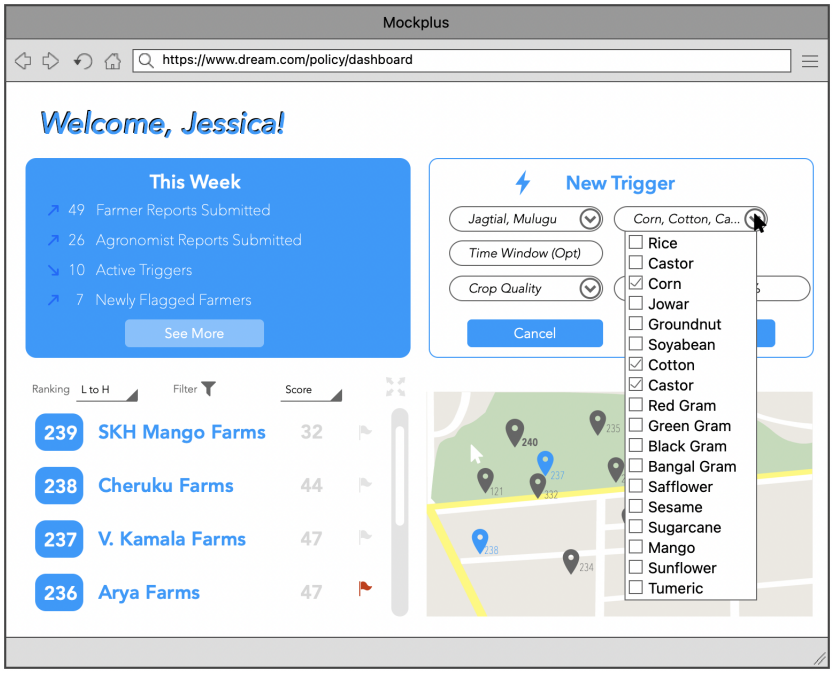

Figure 32: Select crops related to trigger.

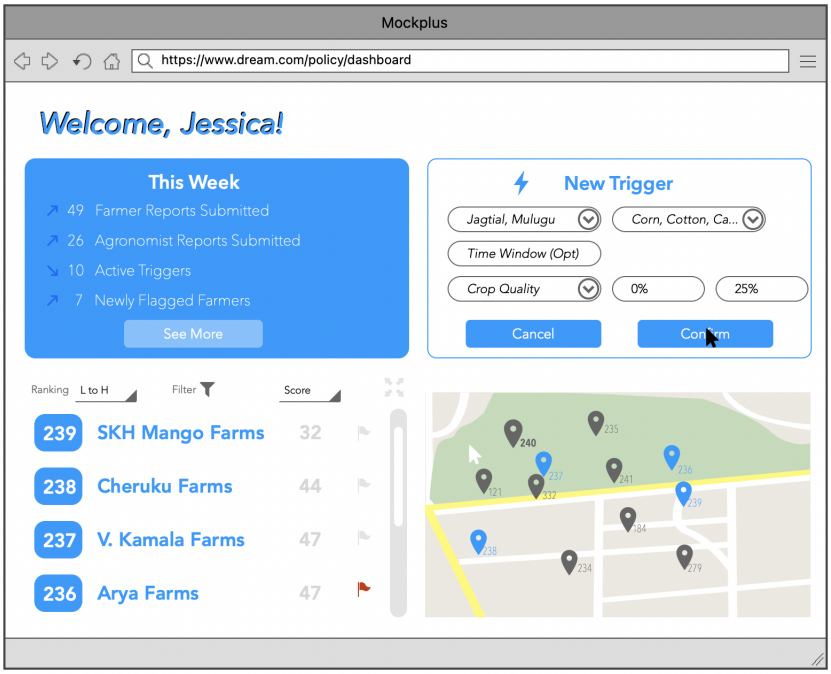

Figure 33: Trigger form complete.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

When the trigger form is submitted, the policy maker can see the newly-added trigger in their Trigger Dashboard. Active triggers can be selected from the policy maker's homepage in order to generate a chart of all the farms eligible for the trigger. From here, the policy maker can see which farmers have satisfied the trigger, and therefore received a flag. From the chart, these farmers are indicated by their data-point being marked as red, whereas in the ranking list-view, these farms are indicated by red flags on the right-hand side.

