POLITECNICO
MILANO 1863

DD

Design Document

DREAM

Salvatore Marragony (10617299)

Alessandro Maranelli (10661029)

Amir Bachir Kaddis Beshay (10659740)

Version 1.0

CONTENTS

1 Introduction

1.1 Purpose

The purpose of this document is to give a precise and exhaustive explanation of the design choices for the deployment of the DREAM application. The analysis has been made to give an overview of the high level architecture, with the main components and their interactions, and then going into the details of the different components and their interactions, the runtime behaviour and the user interfaces provided by the application. Finally, the document includes a plan for the implementation, integration and testing activities. Together with the RASD document, this document will give the developers a complete overview of the DREAM application.

1.2 Scope

Here the scope of the application is briefly described again.
DREAM is an application meant to protect and improve agricultural production in the state of Telangana as it will include helpful online production data and organizational and environmental information.
There are 3 types of users: farmers, agronomists and Telangana's policy makers. Farmers, in order to get a successful production, will get through the application updates on the weather, soil and water conditions. The application will also provide them with discussion forums to reach other farmers and agronomists and suggestions on the most suitable crops to plant.
Agronomists are experts whose job is to periodically visit farm households, answer requests from farmers and monitor the environmental conditions. They are supposed to visit farms for which they are responsible for at least twice a year, so DREAM helps them with a digital agenda to confirm or modify their daily plan. They can also promote steering initiatives to enhance farmers' productions. Finally, Telangana's policy makers need to gather a whole picture of the production system and figure out if the agronomists' initiatives got the expected results. They can visualize the performances of each farmer and the conditions they are operating in to either send incentives to the best ones or give help for those in need.

1.3 Definitions, acronyms and abbreviations

1.3.1 Definitions

Soil moisture	percentage of water with respect to dry soil
Agronomist	professional in the science, practice, and management of agriculture and agribusiness
Best Practices Article	an article written by a TPM after receiving best practices forms by well-performing farmers. It has a title and a list of topics covered in order to be easily searchable by farmers.
Database Management System	it's the system which handles the connection between the application and data in the database
Data Warehouse	it's a secondary unit of storage to save data which are only read and on which aggregate queries are performed.

1.3.2 Acronyms

DBMS	Database Management System
DW	Data Warehouse
BPA	Best Practices Article
TPM	Telangana Policy Maker

1.3.3 Abbreviations

Rx	Functional requirement number x

1.4 Revision History

Version	Date	Description
1.0	08/02/2022	First version

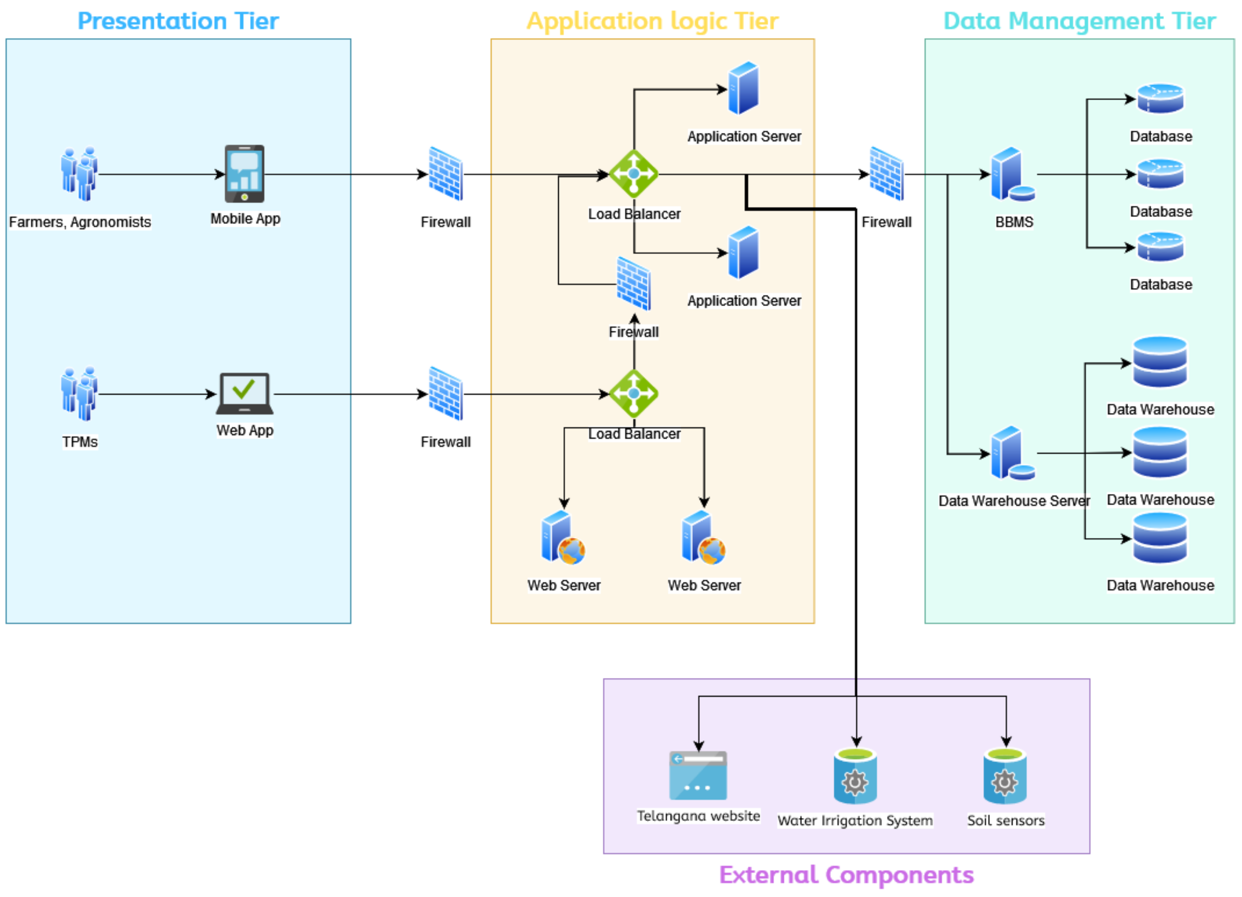
1.5 Reference Document

- Assignment document A.Y. 2021/2022 ("Requirement Engineering and Design Project: goal, schedule, and rules")
- Course slides

2 Architectural Design

2.1 Overview: High-level components and their interaction

The application will be developed using the client-server paradigm on a three-tiered architecture. The three layers of the application (Presentation, Application and Data) are divided into clusters of machines (i.e. tiers) that actually cooperate to provide a specific functionality. In this case we have three tiers and each tier is responsible for one of the three layers. The client tier is responsible (only) for the Presentation layer; therefore, in this architecture, the thin-client has been adopted considering the fact that the required client-side functionalities are limited. The UIs provided are just meant to show results and to allow clients to choose what they want. In particular there are two types of clients: a mobile app for farmers and agronomists who use the DREAM app and a website for TPMs. The Application tier takes care of the application layer encapsulating all is needed concerning the application logic. It receives the requests from the clients and handles them. It is also responsible for sending asynchronous notifications to the presentation layer when certain conditions are met. The Application tier communicates with the Data tier, responsible for the Data Access layer, that is the layer responsible for accessing the Database and the Data Warehouse and performing queries on them. The Data Warehouse is a component in the Data tier able to deal with historical data and aggregate data taken from the operational databases in a more efficient way than a traditional DBMS.
The image below shows a representation of the physical architecture of the DREAM system, highlighting the three-tier division among components. The communication between each tier occurs through the internet so a firewall between each of these connections is to be implemented in order to guarantee a proper level of security. Physical resources are replicated to achieve a high level of reliability and guarantee more availability and fault tolerance. Load balancers are to be implemented in order to manage the high number of concurrent requests properly.


Presentation Tier
Application logic Tier
Data Management Tier
Farmers, Agronomists
Mobile App
Firewall
TPMs
Web App
Firewall
Application Server
Load Balancer
Firewall
Application Server
Load Balancer
Web Server
Web Server
Firewall
BBMS
Database
Database
Database
Data Warehouse Server
Data Warehouse
Data Warehouse
Data Warehouse
Telangana website
Water Irrigation System
Soil sensors
External Components

2.2 Component View

The following diagram shows the component view of the DREAM application system. It highlights the three tiers division and shows how the different components interface between each other. In the presentation tier components of the three types of users are shown, which interface directly or through the Web Server (in the case of the web app) with the Router, which is the front-end component of the application logic tier. It handles all the requests which come from the different user components and routes them to the proper back-end component, providing an authorization-checking mechanism which guarantees that only proper users can access to allowed data. A more detailed description of the back-end components is provided in the following pages.

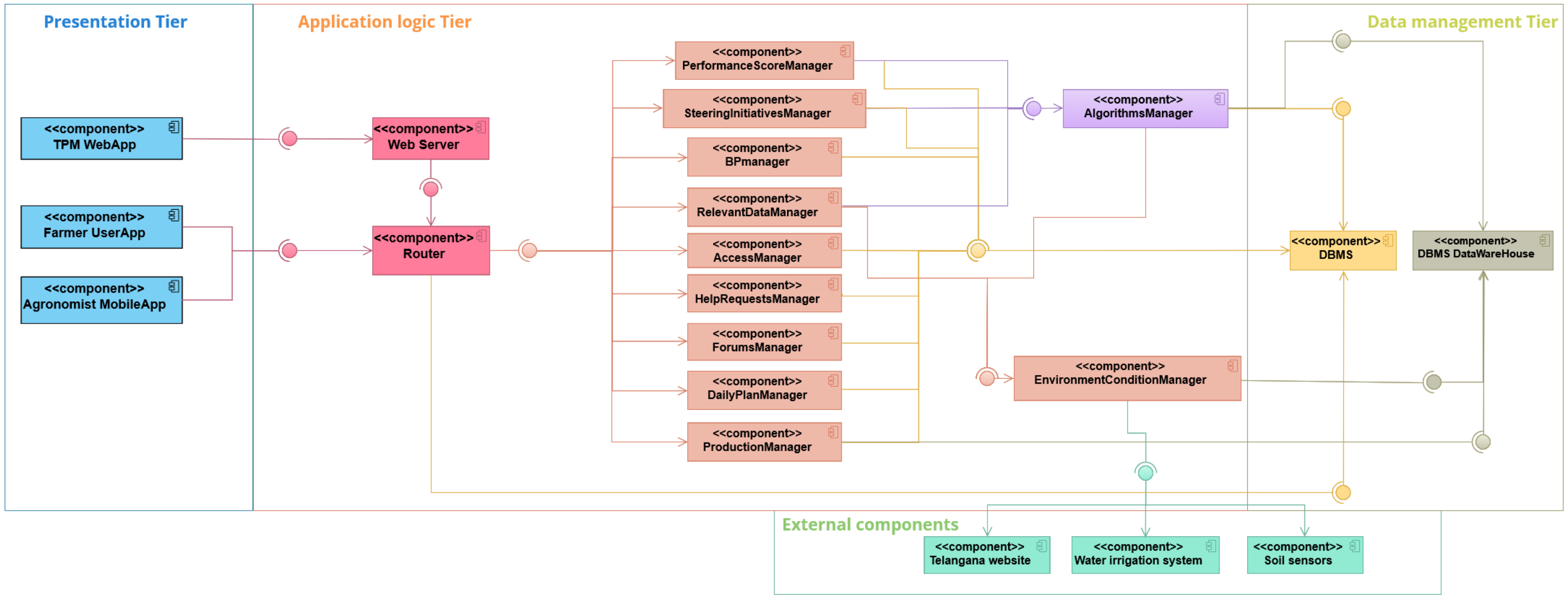
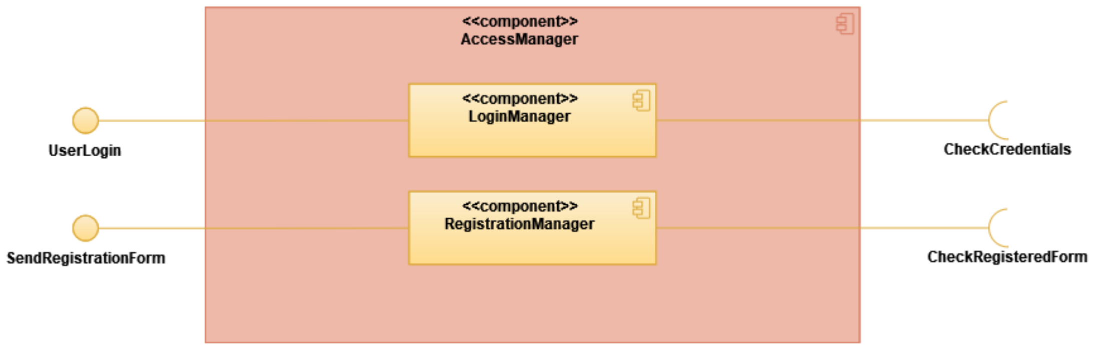

Presentation Tier
Application logic Tier
Data management Tier
External components
<<component>> TPM WebApp
<<component>> Farmer UserApp
<<component>> Agronomist MobileApp
<<component>> Web Server
<<component>> Router
<<component>> PerformanceScoreManager
<<component>> SteeringInitiativesManager
<<component>> BPmanager
<<component>> RelevantDataManager
<<component>> AccessManager
<<component>> HelpRequestsManager
<<component>> ForumsManager
<<component>> DailyPlanManager
<<component>> ProductionManager
<<component>> AlgorithmsManager
<<component>> EnvironmentConditionManager
<<component>> DBMS
<<component>> DBMS DataWareHouse
<<component>> Telangana website
<<component>> Water irrigation system
<<component>> Soil sensors

Figure 2.2.1: Component diagram

- Access Manager: this component handles the access to the application of a user not logged yet. It is made of two subcomponents:
  - LoginManager: it receives the login credentials from a user and access to the DBMS through the *CheckCredentials* interface to grant the access to the user.
  - RegistrationManager: it receives the registration form from an unregistered user with all relevant data and connects to the DBMS through the *CheckRegisteredForm* interface to manage the creation of a new user.



<<component>> AccessManager
<<component>> LoginManager
<<component>> RegistrationManager
UserLogin
SendRegistrationForm
CheckCredentials
CheckRegisteredForm

Figure 2.2.2: Access Manager component

- BPManager: this component handles whatever concerns the creation and publication of BPAs. It is made of two subcomponents:
  - BPFormManager: it is used by TPMs to send best practice forms to best performing farmers and by farmers to fill in the forms received. After filling, TPMs access them through this component. It manages both sending and filling processes. The component interfaces with the DBMS to create new forms, retrieve existing forms and update filled forms.
  - BPAManager: it's used by TPMs after receiving the best practices forms and it manages the writing and publishing process of new BPAs. Farmers also use this component to retrieve the BPAs they're interested in. The component interfaces with the DBMS to create and retrieve BPAs.

