



POLITECNICO
MILANO 1863

DRAM

DD

Requirement Analysis and Specification
Document
AA 2021-2022

Version: 1.0
Date: 6 January 2022

1 Introduction.....	3
1.1 Purpose.....	3
1.2 Scope.....	3
1.3 Definition and abbreviation.....	4
1.3.1 Definition.....	4
1.3.2 Abbreviations.....	4
1.4 revision history.....	4
1.5 Reference document.....	5
1.6 Document structure.....	5
2 Architecture design.....	6
2.1 Overview.....	6
2.2 Component view.....	7
2.2.1 Class diagram.....	7
2.2.2 Component diagram.....	7
2.3 Deployment view.....	11
2.4 Runtime view.....	12
2.4.1 User login.....	12
2.4.2 Add production.....	13
2.4.3 Add production type.....	14
2.4.4 Add soil humidity data.....	15
2.4.5 Search soil humidity.....	16
2.4.6 Search water irrigation data.....	17
2.4.7 Search weather forecast.....	18
2.5 Component interfaces.....	19
2.6 Selected architectural styles and patterns.....	20
2.7 Other design decisions.....	20
3 User interface design.....	21
3.1 Registration page.....	21
3.2 Login page.....	21
3.3 Main page.....	22
3.4 Notification.....	22
3.5 Production data management.....	23
3.6 Weather forecast.....	23
3.7 Water irrigation data.....	24
3.8 Soil humidity data.....	24
3.9 Forum.....	25
3.10 Topic.....	25
4 Requirements traceability.....	26
5 Implementation, integration, and test plan.....	27
5.1 Implementation overview.....	27
5.2 Integration plan.....	28
6 Effort spent.....	29
7 References.....	30

1 Introduction

1.1 Purpose

The pandemic of the COVID-19 has had a huge impact on the world. Because the COVID-19 is spread in population movements, governments in various countries have generally adopted restrictions on travel, restrictions on population movements, and cargo transportation to minimize the spread of the virus. However, using these methods will cause food supply problems in some marginalized communities.

Establish a digital system to collect various data related to agricultural products, such as types of agricultural products, water volume, soil moisture, and other data. Through analysis, and the results of the analysis are sent to farmers, agronomists. Cooperate with various departments to deal with food supply shocks and challenges.

1.2 Scope

To enable the system to collect, analyze data, feedback data, and implement data-driven policy formulation methods. Need to involve multiple stakeholders, the policy

Manufacturers, farmers, and agronomists.

For policy manufacturers, they need to use the data feedback to them by the system to formulate corresponding policies and modify existing policies to adapt to the current food supply situation.

For farmers, they need to upload the type and quantity of their products to the system. The system will give them information about weather forecasts, fertilizer types, and other personalized suggestions to help farmers increase yields.

For agronomists, mainly help/guide farmers. The agronomist will work out a corresponding guidance plan based on the data. At the same time, they will interact with farmers in the discussion forums.

1.3 Definition and abbreviation

1.3.1 Definition

Definition	Description
Interval time-about sensor	Set how often the sensor uploads data
Weather forest	Get information through IT provider and get weather forecast
Personalized suggestions,	Personalized suggestions, the agronomist draws corresponding suggestions based on the analysis of statistical data and sends them to the farmers
Interval time -about the weather forecast	Obtain the latest weather forecast data from IT suppliers regularly
Visualization	Convert data into tables or pictures to view
Release time	The time the farmer/agronomist posted the message on the forum
discussion forums	Provide a platform for interactive communication between agronomists and farmers

1.3.2 Abbreviations

Abbreviation	Description
WP	World phenomena
SP	Shared phenomena
G	Goal
D	Domain assumption
R	Requirement
IT	Information Technology
RASD	Requirement analysis and specification document

1.4 revision history

Version	Date	Description
1.0	05/01/2022	First version

1.5 Reference document

- Specification document: Assignment RDD AY 2021-2022
- IEEE/ISO/IEC 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Requirements engineering
- Course slides.

1.6 Document structure

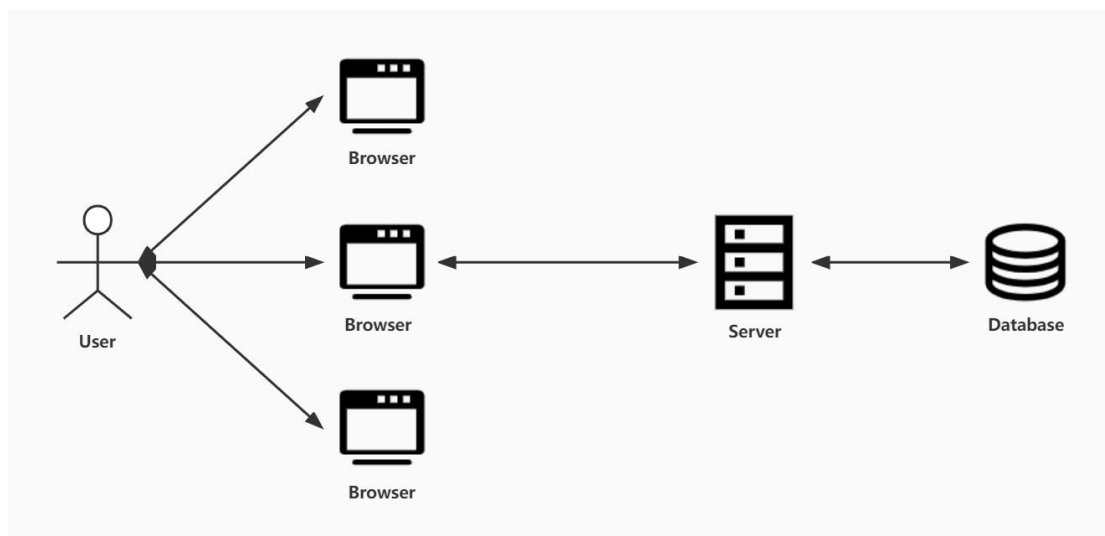
- Section 1
Recapitulation of the scope of the system and goals. It explains the definitions and abbreviations that are used in this document.
- Section 2
The core of the document. It contains an in-depth analysis of the system architecture design, starting from a general overview and then going more in detail with the analysis of the different components, their physical deployment, their interactions, and their interfaces, with an insight into the architectural style and pattern that has been used.
- Section 3
Graphical user interface mock-up for the different views of the system.
- Section 4
Contains a traceability matrix that shows which components contribute to the realization of a certain requirement.
- Section 5
Discussion on the implementation and testing plan, given as a guideline for developers and testers.
- Section 6
Contains the indication of the effort spent by each group member.
- Section 7
Contains the references.

