SAS brought by **MotherCoding**

Smart **A**gricultural **S**ystem

Project Description:

The SAS system tackles various problems concerning agriculture, some of the most illuminating are the deficiency of agricultural land near urban areas like cities and countrysides nowadays, the inefficiency of the agricultural plans and schedules followed by farmers around the country and the miss use of fertilisers and pesticides and many others like power consumption etc., Besides solving problems, the system also aims to increase the efficiency of crop farming and make farming independent of terrain or distance separating the lands farmed and residential areas.

Project members :
Abdelrahman mohamed mostafa kamel (project manager) 21100911
Abdelrahman Abdelkader
Mohamed ibrahim
Ahmed abdo ahmed
Youssef Ahmed 2010251
Ahmed Mohamed abdelhady 20100453

Customer requirements :

1. Eases agricultural activity in deserted places. Example: deserts
2. Make a system capable of identifying minerals in the soil.
3. Make a system capable of recommending the fertilisers and pesticides needed
4. Make a system that sets the best suited agricultural plan no matter what terrain is being cultured
5. Give the customer the option of controlling the system semi-manually or set it to fully automated.
6. System collects data and sets them into a viewable form
7. The system keeps learning which allows it to produce more efficient agricultural plans
8. System supports SDG by making plans more water saving
9. Most of the system should be able to be powered by solar energy
10. The system should have a glossary containing general information about plants

11. The software can advice the user on which plants are best suited to plant in the terrain and which would the most difficult to plant, in addition to more information and support options
12. Software should connect to an industrial company (RainBird) through a shop built into the software, where the user can place orders for more equipment, improvements and additions to his agricultural land.

USER INTERFACE :

Subsystem Name	Subsystem Function	Subsystem interface
UI Subsystem	frontend GUI subsystem which is used to communicate the user with other software components	createButton(name, x, y, z) writeOnScreen(text, x, y, z) createInputBar(x, y, z) createGraph(x-axisData, y-axisData, x, y, z)

UserManagement	a class dedicated for logging in and signing up to the software and it communicates with the network subsystem for user authentication	signUp(name, address, email, dateOfBirth, mobileNo)
		logIn(username, password)
		changeUsername(oldUsername newUsername)
		changeAddress(oldAddress, newAddress)
		changeMobileNo(oldMobileNo newMobileNo)

Network Subsystem	utilises retrieving readings from a dedicated private server connected by the internet or via closed network	logIntoDB(username, password, domain)
		logIntoShop(username, password, domain)
		sendSQL(string)
		retrieveDBTables()
		HTMLView(URL)
		checkWIFIStatus()
		checkServerStatus()
		checkConnectivityStatus()
		checkSensorsStatus()

User Configuration Subsystem	partition the land into different crop fields and identify the types of plant that is grown in each partition as well as selecting the region or location in which the farm is.	setLandCoordinates(x, y):landPartition setSprinklingSchedule(landPartition configurationFile)