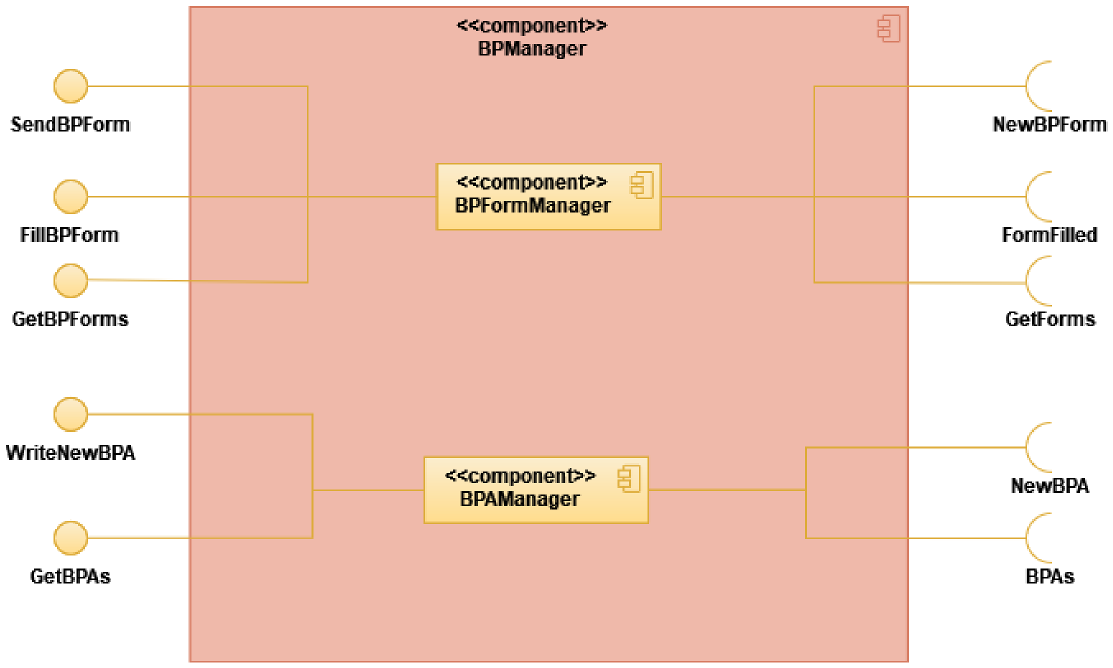


Figure 2.2.3: BPManager component

- PerformanceScoreManager: this component manages whatever concerns the performance analysis of farmers made by TPMs. It is made of three subcomponents:

- ProductionScoreRetriever: it retrieves all the scores of the latest farmers' productions computed by the algorithm implemented in the Algorithm Manager component, and it makes them available to the TPM.
- FarmerScore: it handles the update of farmers overall scores made by the TPM after proper analysis. It changes the scores with update queries in the DW.
- BadScoreReporter: it is used by TPMs when they want to report bad farmers to the agronomists responsible for their area. It manages the reporting phase and the communication with the agronomists

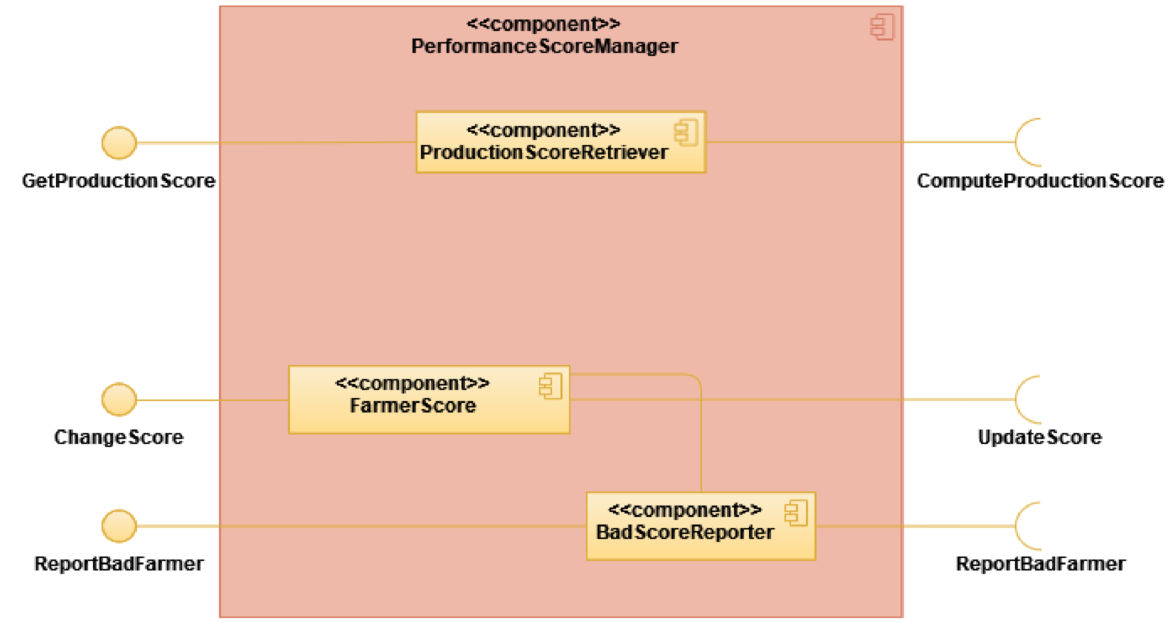


Figure 2.2.4: PerformanceScoreManager component

- RelevantDataManager: this component handles the retrieving of relevant data to different users of DREAM who need them. It is made of three subcomponents:
  - **WeatherForecastsRetriever:** it retrieves weather forecasts for the current day from the Telangana state website, and shows them to agronomists and farmers, who access these subcomponents.
  - **PersonalSuggestionsRetriever:** it retrieves personal suggestions for the current productions of farmers. They are computed in the algorithms manager component.
  - **FarmersDataRetriever:** it retrieves the farmers overall scores to show them to TPMs and to agronomists (who see also the underperforming ones marked with the "!").

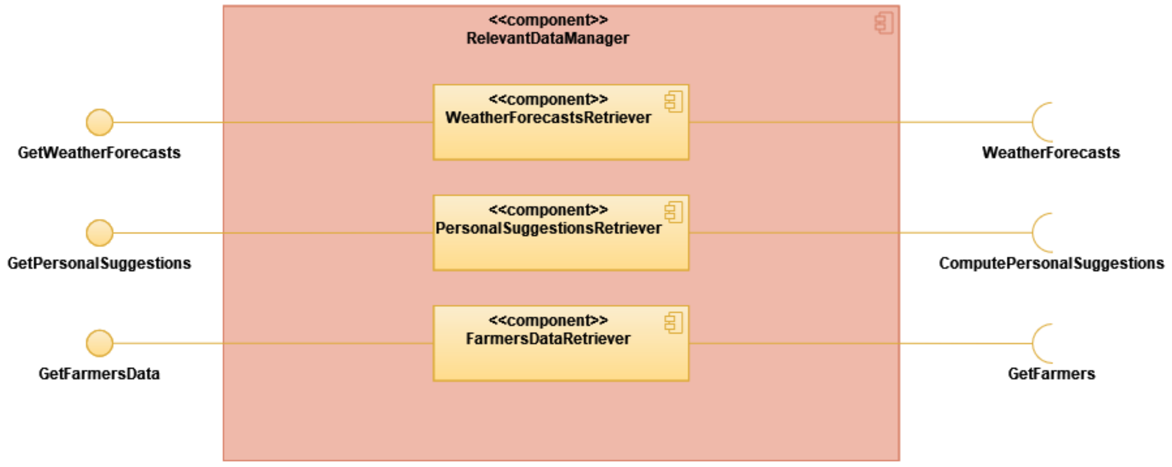


Figure 2.2.5: RelevantDataManager component

- SteeringInitiativesManager: this component handles whatever concerns the steering initiatives carried out by the agronomists. It is used by agronomists, who create new initiatives and add them in the database; and it is used by TPMs who see the scores of the initiatives concluded, computed by the Algorithms manager component

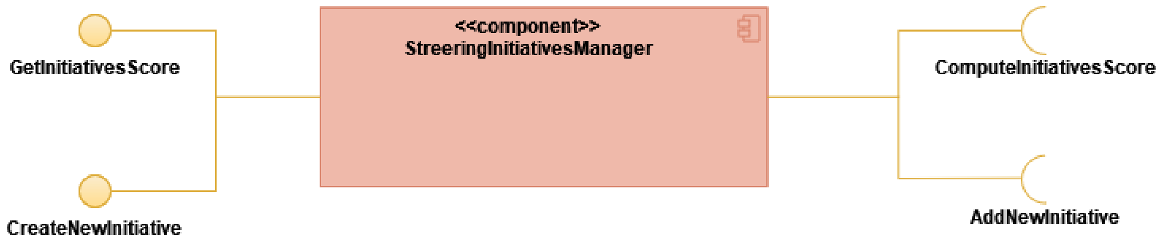


Figure 2.2.6: SteeringInitiativesManager component

- AlgorithmsManager: this component manages all the algorithms used in the application. Three algorithms are computed by this component:
    - **Production score algorithm**: it is the algorithm which produces as outcome the score of every production concluded. It takes data from the data warehouse and gives the results to the *RelevantDataManager.*
    - **Initiative score algorithm:** it gives the score of each steering initiative by computing the differential between the score of the same type of production before and after the initiative. Data is retrieved from the data warehouse while the result is given to the *RelevantDataManager.*
    - **Personal suggestions algorithm:** it creates personal suggestions that are then shown to farmers. It computes data coming both from the data warehouse (concerning past productions) and from the *EnvironmentConditionManager.*

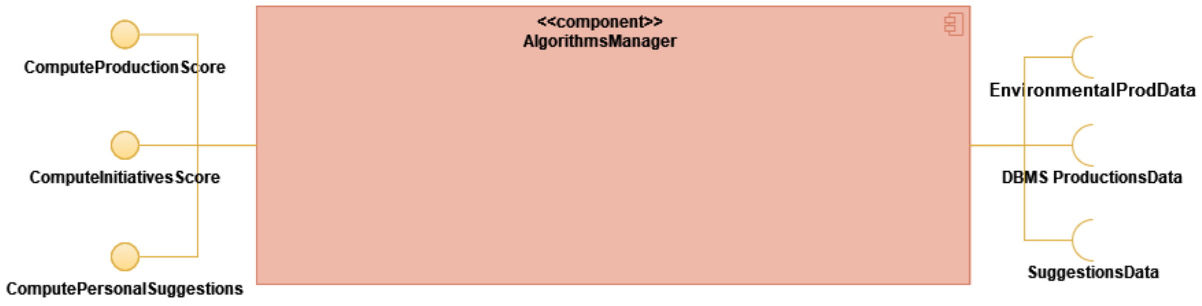


Figure 2.2.7: AlgorithmsManager component

- Production Manager: This component is dedicated to the farmer since she/he is the one who needs to upload data about her/his production. It allows farmers to register a new production, modify an existing one or report a problem encountered during one of them. When a production is completed, they can close it through the CloseProduction interface, that will be then saved in the data WareHouse.

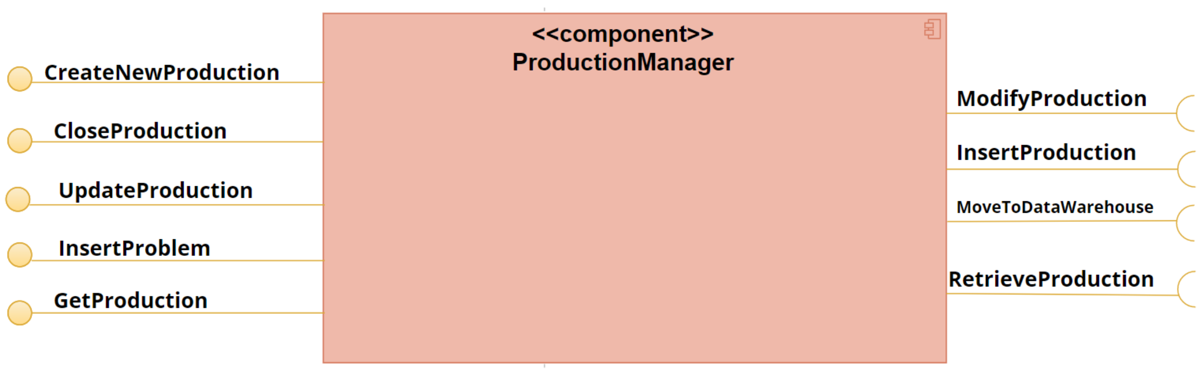


Figure 2.2.8: ProductionManager component

- HelpRequestManager: This component takes care of the management of the farmers' help requests. The requests and their answers are recorded in the DataBase. The component is divided in 2 subcomponents:
  - RequestRecorder: used by farmers to insert a new help request regarding a problem they are facing
  - AnswerRecorder: used by agronomists to answer to open help requests

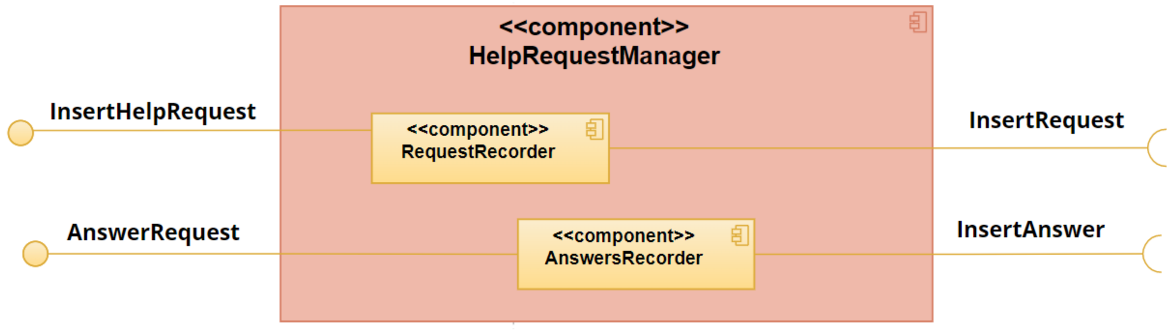


Figure 2.2.9: HelpRequestManager component

- ForumManager: This component accomplishes the task of managing the forums opened by farmers. Farmers can answer in the forum. The answers are saved in the database.

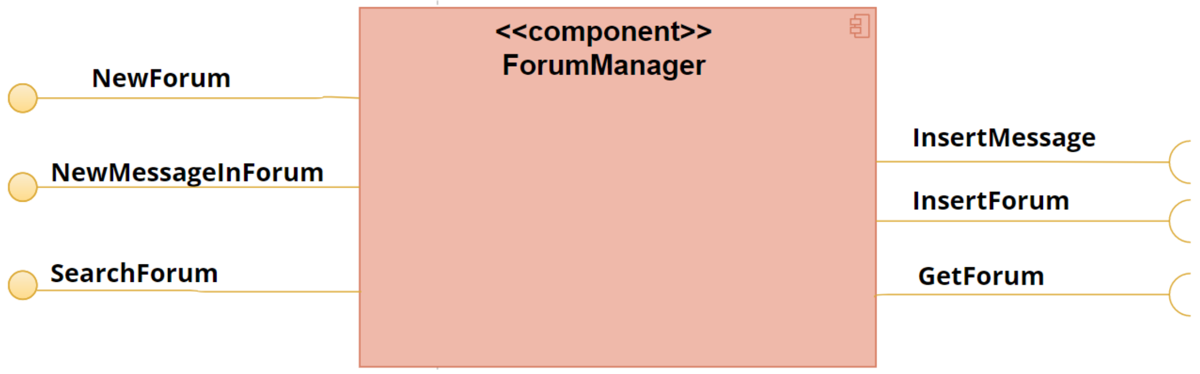


Figure 2.2.10: ForumManager component

- DailyPlanManager: This component is meant to support the agronomists with their visits organization. It has 2 subcomponents:
  - VisitsManager: takes care of adding and modifying visits and saving them in the database
  - PlanManager: used by the agronomist to confirm a whole plan of a day and notifies him/her if there is a non-confirmed daily plan of a past day

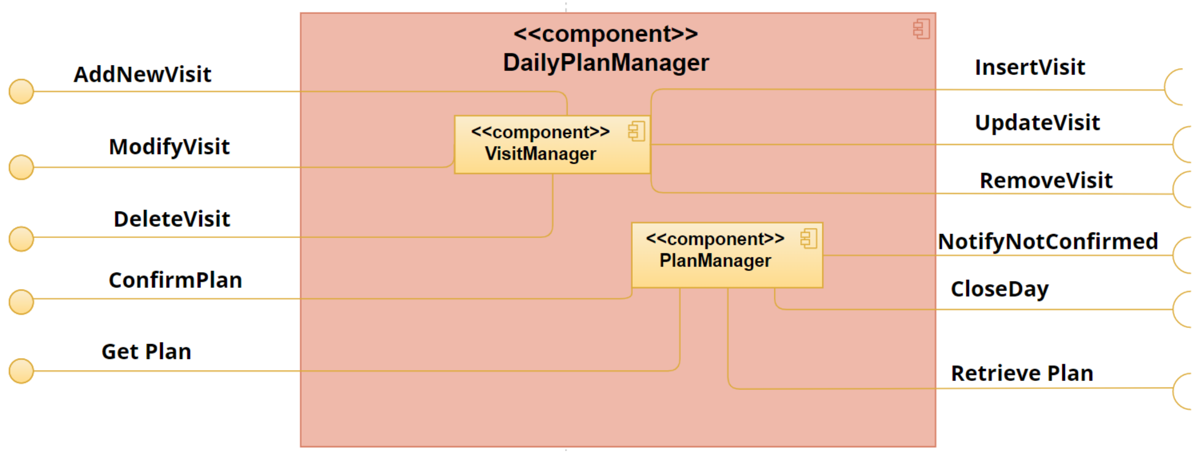


Figure 2.2.11: DeilyPlanManager component

- <u>EnvironmentConditionManager:</u> This component is responsible for the retrieval of environmental and weather conditions from external components and saving them into the data warehouse for future analysis. This is done because we assume that external resources do not maintain an internal record and always update their exposed information. It is composed of 3 subcomponents which all co-operate with external resources and the data:
    - <u>WeatherForecast:</u> retrieves data from Telengana's official website
    - <u>Water:</u> retrieves data from the irrigation sensors
    - <u>Soil:</u> retrieves data from the soil sensors