2 Architecture design

2.1 Overview

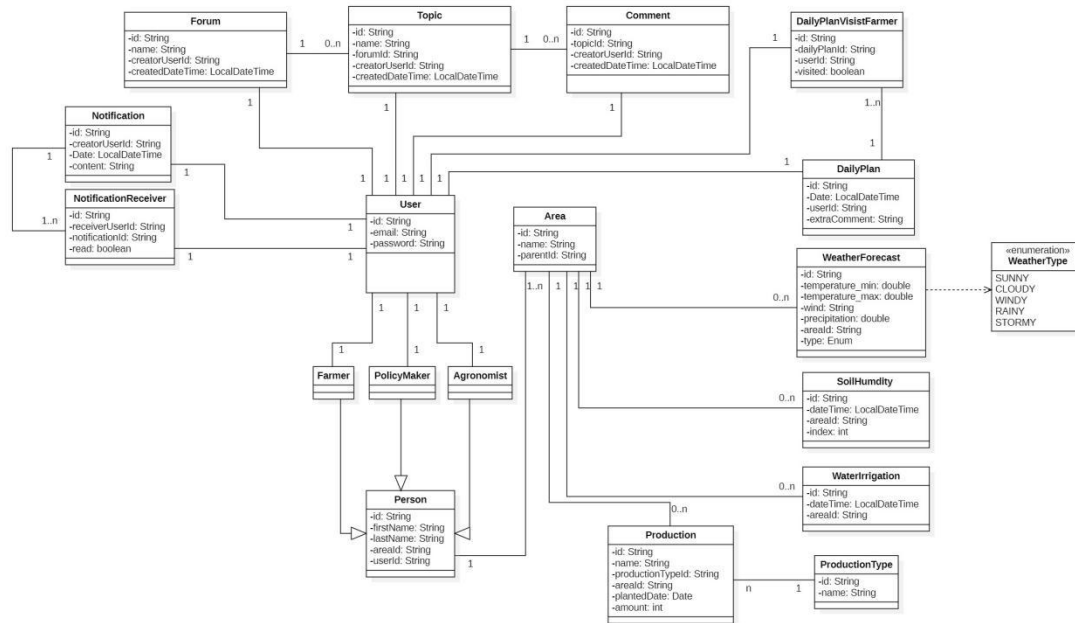
Considering the number of system users, the connection between soil humidity sensors and system, and the development difficulty, B/S (Browser/Server) is a good choice for the structure of DREAM system. B/S is a network structure after the rise of the WEB, this structure divides the system into two main parts, the browser, and the server. It centralizes the core part of the system function realization to the server and simplifies the development, maintenance, and use of the system.

Furthermore, the MVC (Model-View-Controller) also will be used in the system development. It separates the core system into three parts. The basic logic of MVC is that the user uses the controller to manipulate the model, and the model updates the view that the user sees. The Model is responsible for data management(search, updates, delete, etc.). The View is responsible for data and page visualization. The Controller is responsible for the redirection of the system.

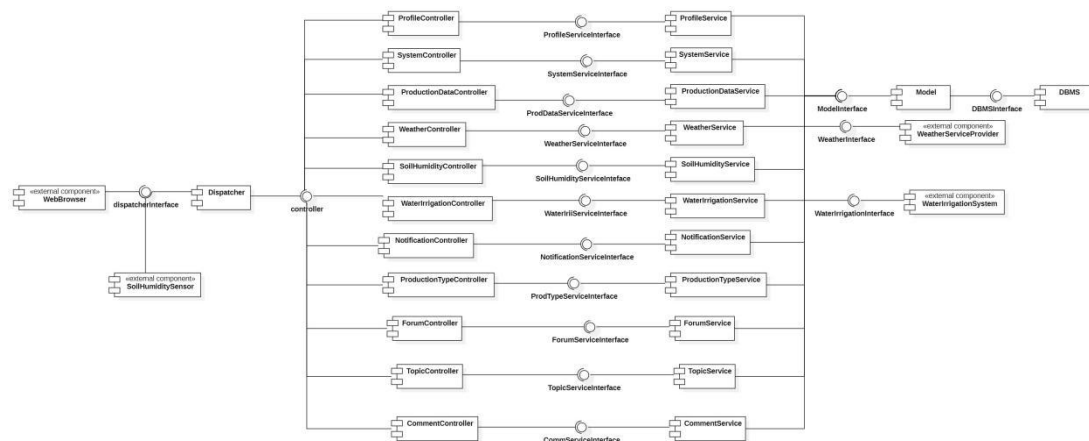


2.2 Component view

2.2.1 Class diagram



2.2.2 Component diagram



This component diagram describes the overall composition of the DREAM system. These components support the most functions of the system. The components are designed and divided into three different hierarchies according to MVC pattern, The first layer is 'Controller', which is responsible for the redirection of the workflow of the system, the second layer is the 'Service', which is responsible for business process or algorithm, finally, the last layer is 'Database manage system' that is in charge for the data management.

The 'Controller' and the 'Service' may be merged into a layer, but two-layer can set a clear

boundary between the operation of system operations(e.g. redirection) and business data operations (e.g. data management), which is good for the development according to Single responsibility principle.

All components are analyzed in detail:

- **SoilHumiditySensor**
This component is used in the soil to detect the humidity of the soil and sends the data to the DREAM system.
- **ProfileController**
This component is used to handle the redirection of the request related to the profile, after it gets the result, sends the result back to the browser.
- **ProfileService**
This component processes the request which is sent by ProfileController, such as modifying the profile, sending the result of the process to ProfileController after finishing all the operations.
- **SystemController**
This component is used to handle the redirection of the request related to login and registration, after it gets the result, sends the result back to the browser.
- **SystemService**
This component processes the request which is sent by SystemController and sends the result back to SystemController.
- **ProductionDataController**
This component is used to handle the redirection of the request related to production management that includes adding, modifying, search and deleting. When it receives the result that is sent by ProductionDataService, it sends it to the browser.
- **ProductionDataService**
This component processes the request which is sent by ProductionDataController. After the process, send the result back to ProductionDataController.
- **WeatherController**
This component is used to handle the redirection of the request related to the weather forecast, after it gets the result, sends the result back to the browser.
- **WeatherService**
This component process the request is sent by WeatherController. For search weather forecast operation, it will request the external component WeatherServiceProvider to obtain weather forecast.

- **WeatherServiceProvider**
This external component provides a weather forecast.
- **SoilHumidityController**
This component is used to handle the redirection of the request related to soil humidity, after it gets the result, sends the result back to the browser.
- **SoilHumidityService**
This component handles the request is sent by SoilHumidityController, which includes saving the data of soil humidity and query.
- **WaterIrrigationController**
This component is used to handle the redirection of the request related to water irrigation, after it gets the result, sends the result back to the browser.
- **WaterIrrigationService**
This component handles the request related to water irrigation. For query, it will request the external component WaterIrrigationSystem to acquire the water irrigation data and send the data back to WaterIrrigationController.
- **WaterIrrigationSystem**
This external component provides water irrigation data.
- **NotificationController**
This component is used to handle the redirection of the request related to notification, after it gets the result, sends the result back to the browser.
- **NotificationService**
This component handles the request for a notification that includes sending, reading operations.
- **ProductionTypeController**
This component is used to handle the redirection of the request related to production type management, after getting the result, sends the result back to the browser.
- **ProductionTypeService**
This component handles the production type management operation that includes adding, modifying, querying, delete operations. After the process is done, it will send to result back to ProductionTypeController.
- **ForumController**
This component is used to handle the redirection of the request related to the forum, after it gets the result, sends the result back to the browser.

- **ForumService**

This component handles the request about a forum that includes adding, modifying, querying, and deleting. After the process is done, it will send to result back to ForumController.

- **TopicController**

This component is used to handle the redirection of the request related to a topic, after it gets the result, sends the result back to the browser.

- **TopicService**

This component handles the request about a forum that includes adding, modifying, querying, and deleting. After the process is done, it will send to result back to TopicController.

- **CommentController**

This component is used to handle the redirection of the request related to comment, after it gets the result, sends the result back to the browser.