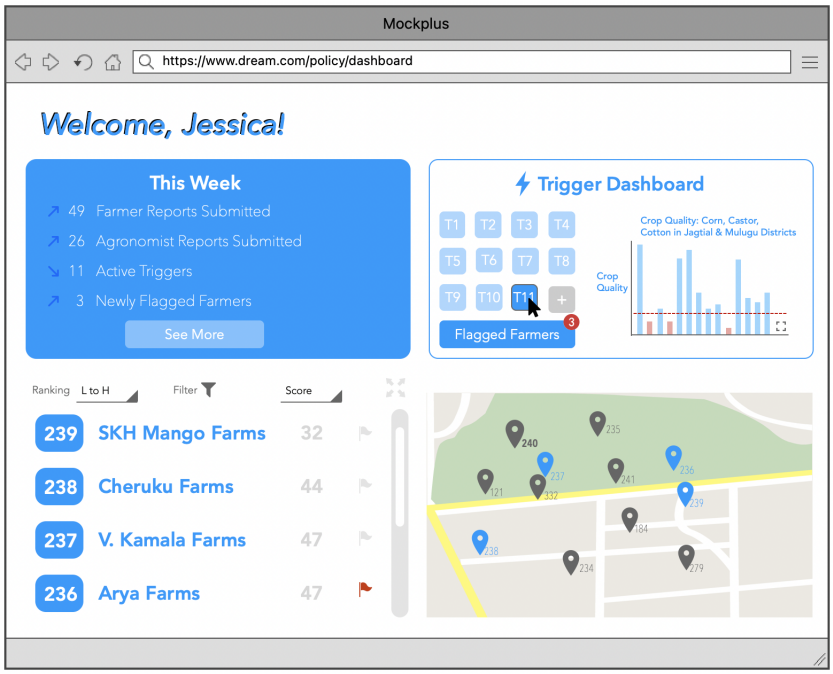

Figure 34: See new trigger from dashboard.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

4 Requirements Traceability

This section explains how the requirements presented in the RASD map to the design components described in the Architectural Design section of this document.

[R1] The system must allow the farmer to set the production types of their fields.

- AuthenticationManager
- FarmManager
- FarmerSuggestionManager
- DataAggregationManager
- RecommenderManager

[R2] The system must allow the farmer to set the position of their fields manually.

- AuthenticationManager
- FarmManager
- DataAggregationManager

[R3] The system must allow the farmer to set the position of their fields through their devices' GPS.

- AuthenticationManager
- FarmManager
- DataAggregationManager

[R4] The system must keep track of the data about farmers

- FarmManager
- FarmerReportManager
- DataAggregationManager
- StatisticsManager
- TriggerManager
- RankingManager

[R5] The system must provide an interface to visualize data

- FarmManager
- DataAggregationManager
- StatisticsManager
- RecommenderManager
- RankingManager
- WeatherManager

[R6] The system must be able to analyze data and show statistics

- DataAggregationManager
- StatisticsManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

[R7] The system must enable farmers to modify their production type.

- AuthenticationManager
- FarmManager
- DataAggregationManager

[R8] The system must enable farmers to report issues they may face.

- AuthenticationManager
- FarmManager
- FarmerReportManager
- FarmerSuggestionManager
- DataAggregationManager
- StatisticsManager
- RecommenderManager
- MessageManager
- WeatherManager

[R9] The system must allow the farmer to report production data at a frequency chosen by the farmer.

- AuthenticationManager
- FarmManager
- FarmerReportManager
- FarmerSuggestionManager
- DataAggregationManager
- TriggerManager

[R10] The system must retrieve the weather forecast data from the data that the Telangana government collects.

- DataAggregationManager
- StatisticsManager
- RecommenderManager
- WeatherManager

[R11] The system must show updated weather forecast data at most 5 minutes from which the data has been published by the Telangana government.