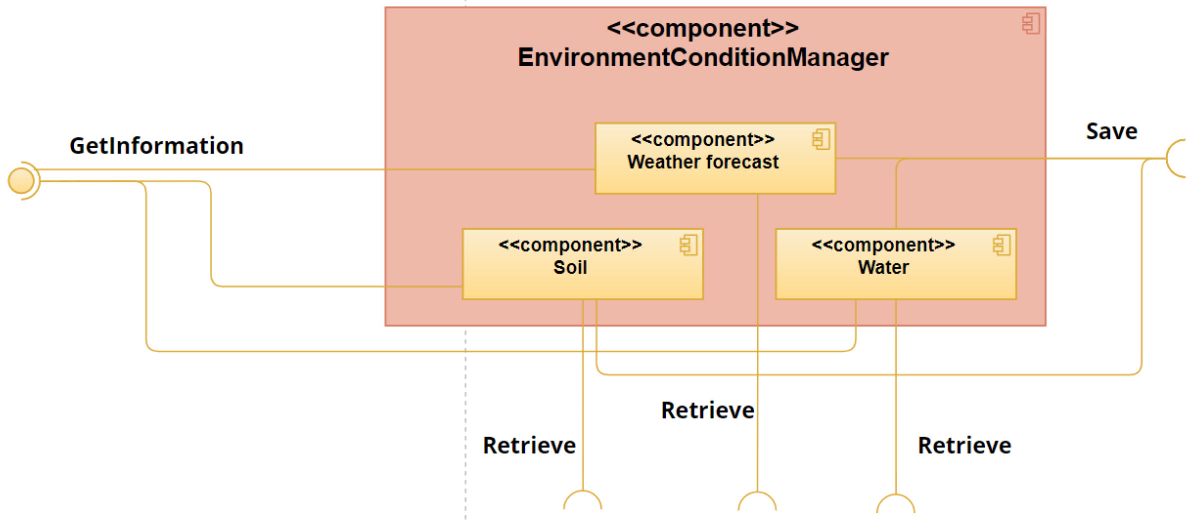
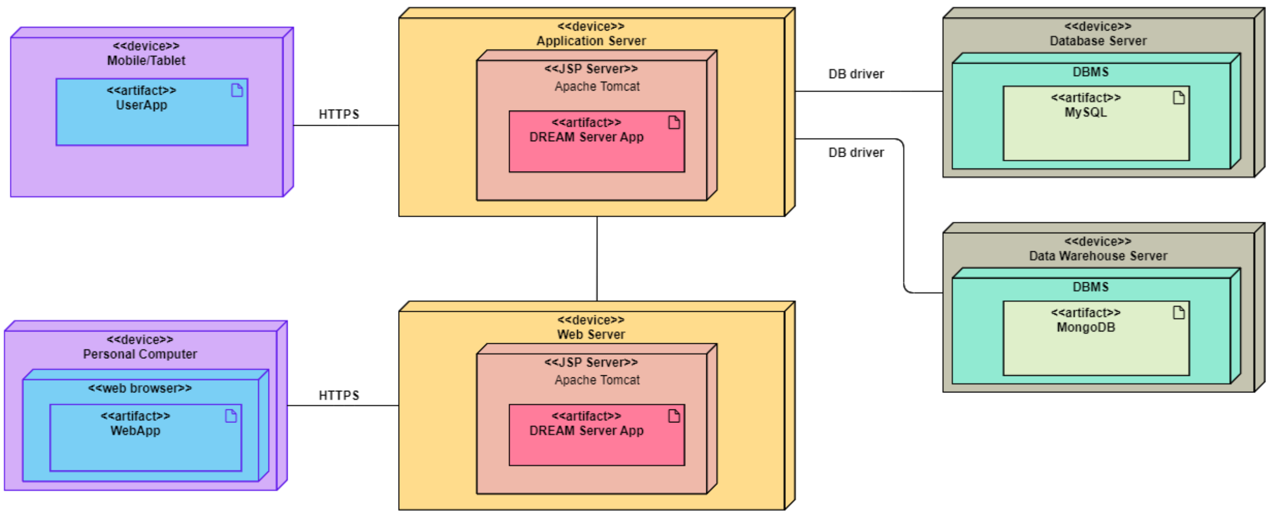


Figure 2.2.12: EnvironmentConditionManager component

2.3 Deployment View

The System architecture is divided in 3 tiers and it is based on the JEE Framework.
The choice of a three-tier architecture has been made for the main reason that since each tier runs on its own infrastructure, each tier can be developed simultaneously by separate developers and can be updated or scaled as needed without impacting the other tiers.
In the following, the DREAM deployment diagram is presented: the image below shows the execution architecture of the system and represents the distribution of software artifacts to deployment targets (nodes). Artifacts represent pieces of information that are used or are produced by a software development process and are deployed on nodes that can represent either hardware devices or software execution environments. In the diagram, external systems are not represented to focus only on the components that host the core functionalities of the application or on the components for which the deployment is effectively executed.



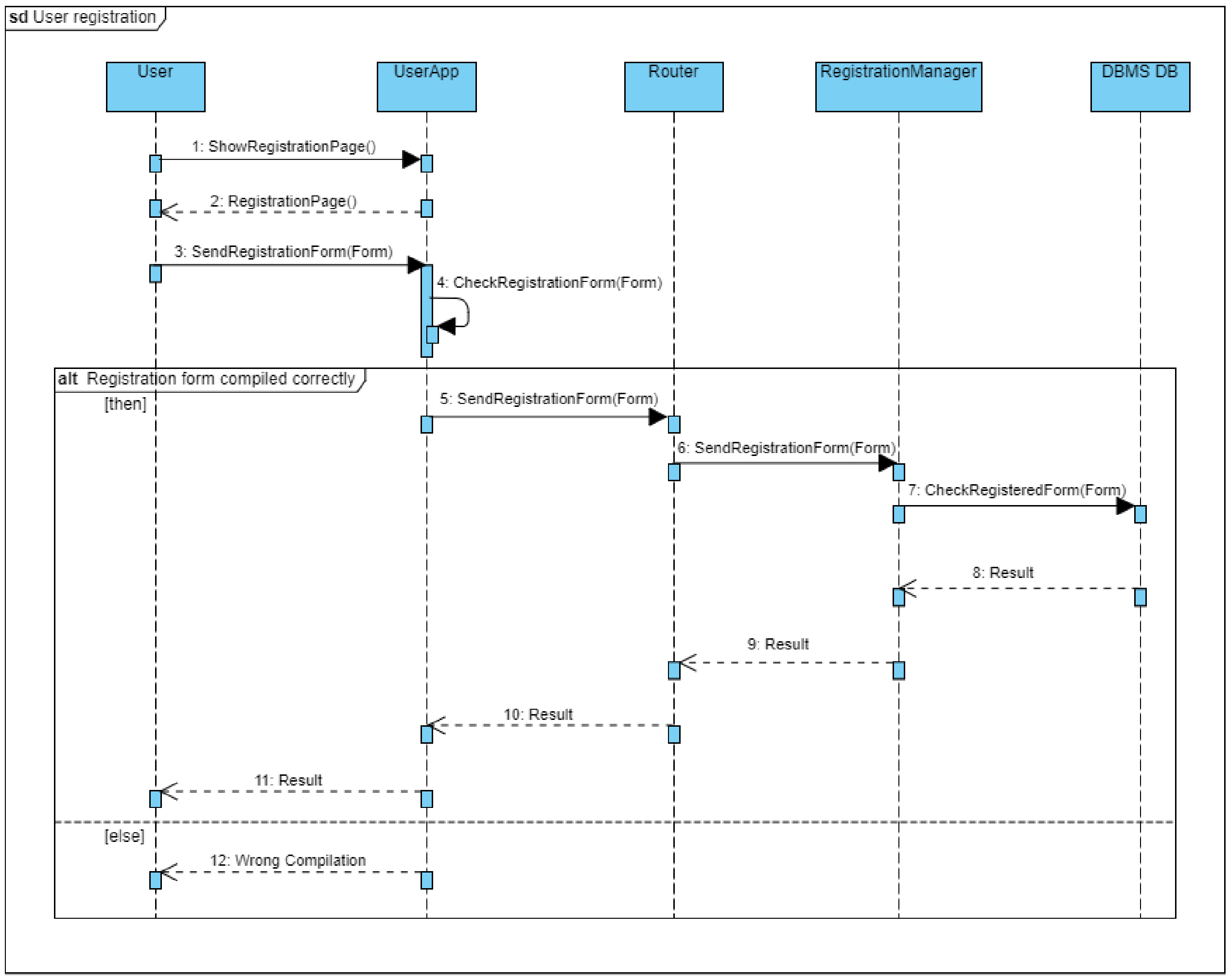
Tier 1: Presentation Tier
In this tier the presentation logic is deployed. Farmers, agronomists and TPMs must be provided with a mobile application on their smartphones. The user App is installed on compatible Android or iOS phones, in order to make it available on most of the devices. TPMs must be provided with a dedicated Web App on their computers. Also in this case, the Web App must be compatible with at least Google Chrome, Safari, Opera, DuckDuckGo, Ecosia, Microsoft Edge and Firefox, most used web browsers.

Tier 2: Application Logic Tier
In this tier the application logic is deployed. Tier 2 includes both the Application and the Web server. The Application server implements the business logic, it handles the user's requests and also provides answers for all the offered services. The application server is directly addressed by Users through the Mobile App. It also handles some requests that are forwarded by the Web Server and sent by TPMs. To provide all these services, the Application Server must communicate with the Database and the DataWarehouse. The DREAM Server App is installed in a dedicated server with a running instance of Apache Tomcat.

Tier 3: Data Management Tier.
It comprises the Database and the Data Warehouse. A DBMS is needed and a good choice for the Database can be Oracle Database because it is widely used, it is secure, it offers good results in terms of performance, scalability and reliability, it also supports large databases and reduces the CPU time to process data. The communication with the DBMS is performed via JPA.
The suggested architecture for the Data Warehouse is instead MongoDb, one of the most used documental databases.

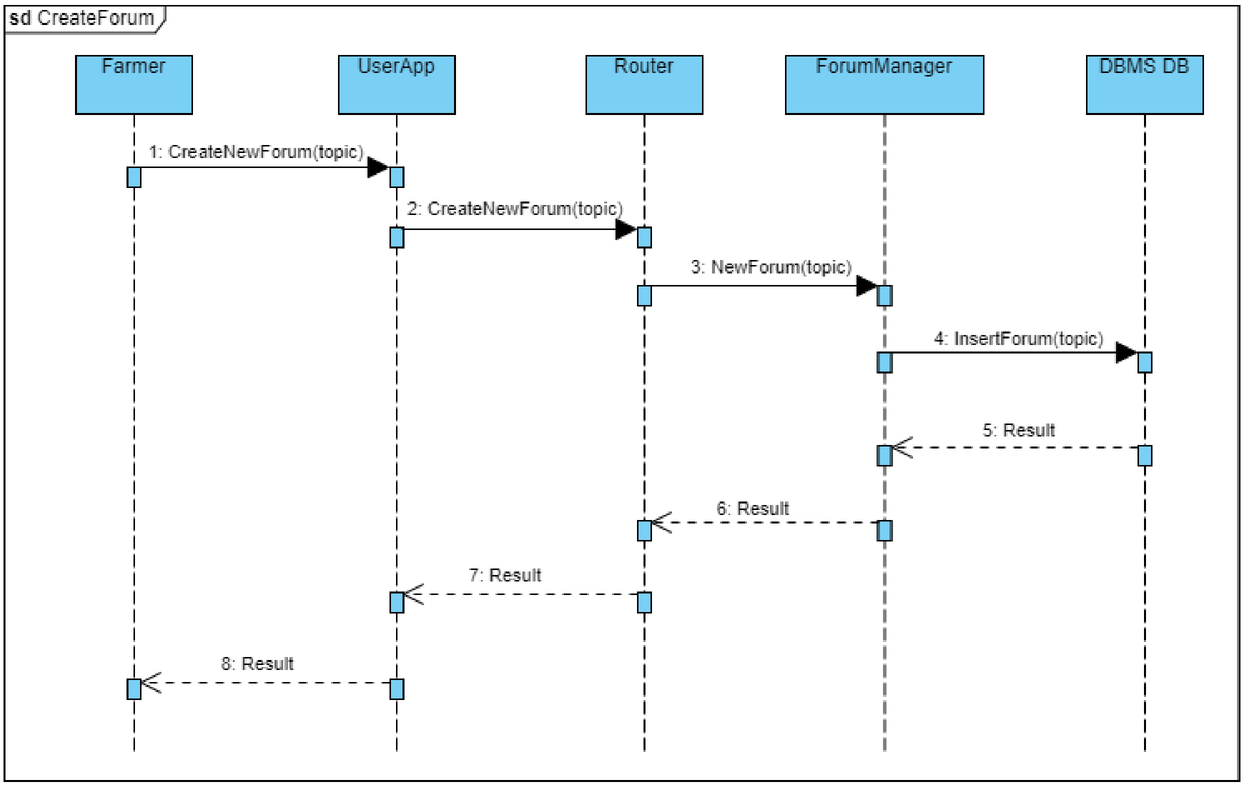
2.4 Runtime View
The following sequence diagrams describe the interactions between the main components of the system when utilizing the most common features. This is still a high-level description of the actual interactions so that they can be slightly modified during the development process.

2.4.1 User Registration
The User asks for the registration page to the UserMobileApp, then compiles the registration form and sends it. Before sending the registration form to the Router, the UserApp controls whether every value in the form is correctly filled. This task is very easy to manage so we decided to let UserApp do it, in order to filter the forms that are sent to the application server.
After this check, if the form was compiled correctly the message is forwarded to the Router and to theRegistrationManager, which queries the Database to check if the values are correct and if they have already been registered. The result, which can be a confirmation of the registration or an error because for instance the selected username is already in use, is then forwarded to the User.


sd User registration
User
UserApp
Router
RegistrationManager
DBMS DB
1: ShowRegistrationPage()
2: RegistrationPage()
3: SendRegistrationForm(Form)
4: CheckRegistrationForm(Form)
alt Registration form compiled correctly
[then]
5: SendRegistrationForm(Form)
6: SendRegistrationForm(Form)
7: CheckRegisteredForm(Form)
8: Result
9: Result
10: Result
11: Result
[else]
12: Wrong Compilation

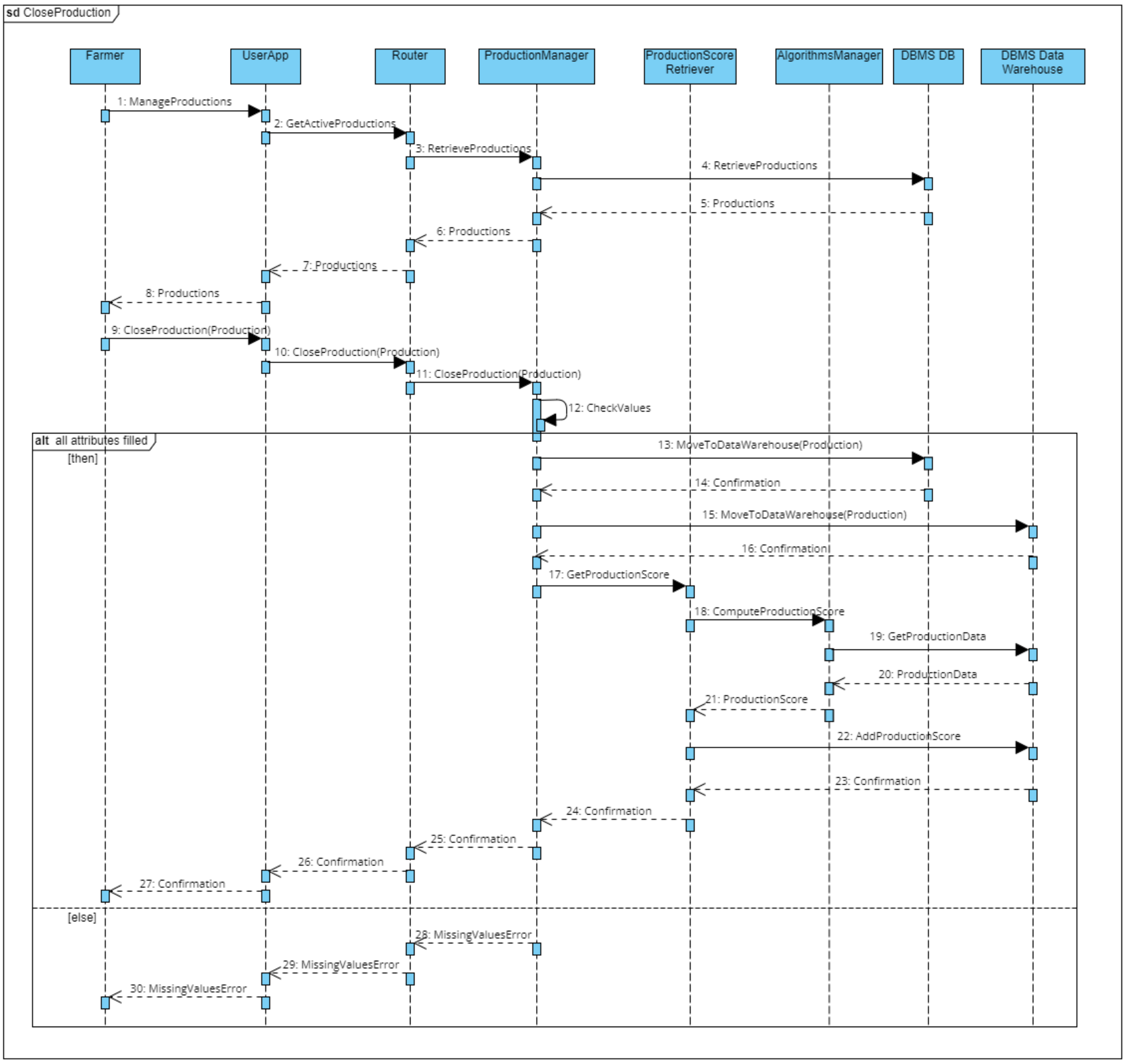
2.4.2 Create Forum

When a farmer wants to create a new forum, he selects the appropriate entry in the UserApp. The command CreateNewForum arrives then to the ForumManager, which inserts the new forum in the database.