- **CommentService**

This component handles the request about a forum that includes adding, modifying, querying, and deleting. After the process is done, it will send to result back to CommentService.

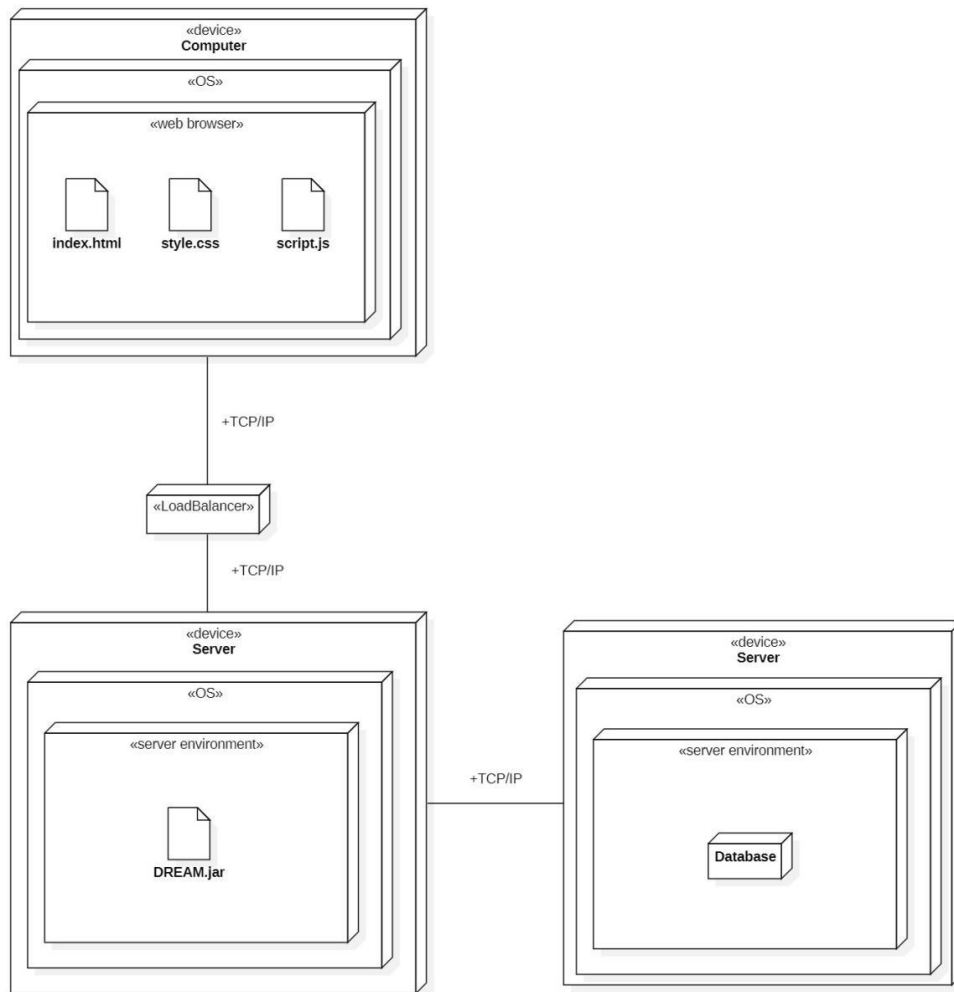
- **Model**

This component is responsible for the single point of access to data, which decouples the data tier to the upper tier appropriately and stably. The interaction between system and database is processed by the Model component.

- **DBMS**

This component is responsible for data management.

2.3 Deployment view



This section describes the deployment of the DREAM system. Due to the B/S (Browser/Server) being the main structure of the DREAM system, so the diagram can be divided into two parts. One is the browser, and another one is the server. Each artifact is described as follow:

- **Computer**

This is an ordinary computer device with a GUI operation system, which is used by a user to connect to the internet and run the browser. When users open the DREAM system website in the browser, the server will send the resource (e.g. HTML file, CSS file, JavaScript files, etc.) that is needed to be loaded on the computer.

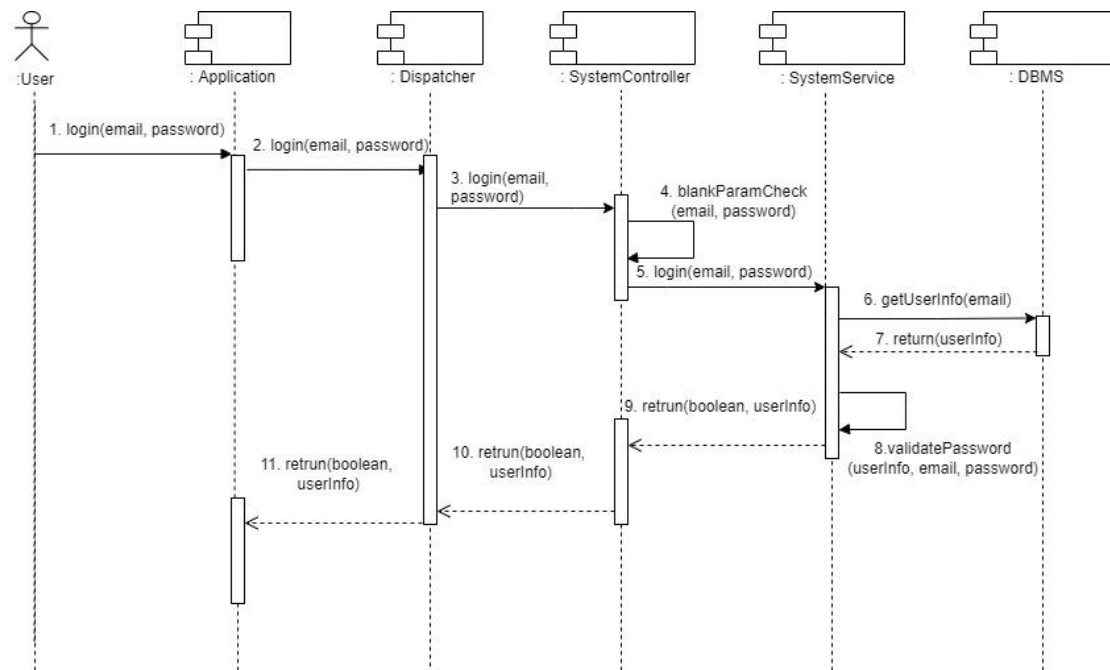
- **Server**

This a device with capable performance to deal with the internet connection and information computing. The most server would be installed Linux operation system because of the stability. Also, the server environment needs to be configured for Java and MySQL databases.

