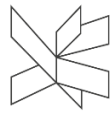

BPR 2- DRONE SURVEY SYSTEM - PROJECT REPORT

Bring ideas to life
VIA University College**Information and Communication Programme***7th Semester*

Group Members: Alin George Cortel (164625)
Alexandru Ungureanu (224469)
Sigmundur Johansen (253916)

Supervisors: Allan Henriksen

Numbers of characters: (79184) (with spaces)

Date: 15.12.2017

Alin George Cortel



Alexandru Ungureanu



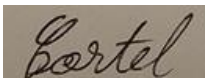
Sigmundur Johansen



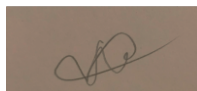
“We hereby declare that our project group prepared this project report and that all sources of information have been duly acknowledged.”

Signatures:

Alin George Cortel:



Alexandru Ungureanu:



Sigmundur Johansen:

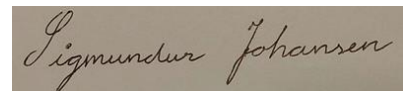


Table of Contents

Table of Figures	4
Table of Tables	4
Abstract	4
1. Acknowledgements	4
2. Executive Summary	5
3. Introduction	5
3.1 Background description	5
3.2 Purpose	6
3.3 Problem Formulation	6
3.4 Delimitation	7
4. Analysis	7
4.1 Initial project assessment	7
4.2 Survey Data gathering	8
4.2.1 Survey data gathering conclusion	8
4.3 Requirements	9
4.3.1 Functional Requirements	9
4.3.2 Non-Functional Requirements	10
4.3.3 User-Interface Requirements	11
4.4 Use Cases Diagram	13
4.5 Use Case Descriptions	14
4.5.1 Login use case	14
4.5.2 Start Survey use case	14
4.5.3 View Field use case	15
4.5.4 Add Field use case	15
4.6 Actor Use Case Scenario	16
4.6.1 “Start Survey” Persona Scenario	16
4.7 Activity Diagrams	17

BPR 2 – Drone Survey System

4.7.1	Login activity diagram.....	18
4.7.2	StartSurvey activity diagram.....	19
4.7.3	View Field activity diagram.....	20
4.7.4	AddField activity diagram	21
4.8	Docking Station Modules	22
4.9	Docking Station Activity Diagram.....	23
4.10	Domain Model Diagram	24
5	Design	26
5.1	System Technologies used	26
5.1.1	ASP.NET framework.....	26
5.1.2	Node.js	26
5.1.3	Bootstrap grid.....	26
5.1.4	Python	27
5.1.5	Ajax	27
5.2	User Authorization Scheme	27
5.3	Class Diagrams.....	28
5.4	Sequence diagram	30
5.5	Deployment Design.....	31
5.6	Database Architecture	32
5.6.1	Database E-R Diagram	32
5.7	Website mockup design	33
5.7.1	User Interface Design	36
5.8	Docking Station Algorithm	37
6	Implementation	45
6.6.1	Login Feature implementation	45
6.1.2	Add Field Implementation	48
6.1.3	View Field Implementation	50
6.1.4	Database E-R diagram implementation	51
6.1.5	Algorithm implementation.....	52

BPR 2 – Drone Survey System

6.1.6	Web-Application-docking station communication	58
6.1.7	Drone-Docking Station Communication	59
6.1.8	Deployment implementation.....	60
7.	Testing.....	61
7.1	In Scope.....	61
7.1.1	Website testing:.....	61
7.1.2	Docking station testing:	62
7.2	Out of scope	62
7.3	Types of testing performed on website	62
7.3	Types of testing performed on Docking Station	63
7.4	Issues and Recommendations.....	63
7.5	Exit Criteria	63
7.6	Conclusion.....	64
8.	Results and discussion	64
9.	Conclusion	65
10.	Project Future Considerations.....	65
10.1	What Is missing	65
10.2	What can be improved.....	65
11.	References.....	66
12.	Glossary	67
13.	List of Appendixes.....	68
13.1	Appendix A Project Description.....	68
13.2	Appendix B User Guide	68
13.3	Appendix C Source code	68
13.4	Appendix D Diagrams and use case descriptions.....	68
13.5	Appendix E Survey Data Sheets.....	68

Table of Figures

Figure 1 Use Cases Diagram.....	13
Figure 2 Login Activity Diagram	18
Figure 3 Start Survey Activity Diagram	19
Figure 4 View Field Activity Diagram	20
Figure 5 Add Field Activity Diagram.....	21

Table of Tables

Table 1 Functional Requirements	10
Table 2 Non-Functional Requirements.....	11
Table 3 Login Use Case	14
Table 4 Start Survey Use Case.....	14

Abstract

Drone Survey System is a software that is created with the purpose of optimizing efficiency among farmers in Denmark. The main objective of this project is to provide the farmers with tools that can enable an objective forecast regarding the healthiness of crops, as well as to identify any actions that should be performed to optimize the amount or size of the harvest. This is done by using drones for surveillance of the fields and having a user interface to initiate this process and check upon its results. Aside this aspect, the main technical choices revolve around having a System of Systems Architecture, which has at its core two components: the drone with its control modules and a 3-tier system which provides an interface for the user. The result is a scalable and adaptable solution that meets all the requirements that were posed and reaches all the objectives of the project.

1. Acknowledgements

Note: For more details about SEGES’s involvement, please visit **Appendix 1 - Minutes of Meetings**, of the “Process Report” folder.

BPR 2 – Drone Survey System

Via University College provides great support in the making of this report. The help is provided through supervisor, Allan Henriksen.

SEGES is involved in the initial phase of requirements gathering process. SEGES itself is not part of the project, nor actively involved. Their involvement consists of one meeting in which the developers express their expectations and assess SEGES's expectations.

2. Executive Summary

Drone Survey System is a software that is created with the purpose of optimizing efficiency among farmers in Denmark. The main objective of this project is to support farmers with tools that can enable an objective forecast regarding the healthiness of crops. This is done by using drones to survey the fields and having a user interface to initiate this process and check upon its results.

Aside this aspect, the main technical choices revolve around having a “System of Systems Architecture”, which has at its core two components: the drone with its control modules and a three-tier system which provides an interface for the user.

The result of the analysis phase is having a clear view on the “What” of the project through proper assessment of necessary requirements. It is followed by the design phase which provide a deeper understanding of connection between concept and actualizing the project.

With the testing phase, all requirements underwent a verification process that deemed them satisfactory based on the exit criteria.

The result is a scalable and adaptable solution that meets all the requirements that were posed and reaches all the objectives of the project.

3. Introduction

3.1 Background description

Danish Agriculture and Food Council state that Denmark has the highest market share of organic products in the world (Danish Agriculture and Food Council, 2017). This fact provides a good insight that the farming industry is partly responsible for a stable economy (CIA, 2017).

Denmark's farming industry is built on a two-layer advisory system which is considered unique worldwide through its support for efficient dissemination of knowledge (SEGES, 2017). The purpose of this advisory system is to build bridges between research and practical farming,

BPR 2 – Drone Survey System

constant developing products and services in relation to its users and ensure that Danish farmers have access to latest technologies (Hørfarter, 2017).

The two-layer advisory system is comprised by the DANISH AGRICULTURAL ADVISORY SERVICE(DLBR) and SEGES. The DLBR represents a nationwide cooperation between thirty-two farmers owned advisory companies which have more than two thousand consultants (SEGES, 2017). SEGES provides IT solutions for farmers and advisers in a variety of domains. (SEGES, 2017)

One of their applications is called CropSAT. CropSAT is a free web application owned by SEGES and the Ministry of Environment and Food (Hørfarter, 2017). This application uses the satellite Sentinel-2A to take pictures once every seven days for all six hundred thousand fields in Denmark (Hørfarter, 2017). Besides the time delay, this application can fail to bring results due to presence of clouds in the sky.

This project's main stakeholders represent the farmers in Denmark, the software developers and afterwards 3rd party companies, such as SEGES, for possible future cooperation.

3.2 Purpose

The purpose of this project is to provide a constant means of assessing farmer's fields. For this, the overview revolves around on building a drone that manages to bypass the obstacles that CropSAT faces. This should furthermore affect the overall efficiency and effectivity of farmer's decisions.

3.3 Problem Formulation

The questions below represent the overall focus of questionable problems to be encountered:

What is required to be initially analyzed?

- 1 Which are the main stakeholders involved within the project?
- 2 Which are the requirements related to the stakeholders?
- 3 Which are the requirements that the drone must meet for optimum working conditions?
- 4 Which are the highest priority requirements to be implemented?
- 5 How should the overall model look?

What does the design process involve?

1. What system technologies should be used?

BPR 2 – Drone Survey System

2. How to transmute static requirements into active design models?
3. How will the website's initial design look?
4. What information does the website store?

Implementation

1. What technologies will be used for implementing?
2. What future considerations can be applied for a further improved product?

Testing

1. Which are the different testing methods that can be used to discover bugs within the software/hardware?
2. Which are the known errors that can be rectified for a further improvement of the drone?

Deployment

1. How to make it possible to deploy the website?

3.4 Delimitation

1. All the legal aspects regarding European regulations will not be covered within this project.
2. The drone will not be required to be able to perform outdoors on the fields due to high expenses and due to the possibility of individuals getting injured due to a faulty software or hardware design, but it will provide an initial insight on basic movements.
3. Through the website, the farmers will have access only to simple commands such as starting the survey and reviewing it.
4. The project will not focus on encryption and decryption of specific areas such as WIFI, radio transmission or website's database.

4. Analysis

4.1 Initial project assessment

Note: Please visit **Appendix D - Diagrams and use case descriptions** for more details about the diagrams that were implemented and more details about the use cases descriptions.

BPR 2 – Drone Survey System

In this case, it is possible to observe that the analysis is applied on a complex of systems which is split in two main parts. Thus, it is concluded that the analysis phase diverges in methodology for each part.

Furthermore, the analysis of the web-application part yields multiple Use Cases. These use cases are further described by specifying the expected flow of each activity, whereas the Docking Station part is divided into conceptual modules.

The encompassing view of the system is presented as a Domain Model Diagram, which summarizes the “Analysis” section. However, it is important to note that all the results of this phase are strictly derived from considerations, such as the scope and the requirements of the system.

System requirements are based on the survey given to farmers, SEGES meeting and overall analysis and interpretation of required concepts.

4.2 Survey Data gathering

Note: Please visit **Appendix E – Survey Data Sheets** for more details about the survey data gathering.

The purpose of creating a survey is to evaluate possible requirements which are overlooked by developers. The survey targets farmers and their relationship with farming technology. The survey is comprised of ten questions:

Survey Disclaimer: Since there are only five actual replies, it is considered that this survey is not viable enough to base all the project’s requirements only on it. The project’s requirements, therefore are based on this survey, the SEGES meeting and the personal analysis and interpretation of developers.

4.2.1 Survey data gathering conclusion

Even though the survey is inaccurate due to lack of replies, it is taken in consideration for a better overview of overlooked requirements.

Farmers lack knowledge regarding drone usage in the farming industry. Despite this fact, they seem eager to practice with modern technology that helps them with work optimization. The optimal way to run a survey is to have an automatic system which can be accessed through all possible internet - access devices.

BPR 2 – Drone Survey System

The information displayed, and the quantity of information displayed is rather irrelevant for the farmers. Their main requirement is to survey the area automatically and expect a simple result that guides them to a better solution for improvement.

4.3 Requirements

4.3.1 Functional Requirements

The functional requirements are listed in the table below. They are split in requirements for the website application and for the drone and docking station. The functional requirements of the website are divided mainly in “user” and “administrator” use cases that focus on covering main functionalities. These functionalities are presumed to fulfill the necessary actions to end a cycle from logging in, modifying relevant data, starting and reviewing a survey and afterwards logging out.

Drone requirements focus on achieving the possibility to survey the field and forward images to the docking station. The purpose of the docking station (can be a laptop, smartphone) is to process the image analysis from the drone.