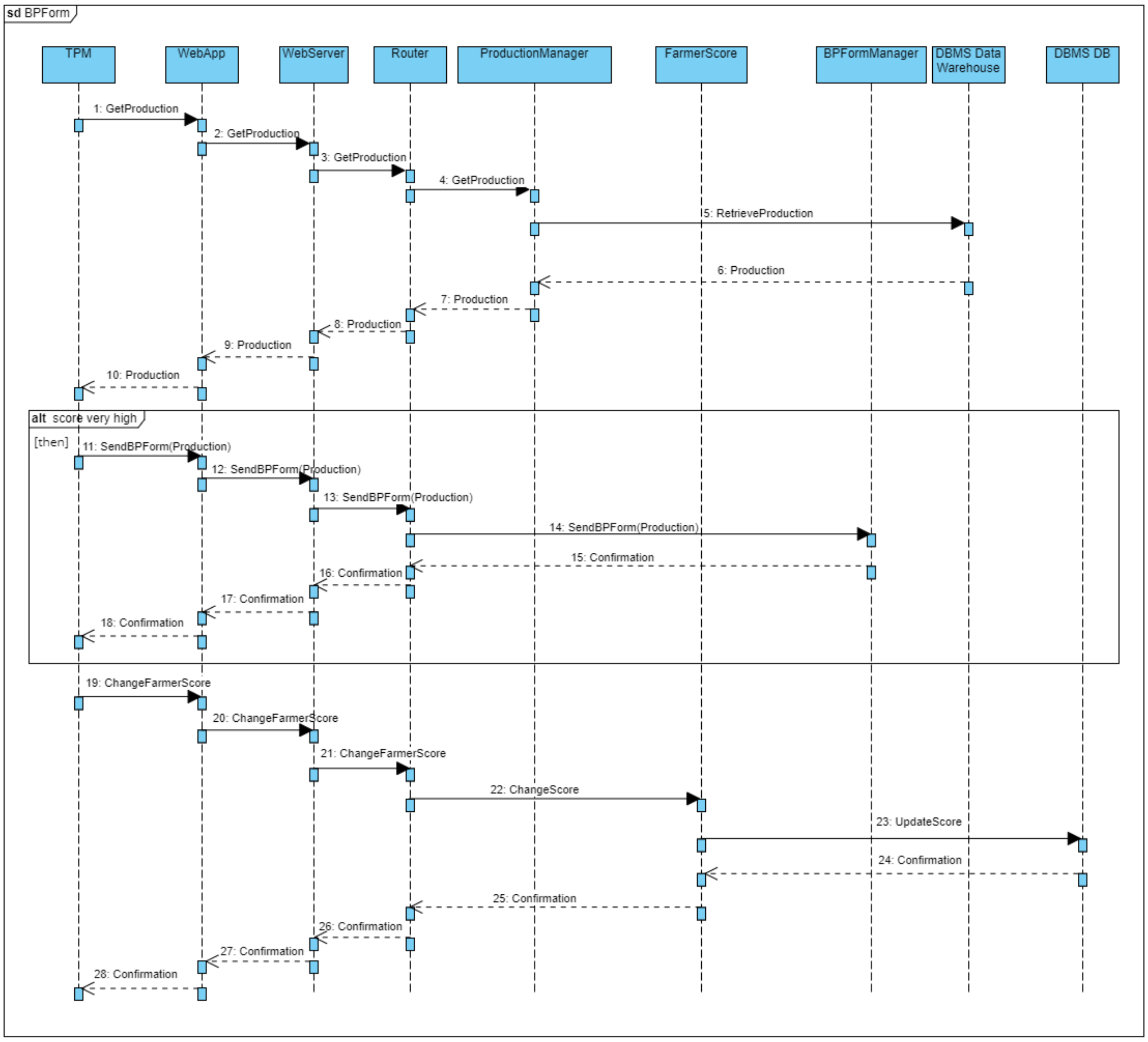
2.4.3 Close Production

To close a production, the farmer has first to query the Database to retrieve his active productions. He then can select one of them and, while launching the command CloseProduction, he has to fill all the needed values in the production. The request is forwarded to the ProductionManager, that checks the values in the production and, if everything is filled and correct, it moves the production from the Database to the Data Warehouse. As soon as this operation ends the AlgorithmsManager also computes the score of the production.



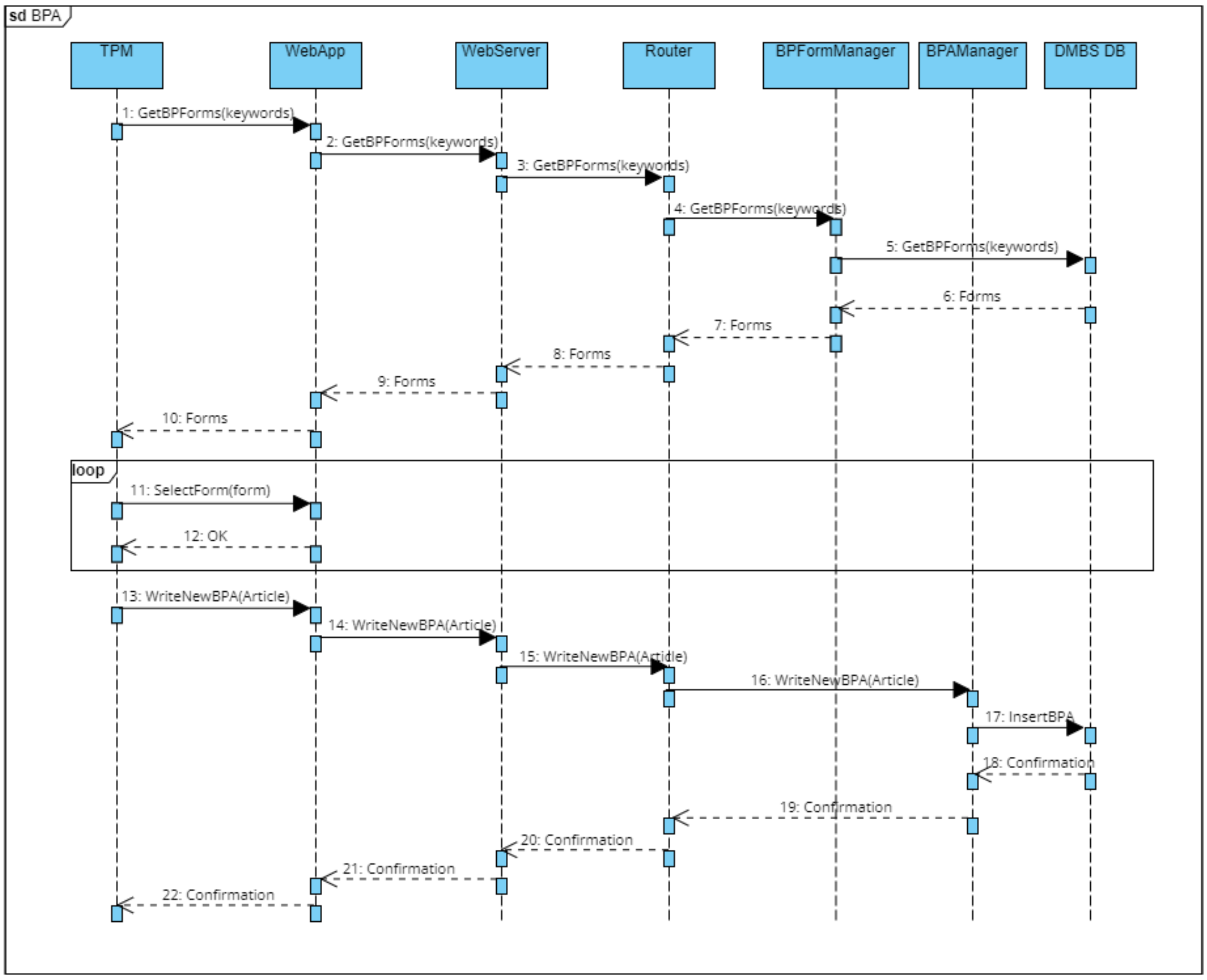
2.4.4 Send BP form

The TPM in the webpage dedicated to productions can see the latest production that has been evaluated by the algorithm with its respective score. He now has the option to send a BestPractice form to compile to the farmer who ended the production and also to change his overall score, considering also this last production.



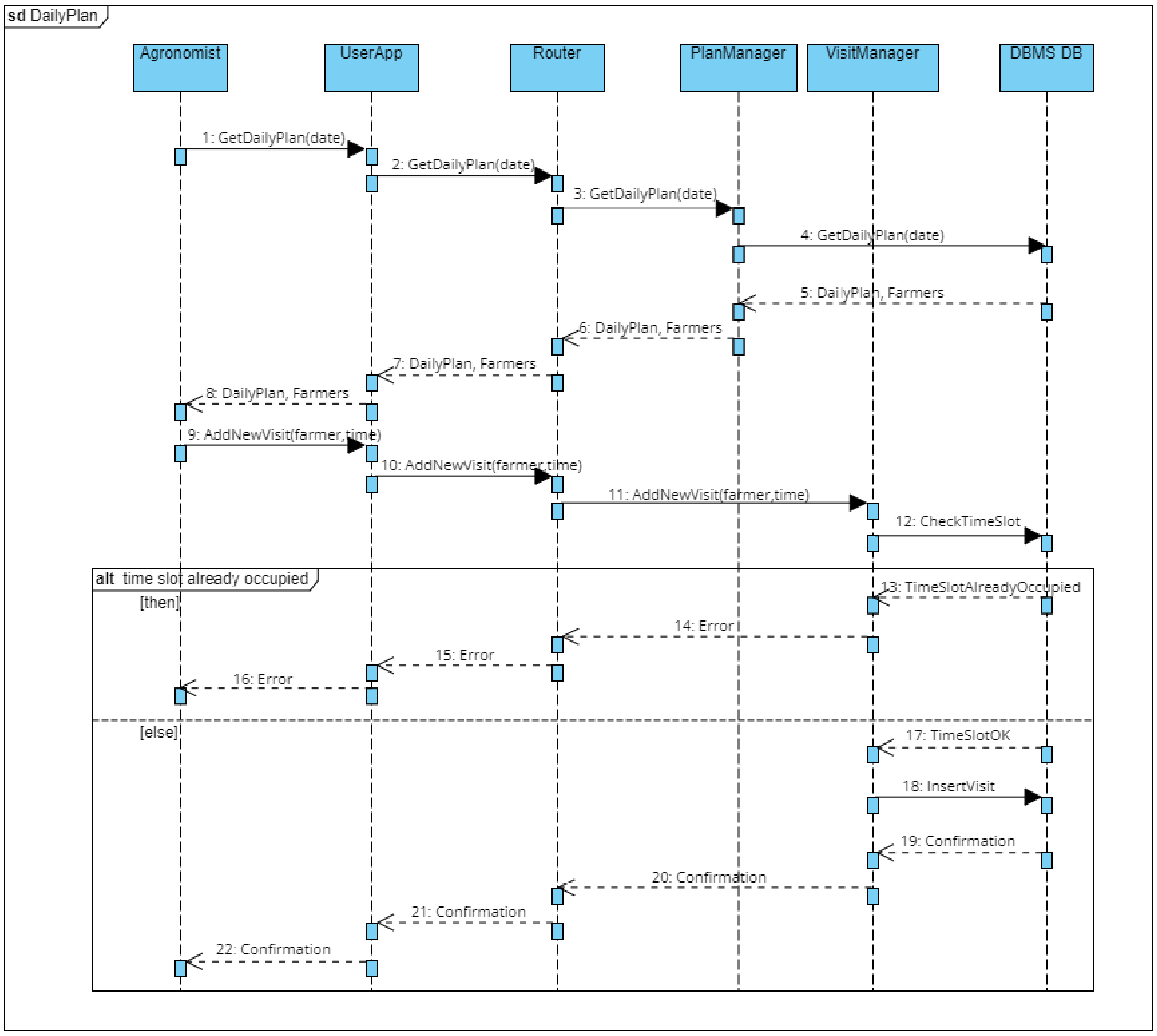
2.4.5 Create BPA

The TPM can retrieve a list of BP forms, and with these he can write a Best Practice Article that is then inserted in the Database.



2.4.6 Add visit Daily Plan

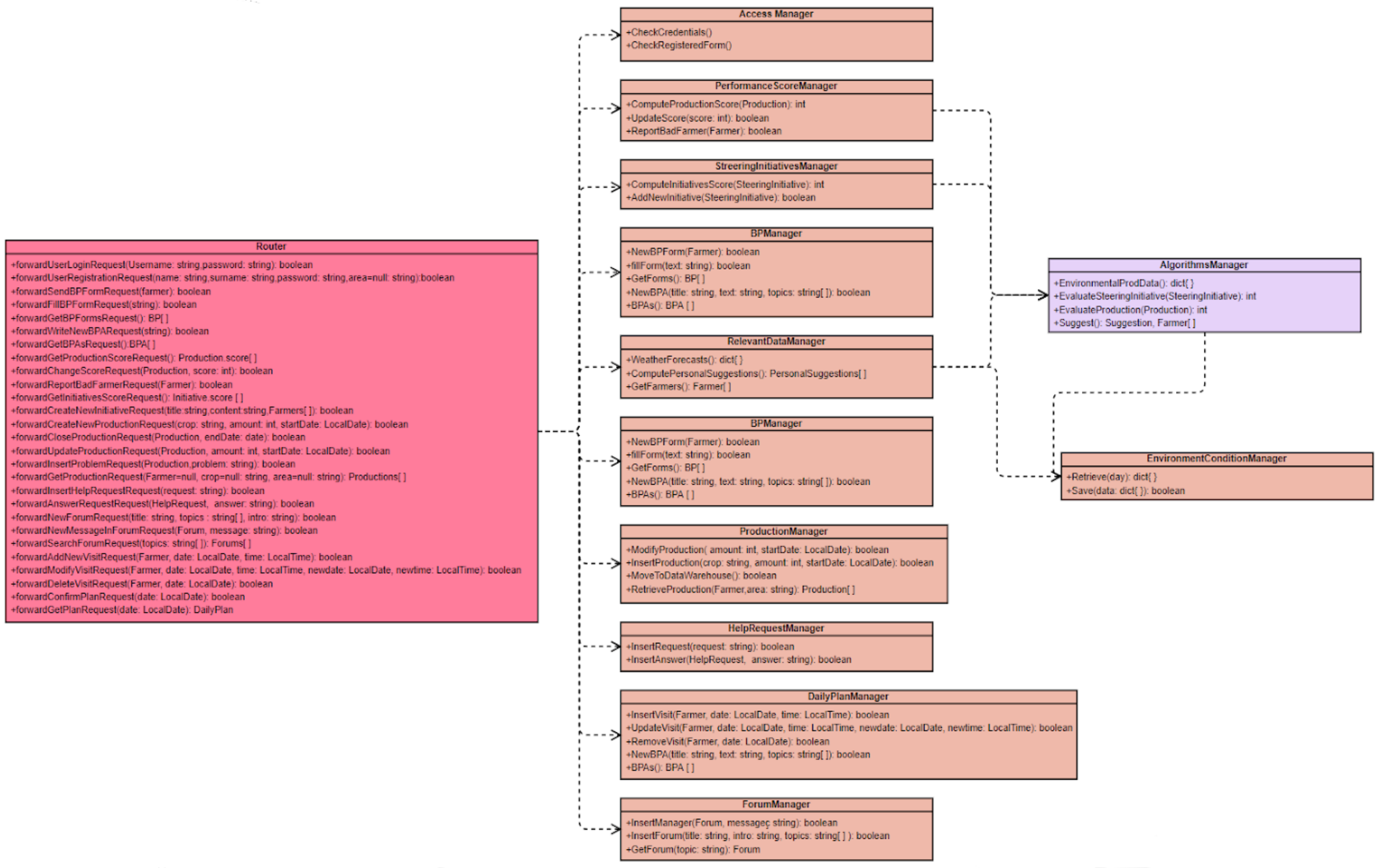
The agronomist asks for the daily plan referred to a specific date and receives as answer a Daily Plan and a list of farmers. He then can add a visit to the Daily Plan by selecting a farmer and a time for the visit. The Database is then queried and if the timeslot selected is free, then the visit is added to the Daily Plan, otherwise an error returns to the agronomist.