2.4 Runtime view

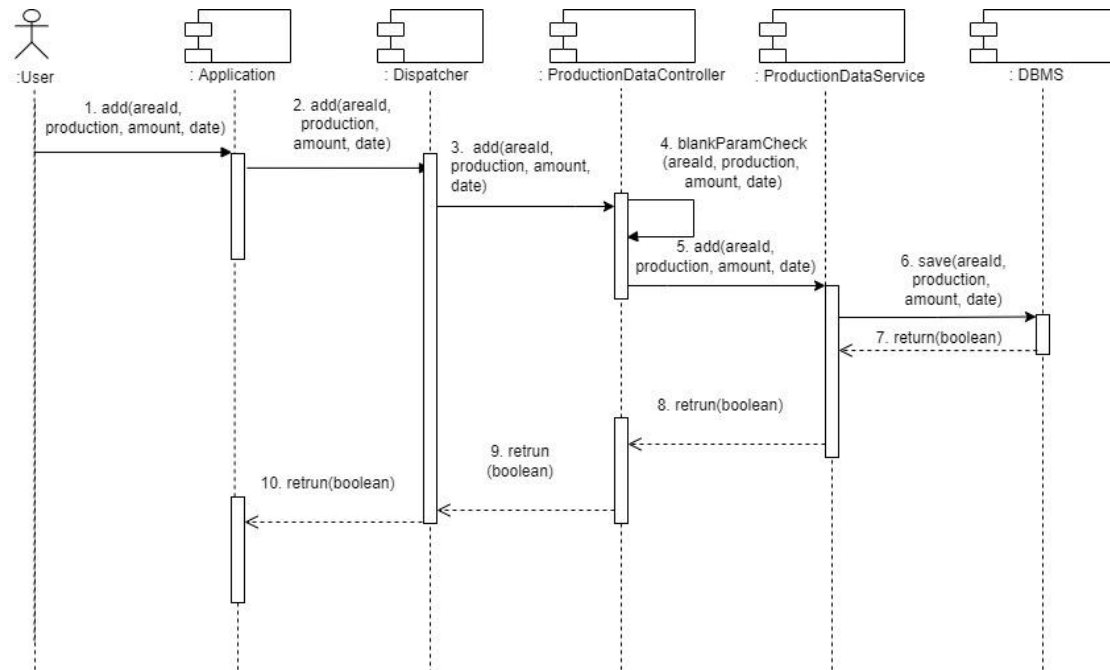
For the following sequence diagrams, we will present the main flow of the events, do not analyze all the possibilities. For each diagram, there is an explanation under the diagram to illustrate the workflow.

2.4.1 User login



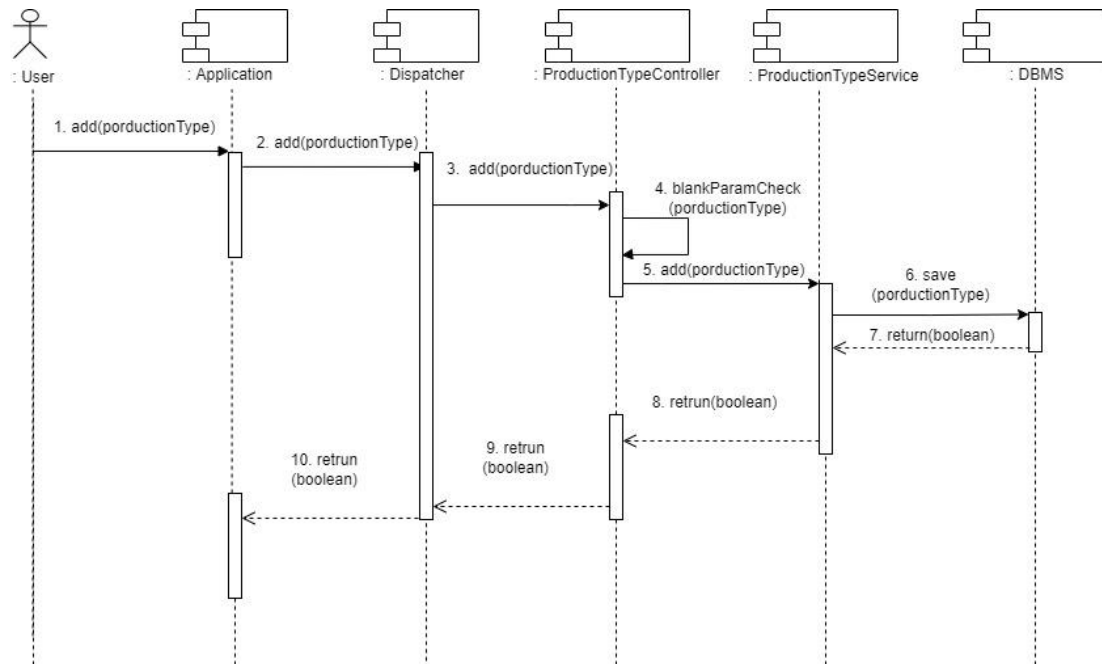
In the login sequence diagram, the user enters the email and password, and clicks the login button, the application will send the request to the server. When the request reaches the server, the request is distributed to the SystemController, the controller redirects to the corresponding SystemService after checking the parameters are blank or not. The SystemService queries the user information by email through the DBMS component and compares parameter and the data from a database is matched or not, then send to result back to SystemController. If the password and email are matched, the controller returns the result and user information, otherwise, only returns the result.

2.4.2 Add production



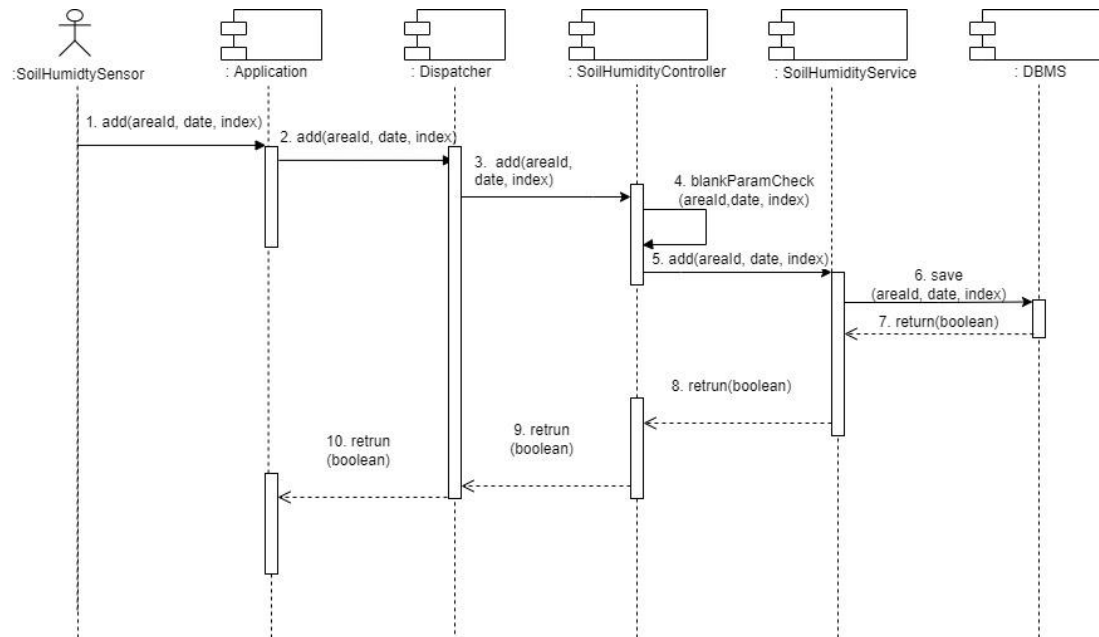
This diagram shows the sequence of adding production data. When a user enters the information, the application sends a request to the server, the request is distributed to ProductionDataController. The ProductionDataController check the parameters are blank or not, then sends the request to ProductonDataService. The ProductionDataService requests the DBMS method to save the production data and obtain the result. Finally, the result will be delivered to the Application.