Each functionality has a level of priority from one to four (high- low). These levels are equivalent to Moscow prioritization of must, should, could, would not. The purpose of prioritization of tasks is to provide focus on most relevant tasks and create a structured workflow. As an example, the task of surveying the field has a precondition of being able to log in, therefore it has a lower priority in relation to the “login” requirement.

BPR 2 – Drone Survey System

Table 1 Functional Requirements

Nr.	Functional Requirements (Moscow prioritization: Must, should, could, would not)	Priority (1 high, 4 low)
1.	User must be able to log in based on given account.	1
2.	User must be able to start the survey.	1
3.	User must be able to view the analysis of the survey.	1
4.	User must be able to log out.	1
5.	User must be able to edit profile.	1
6.	The drone should be able survey the field.	2
7.	The docking station should perform the image analysis.	2
8.	The drone would not be able to record the survey.	4
9.	The drone must be able to forward the images to the docking station.	1
10.	The “docking station” (laptop) must be able to store the result of the survey in a database.	1
11.	Administrator must be able to login.	1
12.	Administrator must be able to logout.	1
13.	Administrator must be able to view Fields.	1
14.	Administrator must be able to view Drones.	1
15.	Administrator must be able to register new user.	1
16.	Administrator must be able to register drone.	2
17.	Administrator could be able to delete docking station.	1
18.	Administrator could be able to delete drone information.	1
19.	Administrator would not be able to edit users.	4
20.	Administrator must be able to associate a drone id with a user id.	1
21.	The website would not be able to notify the user when the task is done.	4

4.3.2 Non-Functional Requirements

Note: Most of non-functional requirements are limited to the drone’s capabilities to perform survey within certain quality and quantity parameters, such as speed of surveying, life of battery and so on. There are many laws that are not taken in consideration due to the project’s different focus.

BPR 2 – Drone Survey System

Table 2 Non-Functional Requirements

Nr	Non-Functional Requirements	Priority (1 high, 4 low)
1.	The estimated time for providing a complete analysis of a field by the docking station, must be less than 10 minutes per square kilometer.	1
2.	The battery of the drone must be functional for a reliable performance.	1
3.	The activity of the system must be logged for each specific user.	1
4.	The system won't be available for usage from any internet browser.	4

4.3.3 User-Interface Requirements

Web design is an important aspect of the website's usability. Because the website application has its own purpose of facilitating the connection between farmers and the drone, certain usability guidelines are used (Shneiderman, 2003).

Usability guidelines that are taken in consideration:

Design process and evaluation:

1. Establish user requirements involving the users in the process
2. Provide useful content for the users
3. Focus on performance as a priority before the preference
4. Use Personas to keep the design team focused on the same types of users.

Optimizing the user experience:

1. Not displaying unsolicited windows or graphics
2. Reduce the user's workload

Hardware and Software:

1. Design for Common browsers

The Homepage:

1. Enable Access to the Homepage
2. Show all major options on the homepage
3. Create a positive first impression of the website

Page Layout:

1. Avoid cluttered displays
2. Place important items consistently
3. Place important items at Top Center

Navigation:

1. Provide navigational options
2. Place primary navigation menus in the left panel
3. Keep navigation-only pages short

Scrolling and Paging:

1. Eliminate horizontal scrolling
2. Use Paging rather than scrolling

Headings, Titles and Labels

1. Provide descriptive page titles
2. Highlight Critical data

Links

1. Use meaningful link label
2. Match link name with their destination pages

Text Appearance

1. Use black text on plain, high-contrast backgrounds
2. Use at least 12-point font
3. Use Familiar fonts

Graphics, Images and Multimedia

1. Use simple background images
2. Graphics should not look like banner Ads

Writing Web Content

1. Avoid Jargon and use familiar words

Content Organization

1. Organize information clearly

BPR 2 – Drone Survey System

2. Ensure that necessary information is displayed
3. Minimize the number of clicks or pages (3-12)

Usability Testing

1. Apply Automatic evaluation methods
2. Choosing laboratory vs remote testing

4.4 Use Cases Diagram

The use cases diagram encompasses all functional requirements that are applied to the web-application only. These use cases are the visual representations of functional requirements listed above. There are two main actors, the admin and the user. They both have in common the “Login”, “Logout” and “View Fields” use cases.

User functionalities are strictly related to viewing, starting the survey and editing user profile. Admin can register new users, drone, new fields, can view drones and can delete docking station, drone information and field.

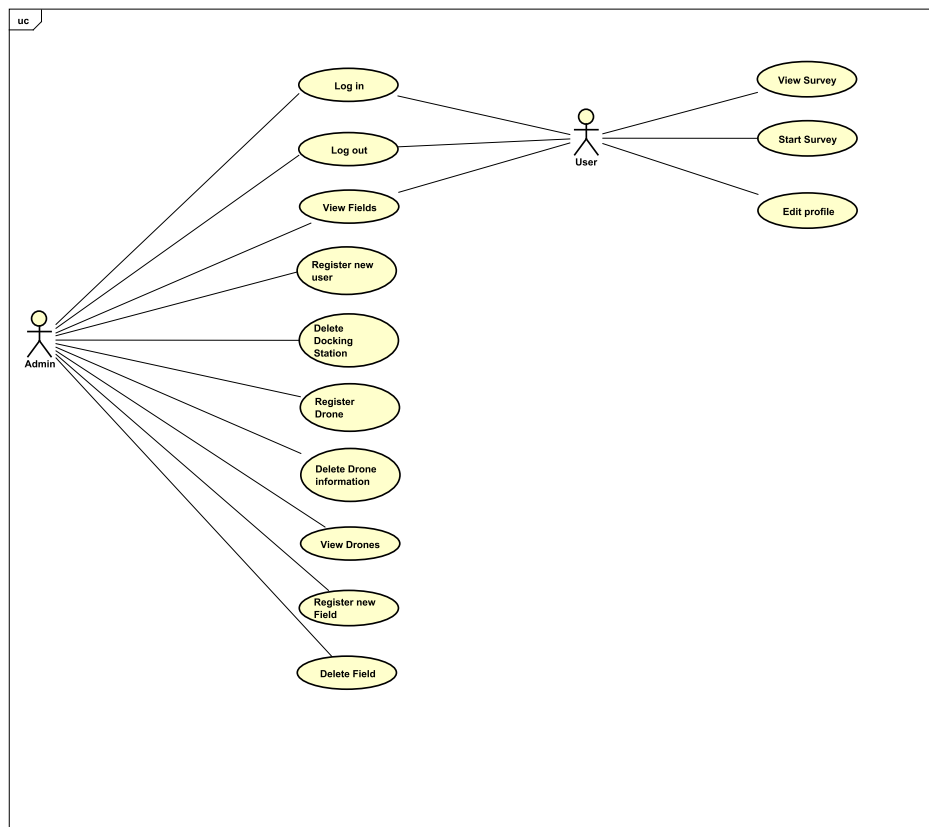


Figure 1 Use Cases Diagram

BPR 2 – Drone Survey System

4.5 Use Case Descriptions

Note: Some of the following use cases were selected in order to provide an overview.

4.5.1 Login use case

Table 3 Login Use Case

Use Case	Log in as basic User
Summary	Users can log in into their account with their credentials.
Actor	Anonymous
Pre-condition	Admin must have sent by email the user's account credentials. User must be logged off.
Post-condition	
Base Sequence	<ol style="list-style-type: none"> 1. Anonymous visits website. 2. Clicks on Login. 3. Into the email field the user writes his email. 4. Into password field the user writes his password. 5. User clicks submit. 6. User is redirected to the homepage.
Branch Sequence	<p>The database does not find the provided email, or password does not match, and the user is informed of his failed login attempt.</p> <p>User must input credentials again.</p>

4.5.2 Start Survey use case

Table 4 Start Survey Use Case

Use Case	Start drone Survey
Summary	Logged in users can start a drone survey from their "My Areas" page.
Actor	User
Pre-condition	Must be logged in
Post-condition	
Base Sequence	<ol style="list-style-type: none"> 1. User clicks on "My Areas" after logging in. 2. User selects desired areas for survey by ticking the empty box. 3. User clicks "Start Survey" 4. User waits, and survey is done.
Branch Sequence	<ol style="list-style-type: none"> 4. An error occurs, and the user is prompted with the error. 5. User must click on "start survey" again.

BPR 2 – Drone Survey System

Sub Use Case	Login Use Case
--------------	----------------

4.5.3 View Field use case

Table 5 View Field Use Case

Use Case	View Field
Summary	Basic users and Administrators can view fields. A user can see his current existing fields and an administrator can see all the fields in the system.
Actor	User, Administrator
Pre-condition	User must be logged in with his credentials as user or administrator.
Post-condition	
Base Sequence for basic user	<ol style="list-style-type: none"> 1. User clicks on “My Areas” after logging in. 2. User clicks on “View Field”. 3. User is redirected to Google Maps with his field area.
Branch Sequence for administrator	<ol style="list-style-type: none"> 1. Administrator clicks on “View Fields” and it receives all fields in the system.
Sub Use Case	Login Use Case

4.5.4 Add Field use case

Table 6 Add Field Use Case

Use Case	Add Field Use Case
Summary	Admins can add a field to a farm.
Actor	Admin
Pre-condition	User must be logged in. User must have administrator rights.
Post-condition	
Base Sequence	<ol style="list-style-type: none"> 1. Administrator clicks on “Add Field”. 2. In the email field, the administrator adds the email of farmer associated with the drone. 3. In the docking station id field, the admin adds the id of the specific drone associated with the farmer. 4. In the google maps, to the right, the administrator clicks on draw tool.

BPR 2 – Drone Survey System

	5. The administrator draws with the tool, line by line the perimeter.
--	---

4.6 Actor Use Case Scenario

4.6.1 “Start Survey” Persona Scenario

Note: The presented character is fictional.

Based on the usability guidelines mentioned earlier in “2.3.3. User-Interface Requirements”, this project includes a “persona” scenario that is used to focus the attention of developers only on one type of user, which is the farmer.

Use Case: Start Survey:

Initial Step-By-Step Description

The user accesses the website after which he enters his credentials. Here he will see the status of the drone, and if it is ready to launch.

If it is ready to launch, then he will only need to click, launch. After the drone is done, the user will be able to review the results.

Actor characteristics

Table 7 Persona Scenario

Name	Henrik Nielsen
Age	48
Location	Horsens Outskirts
Work	Farmer (owns 3 ha of farmable land)
Technological literacy	He is aware of some of the technological developments in the agricultural industry, yet he has not interacted directly with them. He is receiving some feedback regards crops from the Danish agricultural advisory service, yet it is only once per week on an optimum scenario.
Bio	Henrik is a farmer since age 25. He inherited the land and the desire to work the fields from his father.

BPR 2 – Drone Survey System

	He mostly focuses on corn and potatoes crops. Currently, he is married and has two children, them being another motivation for him to succeed even better in the agricultural industry.
Goals	To succeed in the agricultural industry. To provide for his family.
Frustrations	Sometimes, the crops in some areas can be damaged due to a disease and thus affects his goal related to agricultural success. Besides certain diseases, he feels that he lacks information related to how dry his crops are. He feels that the feedback from the Danish agricultural advisory service is not enough.
User Interaction with the software	<p>After finding out about the drone surveillance system, Henrik is eager to test it out. He has heard that through this service, he can receive real time data daily regards his crops, unlike satellite surveillance.</p> <p>He receives the drone and all he must do is to put it on the docking station.</p> <p>The docking station is a small encompassed area in the fields, that supposedly has a device(laptop), that can communicate with the drone. Henrik, all he should do now is to wait for the drone to survey the crops automatically for him.</p> <p>The information from drone is processed, sent through the laptop and uploaded on the website where Henrik can see if his crops are dry, diseased or not optimally functional.</p> <p>To navigate on the website and see the data from the drone, Henrik does not need to create an account because an admin already created one for him. Henrik logs in on the website where he can see his personal profile with all the data from the drones which survey his fields.</p>

4.7 Activity Diagrams

Note: In this report, there are included only four main activity diagrams which represent the previous use cases of the website. It is done in this manner to provide an understanding of overall steps from analysis to testing.