2.5 Component Interfaces

In the following figure the component interfaces, with reference to what was shown in the Component diagram, are represented. The arrows represent a dependency relation. Each component is described by its invocable methods:

- Each of invocable methods of the Router are responsible of forwarding a certain request to the associated component; for example ForwardUserLoginRequestData has the parameters username and password, which must be forwarded to the AccessManagerComponent, invoking its CheckCredentials method.
- The AlgorithmManager’s methods are used to retrieve data - data that concern what the caller component has asked to analyse - from the Data Warehouse, make calculations and save the results, usually scores, in the Database.
- The EnvironmentConditionManager asks at regular intervals of time the external resources to gather information and store it in the Data Warehouse.


Access Manager
+CheckCredentials()
+CheckRegisteredForm()
PerformanceScoreManager
+ComputeProductionScore(Production): int
+UpdateScore(score: int): boolean
+ReportBadFarmer(Farmer): boolean
StreeringInitiativesManager
+ComputeInitiativesScore(SteeringInitiative): int
+AddNewInitiative(SteeringInitiative): boolean
BPManager
+NewBPForm(Farmer): boolean
+fillForm(text: string): boolean
+GetForms(): BP[ ]
+NewBPA(title: string, text: string, topics: string[ ]): boolean
+BPAs(): BPA [ ]
RelevantDataManager
+WeatherForecasts(): dict{ }
+ComputePersonalSuggestions(): PersonalSuggestions[ ]
+GetFarmers(): Farmer[ ]
BPManager
+NewBPForm(Farmer): boolean
+fillForm(text: string): boolean
+GetForms(): BP[ ]
+NewBPA(title: string, text: string, topics: string[ ]): boolean
+BPAs(): BPA [ ]
ProductionManager
+ModifyProduction(amount: int, startDate: LocalDate): boolean
+InsertProduction(crop: string, amount: int, startDate: LocalDate): boolean
+MoveToDataWarehouse(): boolean
+RetrieveProduction(Farmer,area: string): Production[ ]
HelpRequestManager
+InsertRequest(request: string): boolean
+InsertAnswer(HelpRequest, answer: string): boolean
DailyPlanManager
+InsertVisit(Farmer, date: LocalDate, time: LocalTime): boolean
+UpdateVisit(Farmer, date: LocalDate, time: LocalTime, newdate: LocalDate, newtime: LocalTime): boolean
+RemoveVisit(Farmer, date: LocalDate): boolean
+NewBPA(title: string, text: string, topics: string[ ]): boolean
+BPAs(): BPA [ ]
ForumManager
+InsertManager(Forum, messageç string): boolean
+InsertForum(title: string, intro: string, topics: string[ ]): boolean
+GetForum(topic: string): Forum
Router
+forwardUserLoginRequest(Username: string,password: string): boolean
+forwardUserRegistrationRequest(name: string,surname: string,password: string,area=null: string):boolean
+forwardSendBPFormRequest(farmer): boolean
+forwardFillBPFormRequest(string): boolean
+forwardGetBPFormsRequest(): BP[ ]
+forwardWriteNewBPARequest(string): boolean
+forwardGetBPAsRequest():BPA[ ]
+forwardGetProductionScoreRequest(): Production.score[ ]
+forwardChangeScoreRequest(Production, score: int): boolean
+forwardReportBadFarmerRequest(Farmer): boolean
+forwardGetInitiativesScoreRequest(): Initiative.score [ ]
+forwardCreateNewInitiativeRequest(title:string,content:string,Farmers[ ]): boolean
+forwardCreateNewProductionRequest(crop: string, amount: int, startDate: LocalDate): boolean
+forwardCloseProductionRequest(Production, endDate: date): boolean
+forwardUpdateProductionRequest(Production, amount: int, startDate: LocalDate): boolean
+forwardInsertProblemRequest(Production,problem: string): boolean
+forwardGetProductionRequest(Farmer=null, crop=null: string, area=null: string): Productions[ ]
+forwardInsertHelpRequestRequest(request: string): boolean
+forwardAnswerRequestRequest(HelpRequest, answer: string): boolean
+forwardNewForumRequest(title: string, topics : string[ ], intro: string): boolean
+forwardNewMessageInForumRequest(Forum, message: string): boolean
+forwardSearchForumRequest(topics: string[ ]): Forums[ ]
+forwardAddNewVisitRequest(Farmer, date: LocalDate, time: LocalTime): boolean
+forwardModifyVisitRequest(Farmer, date: LocalDate, time: LocalTime, newdate: LocalDate, newtime: LocalTime): boolean
+forwardDeleteVisitRequest(Farmer, date: LocalDate): boolean
+forwardConfirmPlanRequest(date: LocalDate): boolean
+forwardGetPlanRequest(date: LocalDate): DailyPlan
AlgorithmsManager
+EnvironmentalProdData(): dict{ }
+EvaluateSteeringInitiative(SteeringInitiative): int
+EvaluateProduction(Production): int
+Suggest(): Suggestion, Farmer[ ]
EnvironmentConditionManager
+Retrieve(day): dict{ }
+Save(data: dict{ }): boolean

2.6 Selected architectural styles and patterns

Dream will be a 3-tiers application in which the tiers respectively consist of Client (thin), Server (with both the Application and the Web Server) and Data tier.
These 3 tiers will be connected via internet and will exchange data according to the following protocol:
Concerning the client-server architecture, TPMs will use the dedicated web app connected to the web server which communicates with the application server. Agronomists and farmers will instead use the mobile application that communicates directly with the application server. All messages exploit the HTTP protocol, using TLS when dealing with sensitive data in order to guarantee the security and the reliability of the connection and also to authenticate the identity of the communicating parties. Regarding the format in which data will be transmitted, JSON will be used because it is suitable for the data interchange between client and server and, despite its simplicity, it is enough to satisfy our purposes.

2.6.1 Design Patterns

MVC (Model–view–controller)

The Model-View-Controller pattern is used in order to obtain a clear division between the internal representation of information and the way in which information is presented to the user and the business logic of the system. This is one of the most common and effective ways to drastically reduce the level of coupling between the various parts of the system.

Other design patterns

• Facade pattern: supplies a single interface to a set of interfaces within the system.
• Observer pattern: notifies the subscribed objects when the state of the observed object is updated.
• Proxy pattern: allows object level access control by acting as a pass through entity or a placeholder object.
• Singleton pattern: provides a single instance of a class that is able to coordinate all the actions across the system.

Other design decisions

Thin Client

Just little of the logic is implemented client-side, for instance the functions that check if the fields of a form (es. Registration) are filled with proper values and that mandatory fields are not empty. All the computation is to be made on the Application Server. The choice of using Thin Clients it's also due to the fact that the Mobile App is intended to run on all kinds of smartphones, also the ones with underperforming hardware.

Relational Database

We opted for a relational database to make it easy to query the system and to deal with many transactions. Furthermore, a relational database can guarantee the ACID properties for the transactions.

NoSQL Data Warehouse

We chose to store closed productions in a Data Warehouse combined with MongoDB in order to make it easier to analyse those data. The combination of a documental database and a traditional data warehouse allows traditional analytical applications to come alive with the flexibility and new data from MongoDB and the MongoDB repository can easily access and import data from the data warehouse.
In this way production data are easily retrieved and all together when needed.

RESTful API

Representational State Transfer (REST) is a set of commands commonly used with web transactions. It allows the usage of HTTP to obtain data and generate operations on that data in all possible formats, such as XML and JSON. Since the application is data-centric and the main goal of DREAM users is to learn information from data, this API perfectly fits for this purpose. In fact, it is resource-centric and guarantees strong separation between client and server. It is stateless, in the sense that no client context is saved in the server memory, which is good because it doesn't overload the application server (considering the big number of users that can access the application simultaneously) but still suits with the application requirements since users only dialogue with the server to obtain data or to put data in the database, so every request can be handled as standalone. Authentication is still provided through OAuth protocol, guaranteeing that no unauthorized user can access data.

2.7 Other design decisions

2.7.1 Algorithms in the application

In this section are described the algorithms used to automate some calculations, ease the performances' analysis process for the TPMs and create, exploiting an AI, useful suggestions for farmers.

2.7.1.1 Production score algorithm

The score is a positive number that indicates how well a production has performed, in particular its efficiency. The factors that affect the score, excluding the amount produced, can be divided into 2 groups: human controllable and environmental. The crop type, area cultivated and amount of water used are decided by the farmer. On the other hand, (critical) weather conditions and soil humidity do not depend on the farmer. The crop type is translated in the formula as a coefficient that expresses the intrinsic difficulty of cultivating it. The efficiency should not be affected by the ground extent so, the score is divided by the cultivated area. Weather conditions and soil humidity are taken into account considering some thresholds that boost the production's score in case they have been unfavourable.

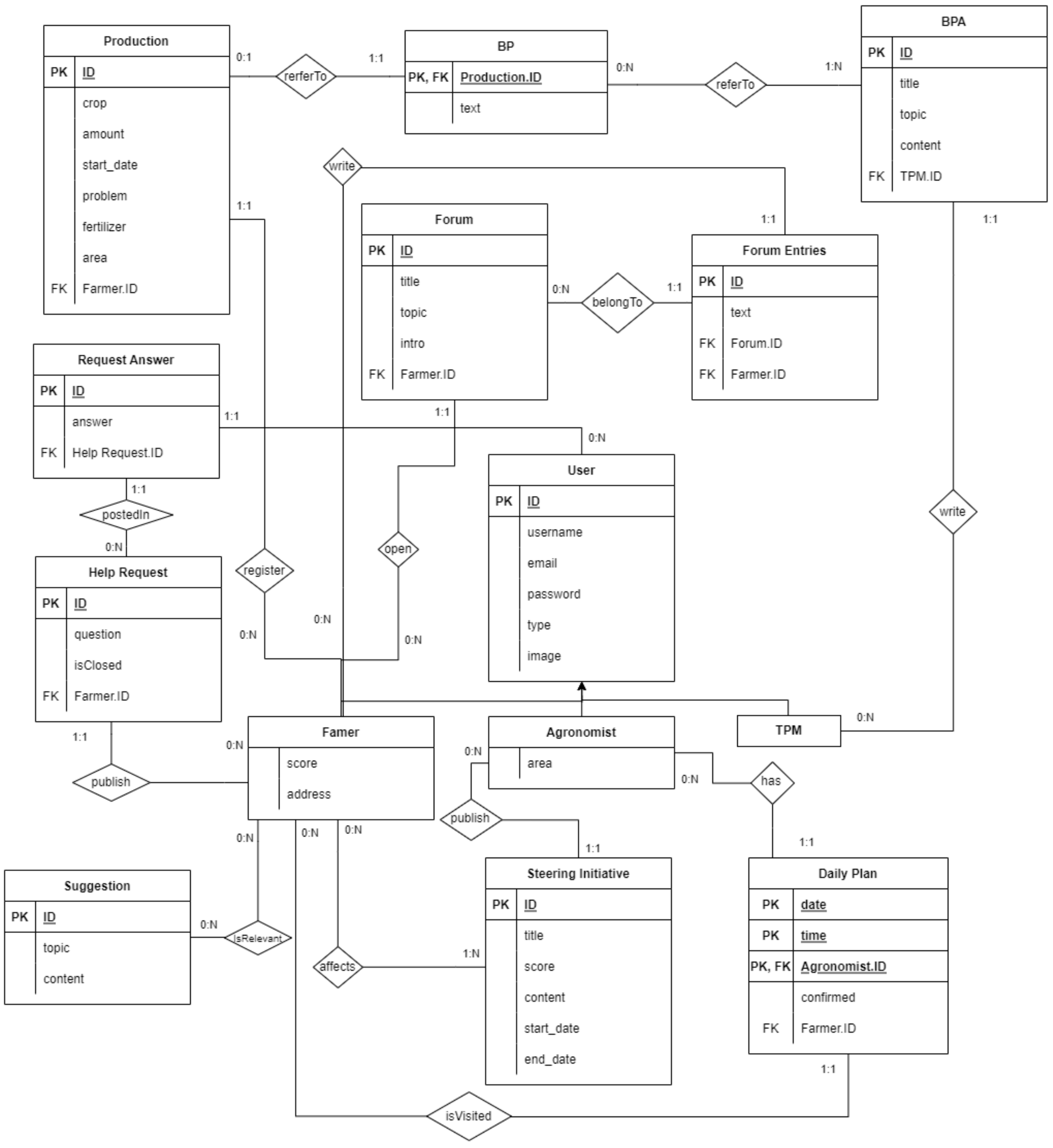
2.7.1.2 <u>Steering initiative score algorithm</u>
The score shows how a proposed initiative suggested by an agronomist affected the farmers' productions. The initiative score is a delta of the productions score - of those farmers affected by it - that began after the introduction of the initiative. The algorithm is rather simple: consider only the farmers affected by the initiative, sum the production scores of the previous period of all the farmers and subtract it from the sum of all the production scores of the considered period.

2.7.1.3 <u>Exceptional weather conditions algorithm</u>
This algorithm works in parallel with the Production score algorithm when a production ends. The aim of this algorithm is to tell the TPM if a production was made during exceptional weather conditions which affected its score. To do so, the algorithm retrieves the document of that specific production from the data warehouse, in which meteorological information for each day of production is saved. The algorithm takes this information and calculates if adverse conditions occurred, based on parameters that have to be decided in accordance with members of the Telangana government before deploying the application. For instance, it can be assumed that a production took place in adverse conditions if the weather was rainy for more than a third of the total days of the production, or if the temperature was over/under a certain value (e.g. 35 °C/5°C) for more than a third of the total days of the production. If one or more of these conditions happens, it is shown to the TPM that the production was in adverse meteorological conditions, and he/she can see in detail which condition occurred.

2.7.1.4 <u>Suggestion algorithm</u>
A recommender system based on collaborative filtering will take care of providing pertinent suggestions to farmers. Collaborative filtering approaches build a model from a user's past behaviour and also his context (in this case the past farmer's productions, long term forecasts, BPAs) as well as similar decisions made by other farmers. This model is then used to provide practices and suggestions that may boost the farmer's performance. Examples of suggestions can be: specific crops to plant, specific fertilizers to use soil, or even some harvesting methods.

2.7.2 ER diagram
The following entity relationship (ER) diagram is a high-level design of how the database is structured. In the diagram, entities and their relationships with the respective cardinalities are represented. Some clarifications and possible implementation:

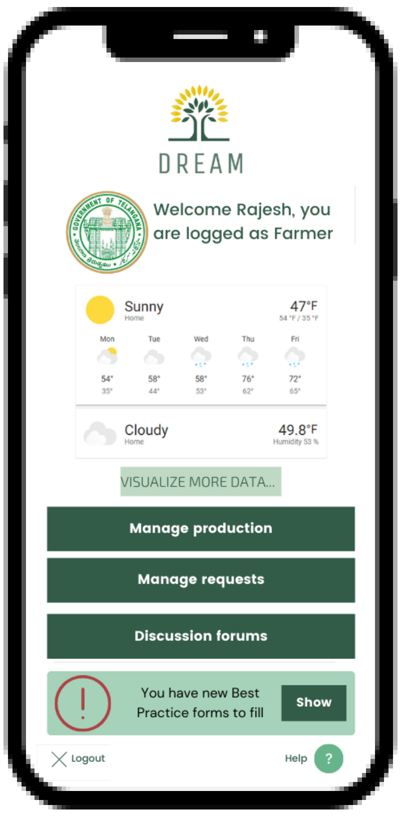
- User is the super type of Farmer, TPM and Agronomist and their Primary Key is ID
- Steering Initiative will be the owner of the relationship with the Agronomist and will hold the FK (Agronomist.ID)
- There will be an additional table that records the many to many relationships between Farmer and Steering Initiative, Farmer and Suggestions, BPA and BP



3 User Interface Design

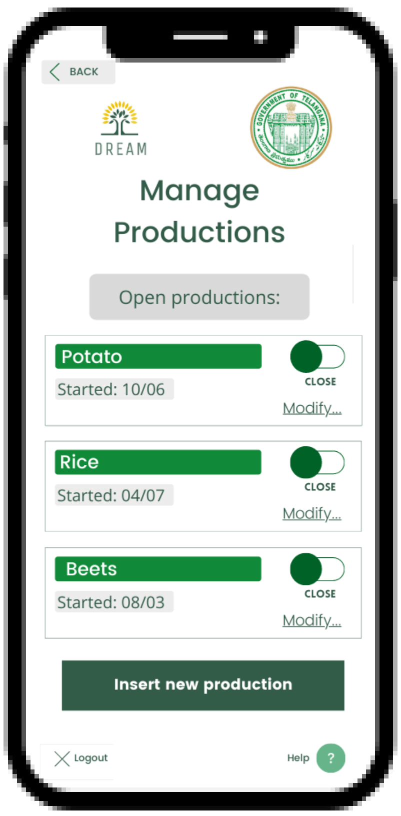
In this section we provide mock-ups of user interfaces divided by the type of user. Login interfaces were already provided in the RASD document, so they're not provided here, while main pages are shown in this document as well for completeness. We chose to show interfaces regarding all the main actions a user can do, without showing all the possible interfaces (such as error interfaces or minor action interfaces) since they would make the document too verbose without adding real interest.