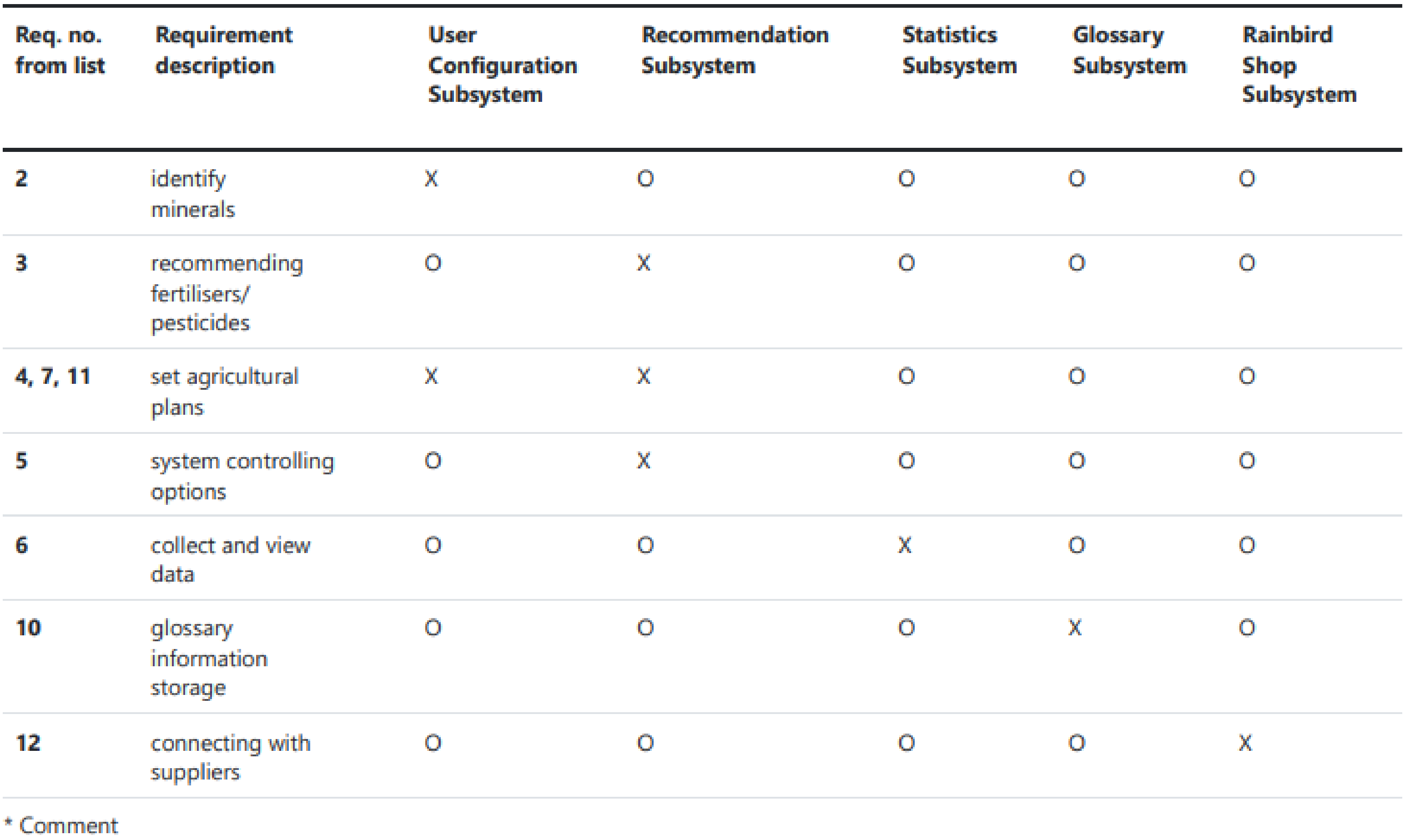
System Recommendation Subsystem	The system should recommend pesticide and fertilizer type based on the type of the grown plant by accessing a specific database for the corresponding recommendations. The system can make agricultural plan for the specific plant based on various data including sensor readings, region and biome, weather data (AccuWeather web scraping), and plant type. The recommendations include the best time	getFertilizerRecommendations ype)
		getPesticideRecommendations ype)
		getPlanRecommendation(PlantType biomeType)
		getPowerConsumptionInfo()
		setRegionBiome(biomeType)
		getTodaysWeather()

	for seeding, the best period for harvesting, the optimal sprinkling schedule, when to use fertilizers, and how often should the user remove weed from crops.	activateAutoMode()

Statistics and Analysis Subsystem	Shows a summary that includes the dates for harvesting and the expected growth rate.	getFullSummary()
		getFullStatistics()

Glossary Subsystem	Shows info pages for searching plant types, fertilizer types, and pesticide types.	viewPlantAtlasWebpage() viewFertilizerTypesWebpage viewPesticideTypesWebpage
Rainbird Shop Subsystem	An interactive shop that the user can interact with and buy specific addons for the system.	

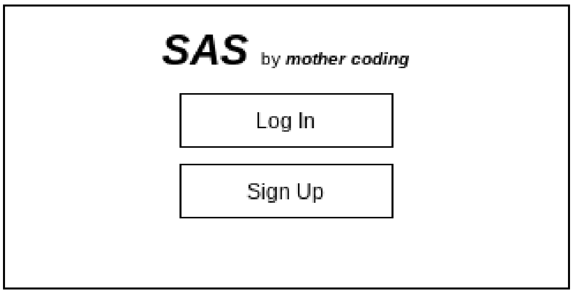
C Requirements trace matrix

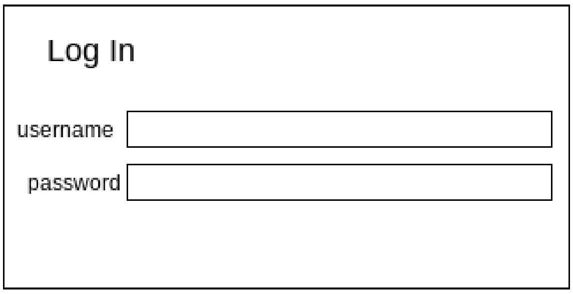


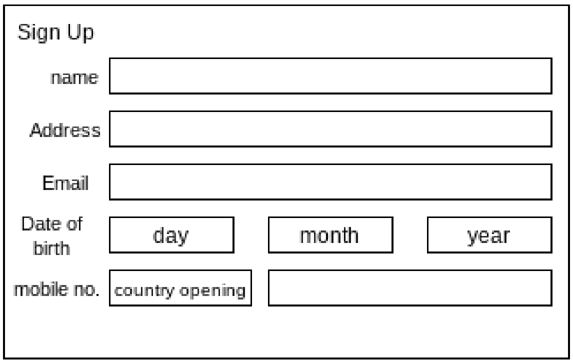
Req. no. from list	Requirement description	User Configuration Subsystem	Recommendation Subsystem	Statistics Subsystem	Glossary Subsystem	Rainbird Shop Subsystem
2	identify minerals	X	O	O	O	O
3	recommending fertilisers/ pesticides	O	X	O	O	O
4, 7, 11	set agricultural plans	X	X	O	O	O
5	system controlling options	O	X	O	O	O
6	collect and view data	O	O	X	O	O
10	glossary information storage	O	O	O	X	O
12	connecting with suppliers	O	O	O	O	X