BPR 2 – Drone Survey System

Another reason for omitting from the main report other activity diagrams is to maintain the focus on the overall system. For more details regarding other activity diagrams, please visit **Appendix D-Diagrams and use case descriptions**, in “Project Report” folder.

To describe activities with higher accuracy, all activity diagrams include swimming lanes which stand for the client and the server.

4.7.1 Login activity diagram

The “Login” activity diagram sums the possible actions that take place between the client and the server for the user to log in. From the client side, the user clicks on “login”, where he inputs his credentials and submits.

On the server side, the database is searching for the specified user. If the specified user is not found, or if the provided password is not valid, the user is informed about failed login attempt and is redirected to input his credentials.

If the credentials match, the login is succeeded, and the user is redirected to My Area.

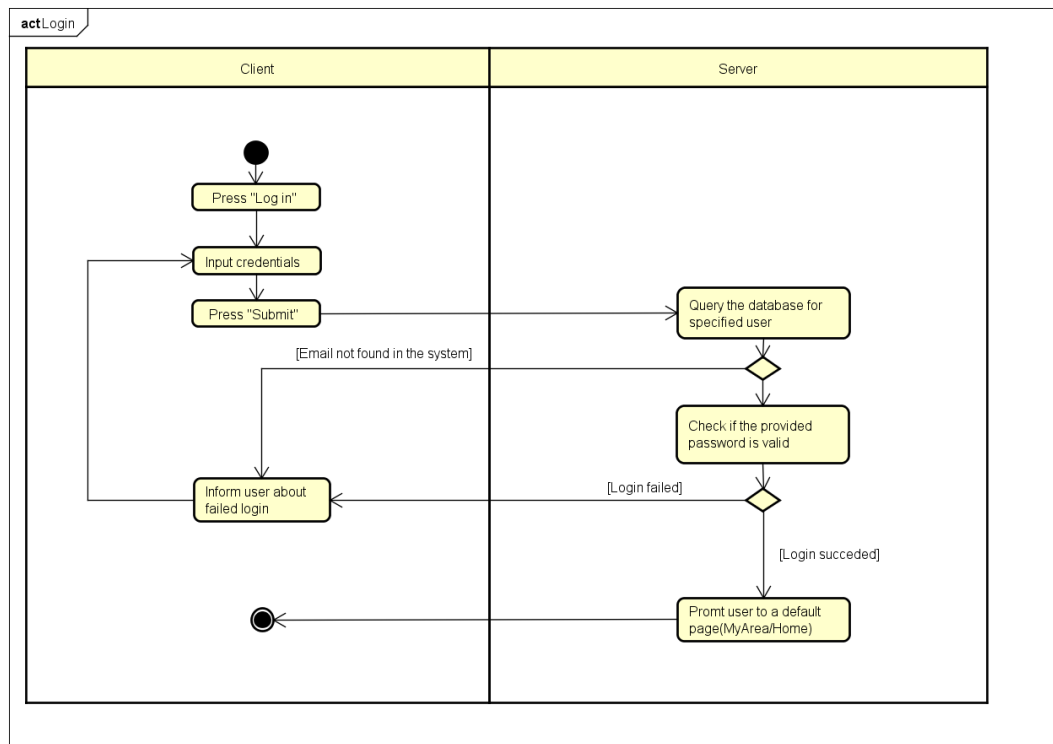


Figure 2 Login Activity Diagram

BPR 2 – Drone Survey System

4.7.2 StartSurvey activity diagram

The “Start Survey” activity has the precondition of the user to be logged in to be granted the necessary rights. After logging in, the user accesses the “My Areas” page. User clicks on “Start Survey” which triggers in the database a new query that will add the new survey for the specified field.

If the database fails to add a new survey, the user is informed about the error and is redirected to redoing the survey.

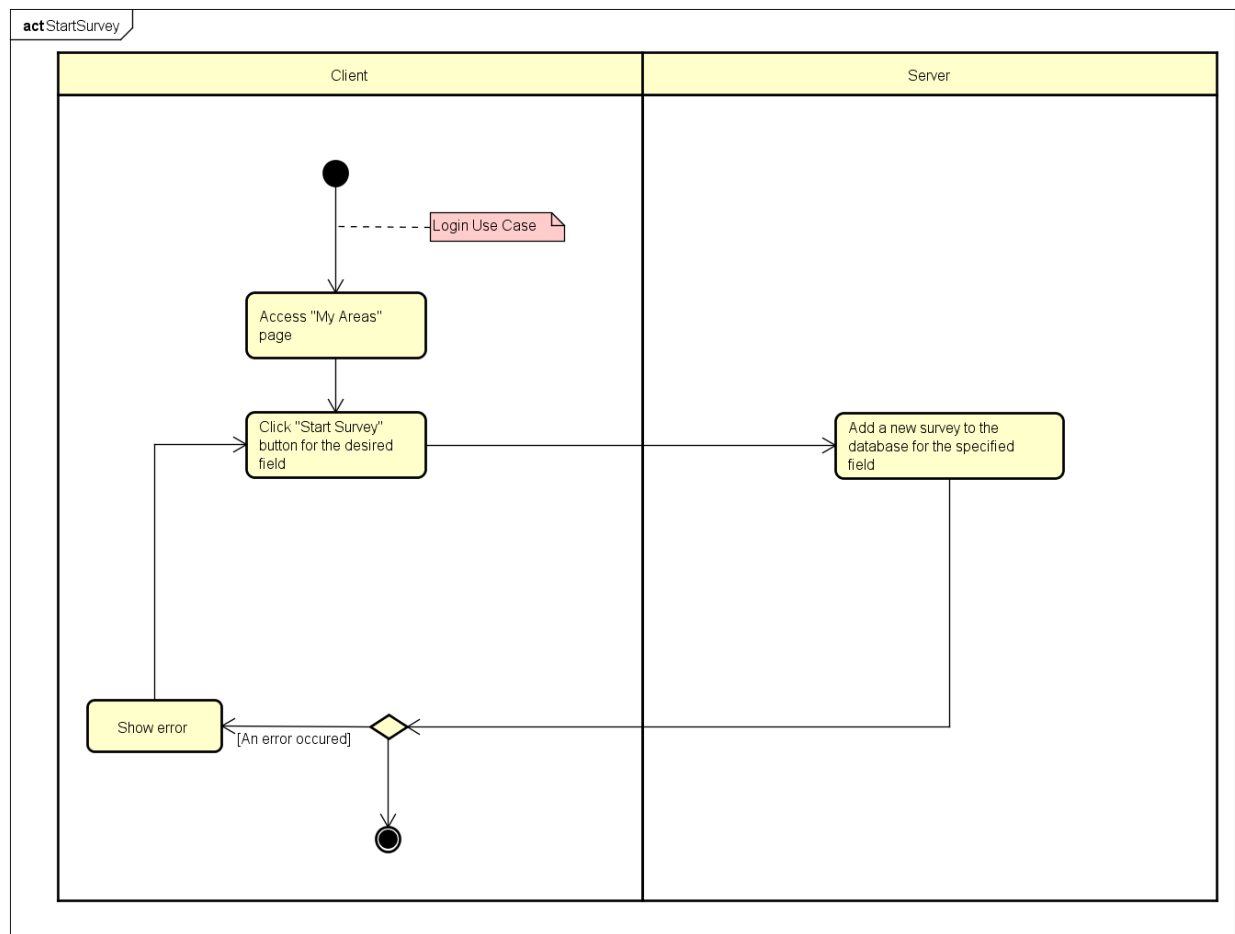


Figure 3 Start Survey Activity Diagram

BPR 2 – Drone Survey System

4.7.3 View Field activity diagram

The “View Field” activity diagram is based on the “View Field” use case which is both an administrator and a basic user functionality. The difference is that the administrator has access to all the fields in the database, unlike the user which has access only to his current fields.

On the client side, the admin clicks on the “View Fields” button, and on the server side the request is processed by the database and the administrator receives its requested information.

The basic user must click “My Areas” button to redirect him to his displayed fields. If there are no current fields, an error will be displayed. After having the overview of fields, the user can click on “View Field” and receive a page with google maps, displaying the chosen field.

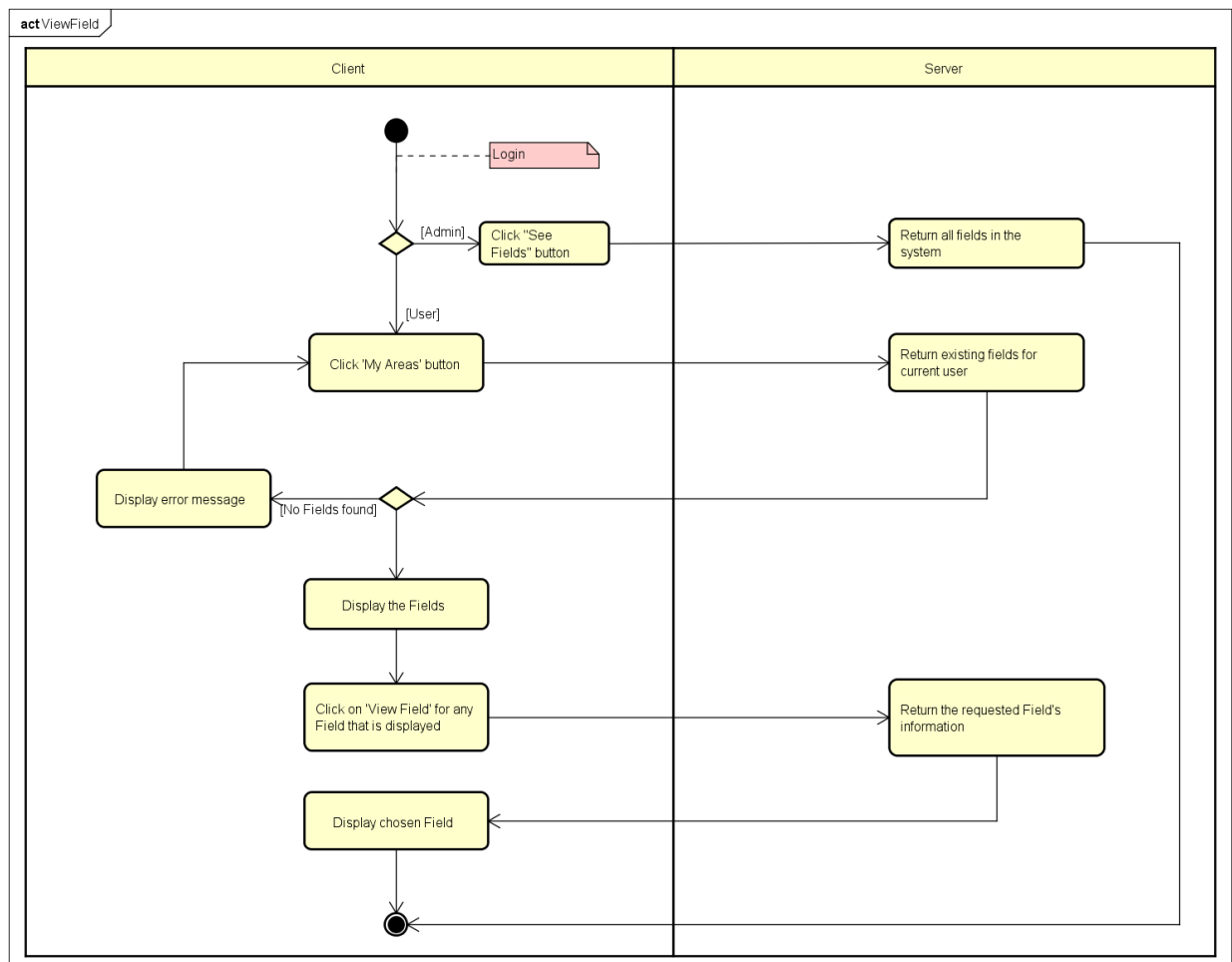


Figure 4 View Field Activity Diagram

BPR 2 – Drone Survey System

4.7.4 AddField activity diagram

The “Add Field” functionality is only available for administrators. On the client side, the administrator selects the option to add a new field. To add a new field, the administrator must add the email of the user and the docking station id. In this way, the user is associated with its specific docking station.