3.1 Farmers user interfaces


Welcome Rajesh, you are logged as Farmer

You have new Best Practice forms to fill

Figure 3.1: Farmer main page


Manage Productions

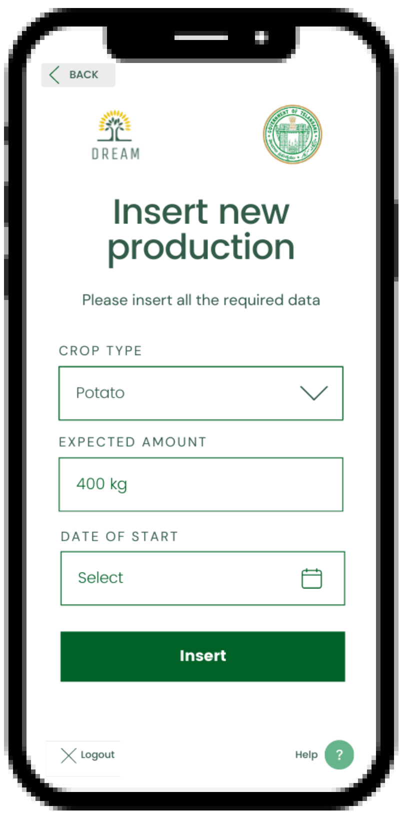
Open productions:

Potato — Started: 10/06

Rice — Started: 04/07

Beets — Started: 08/03

Figure 3.2: Manage production page


Insert new production

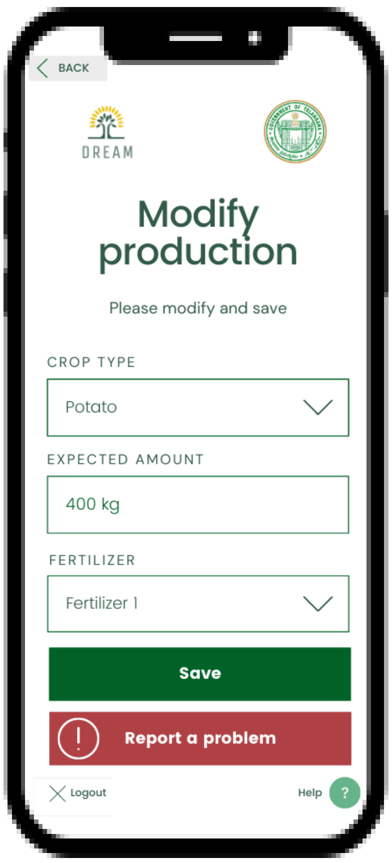
Please insert all the required data

CROP TYPE: Potato

EXPECTED AMOUNT: 400 kg

DATE OF START: Select

Figure 3.3: New production page


BACK

DREAM

Modify production

Please modify and save

CROP TYPE

Potato

EXPECTED AMOUNT

400 kg

FERTILIZER

Fertilizer 1

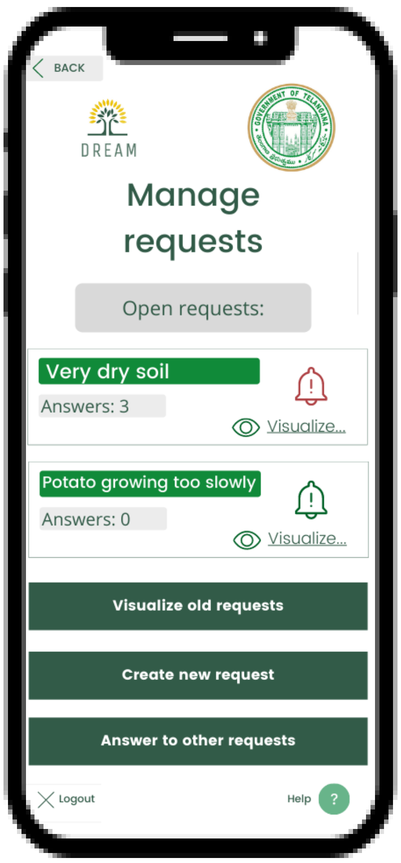
Save

Report a problem

Logout

Help

Figure 3.4: Modify production page


BACK

DREAM

Manage requests

Open requests:

Very dry soil

Answers: 3

Visualize...

Potato growing too slowly

Answers: 0

Visualize...

Visualize old requests

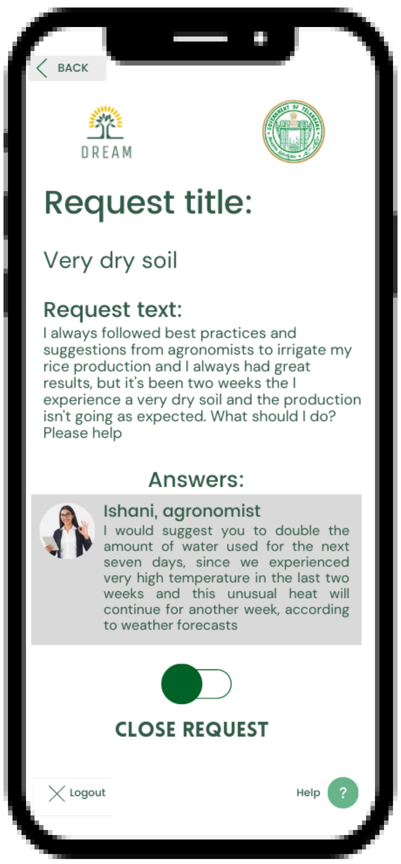
Create new request

Answer to other requests

Logout

Help

Figure 3.5: Manage requests page


BACK

DREAM

Request title:

Very dry soil

Request text:

I always followed best practices and suggestions from agronomists to irrigate my rice production and I always had great results, but it's been two weeks the I experience a very dry soil and the production isn't going as expected. What should I do? Please help

Answers:

Ishani, agronomist

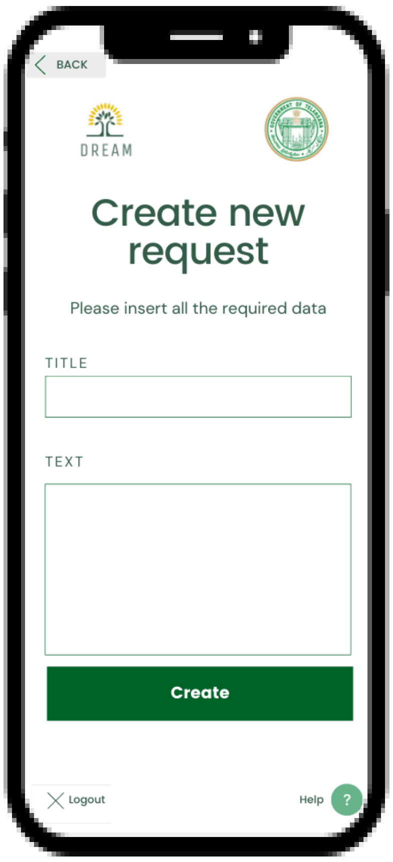
I would suggest you to double the amount of water used for the next seven days, since we experienced very high temperature in the last two weeks and this unusual heat will continue for another week, according to weather forecasts

CLOSE REQUEST

Logout

Help

Figure 3.6: Visualize requests page


BACK

DREAM

Create new request

Please insert all the required data

TITLE

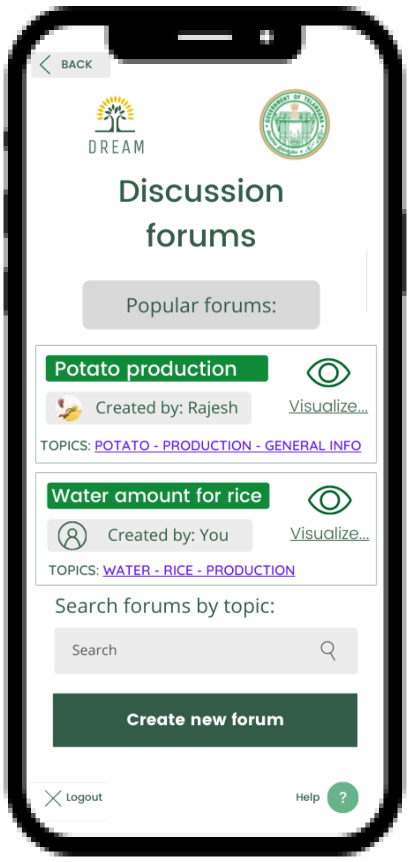
TEXT

Create

Logout

Help

Figure 3.7: New request page


BACK

DREAM

Discussion forums

Popular forums:

Potato production

Created by: Rajesh

Visualize...

TOPICS: POTATO - PRODUCTION - GENERAL INFO

Water amount for rice

Created by: You

Visualize...

TOPICS: WATER - RICE - PRODUCTION

Search forums by topic:

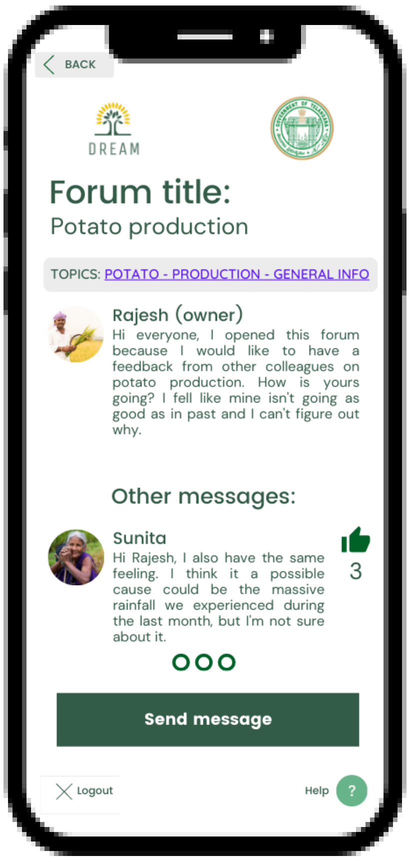
Search

Create new forum

Logout

Help

Figure 3.8: Discussion forums page


BACK

DREAM

Forum title:

Potato production

TOPICS: POTATO - PRODUCTION - GENERAL INFO

Rajesh (owner)

Hi everyone, I opened this forum because I would like to have a feedback from other colleagues on potato production. How is yours going? I fell like mine isn't going as good as in past and I can't figure out why.

Other messages:

Sunita

Hi Rajesh, I also have the same feeling. I think it a possible cause could be the massive rainfall we experienced during the last month, but I'm not sure about it.

3

Send message

Logout

Help

Figure 3.9: Visualize forum page

25

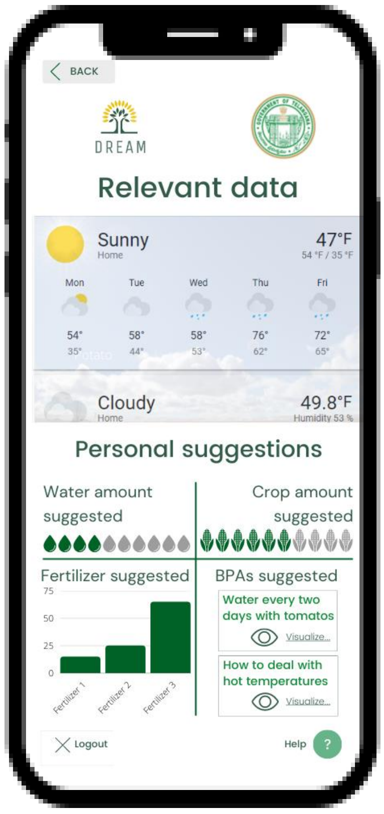


Figure 3.10: Relevant data page

3.2 Agronomists user interfaces

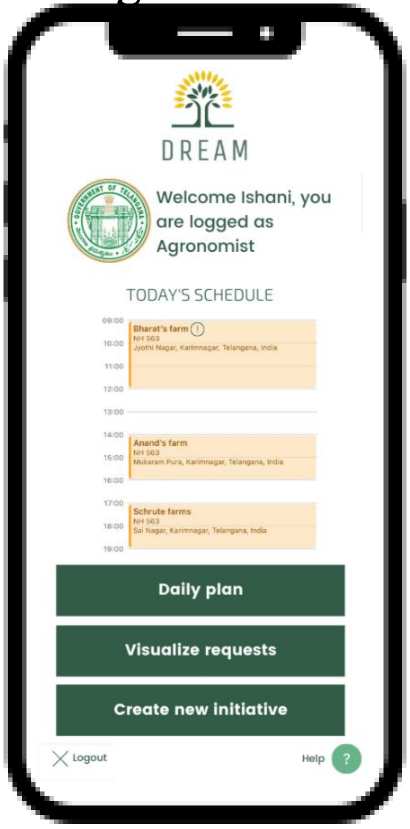


Figure 3.11: Agronomist main page

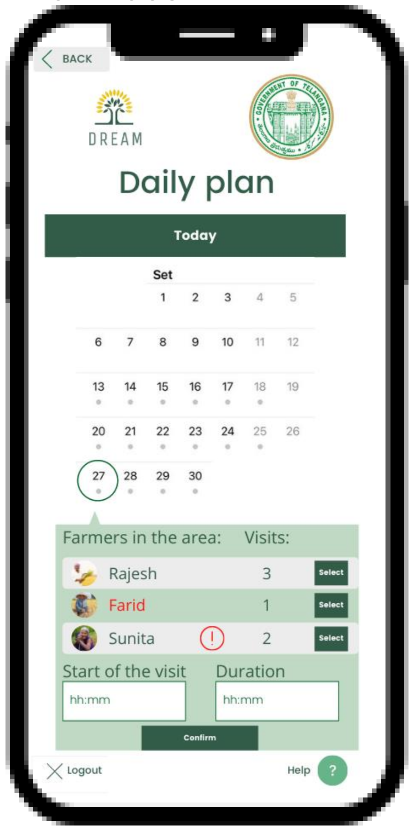


Figure 3.12: Daily plan page

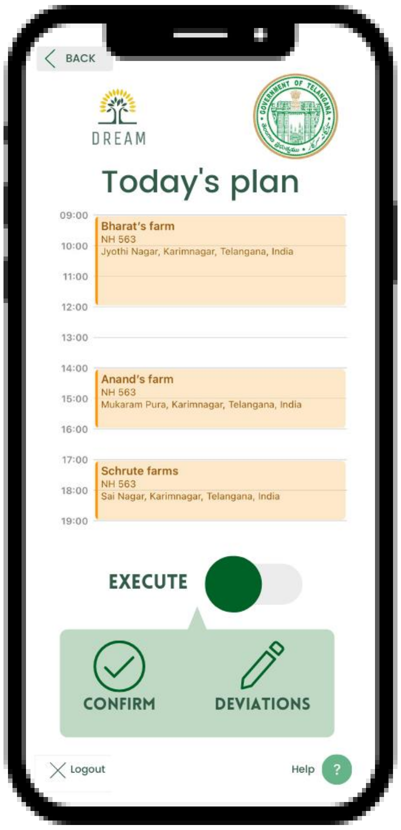

BACK
DREAM
Today's plan
Bharat's farm
NH 563
Jyothi Nagar, Karimnagar, Telangana, India
Anand's farm
NH 563
Mukaram Pura, Karimnagar, Telangana, India
Schrute farms
NH 563
Sai Nagar, Karimnagar, Telangana, India
EXECUTE
CONFIRM
DEVIATIONS
Logout
Help