- DataAggregationManager
- StatisticsManager
- RecommenderManager
- WeatherManager

[R12] The system must provide weather data that forecasts at least 3 days ahead

- DataAggregationManager
- StatisticsManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

  - RecommenderManager
  - WeatherManager

[R13] The system must allow agronomists to access weather forecast data specific to their responsible area

  - AuthenticationManager
  - StatisticsManager
  - RecommenderManager
  - WeatherManager

[R14] The system must allow farmers to access weather forecast data based on their GPS location [or from the location of their farm on record]

  - AuthenticationManager
  - DataAggregationManager
  - WeatherManager

[R15] The system must provide an interface for farmers to request help and suggestions from other farmers

  - NotificationManager
  - AuthenticationManager
  - ForumManager
  - FarmManager
  - FarmerSuggestionManager
  - DataAggregationManager
  - RecommenderManager

[R16] The system must provide an interface for farmers to receive help requests and receive suggestions sent to them from other farmers

  - NotificationManager
  - AuthenticationManager
  - ForumManager
  - FarmManager
  - FarmerSuggestionManager
  - DataAggregationManager
  - RecommenderManager

[R17] The system must provide an interface for farmers to provide suggestions to other farmers

  - NotificationManager
  - AuthenticationManager
  - ForumManager
  - FarmManager
  - FarmerSuggestionManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

  - DataAggregationManager
  - RecommenderManager

[R18] The system must provide an interface for farmers to respond to help requests sent to them from other farmers

  - NotificationManager
  - AuthenticationManager
  - ForumManager
  - FarmManager
  - FarmerSuggestionManager
  - DataAggregationManager
  - RecommenderManager

[R19] The system must provide an interface for farmers to request help and suggestions from other agronomists

  - NotificationManager
  - AuthenticationManager
  - FarmManager
  - FarmerSuggestionManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - RecommenderManager
  - MessageManager
  - RankingManager

[R20] The system must provide an interface for agronomists to receive help requests sent to them from other farmers

  - NotificationManager
  - AuthenticationManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - MessageManager

[R21] The system must provide an interface for agronomists to respond to help requests sent to them from other farmers

  - NotificationManager
  - AuthenticationManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

  - RecommenderManager
  - MessageManager
  - RankingManager

[R22] The system must provide an interface for agronomists to provide suggestions to other farmers

  - NotificationManager
  - AuthenticationManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - RecommenderManager
  - MessageManager
  - RankingManager

[R23] The system must provide a forum interface

  - AuthenticationManager
  - ForumManager

[R24] The system must allow the farmer to create discussion forums.

  - AuthenticationManager
  - ForumManager
  - DataAggregationManager

[R25] The system must allow farmers to view all posts in the discussion forum.

  - AuthenticationManager
  - ForumManager

[R26] The system must allow farmers to post replies in the discussion forum

  - NotificationManager
  - AuthenticationManager
  - ForumManager

[R27] The system must keep track of all the forum discussion

  - ForumManager
  - DataAggregationManager

[R28] The system must allow agronomists to specify the geographic area in which they are responsible for.

  - AuthenticationManager
  - DailyPlanManager

[R29] The system must allow agronomists to modify their responsible geographic area.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

  - AuthenticationManager
  - DailyPlanManager

[R30] The system must allow agronomist to view the list of all farmers in their area.

  - NotificationManager
  - AuthenticationManager
  - FarmManager
  - FarmerReportManager
  - DataAggregationManager
  - RankingManager

[R31] The system must provide an evaluation of farmers such that the evaluation reflects the quality and quantity of their crop production

  - FarmManager
  - FarmerReportManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - RankingManager

[R32] The system must enable agronomists to access farmer evaluations from their specific area

  - AuthenticationManager
  - FarmerReportManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - RankingManager

[R33] The system updates farmers' evaluation when new data is available (ie, new farmer event entries or after an agronomist visit, etc)

  - FarmManager
  - FarmerReportManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - TriggerManager
  - RankingManager

[R34] The system must provide an interface for daily plans

  - AuthenticationManager
  - DailyPlanManager

[R35] The system must recommend which farmers should be included in the agronomist's daily plan

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RecommenderManager
- DailyPlanManager
- RankingManager

[R36] The system must generate recommendations such that farmers are visited by their respective agronomists at least twice a year

- AuthenticationManager
- AgronomistReportManager
- RecommenderManager
- DailyPlanManager

[R37] The system must generate recommendations such that farmers with low evaluation are visited more often than twice a year

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RecommenderManager
- DailyPlanManager
- RankingManager

[R38] The system must allow agronomist to view the list of all farms to visit on a specific day.