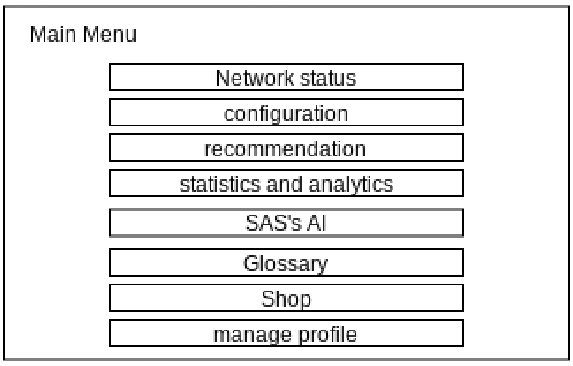
* Comment

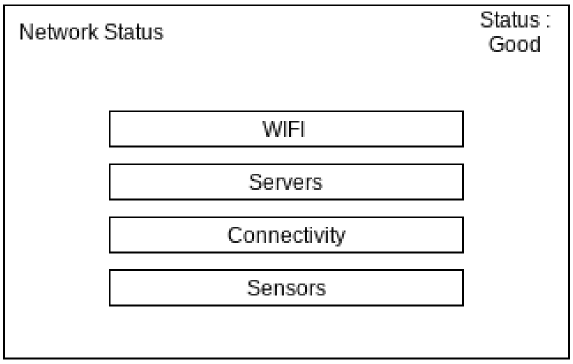
GUI :

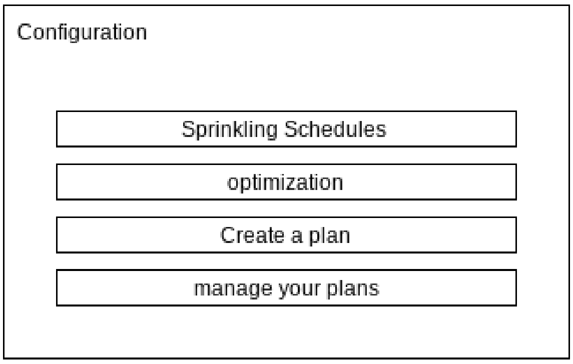












1. SignUp/LogIn page:
   This screen directs the user to one of two pages. Login or sign in.

2. Log In :
   Helps the user create his new account, the user inputs his information and it is then saved into the Database.

3. Sign In :
   User enters his name and the password he set to get access into the system.

4. Main Menu:

The main menu contains all the features the system has to offer, where the user gets to direct to any of them, and the features include network status, configuration, recommendation, statistics and analytics, the AI, the glossary and his profile management centre.

5. Network Status:
   Helps the user monitor everything from the WIFI connection to the sensors.

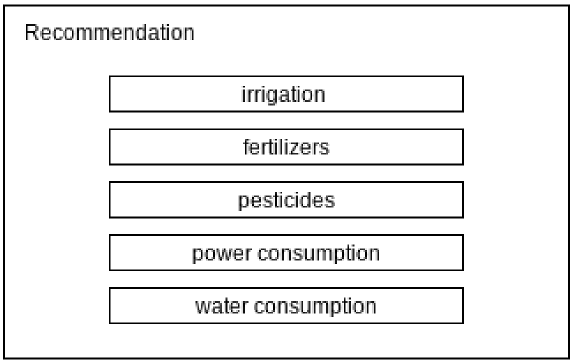
5.1. WIFI: where the user chooses his connection to the internet

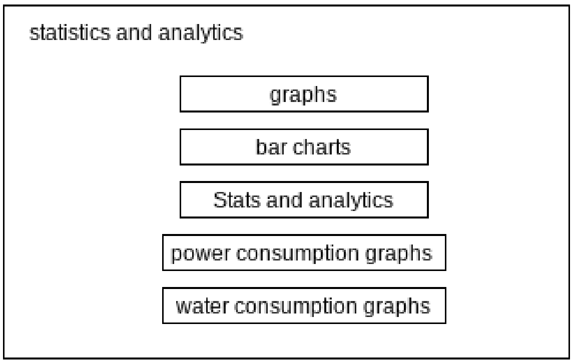
5.2. Servers: where the user displays and monitors the servers which collect data and facilitate connectivity

5.3. Connectivity: where all the connections between devices and database systems are displayed and monitored

5.4 sensors: where the user can display the status, condition and the data collected by the sensors, also displays errors and shows the user where the malfunction is coming from