Figure 3.13: Current day plan

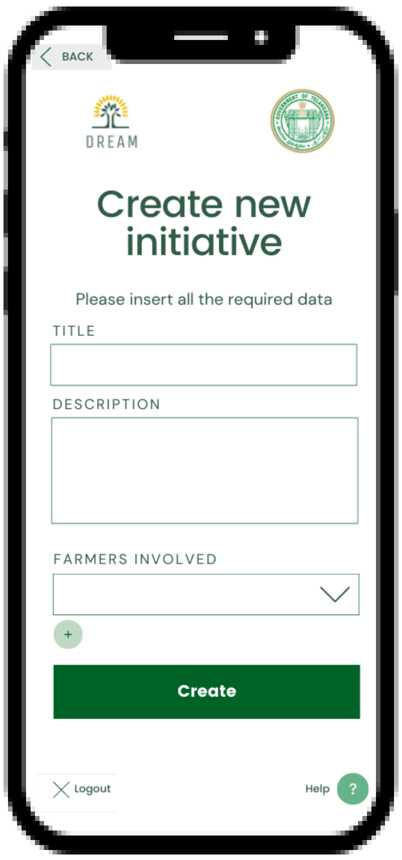


Figure 3.14: Create initiative page

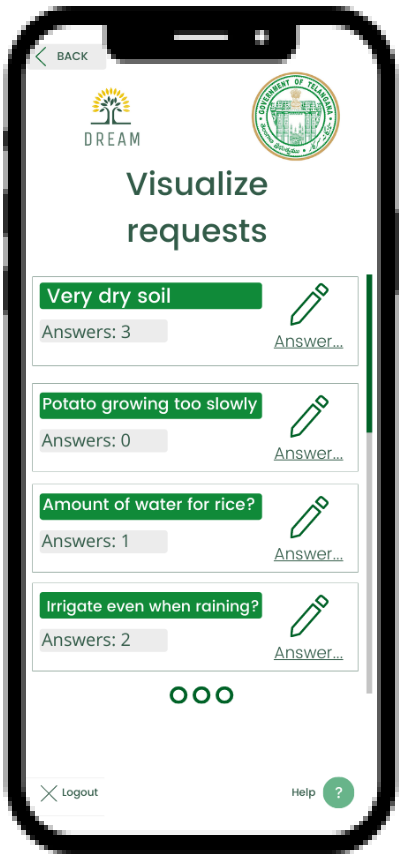


Figure 3.15: Visualize requests page

3.3 TPMs user interfaces

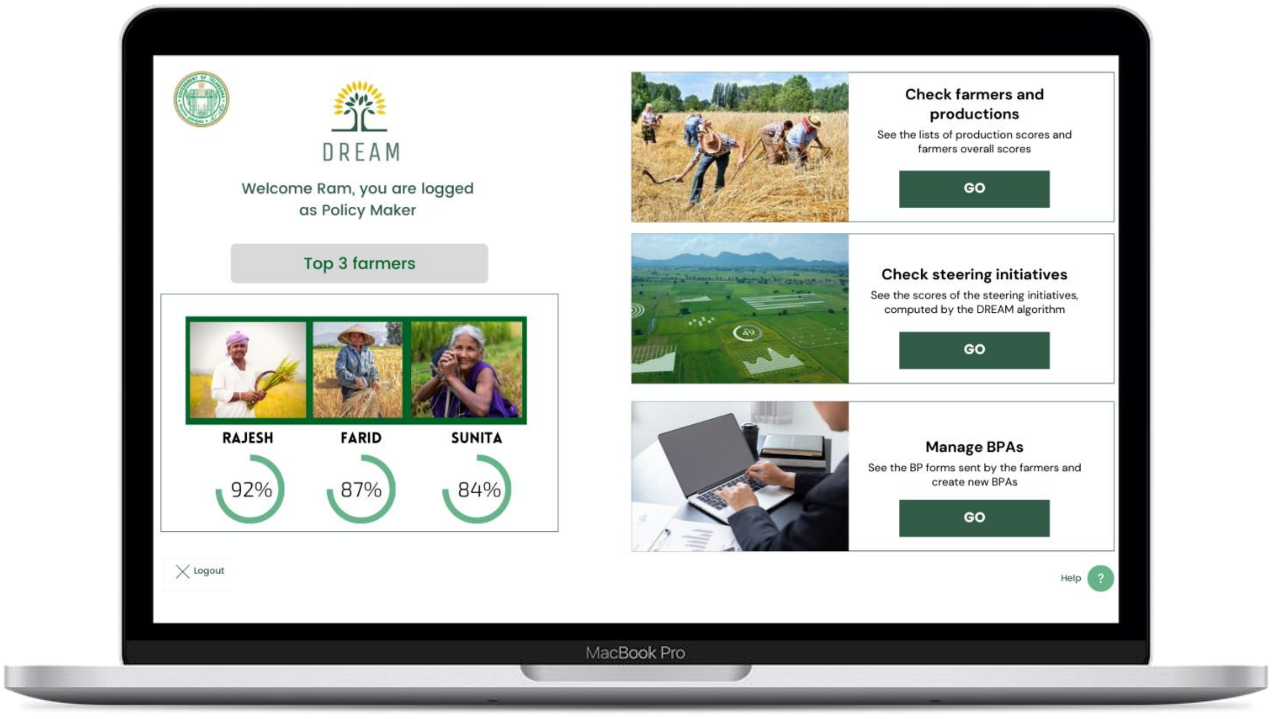

DREAM
Welcome Ram, you are logged as Policy Maker
Top 3 farmers
RAJESH
FARID
SUNITA
92%
87%
84%
Check farmers and productions
See the lists of production scores and farmers overall scores
GO
Check steering initiatives
See the scores of the steering initiatives, computed by the DREAM algorithm
GO
Manage BPAs
See the BP forms sent by the farmers and create new BPAs
GO
Logout
Help
MacBook Pro

Figure 3.16: TPM main page

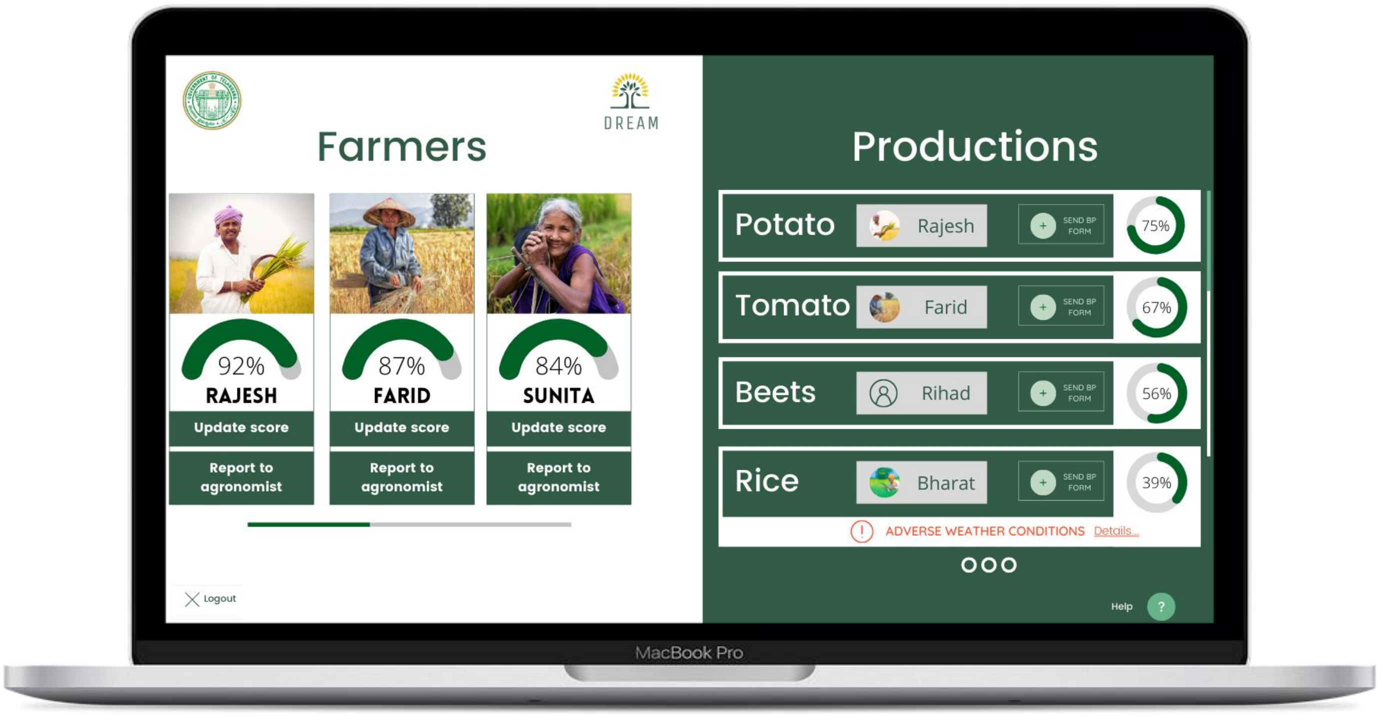


Figure 3.17: Farmers and productions page

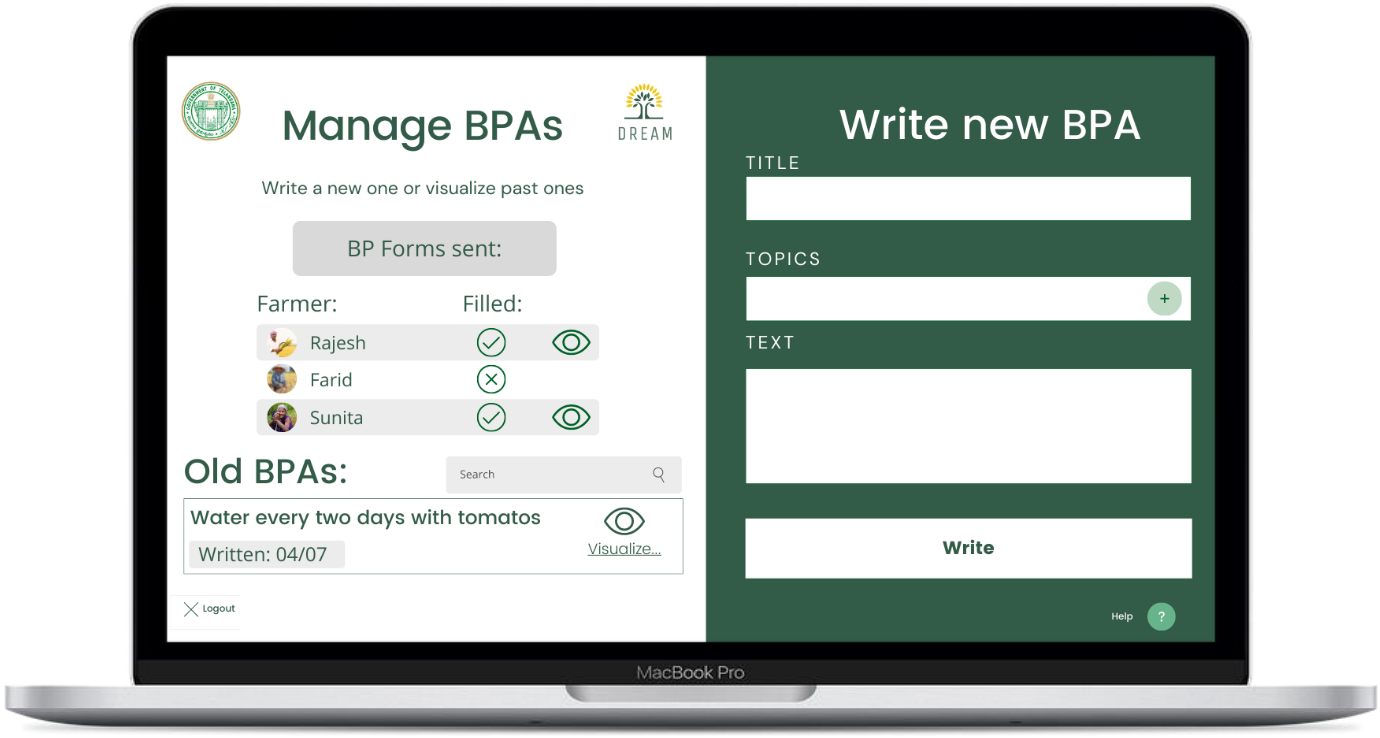


Figure 3.18: Manage BPAs page

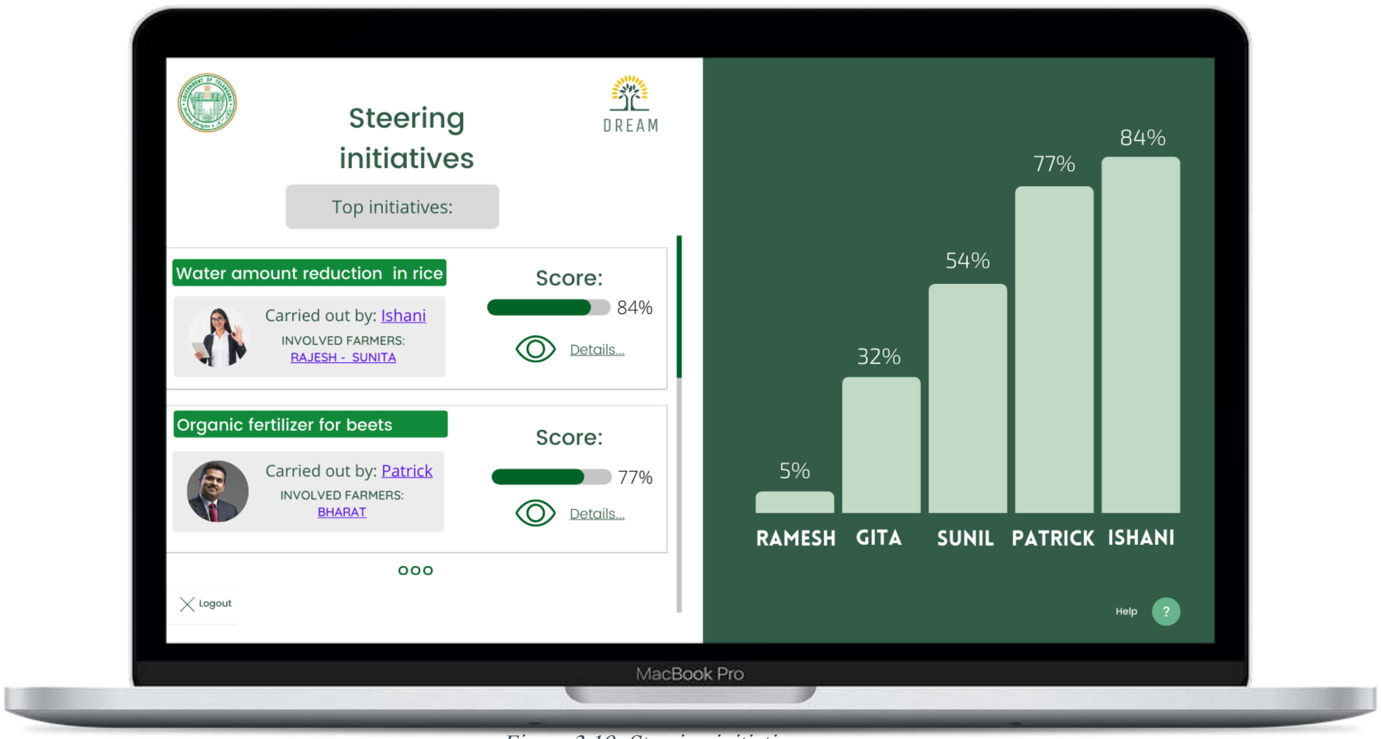


Figure 3.19: Steering initiatives page

4 Requirements traceability

Below is shown the component mapping for every requirement mentioned in the RASD document. Components Router and Web Server were not included in this mapping since they only work as means to forward messages to other components.