2.4.3 Add production type



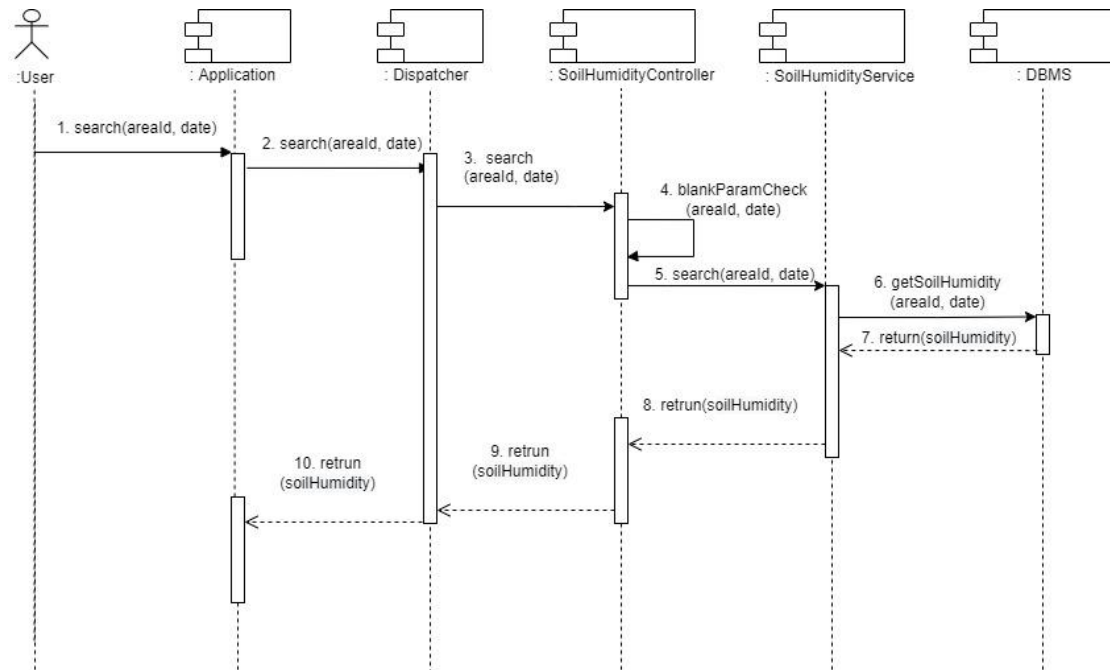
This diagram shows the sequence of adding a production data type. When a user enters the information, the application sends a request to the server, the request is distributed to ProductionTypeController. The ProductionTypeController check the parameters are blank or not, then sends the request to ProductonTypeService. The ProductionTypeService requests the DBMS method to save the production data and obtain the result. Finally, the result will be delivered to the Application.

2.4.4 Add soil humidity data



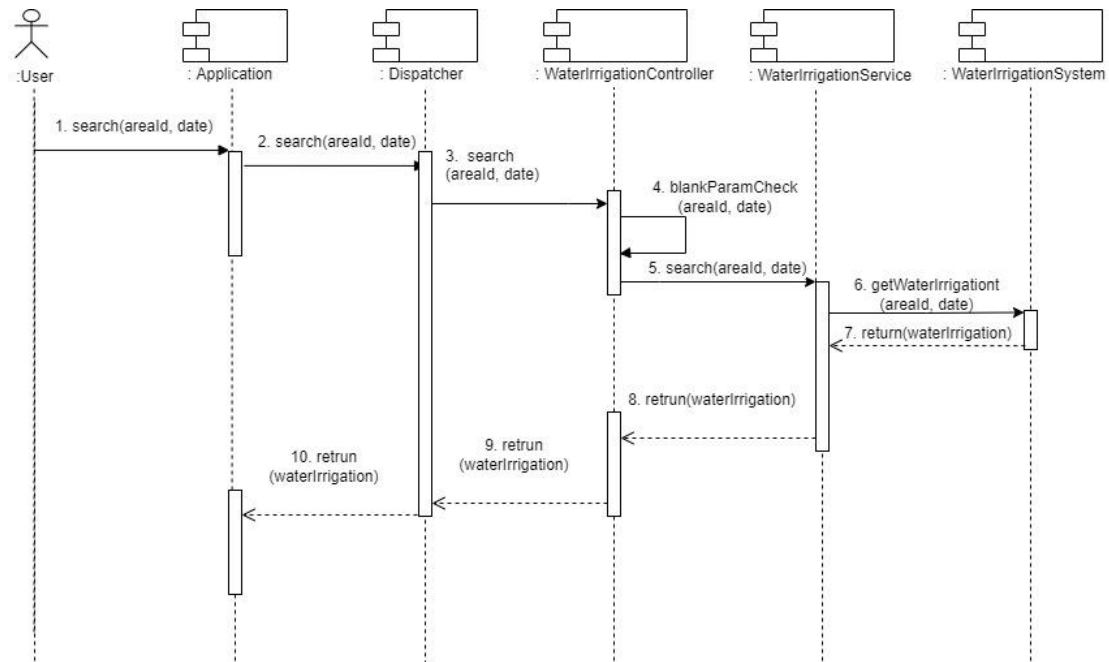
This diagram presents the sequence of adding a production data type. When SoilHumiditySensor collects the information, the application sends a request to the server, the request is distributed to SoilHumidityController. The SoilHumidityController check the parameters(areald, date, index) is blank or not, then sends the request to SoilHumidityService. The SoilHumidityService requests the DBMS method to save the production data and obtain the result. Finally, the result will be delivered to the Application.

2.4.5 Search soil humidity



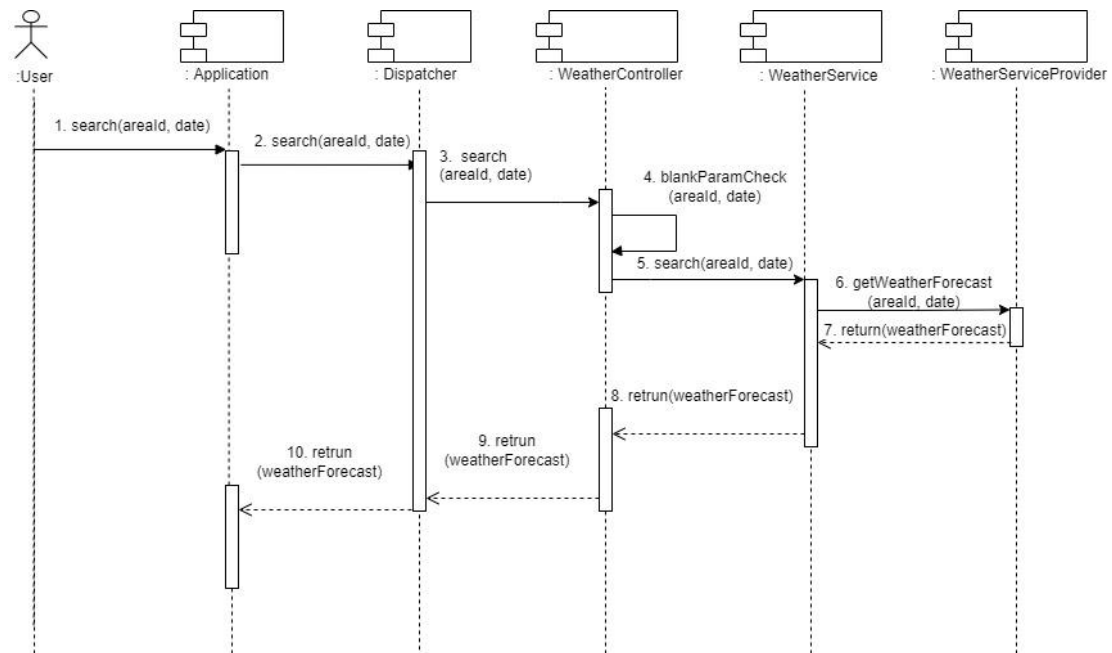
This diagram shows the sequence of search soil humidity data. A user enters the area and date and clicks the search button, the request will be sent to the server, the system distributes the request to the corresponding controller. The SoilHumidityController checks the parameters are blank or not, then sends the request to the SoilHumidityService. The SoilHumidityService queries the data by areald and date through DBMS. After getting the result, the result will be sent to the Application.