After the administrator clicks on the “Save” button, on the server side the database checks if User(owner) exists. If it exists, it adds a new field into the database and display a “Success” message.

If the user does not exist, it displays an error message which redirects the administrator back into filling the proper credentials.

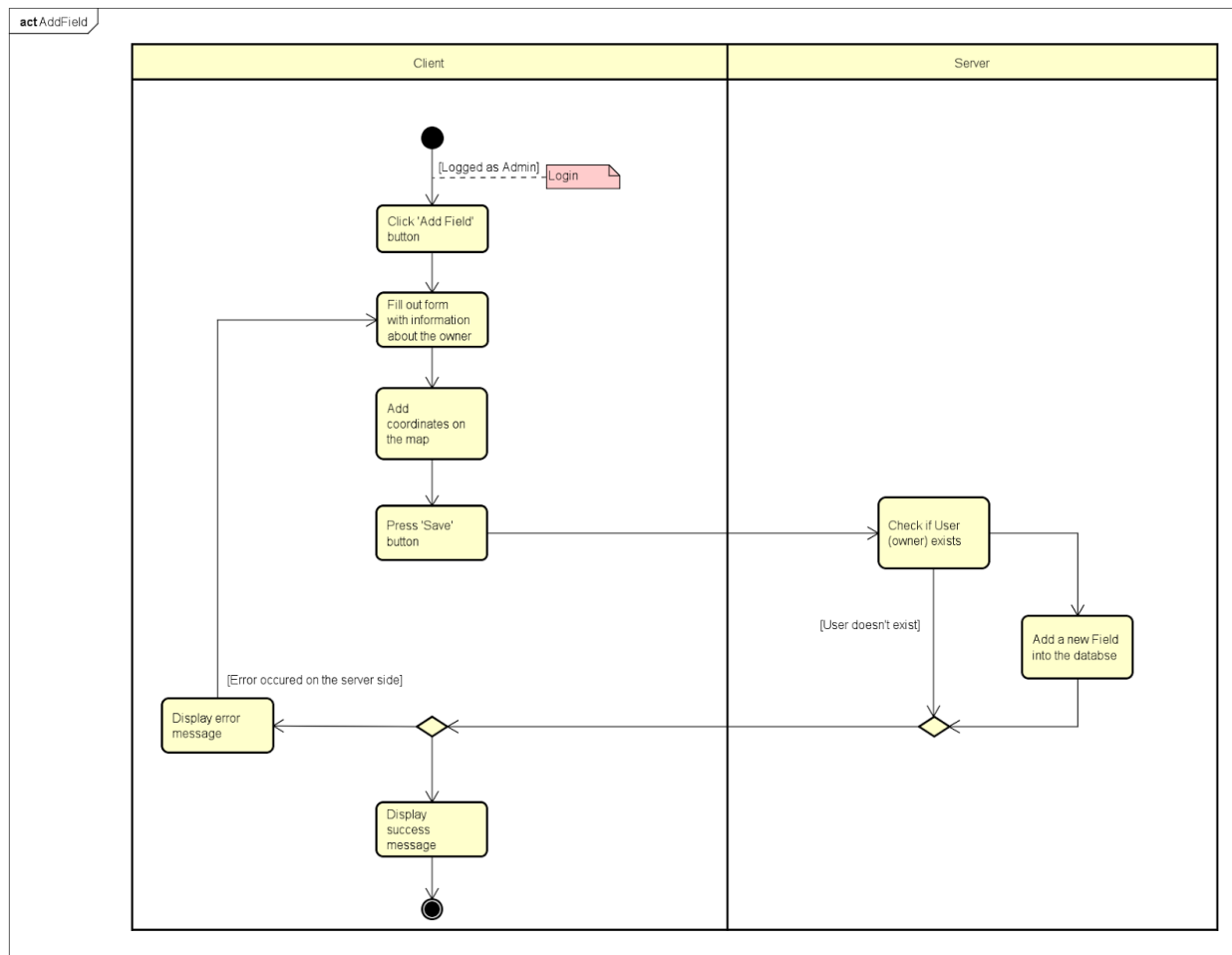


Figure 5 Add Field Activity Diagram

4.8 Docking Station Modules

The second component of the Drone Surveillance System is the Docking Station. If the web-application part has the role of centralizing the data within the system and provide an interface to the end-user, the Docking Station represents the controller of the drone and is responsible for the communication with the web-application.

It is important to note from the delimitations of this project that the docking station should be as independent from the drone as possible, which is crucial for the adaptability of the system.

Therefore, the Docking Station should have a modular structure, where each module embodies a certain data flow, similarly to what use cases represent within the web-application. In this context, it is possible to draw an analogy with the analysis of the web-application by creating data flow diagram with all the processes that are expected to run.

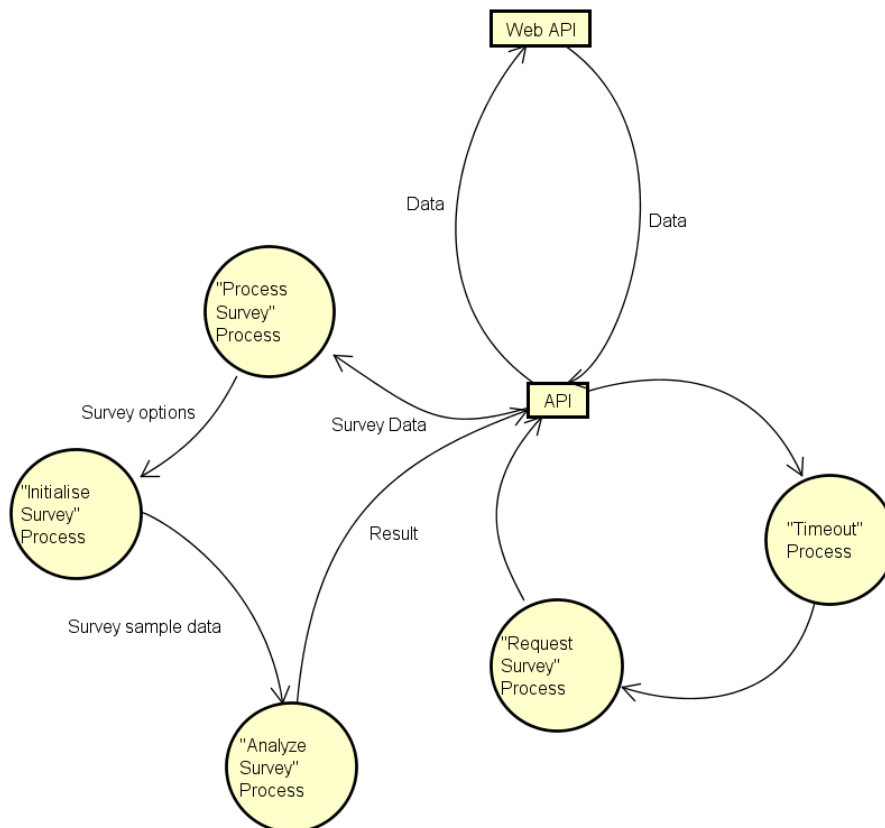


Figure 6 Docking station modules

BPR 2 – Drone Survey System

In the diagram above, it is possible to observe all the processes (modules) that are within the Docking Station, as well as the gateways between it and the web-application. The data flow represents the logical sequentiality of the modules, which are deduced from the requirements of the system.

The starting point in the flow is the “Timeout” process, which is responsible for triggering the “Request Survey” activity. It is assumed that the “Timeout” process should run periodically if there are no surveys to perform.

If a survey must be performed, the “Process Survey” process is responsible for analyzing the survey request and create any options that are needed for the drone to operate with. Next, the “Initialize Survey” module enables the drone to perform the survey and gets the raw survey data.

The last module, the “Analyze Survey” process, yields a result that is sent back to the web-application. This is like Use Case modeling, except for the fact that it is providing information regarding the data which is passed between the activities.

4.9 Docking Station Activity Diagram

In this context, activity diagrams can be created for each of the processes listed in the data flow diagram, thus providing a better overview of them. As it can be seen from the previous part, there are roughly two branches that the execution can take: waiting for surveys and performing one.

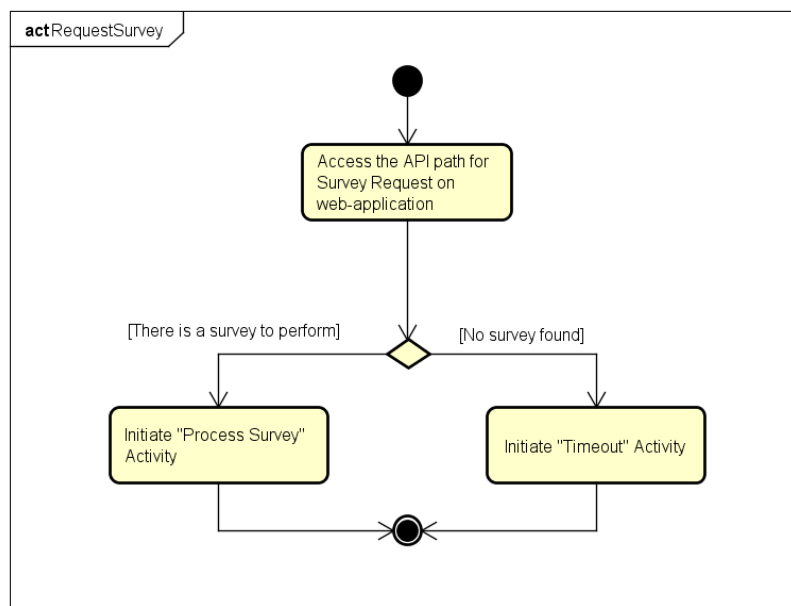


Figure 7 Docking Station Activity Diagram

BPR 2 – Drone Survey System

For the first branch, the “RequestSurvey” Activity Diagram illustrates best how the flow of the application is. If there is a response from the web-application API saying that there are no surveys to perform, the application should retry the operation later, by starting the “Timeout” process.

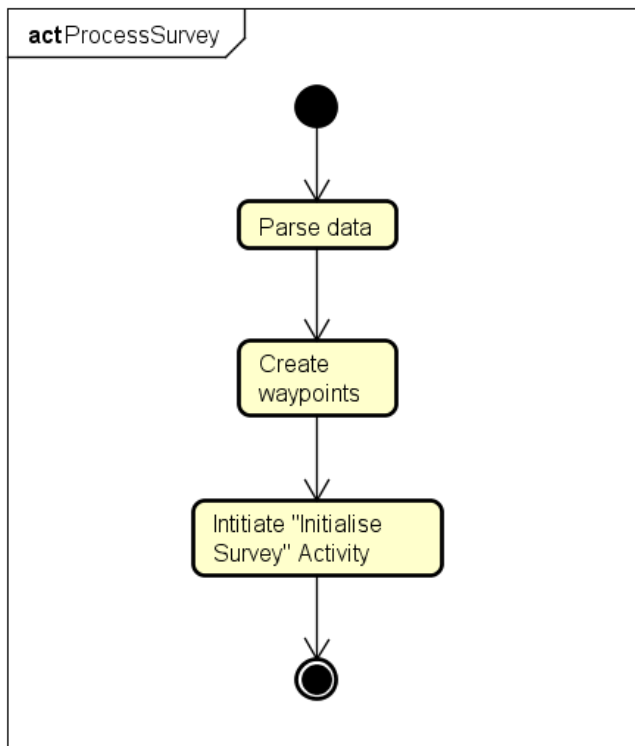


Figure 8 Process Survey

On the other hand, when a survey is requested, the processes from the second branch should execute. This starts with the “ProcessSurvey” process, where any input data is parsed from the inter-application exchange format into a manageable one. Next, the waypoints for the drone should be calculated so that the next process can handle them.