- AuthenticationManager
- FarmManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- DailyPlanManager

[R39] The system must allow agronomists to modify which farmers they visit in their plan

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

  - RecommenderManager
  - DailyPlanManager
  - RankingManager

[R40] The system must allow agronomists to specify and modify the duration of the visits in their plan

  - AuthenticationManager
  - DailyPlanManager

[R41] The system must maintain a record of farmers who have been visited by their respective agronomists

  - AuthenticationManager
  - AgronomistReportManager
  - DataAggregationManager
  - DailyPlanManager

[R42] The system must allow agronomists to modify the daily plan at the end of the day.

  - AuthenticationManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - DailyPlanManager

[R43] The system must allow agronomists to confirm that the daily plan was executed that the end of that day

  - AuthenticationManager
  - DataAggregationManager
  - AgronomistReportManager
  - StatisticsManager
  - DailyPlanManager

[R44] The system must not allow anymore modifications to the plan after the plan is confirmed by the agronomist

  - AuthenticationManager
  - DataAggregationManager
  - DailyPlanManager

[R45] The system must only generate a new plan for a new day after the plan from the preceding day was confirmed by the agronomist.

  - AuthenticationManager
  - DataAggregationManager
  - DailyPlanManager

[R46] The system must allow Telangana's policy makers to view the list of all farmers.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RankingManager

[R47] The system must allow Telangana's policy makers to view the performance and evaluation of the farmers.

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RankingManager

[R48] The system must allow Telangana's policy makers to view the ranking of the farmers.

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RankingManager

[R49] The system must allow Telangana's policy makers to view well-performing and poor-performing farmers.

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RankingManager

[R50] The system must allow Telangana's policy makers to flag the farmers that need to be helped based on their performance.

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- TriggerManager
- RankingManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

[R52] The system must allow policy makers to view the history of farmers' performance/ evaluation (score over time)

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RankingManager

[R53] The system must allow Telangana policy makers to view this measure of support designated to each farmer.

- AuthenticationManager
- FarmerReportManager
- DataAggregationManager
- AgronomistReportManager
- StatisticsManager
- RankingManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

5 Implementation, Integration, And Test Plan

5.1 Overview

To comply with standard design practices it is suggested to undergo unit testing as each component is built, integration testing as components get pieced together, and system testing to validate the system as a whole. This section will detail the order in which the components should be built, the method to carry out testing, and the manner in which to integrate the components.

5.2 Implementation Plan

5.2.1 DBMS

The structure of the database and storage servers should be implemented first. Investing in the implementation of these components first lays out the foundation of the structure of the data expected by the system before starting the implementation of the application components as described in Section 5.2.2. Additionally, after the DBMS and storage components are implemented, these components can serve as a bed for continuous integration testing as the system gets incrementally built. This is explained in further detail in Section 5.3.

5.2.2 DREAM Application

As stated previously, the entire system can be decomposed into 15 components which shall be built in the following order:

1. AuthenticationManager
2. FarmerReportManager
3. AgronomistReportManager
4. DataAggregationManager
5. StatisticsManager
6. DailyPlanManager
7. MessageManager
8. FarmManager
9. RankingManager
10. TriggerManager
11. NotificationManager
12. ForumManager
13. FarmerSuggestionManager
14. RecommenderManager
15. WeatherManager

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

This ordering of components prioritizes the most important features expected by the DREAM system and core components that serve as a foundation or basis for other components. Namely, the ≪AuthenticationManager≫ is the first component to be built to emphasize the importance of security. Before any user can do anything they must be able to login. Then, components supporting the generation of reports and handling the data are listed. This is a core function of the DREAM system and the foundation of the value-add of the DREAM system. After that, additional components that generate more data and support the aggregation of diverse data sources, such as ≪DailyPlanManager≫, ≪MessageManager≫, and ≪FarmManager≫, are listed. These component just build on the core functionalities that have already been established. Then, the remaining components refine the features that the DREAM system provides and fulfill the goals committed to by the system.

These four phases are also shown in Figure 35. The components have been seperate in such a way so that when each phase is complete, the subsequent or precedent phase can be iterated over. For example, the ≪DataAggregationManager≫ and ≪StatisticsManager≫ are both included in Phase 1. The components in Phase 3, such as the ≪FarmManager≫, however, may offer more data to plug into the ≪DataAggragationManager≫ or may tie into elements built in Phase 1 such as the ≪DailyPlanManager≫ tying into the reports generated in the ≪AgronomistReportManager≫. The segmentation of the phases not only considers priority but also establishes reasonable points in the implementation effort to consider iterating over other components that may or may not have been built, yet.