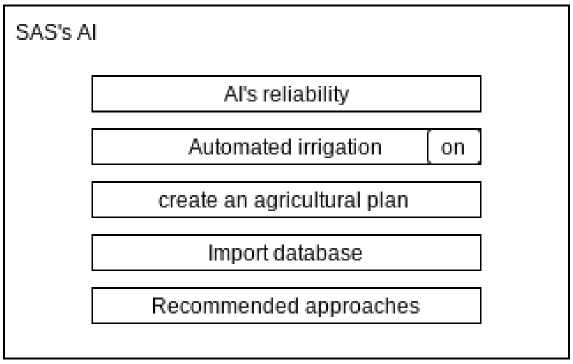
6. Configuration:
   Where the user goes to set and reset, review and display his plans and configurations and it includes:
   6.1. Sprinkling schedules: where the user goes to set new sprinkling schedules, reset them or make tweaks.
   6.2. Optimization: where the best setting for the plan are set, with a collection of filter settings like best power consumption and water consumption
   6.3. Create plan: where the user creates new self prepared agricultural plans
   6.4. Manage plans: where the user edits already set plans

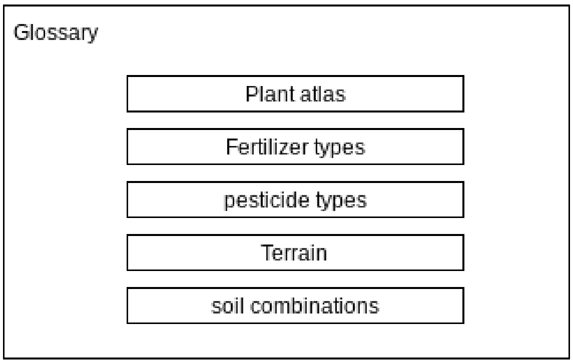


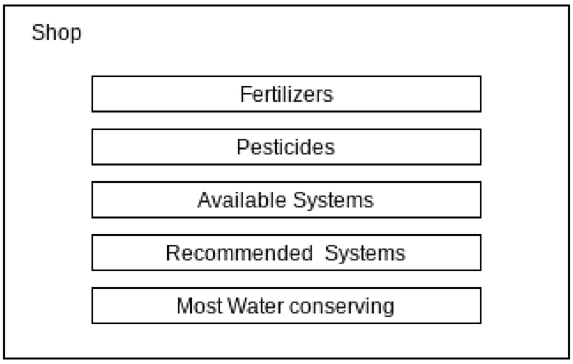


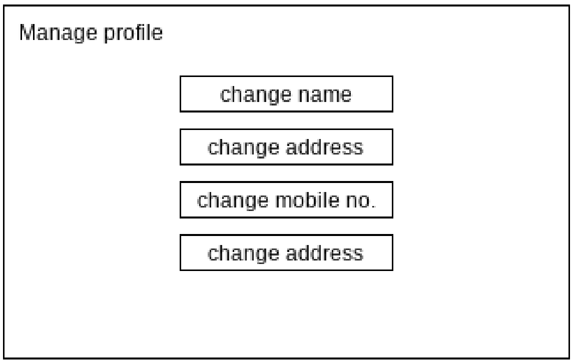
7. Recommendations:
   It is where the best recommendations by the AI are displayed for the user to consider and take advantage of the Ai’s capabilities of making suggestions

8. Statistics and analytics:
   Where the data collected by the sensors are sent to the AI and turned into information, the AI creates everything from graphs to charts to best performance plans









9. SAS’s AI:
   This partition lets the AI control the system 100% without any human intervention, and can also with the help of AI, create a wholly automated agricultural plan in consideration to all the kinds of data collected by the sensors, which the AI then processes.

10. The Glossary:

The glossary is where all the information collected by the system from the sensors, the web and other third parties is contained, categorised and displayed, the glossary contains all and any information the user might want to search or look for.

11. Shop:
This is where the user gets access to buying supplies, support items and more.

12. Manage profile:
Here the user can add, change or delete information

Functional requirements:

1. The user will use the Subsystem {Configuration} to partition the agricultural land, they shall also review, reset and display their entries from the database.

1.1. The user will not be able to use this subsystem if they didn't have their soil data and readings connected or manually entered into the database.

2. The Subsystem{systemRecommendation} will recommend pesticide and fertilizer type based on the type of the grown plant by accessing specific parts of the database for the corresponding recommendations.

2.2. The system can make agricultural plan for the specific plant based on various data including sensor readings, region and biome, weather data (AccuWeather web scraping), and plant type. The recommendations include the best time for seeding, the best period for harvesting, the optimal sprinkling schedule, when to use fertilizers, and how often the user should remove weeds from crops.

2.3. The user can not use this subsystem unless they have configured their agricultural land.

3. The subsystem{StatisticsAndAnalytics }will facilitate the display of all the data mined, read or collected into the database in a vast number of presentable charts and graphs.

4. The Subsystem {SASAI} will facilitate the AI and control the system without nearly any human intervention, will also with the help of AI, create a wholly automated agricultural plan in consideration to all the kinds of data collected by the sensors, which the AI then processes into testing data for a machine learning process that will later improve crop quality by increasing the systems accuracy and efficiency.