4.10 Domain Model Diagram

All in all, as a complete overview of the Analysis phase, the Domain Model Diagram is presented.

BPR 2 – Drone Survey System

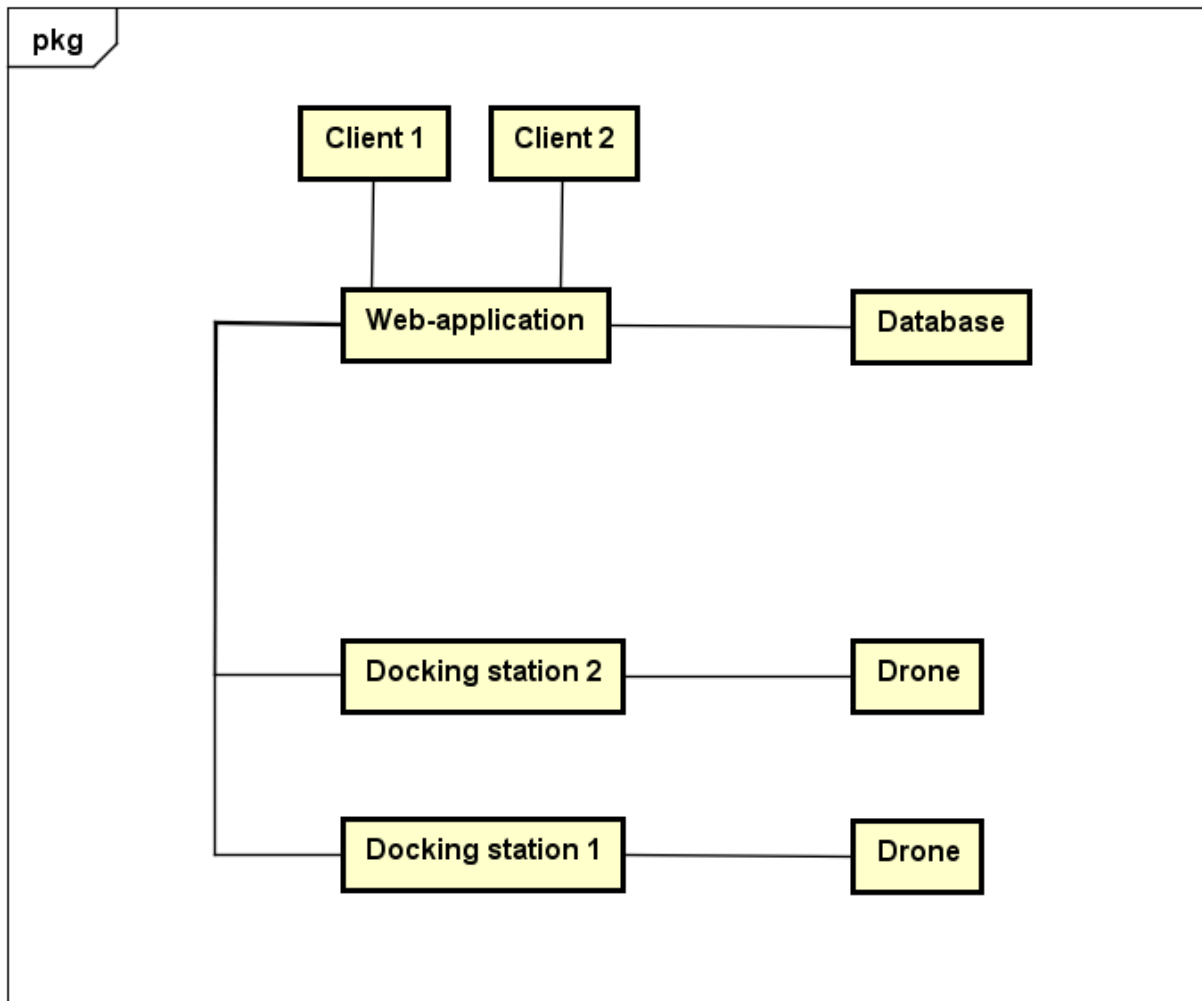


Figure 9 Domain Model Diagram

As it can be seen, the 2 main parts of the system are the Web-application and the Docking Station. All the data within the system is stored in a database, which can be accessed through the web-application. There must be more Docking Stations in the system and thus, this is indicated in the diagram by the presence of 2 entities of the same type which are numerated. Each docking station should control only one drone.

Finally, by having a clear view on the “What” of the project, it is possible to proceed to the “How”, which is the Design phase.

5 Design

5.1 System Technologies used

The choice of tools and languages is an important aspect when designing an application. Because there are always tools that have priority depending on the situation, it is crucial to make the best possible choices to maximize the quality of the built system.

5.1.1 ASP.NET framework

NET comes with multiple technologies and authoring tools that help developers put their application together with various programming tools. This framework contains a collection of functionalities into groups of classes. One big advantage for using it is represented by the extensive documentation and the possibilities in terms of design choices.

5.1.1.1 ASP.NET CORE MVC with VS Code

Visual Studio code is chosen as the source code editor for developing in ASP.NET MVC framework. This is done due to the advantage of having a debugger tool and it includes Git version control support.

ASP.NET MVC represents one of the three frameworks for web applications development. Compared to the other frameworks, the MVC framework makes it easier for developers to manage complex applications (Microsoft, 2017).

MVC focuses on having the code for the business logic layer separated from the presentation layer code. It provides further flexibility in enabling control over what it is required of the application to do and its behavior in the web environment. This provides the advantage of working simultaneously on the same application without any interference from other teams.

5.1.2 Node.js

Being a JavaScript backend framework, Node.js is gaining big popularity, its major assets being the high performance and big community support. Working with it constricts to developing software in a modular way, where all features are sorted into basic conceptual units.

The main reason for using it revolves around the experience the group already has with the tool, as well as the fact that a large variety of drones can be programmed in this way, including the prototype drone used for testing in the context of this project.

5.1.3 Bootstrap grid

The bootstrap framework is an open source toolkit that is used to develop on the front-end side of the website.

BPR 2 – Drone Survey System

Additionally, bootstrap provides a collection of JavaScript files and CSS styling that can be customized (W3schools, 2017). Due to the added features, any developer that uses bootstrap saves time spent on CSS customization. An example of such a feature is the “Carousel”, which is made based on premade classes from the collection. Such classes have similar namespaces as “carousel slide”, “carousel-indicators” and so on.

Overall, bootstrap represents a great tool for time-efficiency tasks. Even though it has its pluses, the disadvantage of using bootstrap is that it provides a generic view of a website due to its high usability and its inflexibility in modifying bootstrap classes for further customization.

5.1.4 Python

The docking station logic requires to be written in a language that is platform independent. The docking station can be a laptop, beagle bone black, or any other device capable of image processing. Python, in regard to this aspect provides portability.

Python is platform independent and is supported on the beagle bone black. It is convenient for technical projects since it has many libraries that are needed in this project, such as “PIL” (python image library). This image library is used for manipulating images taken by the drone.

Another useful library called “NumPy”, is used for scientific computing. Also “Matplotlib” is a 2D plotting library.

5.1.5 Ajax

Being part of the JavaScript library, jQuery, AJAX offers the possibility of making http requests without the need to reload the page of a web-site. It is exactly this aspect that is useful in the context of this project because of the need of loading dynamic content.

One more advantage of using it is the fact that the developer team has experience with working with this tool, which is an important aspect.

5.2 User Authorization Scheme

As stated in the requirements, there are multiple type of users that have access to the system and they should be able perform actions such as “Login” and “Logout”. Therefore, an authorization scheme is necessary to be designed.

In this sense, it is important to identify what are the options and which of them suits the purposes of the project best. Because of the delimitation that the security is not the most important aspect to consider regarding authorization, a lightweight and easy-to-use solution that is able to provide the needed outcome is preferred.

BPR 2 – Drone Survey System

In this way, the best candidate would be a token-based authorization scheme, where on the server-side a token would be created using an algorithm and then be assigned to a client's browser. Afterwards, with each request, the user can be identified by the token.

The main advantage of this scheme is that it is robust enough to satisfy the requirements of the system and is easy to integrate with the chosen development framework. To design the authorization scheme, it is crucial to describe the expected steps of the authorization process:

The server should be able to create tokens that are unique for each user. These tokens should be derived from users' credentials.

- The server should be able to issue the created tokens to the users.
- The server should be able to have access to each issued token, when authorization is required.
- The server should be able to verify the validity of the tokens.
- The server should be able to grant access to restricted resources if the user is authorized to do so.

From this list of steps, the third one indicates that each http request should contain the token. However, it is not possible to append it to the request with ASP.NET Core. Therefore, there would be need for other ways of accessing the token.

From this perspective, the best solution would be to design a way for the server to request the token before processing the initial user request. This could be easily done with the middleware functionality provided by the ASP.NET Core framework. As for the other steps, in the next section it is possible to see a more comprehensive view upon their design, by presenting the class diagram.

5.3 Class Diagrams

From the results of the Analysis phase, a class diagram is created to illustrate the entities that are expected to be part of the Web-Application. Because of the choice of using ASP.NET Core MVC as the development framework for this part of the system, this diagram includes .NET specific classes and elements to show how the system fits in with the framework.

BPR 2 – Drone Survey System

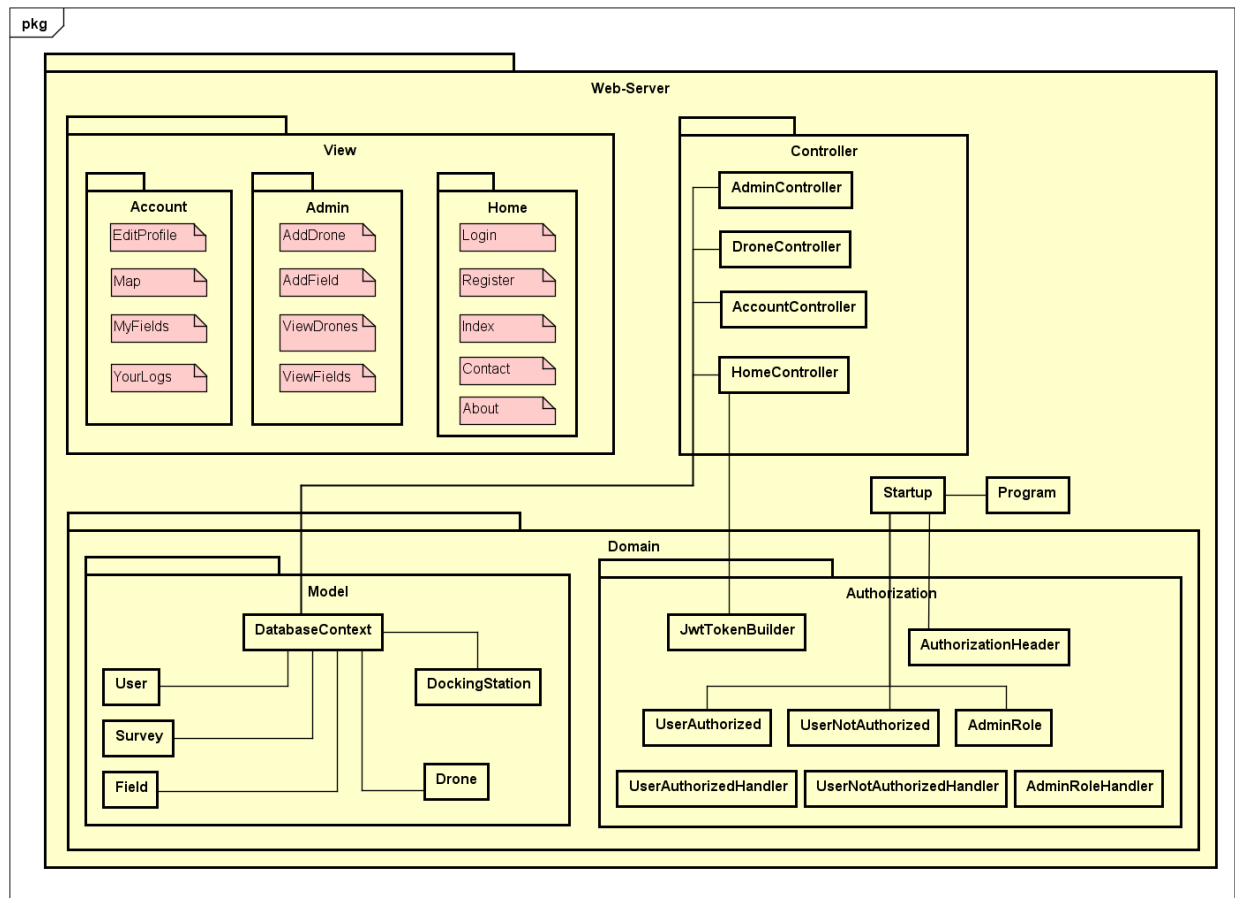


Figure 10 MVC model

The MVC framework states that a system should be divided into 3 main components: Model, View, Controller. That is why these parts are included into the above diagram. The .NET MVC framework can handle automatically the communication between the View and the Controller and provide the Model into the context of the Controller.