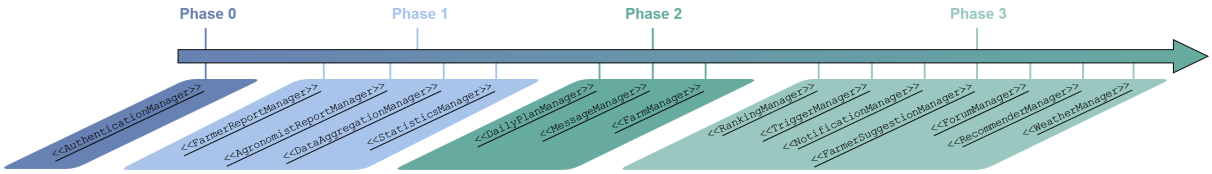

Figure 35: Implementation Phases.

5.2.3 Web Server

As stated in Section 2.2.1 the web server component can be implemented using a commercial-off-the-shelf product that can receive and handle HTTP requests. This should be implemented early in the development stages in order to enable early testing of the application components.

5.3 Integration and Testing Plan

5.3.1 Continuous Integration

Due to the complexity and size of this effort, it is best to adopt a continuous integration methodology to ensure that integration and testing is exhaustive, thorough, and complete. Continuous integration enable on-going testing; each time new code is developed and pushed, the entire software undergoes a full suite of tests. This practice requires a lot of set-up and maintenance, but it is preferable in order to successfully integrate the system and catch errors early.

With CI, testing occurs frequently even with small increments of development. As such a cadence, developers can find and address code and integration errors early in the development process. This does, however, mean that whenever new components are developed, the corresponding test modules need to be developed in parallel to support CI. This model tightly couples the implementation and the test phases of the development.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

5.3.2 System Testing

As explained in Section 5.3.1, the system will continuously undergo testing as the implementation progresses. This lightens the load of final system testing. Once the system has been fully implemented and integrated with the CI testing model, the system will have undergone and passed all the CI tests as a whole. The final system testing phase would simply be a final validation of the system; to ensure that the system successfully meets the customer's expectations.

Copyright © 2022, Marra, Miceli, Mora – All rights reserved

6 Effort Spent

Gabriele Marra		
Task	**Time**	**Date**
Chapter 2 Meeting 1	1.5h	2021/12/27
Chapter 2 Meeting 2	1.5h	2021/12/28
Architecture Overview Text	2.0h	2022/01/02
Meeting/ Working Session	2.5h	2022/01/03
Architecture Component Diagrams	3.0h	2022/01/03
Architecture Component Text	3.5h	2022/01/04
Sequence Diagrams	1.5h	2022/01/05
Meeting	3.5h	2022/01/06
Diagrams	3.5h	2022/01/07
Section 2.6 Writing	1.5h	2022/01/08
Final Meeting	3.5h	2022/01/08
Total	**27.5h**	

Destiny Mora		
Task	**Time**	**Date**
Chapter 2 Meeting 1	1.5h	2021/12/27
Chapter 2 Meeting 2	1.5h	2021/12/28
Chapter 1 Writing	2.5h	2021/12/29
Meeting/ Working Session	2.5h	2022/03/01
Diagram Making	1.0h	2021/01/03
Chapter 5 Writing	3.0h	2022/01/04
Sequence Diagram	2.0h	2022/01/05
Meeting	3.5h	2022/01/06
Mock Ups	6.0h	2022/01/07
Section 2.7	2.0h	2022/01/08
Final Meeting	3.5h	2022/01/08
Total	**29.0 h**	

Matteo Miceli		
Task	**Time**	**Date**
Chapter 2 Meeting 1	1.5h	2021/12/27
Chapter 2 Meeting 2	1.5h	2021/12/28
Requirement traceability	3.0h	2022/01/02
Requirement traceability	2.0h	2022/01/03
Meeting/ Working Session	2.5h	2022/01/03
Sequence diagram	2.5h	2022/01/05
Sequence diagram	1.0h	2022/01/06
Meeting	3.5h	2022/01/06
Section 2.4 Writing	2.0h	2022/01/07
Section 2.4 Writing and text reviewing	2.5h	2022/01/08
Final Meeting	3.5h	2022/01/08
General reviewing	1.0h	2022/01/08
Total	**26.5 h**	

Copyright © 2022, Marra, Miceli, Mora – All rights reserved