5. The glossary is where all the information collected by the system from the sensors, the web and other third parties is contained, categorised and displayed, the glossary contains all and any information the user might want to search or look for, it categories and filters the data provided by the database, the web and third party sources for future use and processing.

6. The subsystem{RainBirdShop} displays an interactive shop where the user can interact with and buy specific addons for the system, this subsystem should be able to access the database and to identify which or what addons the user might need most according to his current status.

Non-Functional requirements:

1. UserManagement

subsystem shall provide the user with vast profile customization and personalization, they can read, delete, import and replace profile data whenever they choose.
1.1. The use cant access the subsystem if they are not signed into the system

1.2 the passwords should be two step authenticized

2. **Security**

   2.1.Account creation: Systems may require users to create accounts to access applications that store information and display profiles. A security system typically grants access to accounts when users enter the correct username and password.

2.2. Password generation: An application may not grant access until the user creates a strong password. For example, a strong password might contain a certain number of characters and a capital letter.

   2.3. Security question answering: A security system for a product may ask questions that only the user knows the answer to. This can help verify a user's identity when they log into an account. Examples of security question topics include the color of your first car or your mother's maiden name.

   2.4. Account locking: After a certain number of login attempts, a security system may lock an account to protect a user's information from potential hackers. To unlock their account, a user can typically call the company to verify their identity and set a new password.

3. **Localization**

Software must have features that match the geographical location of its users, including aspects such as:

- Languages
- Currencies
- Measurements, such as pounds vs. kilograms
- Time zones

4. NetworkConnectivity:

4.1. The subsystem {NetworkConnectivity} should be able to monitor the connections between all physical hardware components and the software's database

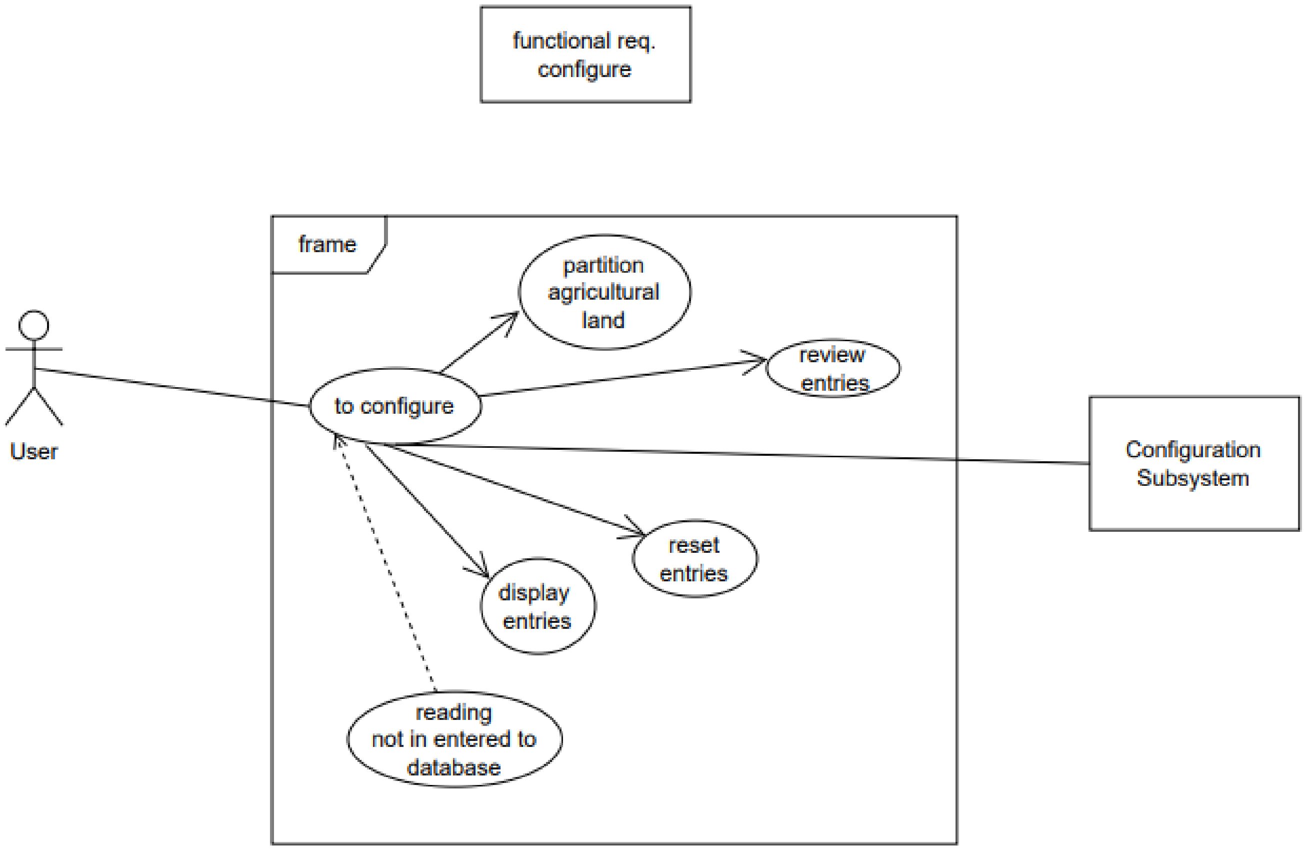
4.2.the subsystem should display the strength and status of each component whether it is a hardware component or just the wifi connection to the device used for navigating the software.