This is illustrated by having a connection between the Model and all of the Controllers. The View section should contain templates that are rendered upon request. It is important to note that the template names are derived from the Use Cases and that the Controllers represent activity flows which are similar conceptually: the “DroneController” is intended to handle the interaction with the “Docking Station” part of the system, the “HomeController” is designed to manage the interaction of the Web-Application with not-logged users, etc.

BPR 2 – Drone Survey System

The “Domain” namespace should contain the classes for the “Model” and the “Authorization” parts of the system. The Model refers to the entities of the system, which should be eventually mapped into a database.

The “DbContext” should inherit from the “DbContext” class which is part of the Entity Framework. On the other side, the Authorization namespace contains classes necessary for providing the authentication scheme for the Web-Application. The ASP.NET Core framework allows a robust way to integrate custom authentication to the system by providing classes that embody each type of desired user as well as handler classes that have methods for validating based on needed requirements. These should be integrated in the “Startup” class, which is specific for this framework. The “Program” class serves as the start point for the execution of the Web-Application part.

5.4 Sequence diagram

Note: For more details regards sequence diagrams, check **Appendix D-Diagrams and use case descriptions**.

With the purpose of a better illustration of how the classes from the class diagram work together, sequence diagrams were created. Here, it is possible to view the flow of the activities in a more detailed way.

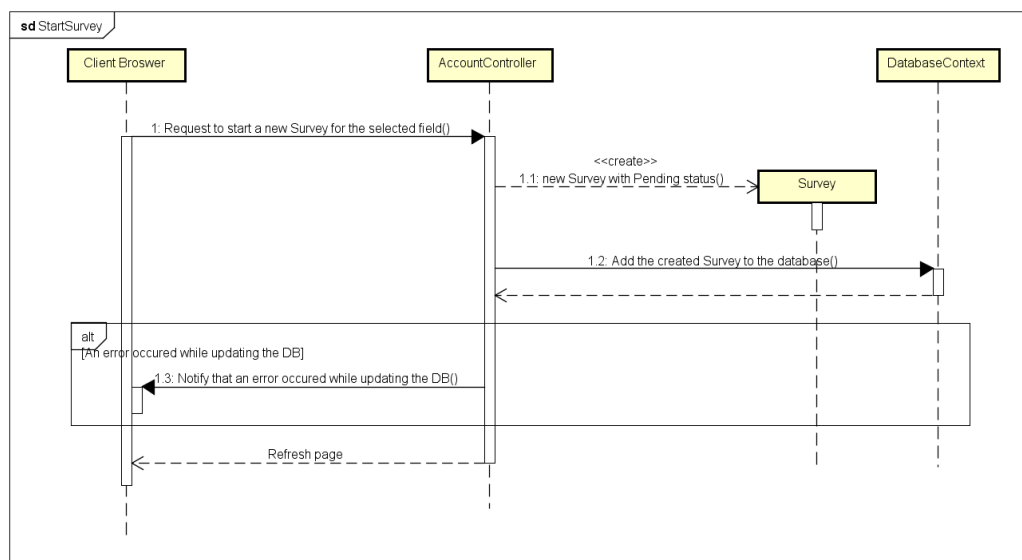


Figure 11 Start Survey Sequence Diagram

BPR 2 – Drone Survey System

As previously shown in the Analysis phase, the “StartSurvey” activity diagram was translated into a sequence diagram. As it can be seen, the main classes involved are the “AccountController” (responsible for the features of the basic user), the “Survey” (model entity) and “DatabaseContext” (which allows working with the database).

Once a request for starting a “Survey” is received, the “AccountController” creates an instance of the “Survey” class and sets all the needed parameters. Afterwards, a call to the “DatabaseContext” is performed to create a new record. If any errors occurred, the user would be notified. Once the operation is successful, the page is refreshed.

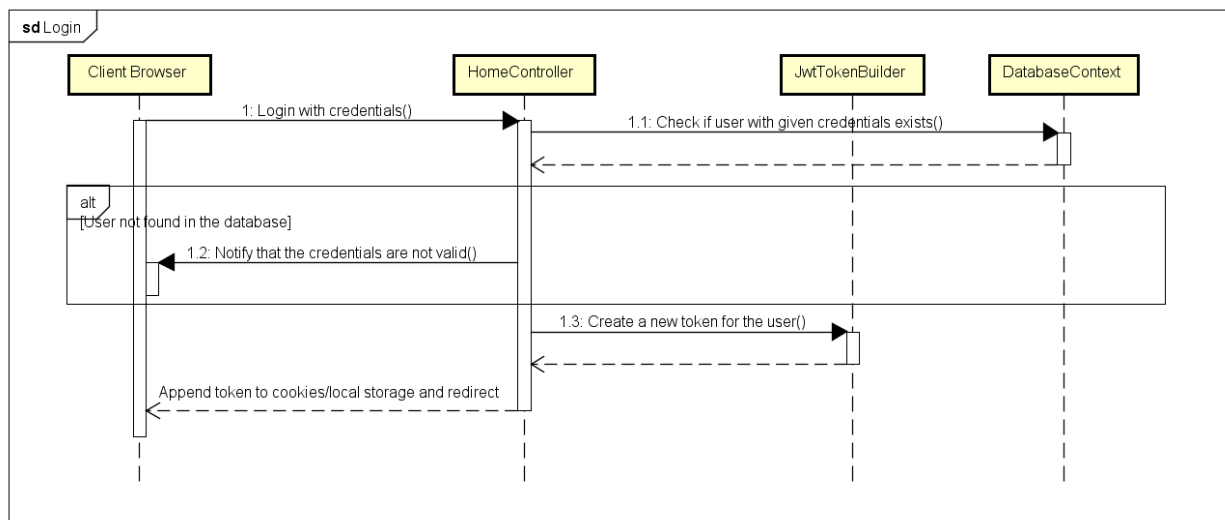


Figure 12 Login Sequence Diagram

Similarly, to the previous sequence, on the “Login” diagram it is possible to see the “Controller” and “DatabaseContext” classes. However, as an addition, the “JwtTokenBuilder” class from the “Authorization” namespace is used. Once a record of the user with the inputted credentials is found, a token is created for the user that is trying to log in. The sequence will finish with the user getting its token and being redirected to some default landing page.

5.5 Deployment Design

To have the system running in an environment as close as possible to the real world, it has been decided that the Web-Application part should be deployed on an external server and be configured that it can be accessed at any time.

This would provide a more insightful way of designing the application, which is relevant for the understanding of how to eventually use the software in the real world and how to avoid making it tied to the local development environment.

BPR 2 – Drone Survey System

For the Web-Application to be deployable, it is important to configure all the relations between the components and classes in a way that is not dependent on the environment. In this sense, the database layer should be accessible from any platform or device the Web-Application might run on.

5.6 Database Architecture

5.6.1 Database E-R Diagram

The database consists of all relations between all required entities so that all main requirements can be met.

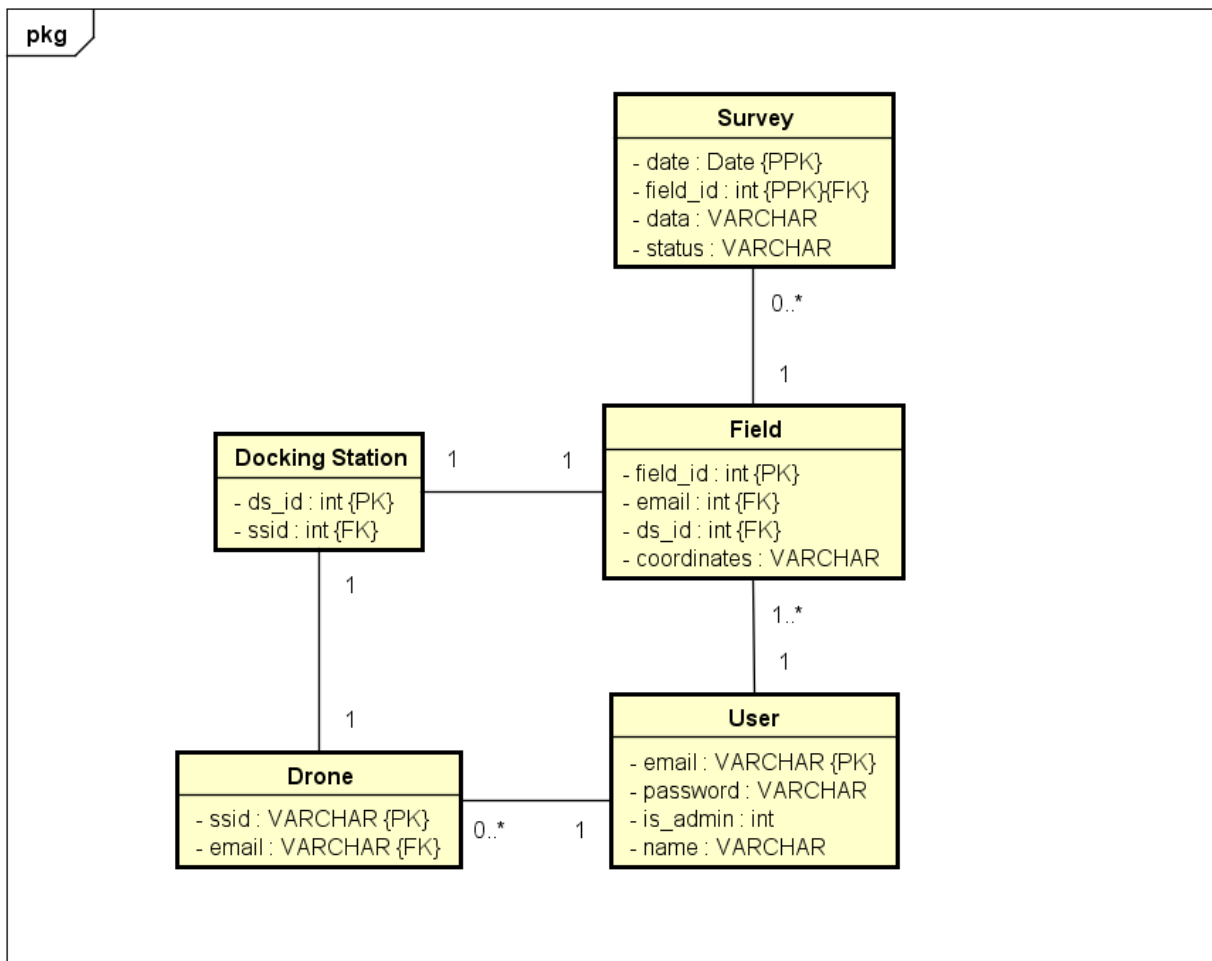


Figure 13 Database Architecture

There are five main entities: “Survey”, “Field”, “User”, “Docking Station” and “Drone”. From the previous list, only the “User” and “Docking Station” are strong independent entities. Dependent

BPR 2 – Drone Survey System

entities are enlisted as follows: “Drone” depends on “User”, “Field” depends on “User” and on “Docking Station” and “Survey” depends on “Field”.