2.4.6 Search water irrigation data



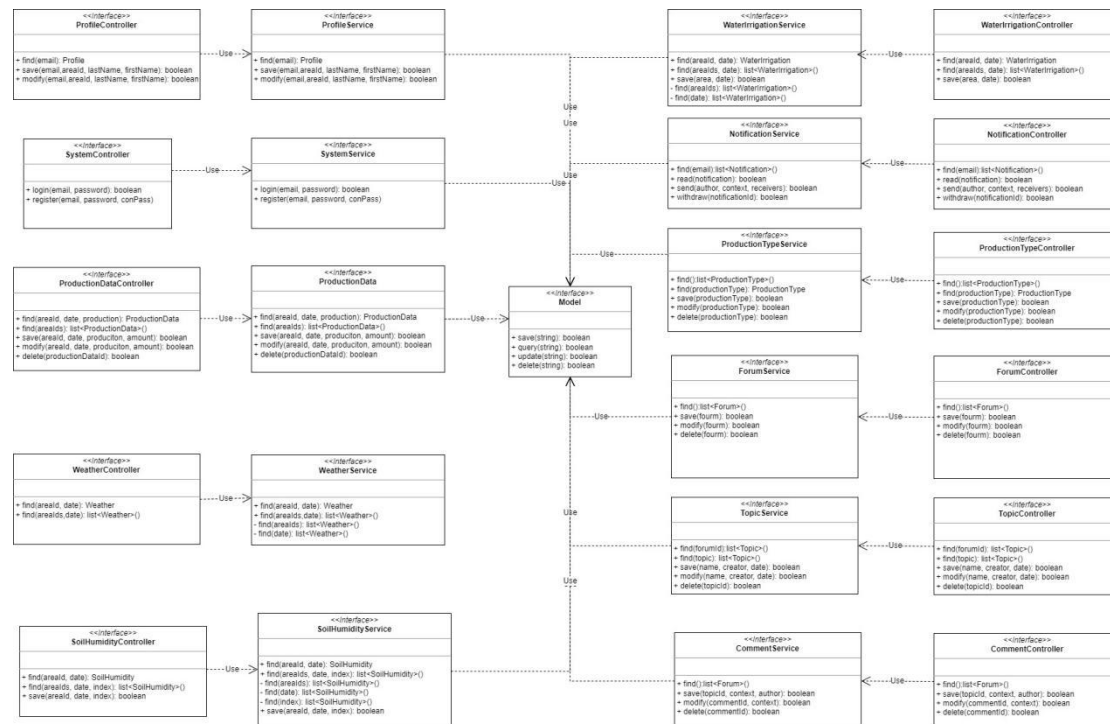
This diagram shows the sequence of user search data of water irrigation. When a user enters the areald and date and clicks the search button, the Application will send a request to the server. The request is distributed by the dispatcher to WaterIrrigationController. The controller checks the parameters are blank or not, then sends them to the WaterIrrigationService. The service searches the data by the parameters through the WaterIrrigationSystem. When it obtains the result, the result is sent the result back to the last invoker and finally reaches the Application.

2.4.7 Search weather forecast



This diagram shows the sequence of searching weather forecasts. A user enters the information and clicks the search button. The request will be sent to the server, when the server receives the request, the request will be distributed to the WeatherController. The controller checks the parameters are blank or not and sends the call to the search method to get the result. After obtaining the result, the result will be sent to the Application.

2.5 Component interfaces



The diagram above shows more detail of the interface that is offered by each component, also it presents the dependency between each component. For **WeatherService**, this component has no dependency on DBMS, because it uses **WeatherServiceProvider** as the data source.

2.6 Selected architectural styles and patterns

- B/S(Browser/Server) Architecture
B/S Architecture is platform-careless. Its application can run if the device has a browser and internet connection, despite the operation system(e.g. Mac, Windows, Linux, Android, IOS). Furthermore, it can run without configuring users' devices compare with C/S Architecture.
- MVC (Model-View-Controller) pattern
MVC is the pattern that separates the system input, processing of system, and system output into three individual parts without overlap, which leads to the benefit that these three parts can handle their tasks.

The model represents enterprise data and business rules, In three parts of MVC, the model has the most processing tasks.

The view is the interface that the user sees and interacts with. A benefit of MVC is that it can handle different views. There is no processing in the view, it is just a way to show data and allow the user to manipulate it.

The controller takes input from the user and calls models and views to fulfill the user's requirements. The controller itself does nothing but receive a request and invokes methods, and then determines which view to use to display the returned data.

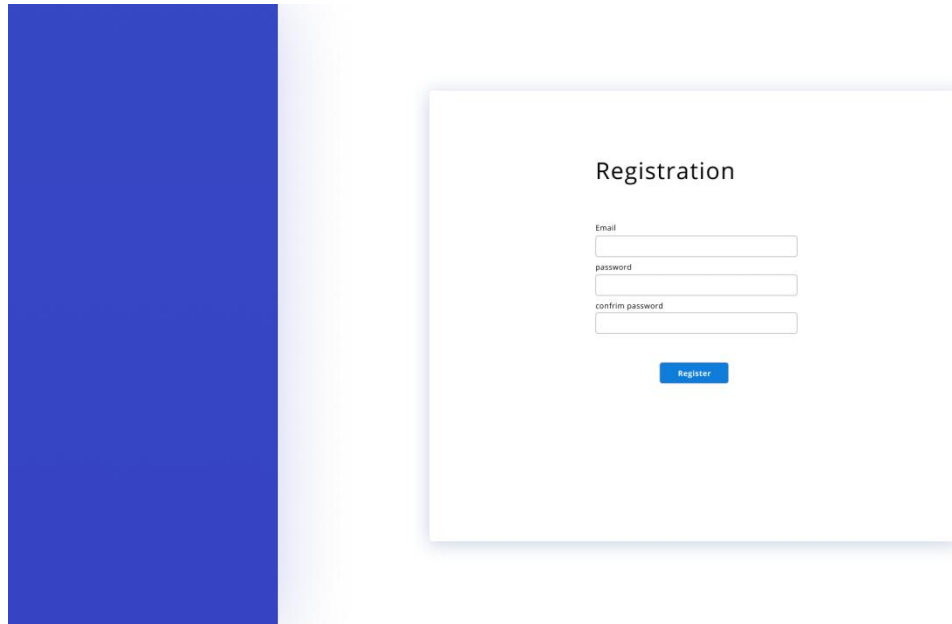
2.7 Other design decisions

- Single responsibility principle
Each class or method just deals with their strong-connect data or processes. If a class has too many responsibilities, a change in one responsibility may impair or inhibit the class's ability to perform other responsibilities. This coupling can lead to a fragile design that can suffer unexpected damage or problem, especially during the refactoring.