R1: The system shall allow an app customer to login	*AccessManager*
R2: The system shall allow an app customer to logout	*AccessManager*
Telangana Policy Maker	
R3: The system shall allow a TPM to visualize an ordered list with the farmers and their performances	*PerformanceScoreManager RelevantDataManager TPM WebApp*
R4: The system shall allow a TPM to update the overall score of a farmer	*PerformanceScoreManager TPM WebApp*
R5: The system shall allow a TPM to visualize a list with the production performances	*PerformanceScoreManager RelevantDataManager TPM WebApp*
R6: The system shall allow a TPM to send a form to the top performing farmers in order to collect their best practices	*BPManager TPM WebApp*
R7: The system shall allow a TPM to notify an agronomist with the worst performing farmers who need to be visited as soon as possible	*RelevantDataManager TPM WebApp*
R8: The system shall allow a TPM to visualize an ordered list with steering initiatives carried out by agronomists associated with their performances	*PerformanceScoreManager SteeringInitiativesManager TPM WebApp*
R9: When a farmer inserts the last update and closes a production, his/her production performance is computed by the algorithm	*PerformanceScoreManager AlgorithmsManager EnvironmentConditionManager*

R10: When a production performance is computed, the ordered list of the performances is updated	*PerformanceScoreManager* *RelevantDataManager*
R11: The system shall allow a TPM to receive the best practices	*BPManager* *TPM WebApp*
R12: The system shall allow a TPM to create BPAs	*BPManager* *TPM WebApp*
R13: The system shall send out the BPAs to all farmers	*BPManager* *TPM WebApp*
Farmer	
R14: The system shall allow farmers to insert their location	*AccessManager* *Farmer MobileApp*
R15: The system shall allow farmers to fill in the form sent by TPM with their best practices	*BPManager* *Farmer MobileApp*
R16: The system shall allow farmers to visualize weather forecasts	*RelevantDataManager* *EnvironmentConditionManager* *Farmer MobileApp*
R17: The system shall compute personalized suggestions based on the designed algorithm	*RelevantDataManager* *EnvironmentConditionManager*
R18: The system shall allow farmers to visualize personalized suggestions	*RelevantDataManager* *Farmer MobileApp*
R19: The system shall allow farmers to insert data about a new production	*ProductionManager* *Farmer MobileApp*
R20: The system shall allow farmers to insert information on problems they face during the production	*ProductionManager* *Farmer MobileApp*
R21: The system shall allow farmers to close a production when they finish harvesting	*ProductionManager* *Farmer MobileApp*
R22: The system shall allow farmers to create help requests	*HelpRequestManager* *Farmer MobileApp*
R23: The system shall allow farmers to create discussion forums	*ForumManager* *Farmer MobileApp*
R24: The system shall allow farmers to send messages in existing discussion forums	*ForumManager* *Farmer MobileApp*
Agronomist	
R25: The system shall allow an agronomist to insert the area he is responsible of	*AccessManager* *Agronomist MobileApp*
R26: The system shall allow an agronomist to receive information about requests for help	*HelpRequestManager* *Agronomist MobileApp*
R27: The system shall allow an agronomist to answer to requests for help	*HelpRequestManager* *Agronomist MobileApp*
R28: The system shall allow an agronomist to visualize data concerning weather forecasts in the area	*RelevantDataManager* *EnvironmentConditionManager* *Agronomist MobileApp*
R29: The system shall allow an agronomist to visualize the best performing farmers in the area	*RelevantDataManager* *Agronomist MobileApp*
R30: The system shall allow an agronomist to create a daily plan to visit farms in the area	*DailyPlanManager* *Agronomist MobileApp*
R31: The system shall allow an agronomist to update the daily plan	*DailyPlanManager* *Agronomist MobileApp*
R32: The system shall allow an agronomist to visualize highlighted in red the farms that have not been visited at least twice during the year	*DailyPlanManager* *Agronomist MobileApp*
R33: The system shall allow an agronomist to visualize with a “!” the most underperforming farms	*DailyPlanManager* *Agronomist MobileApp*
R34: The system shall allow an agronomist to confirm the execution of the daily plan	*DailyPlanManager* *Agronomist MobileApp*

R35: The system shall allow an agronomist to specify deviations from a daily plan previously created	*DailyPlanManager Agronomist MobileApp*
R36: The system shall allow an agronomist to insert a place, a period and a description for a new steering initiative	*SteeringInitiativesManager Agronomist MobileApp*
R37: The system shall allow unregistered users to sign up using their ID	*AccessManager*
R38: The system shall allow to set a user profile type	*AccessManager*

Requirements – Components Matrix

AccessManager	A
BPManager	B
PerformanceScoreManager	C
RelevantDataManager	D
SteeringInitiativesManager	E
AlgorithmsManager	F
ProductionManager	G
HelpRequestManager	H
ForumManager	I
DailyPlanManager	L
EnvironmentConditionManager	M
TPM WebApp	N
Farmer MobileApp	O
Agronomist MobileApp	P

	A	B	C	D	E	F	G	H	I	L	M	N	O	P
R1	X													
R2	X													
R3			X	X								X		
R4			X									X		
R5			X	X								X		
R6		X										X		
R7				X								X		
R8			X		X							X		
R9			X			X					X			
R10			X	X										
R11		X										X		
R12		X										X		

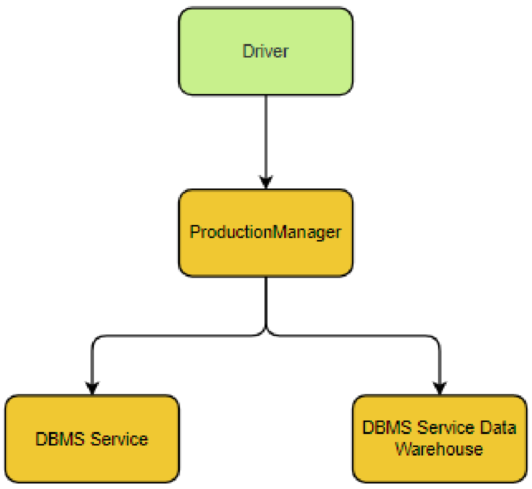
R13		X										X		
R14	X												X	
R15		X											X	
R16				X							X		X	
R17				X							X			
R18				X									X	
R19							X						X	
R20							X						X	
R21							X						X	
R22								X					X	
R23									X				X	
R24									X				X	
R25	X													X
R26								X						X
R27								X						X
R28				X							X			X
R29				X										X
R30										X				X
R31										X				X
R32										X				X
R33										X				X
R34										X				X
R35										X				X
R36					X									X
R37	X													
R38	X													

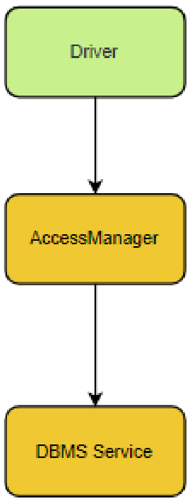
5 Implementation, Integration and Test Plan

This section defines the implementation order for the different components of the DREAM application that should be followed by the development team.

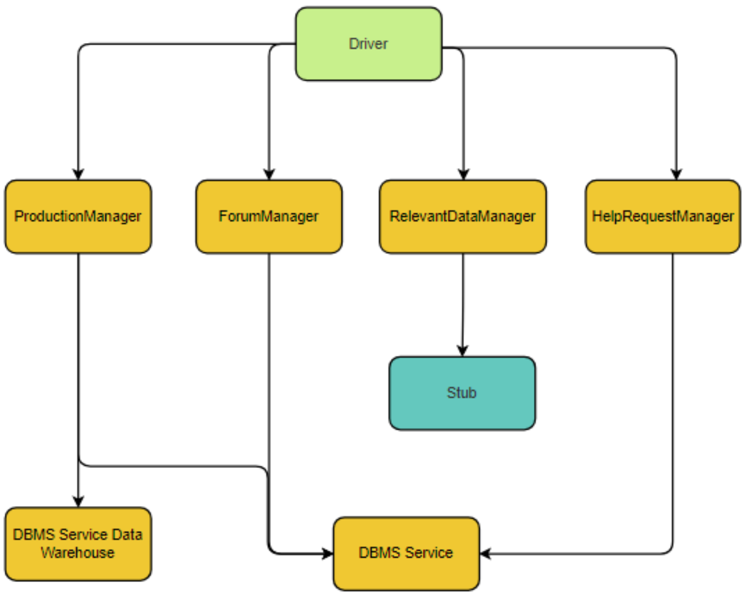
The implementation, integration and testing of the system will mainly follow a bottom-up approach, with just a few exceptions. This approach is chosen for the server side while the client side will be first tested with the help of a stub. Client and server will be implemented and tested in parallel, and integrated at the end of the process.
An incremental integration facilitates bug tracking and generates working software quickly during the software life cycle. External services do not need to be unit tested since it is assumed that they are reliable.

5.1 The first step in the implementation and integration process is to implement and unit test the ProductionManager component, which interacts with both the Database and the Data Warehouse. A driver will be used to simulate the module calls made by the Farmer.
In parallel, also the AccessManager module can be implemented and unit tested.

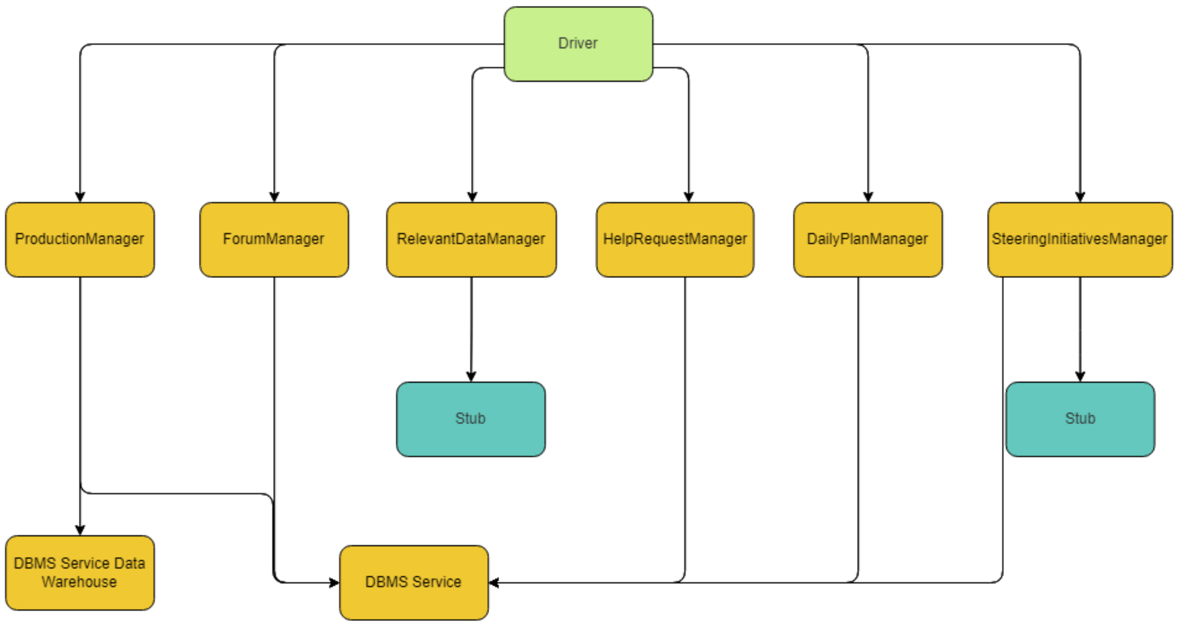




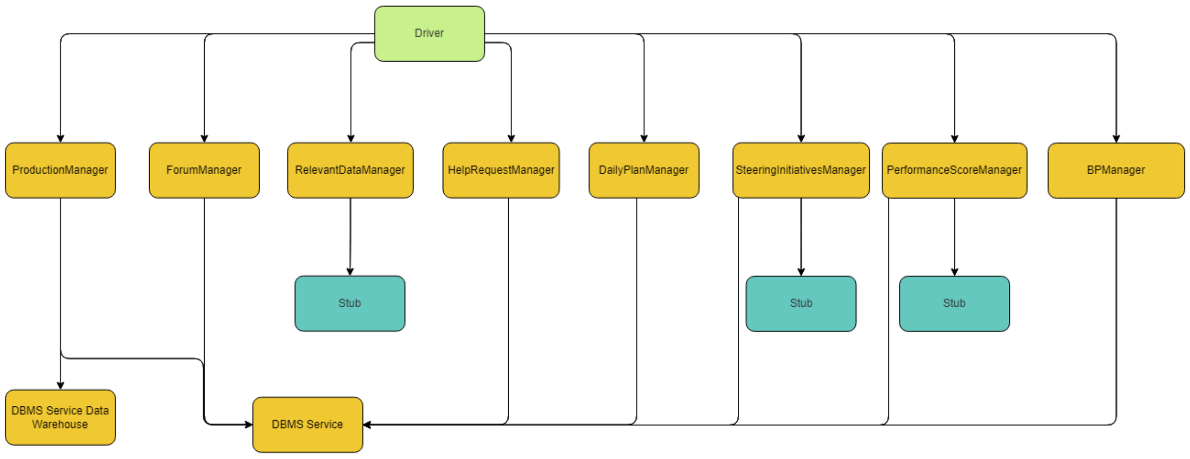
5.2 Then all the other modules to which farmers relate to can be implemented. Here a driver has to be used to simulate the modules calls done by both Farmers and Agronomists (some components have functions called by more actors) through the Router. A stub is used to simulate the behavior of the EnvironmentConditionManager and the AlgorithmsManager, still to be integrated at this point.



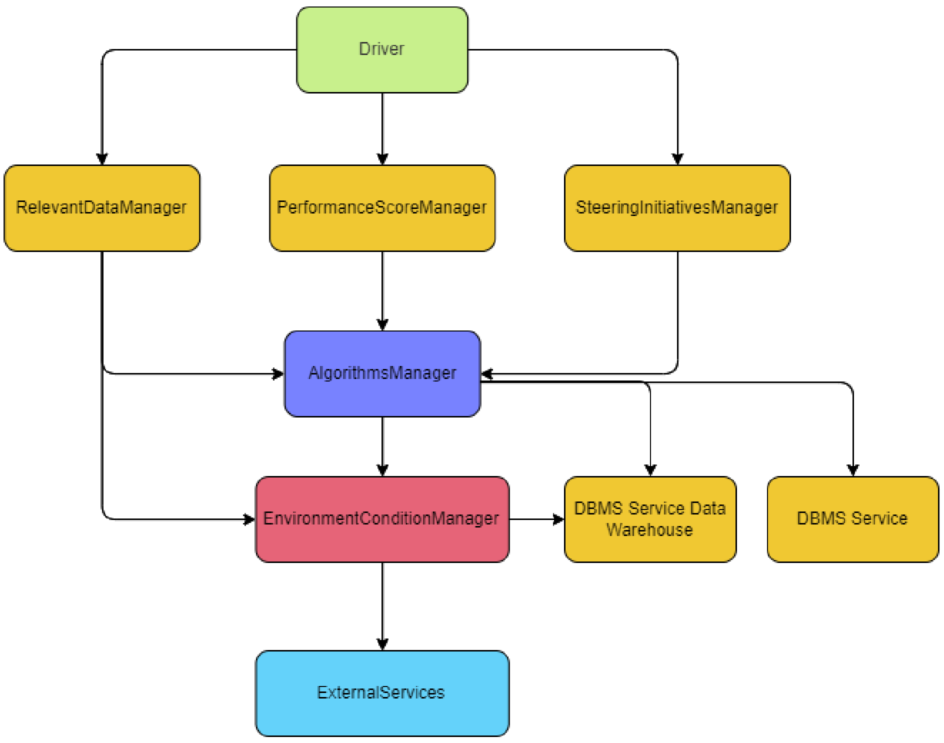
5.3 After this step modules developed especially for agronomists can be implemented and integrated with the already built system. A driver like in the steps before will simulate the calls done by the Router that come from the three different Users of Dream (as in step 2, here TPMs share some components with the two actors). EnvironmentConditionManager and AlgorithmsManager will be integrated in a later stage, so stubs will simulate their behavior.



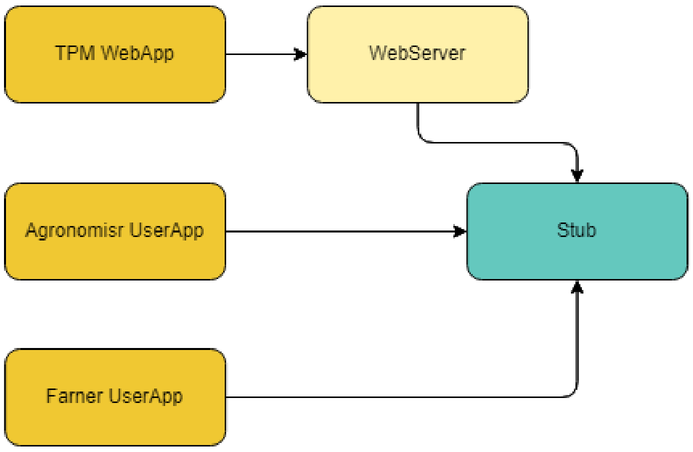
5.4 Then modules for TPMs are to be implemented and integrated. Here the same strategy of the previous steps for drivers and stubs is applied.



5.5 In the end the components AlgorithmsManager and EnvironmentConditionManager can be integrated and the whole Model can be tested.



5.6 In parallel the user interfaces and the WebServer have been implemented. A stub here simulates the Router behavior.



5.7 Finally, the component Router is implemented and the Client side and the Server side are integrated. This happens only when all the components of both sides have been implemented and tested (indeed, we cannot perform this integration before without the Router that is the key-component for the communication between server and client).

Once all the components have been integrated, we proceed with the system testing in order to verify if the complete application works properly. Our purpose is to check if all the requested functionalities are effectively offered by our system (functional testing) and eventually to identify bottlenecks that affect the performance of our system in terms of throughput and the response time. Finally, it's important to perform a Load Testing and a Stress Testing to check how our system behaves in extreme situations.
Before making the system generally available (GA), in order to catch any software bugs and errors that have snuck through the testing process, to understand and to improve the product's full end user experience, beta testing is strongly suggested.

6 Effort spent

Team

Topic	Hours
Initial briefing	4
General reasoning	4
Document revision	6
Total	**14**

Alessandro Maranelli

Topic	Hours
Deployment View	2
Runtime View	7
Architectural styles and patterns	1
Algorithms	1
Requirements traceability	1
Implementation and integration plan	6
Additional revision	2
Total	**20**

Amir Bachir Kaddis Beshay

Topic	Hours
Architectural Design	4
Component Diagrams	3
ER diagram	4
Algorithms	1
Component Interfaces	3
Additional revision	3
Total	**18**

Salvatore Marragony

Topic	Hours
Introduction	1
Component diagrams	5
Algorithms	2
User interfaces	10
Additional revision	3
Total	**21**

7 References

Websites and tools used while writing the Design document:

- Github with information about the real UN project: https://github.com/UNDP-India/Data4Policy

- Telangana government website for weather forecasting: https://www.tsdps.telangana.gov.in/aws.jsp

- Tools for diagrams:
    - Visual Paradigm Online - Suite of Powerful Tools (visual-paradigm.com)
    - draw.io