The “Survey” entity is dependent on the “Field” entity. Their relationship states that no one to many “Survey” entities can be associated to only one specific “Field”. The relation is done based on the primary key of the “Field” which has the role of being a primary key for the “Survey” and the role of a foreign key that refers to the field_id of the “Field”.

The relation between the “Drone” entity and “Docking Station” entity is a one-to-one relationship. This is equivalent to the fact that there is only one drone for one docking station and vice versa. On the other hand, the relationship of one-to-many “Drone” entities to one “User” entity expresses the fact that a user can have more than one drone associated with the user.

The “Drone” entity has a primary key called “ssid” which is used to reference the docking station. This is done to uniquely identify each drone. In relation to the “User” entity, the “Drone” entity contains an “email” parameter which stands as a foreign key between the “Drone” and “User entity”. The “email” parameter has the purpose of being a primary key for the “User” entity and to reference the “Field” entity about uniquely identified users based on email.

The “Field” entity communicates primarily with the “User”, “Docking Station” and “Survey”. In order for the user to create a new entity called “Field”, it is required that the field is associated with the “email” parameter of the user and “ds_id” parameter of the docking station.

The relation with “Survey” entity is strictly based on providing a “field_id” foreign key so that “Survey” entity can be added to a “Field” entity.

5.7 Website mockup design

Note: For more details regards the aspect of the website, please visit **Appendix B - User Guide**, under “Project Report” folder.

The website is initially designed in the pattern that it provides a conceptual integration of both functionalities and usability guidelines. Main functionalities of the initial design focus on user use cases.

In the figure below, there is a mockup of the view of the homepage for an anonymous user. It has a navbar that contains the Title, Home, About, Contact and Login.

BPR 2 – Drone Survey System

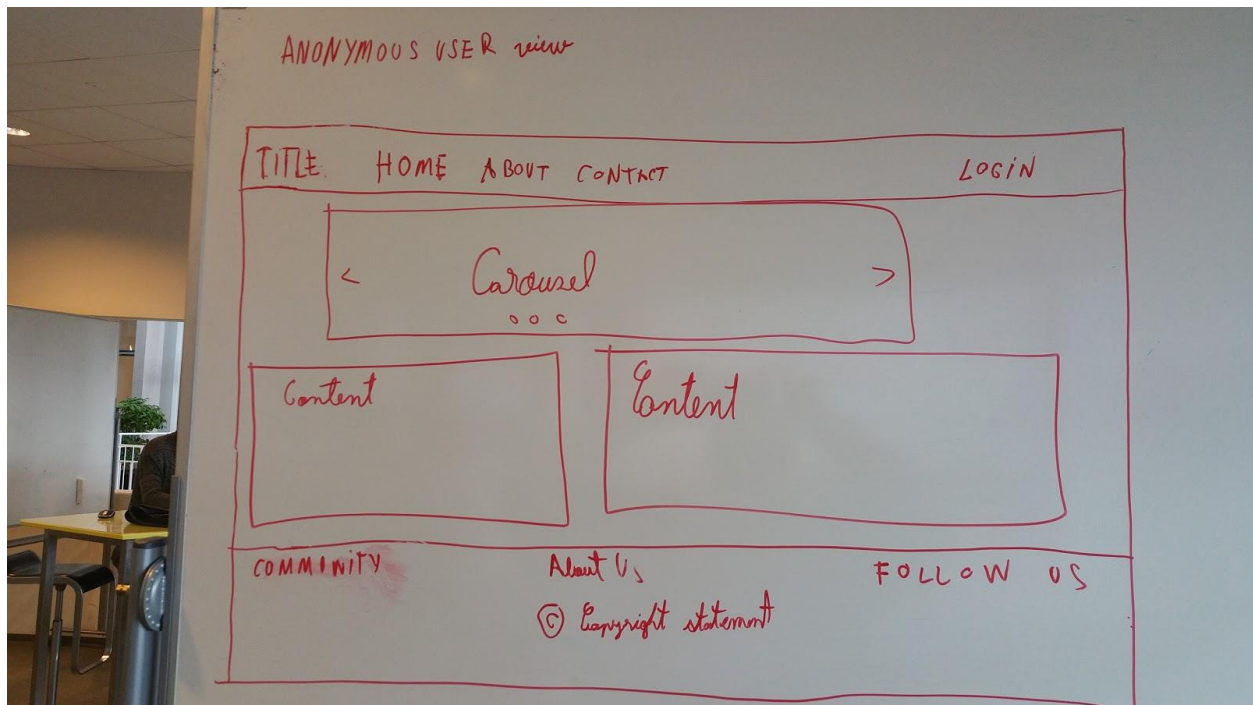


Figure 14 Anon User View

This is the preview of the homepage, being logged in as a basic user:

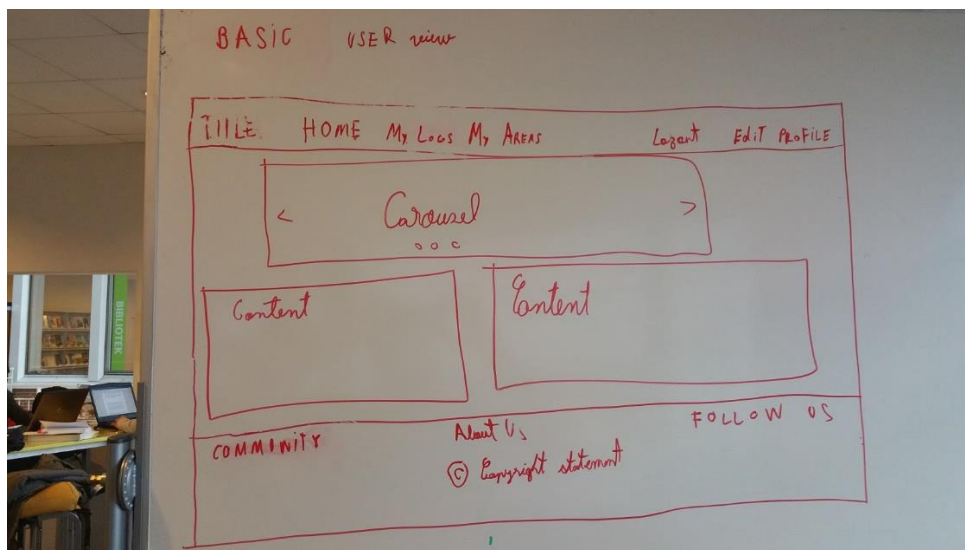


Figure 15 Basic user homepage view

BPR 2 – Drone Survey System

In the picture below, there is the “My Areas” page from where the user can start his survey

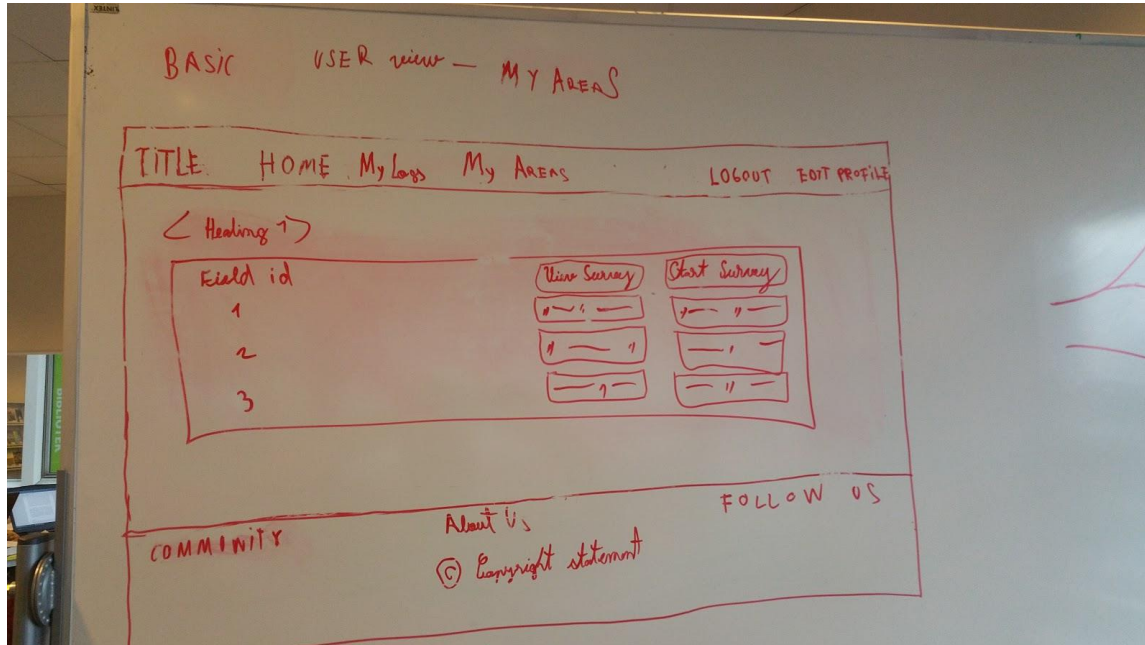


Figure 16 My Areas- Basic User view

This page is supposed to reflect the view of users of their survey Logs. It is expected that after a survey is done, the user is redirected to the “My Logs” page.

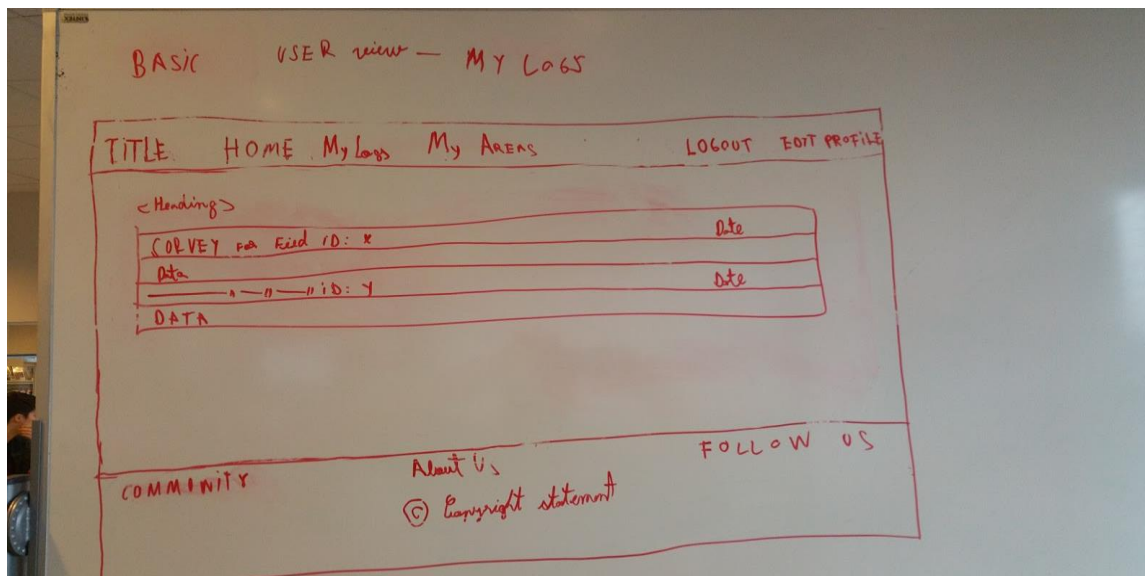


Figure 17 My Logs- Basic User view

BPR 2 – Drone Survey System

The figure below shows how the “add field” page is previewed for the administrator.