There are two main benefits, which are decoupling and cohesion. These two benefits can increase the reusability and maintainability of the code.

3 User interface design

3.1 Registration page



The registration form is presented as a white card with a subtle drop shadow, set against a background with a solid blue vertical bar on the left and a light blue gradient on the right. The card is titled "Registration" in a bold, black font. Below the title, there are three input fields: "Email", "password", and "confirm password", each with a small label above it. A blue "Register" button is positioned at the bottom center of the card.

Registration

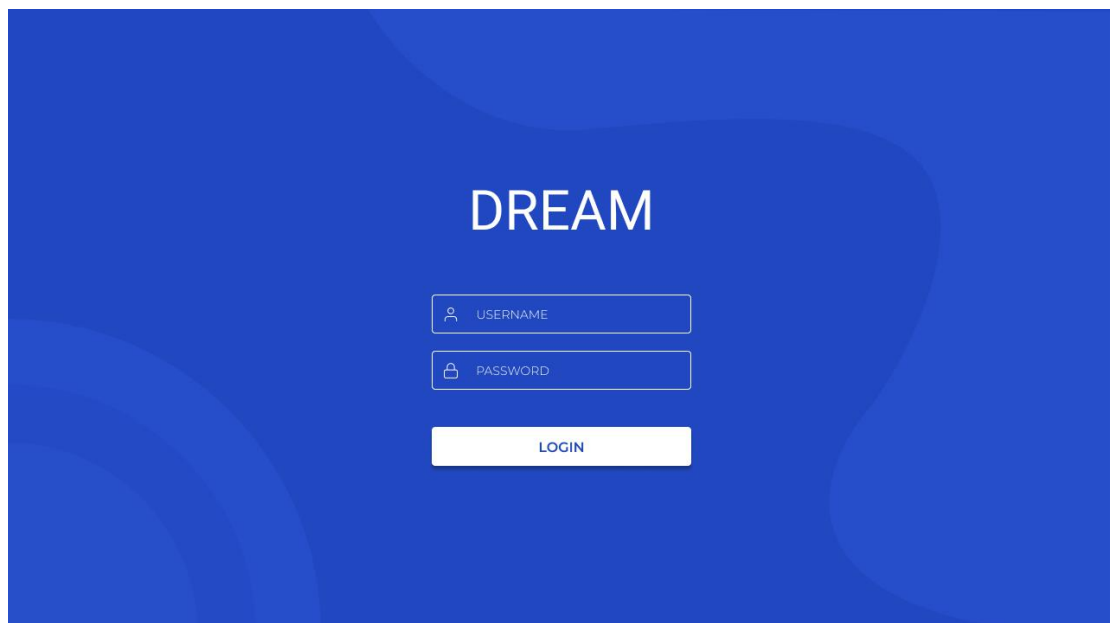
Email

password

confirm password

Register

3.2 Login page



The login page features a solid blue background with abstract, lighter blue organic shapes. The word "DREAM" is centered at the top in a large, white, sans-serif font. Below it, there are two input fields: "USERNAME" with a user icon and "PASSWORD" with a lock icon. A white "LOGIN" button is centered at the bottom.