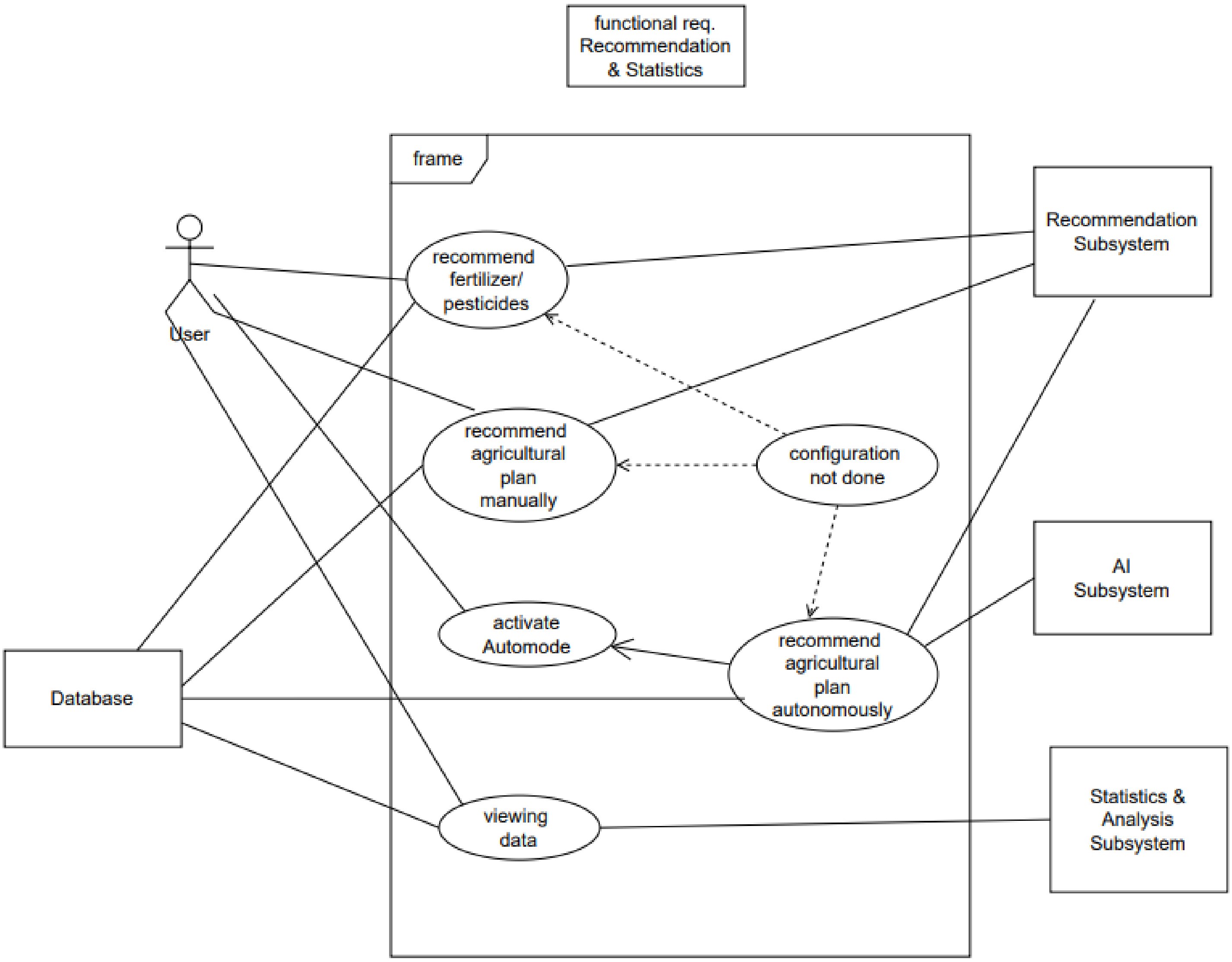
5. API

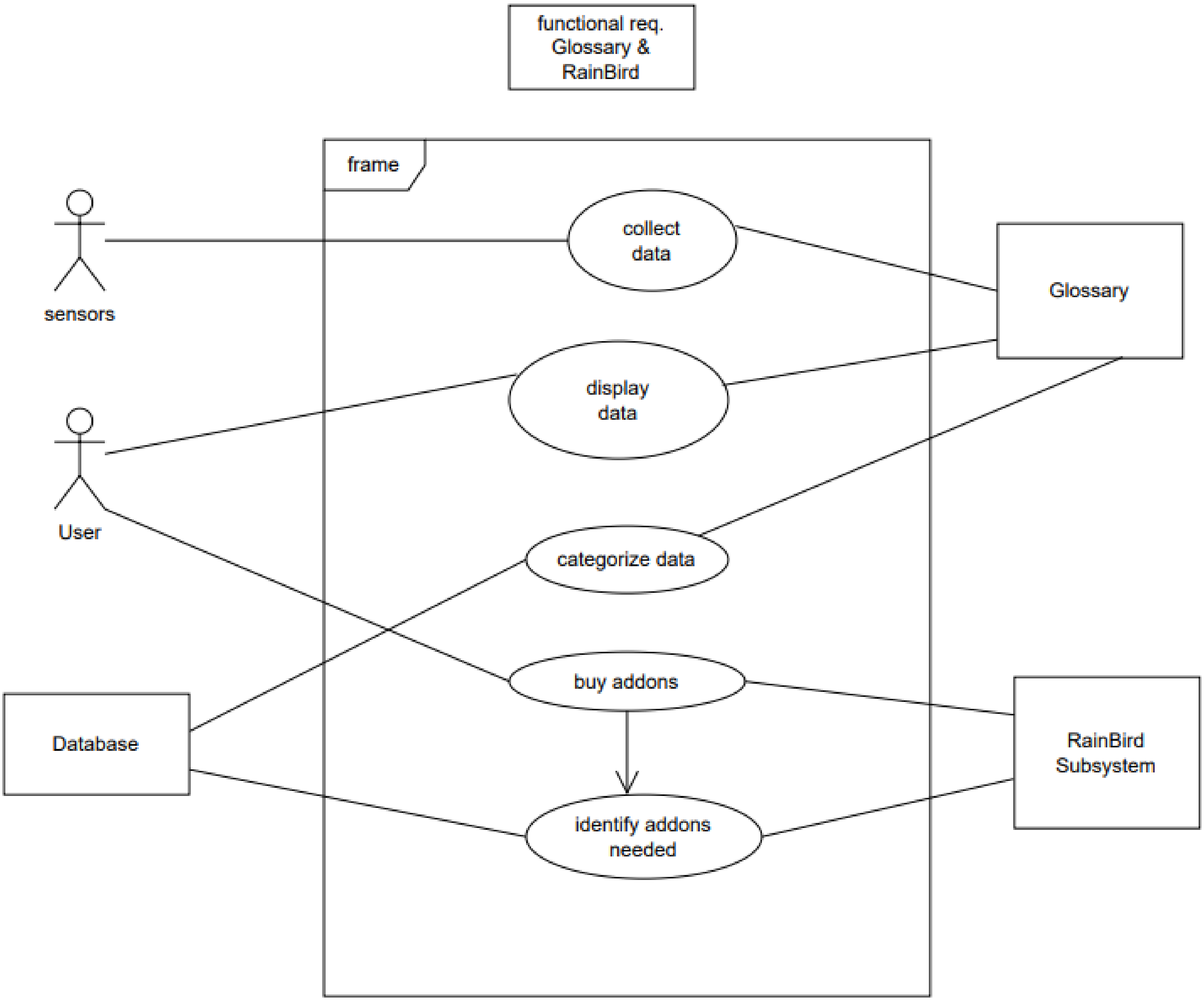
5.1. Should facilitate login and registration services by at first collecting the users data and personal information

5.2 password replacement and retrieval should be provided, but only accessible when the user passes two step authentication.

Use Case Diagrams:




functional req.
Recommendation
& Statistics
frame
User
recommend fertilizer/ pesticides
recommend agricultural plan manually
configuration not done
activate Automode
recommend agricultural plan autonomously
viewing data
Database
Recommendation Subsystem
AI Subsystem
Statistics & Analysis Subsystem


functional req.
Glossary &
RainBird
frame
sensors
User
Database
collect data
display data
categorize data
buy addons
identify addons needed
Glossary
RainBird Subsystem

Use case scenarios:

Use case: **viewing data**
Goal and context: user view data
Preconditions: data present in database
Success End Conition: data is successfully viewed to user
Primary Actor: User
Secondary Actors: database, Statistics & Analysis Subsystem
Trigger: User want to view the data

Use case: **to configure**
Goal and context: review entries
Preconditions: readings entered to database
Success End Conition: user reviewed entries
Primary Actor: User
Secondary Actors: Configuration Subsystem
Trigger: User want to review entries

Use case: **display data**
Goal and context: to display data in glossary
Preconditions: data is categorized
Success End Conition: data is successfully displayed to user
Primary Actor: User
Secondary Actors: database, glossary
Trigger: User want to display the data in glossary

Use case: **recommend agricultural plan**
autonomously
Goal and context: recommending and controlling agricultural plan without human interaction
Preconditions: configuration already done
Success End Conition: agricultural plan is automated
Primary Actor: AI Subsystem, Recommendation Subsystem
Secondary Actors: database, User
Trigger: User Activate Auto Mode