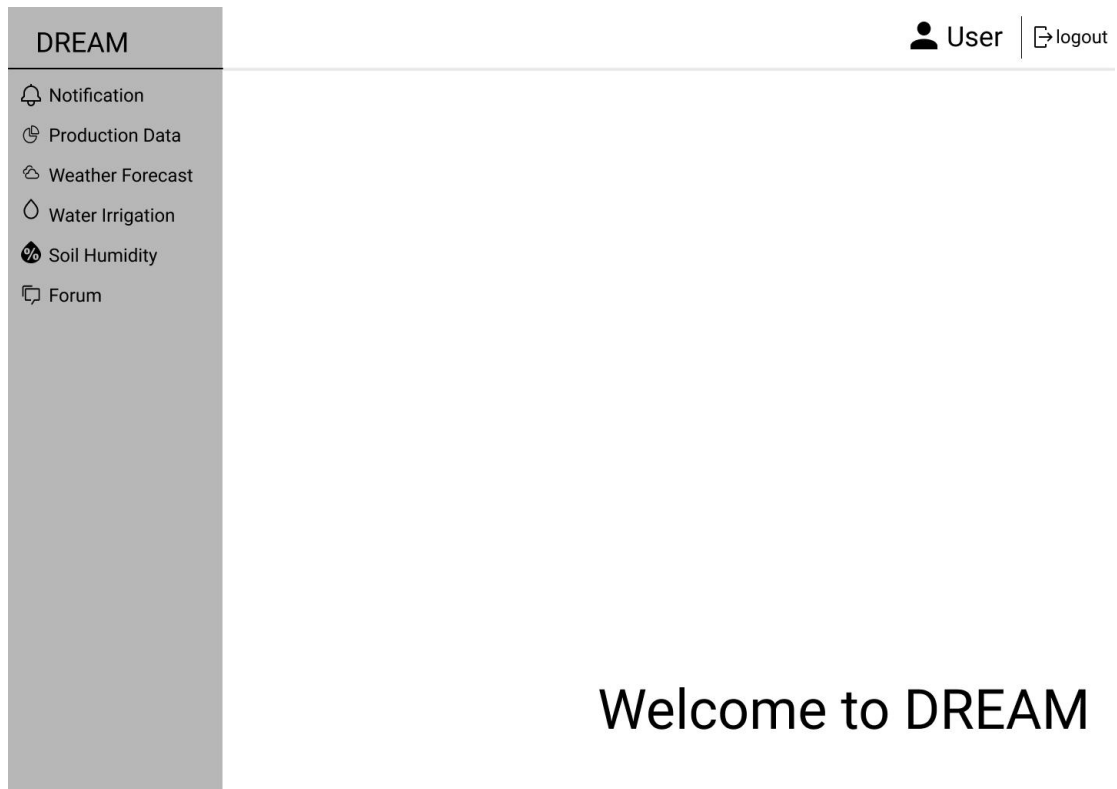
DREAM

USERNAME

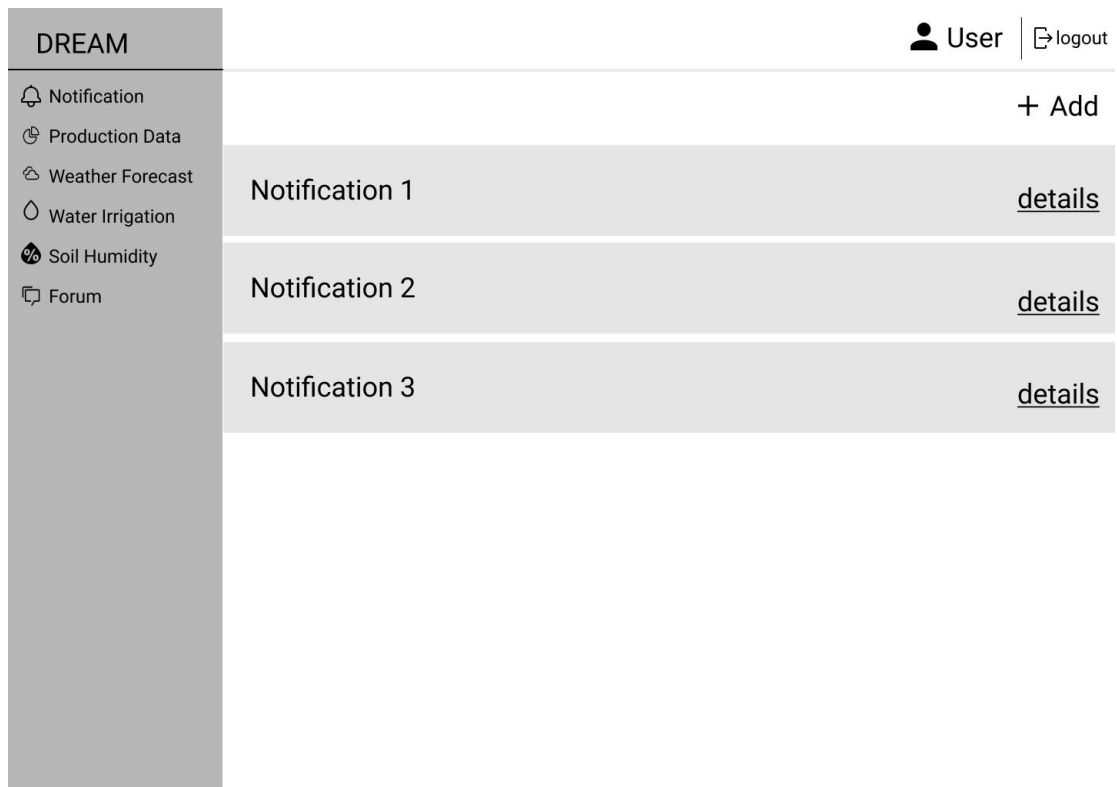
PASSWORD

LOGIN

3.3 Main page



3.4 Notification



3.5 Production data management

DREAM

Notification

Production Data

Weather Forecast

Water Irrigation

Soil Humidity

Forum

Search an area

+ Add

Production name	Production type	Area	Planted date	amount
onion	vegetable	area1	01/12/2021	10
apple	fruit	area2	01/05/2021	10
banana	fruit	area3	01/06/2021	10
rice	Oryza L.	area4	01/07/2021	10
onion	vegetable	area5	01/09/2021	10
apple	fruit	area6	01/10/2021	10
banana	fruit	area7	01/02/2021	10
rice	Oryza L.	area8	01/01/2021	10
onion	vegetable	area9	01/03/2021	10
apple	fruit	area10	01/04/2021	10

3.6 Weather forecast

DREAM

Notification

Production Data

Weather Forecast

Water Irrigation

Soil Humidity


Forum


Search an area





3.7 Water irrigation data


DREAM


 Notification


 Production Data

 Weather Forecast


 Water Irrigation


 Soil Humidity

 Forum

 Search an area


Area	Date	Amount
area1	01/22/2021	300
area2	01/23/2021	400
area3	01/22/2021	361
area4	01/22/2021	324


 User


 logout


3.8 Soil humidity data


DREAM


 Notification


 Production Data

 Weather Forecast


 Water Irrigation


 Soil Humidity

 Forum

 Search an area

Area	Date	Index
area1	01/22/2021	20%
area2	01/22/2021	21%
area3	01/22/2021	22%
area4	01/22/2021	19%

 User

 logout

3.9 Forum

DREAM	<div>User logout</div>
<div><div><div>Notification</div><div>Production Data</div><div>Weather Forecast</div><div>Water Irrigation</div><div>Soil Humidity</div><div>Forum</div></div></div>	<div><div><div>+ Create a forum</div></div></div>
	<div><div>Forum 1</div><div>Topics</div></div>
	<div><div>Forum 2</div><div>Topics</div></div>
	<div><div>Forum 3</div><div>Topics</div></div>

3.10 Topic

DREAM	<div>User logout</div>
<div><div><div>Notification</div><div>Production Data</div><div>Weather Forecast</div><div>Water Irrigation</div><div>Soil Humidity</div><div>Forum</div></div></div>	<div><div><div>+ Create a topic</div></div></div>
	<div><div>Topic 1</div><div>comments</div></div>
	<div><div>Topic 2</div><div>comments</div></div>
	<div><div>Topic 3</div><div>comments</div></div>

4 Requirements traceability

The following matrix shows which component contributes to the realization of a certain requirement. It contains components and requirements, when the component fulfills the requirement, there will be a x in the intersection cell to indicate.

Due to the ‘Service’ components being responsible for the data operation, we use the “Service” component to check the requirement traceability.

In order to all the component under the page width constrain, the abbreviations are used as follow:

SystemService: SS

ProfileService: PS

ProductionDataService: PDS

WeatherService: WS

SoilHumidityService: SHS

WaterIrrigationService: WIS

NotificationService: NS

ForumService: FS

TopicService: TS

CommentService: CS

Model: M

DBMS: DBMS

[illegible]

R17			x								x	x
R18			x								x	x
R19						x					x	x
R20					x						x	x
R21							x				x	x
R22							x				x	x
R23							x				x	x
R24								x			x	x
R25									x		x	x
R26										x	x	x

5 Implementation, integration, and test plan

5.1 Implementation overview

This section talks about the implementation of the DREAM project. We will use the Spiral Model and bottom-up theory to control the process of implementation.

For coding, the DBMS-related component should be put at the first place to develop, because it provides a fundamental method to support the data management. After finishing the development of DBMS-related coding, we can start to code the upper layer component.

For integration, according to bottom-up theory, when the children-component is done, the parent-component can start to integrate.

For testing, it ought to start as soon as possible, because a bug was found more lately, it would cost more effort to fix the bug. There are unit testing, integration testing, system testing, and acceptance testing. Every developer should unit test when the developer finishes a feature, and take an integration test when they are integrating different components.

5.2 Integration plan

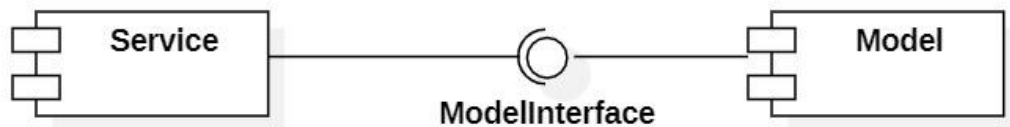
The integration of different components is carried out by the base bottom-up theory. The order of component integration is described as follow:

1. The DBMS and model component



Model components play as single access to the database, the priority of integration of DBMS and model components should be carried out first.

2. The model component and the “service” components



The model component is responsible for the data access and operation, and the “service” components handle all the business processes. So these two types of components should be integrated first, and there is no specific priority between “service” components.

3. The “service” components and the “controller” components



Each “controller” component has a corresponding “service” component. After integration between the model component and the “service” components, the integration between the “controller” component and “service” should be carried out.

6 Effort spent

ZHENGLIANG MA(10824853)

Topic	Hours
General reasoning	8:00h
Component view	7:00h
Deployment view	1:30h
Component interfaces	9:00h
Runtime view	5:00h
Selected architecture	2:00h
Other design decisions	1: 00h
User interface	6: 00h
Implementation, integration, and test plan	3:00 h
Requirement traceability	1:00h
Document organization	5:00h

BOREN JIANG(10825686)

Topic	Hours
General reasoning	8:00h
Component view	2:30h
Deployment view	3:00h
Component interfaces	1:30h
Runtime view	2:00h
Requirement traceability	1:00h
Document organization	3:00h

7 References

UML diagrams are made with [StarUML](#) and [diagrams](#)

User interfaces are made with [Figma](#)