Use case: **buy addons**
Goal and context: to buy needed addons
Preconditions: needed addons are identified
Success End Condition: needed addons are buyed successfully
Primary Actor: User
Secondary Actors: database, RainBird Subsystem
Trigger: User want to buy missing addons
after the Subsystem identify them to him

Use case: **categorize data**
Goal and context: to categorise data collected from sensors
Preconditions: data is collected from sensors

User Interface Validation:

0.0.1 Verify that there is a menu option for viewing data.

0.1.1 Verify that the data viewing button in the main menu returns the right submenu interface.

0.1.2 Verify that the new submenu has the data viewed in tabular form.

0.2 System Connectivity Validation:

0.2.1 Verify that the local database is connected before calling the required readings data.

0.2.2 Verify that the sensors are connected to the local database.

0.2.3 Verify that the sensors are sending readings in real time to the database.

0.3 Data Integrity Validation:

0.3.1 Verify that the sensor readings data are still up to date (Last 15 minutes)

0.3.2 Verify that the required data isn’t corrupted or of different type.

2.1 Configuration Saving/Reading Validation:

2.1.1 Verify that the changed system configurations are being saved to file.

2.1.2 Verify that the system configuration file is being read by the system.
2.1.3 Verify that the settings change the system functions and behaviors accordingly.
2.2 Hardware changes validation:
2.2.1 Verify that the system changes apply to hardware that activates sprinklers as well.
2.2.2 Verify that the settings give the proper sprinkling scheduled times.

3.1 System Connectivity Validation:
3.1.1 Verify that the local database is connected before calling the required readings data.
3.1.2 Verify that the sensors are connected to the local database.
3.1.3 Verify that the PC or Mobile in which the software is running is connected to the internet.
3.1.4 Verify If the system has sent log-in info to third parties for accessibility.
3.2 Search Validation:
3.2.1 Verify that case matching is disabled.
3.2.2 Verify that numbers are allowed.
3.2.3 Verify that the query keyword entered is in English.
3.2.4 Verify that the entered query doesn’t interfere with the system’s functions.
3.2.5 Verify that the query keyword entered is present in the glossary.
3.3 Results Validation:
3.3.1 Verify that search page returns the corresponding search pages.
3.3.2 Verify that the user gets presented with “no results” if the entered query doesn’t match anything present.
3.3.3 Verify that the results with title matching has higher priority than content matching.

4.1 User interface Validation:
4.1.1 Verify that the system recommendations sub-menu is accessible and displayed correctly.
4.1.2 Verify that the sub-menu provides options for selecting soil and plant partitions.

4.2 Functional Validation:
4.2.1 Verify that the AI subsystem receives the user's selections of soil and plant partitions correctly.
4.2.2 Verify that the AI subsystem provides customized fertilizer recommendations for each plant type based on the selected soil and plant partitions.
4.2.3 Verify that the AI subsystem provides sprinkling schedules for each plant type based on the customized fertilizer recommendations.
4.3 Reliability Validation:
4.3.1 Verify that the system provides reasonable amount of pesticides and fertilizers to be used for the specific area.

5.1 User Interface Validation:
5.1.1 Verify that the user can access the addons submenu from the main menu of the application.
5.1.2 Verify that the store is displayed with all available addons and their corresponding prices.
5.2 Connectivity Validation:
5.2.1 Verify that the store is connected to the rainbird warehouse and that the items are in stock.
5.2.2 Verify that the user is presented with an error message if they try to access the store submenu without an internet connection.
5.3 Transaction Validation:
5.3.1 Verify that the user can complete a transaction even with intermittent internet connectivity.
5.3.2 Verify that the application can handle unexpected network disconnections without data loss or errors.
5.3.3 Verify that the credit card payment process requires bank authentication.
5.3.4 Verify that the payment details are securely processed and stored.
5.3.5 Verify that the payment gateway is secure and encrypted to prevent unauthorized access.
5.3.6 Verify that the application can handle multiple concurrent transactions without issues.
5.4 Usability Validation:
5.4.1 Verify that the user can view their purchase history.

5.4.2 Verify that the user can cancel an order before it is shipped.
5.4.3 Verify that the user can track their order once it has been shipped.

6.1 User Interface Validation:
6.1.1 Verify that the diagram representing the land area is displayed correctly.
6.1.2 Verify that the user can partition the land into different areas by drawing boundaries using the cursor.
6.1.3 Verify that the user can assign different crop types to each area.
6.2 Functionality Validation:
6.2.1 Verify that the system can save the configurations after the user assigns crop types to each area.
6.2.2 Verify that the system can communicate with the sprinklers to water the specific crop depending on its location.
6.3 Performance Validation:
6.3.1 Verify that the system can handle a large number of areas and crops.
6.3.2 Verify that the system can communicate with the sprinklers in a timely manner.
6.4 Integration Testing:
6.4.1 Verify that the system can integrate with different types of sprinklers and watering systems.