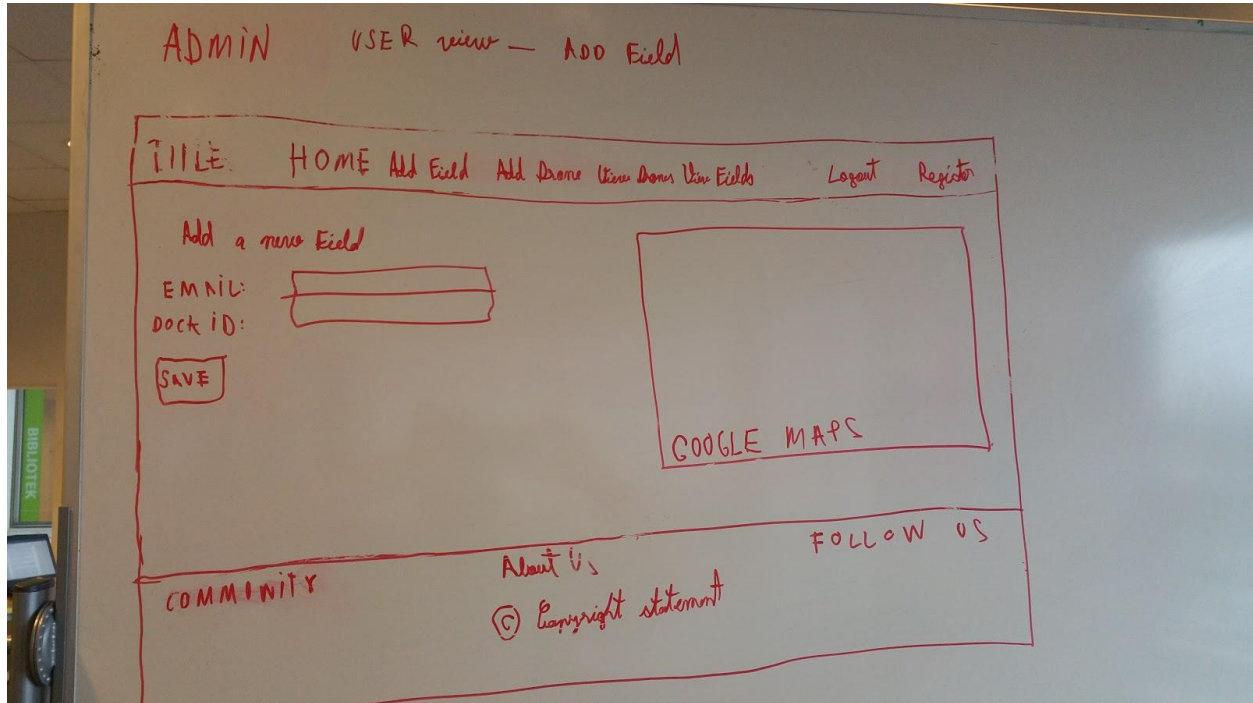


Figure 18 Admin - add field view

5.7.1 User Interface Design

User interface applies the usability guidelines stated in 4.3.3 - User-interface requirements. The content provided for users targets the farmers. While designing the website, a persona scenario is used to provide a better view on whom are the targeted people.

The content does not provide unsolicited windows and the design of the website is expected to be responsive to more common browsers as well.

The homepage follows the principles of enabling access to it for all users. It provides all major options after logging in. Options are provided differently based on login rights.

The pages are overall simple and have the primary navigation on top. There is no need for users to scroll horizontally because the website is responsive from that point of view. All pages have a descriptive title page and highlight critical data using colored lists or tables.

All provided links have a matching name with their destination page. “Text” on the page is black because the background is mostly white.

5.8 Docking Station Algorithm

As presented in the Analysis phase, the Docking Station has modules that each perform a clearly defined task. In the Design phase, the “Process Survey”, “Initialize Survey” and “Analyze Survey” processes are mapped to the blueprints of three algorithms that represent the logic behind the modules.

The “Waypoint Generator” algorithm creates virtual coordinates for the drone to follow. The “Mission Planned” algorithm creates the route using GPS coordinates for the drone to follow during surveys. The “Survey” algorithm implies image analysis for yielding visually interpretable results. In following section, the design of these algorithms is presented.

Waypoint Generator

To establish the pattern of the algorithm, it is firstly required to consider the waypoint generation process. This is done based on iteration of empirical situations, from simple to complex scenarios.

The purpose of the waypoints is to provide the drone a certain path that should cover all necessary fields without overlapping pictures unnecessary.

Five by Five Grid

Note: Position (X, Y) is due to respect to Position (0,0) which is in the left, top corner.

In the figure below, the black square represents a waypoint. The waypoint is the center of a three by three square which symbolizes a survey image.

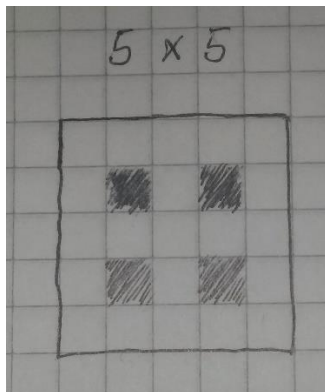


Figure 19 5 by 5 grid

BPR 2 – Drone Survey System

The first pattern focuses on minimizing redundant overlapping on the edges (top and left area of the survey image). Steps that are taken:

1. Start at First Cell (1,1)
2. Go one cell down
3. Go one cell to the right
4. Reach Waypoint (2,2)
5. Go three cells to the right (5,2)
6. (Condition): next movement on X axis implies an overlap with the edge
7. (Condition True): Make new Waypoint at (4,2)

This condition minimizes edge overlapping but does not avoid waypoints one (2,2) and two (4,2) overlapping.

Next step involves the iteration process.

8. Go back to the beginning of the next row and two down and one to the right (2,5).
9. (Condition): previous movement on Y axis implies an overlap with the edge
10. (Condition True): Make new Waypoint at (2,4)
11. Go three cells to the right (5,4)
12. (Condition): next movement on X axis implies an overlap with the edge
13. (Condition True): Make new Waypoint at (4,4)
14. Repeat process of the linear movement in rows, explained previously

Six by Six Grid

Next step is to test if this pattern works on a larger grid, such as a six by six grid. The movement starts at the first waypoint (2,2). It follows the pattern explained above and reaches the second waypoint at (5,2).

The path follows the next row where the third waypoint (2,5) and the forth (5,5) are established. The current case is an ideal result due to no overlapping.

BPR 2 – Drone Survey System

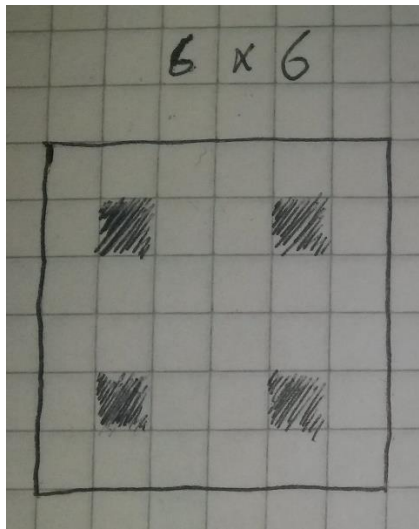


Figure 20 6 by 6 grid

Eight by Eight Grid

The pattern's scalability is tested once more on an eight by eight grid. The result is that only the first waypoint does not overlap. The other waypoints are overlapping in the same manner as in the 5 by 5 grid. This concludes that the pattern is scalable if the shape of the area is a rectangle.

Eight by six Grid

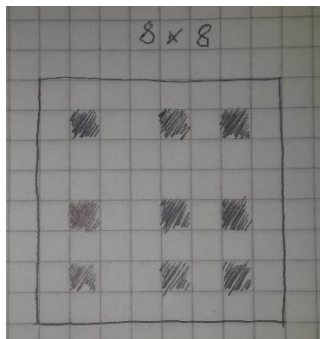


Figure 218 by 6 grid

The purpose of the eight by six grid is to simulate both an irregular terrain and unviable cells for movement. These cells can be either outside the field or be part of a non-fly zone. Under these circumstances, it is required that the waypoint avoid them.

BPR 2 – Drone Survey System

By analyzing this pattern, the algorithm provides an efficient way of distributing waypoints. The downside is that the waypoints are not in line in respect to other waypoints. In a real-life situation, the drone faces the problem of constantly changing the direction if it is required to follow the waypoints. The solution provided for the drone in this case has flaws in optimization, scalability and flight time duration.

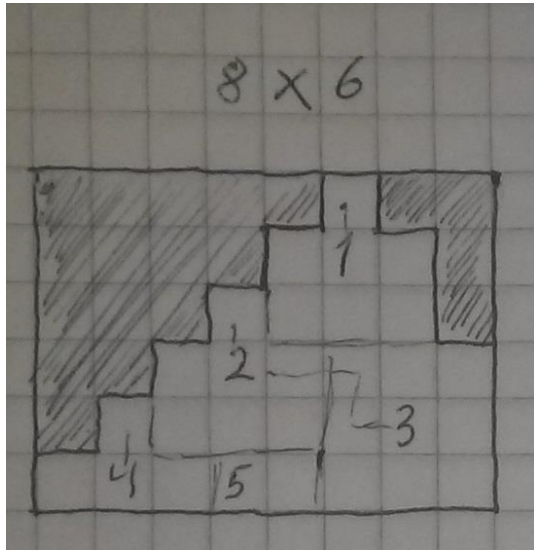


Figure 22 8 by 6 grid

Overall steps taken regards the flow of the path:

1. First viable cell is on Row 1 (6,1)
2. (Condition): Edges around the first cell except at the bottom.
3. (Condition True): go down and make first Waypoint (6,2)
4. Next viable cell is (4,3)
5. (Condition): Edges around the first cell except at the bottom.
6. (Condition True): go down and make Waypoint at position (4,4)
7. Next viable cell is (6,4).
8. (Condition): Edge Top
9. (Condition True): Path goes down one cell (6,7)
10. (Condition): Edge Left

BPR 2 – Drone Survey System

11. (Condition True): Path advances one row to the right and creates the third waypoint (7,5)
12. Validation: Third waypoint covers all cells without any overlapping.
13. Next available cell is (2,5)
14. (Condition): Surrounded by edges except at bottom
15. (Condition True): go down and make Waypoint (2,6).
16. Last waypoint created is at next viable cell (4,6).

Eight by six Grid, second pattern

In the figure below there is another algorithm that uses the same 8x6 grid. In this case, the algorithm focuses on lining up rows.

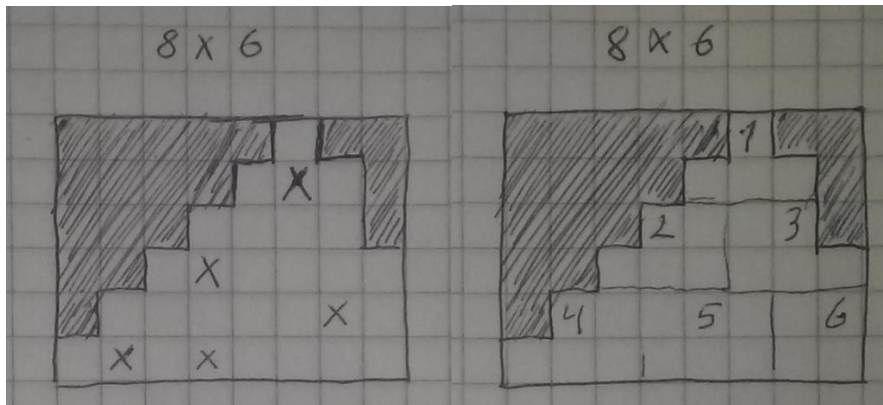


Figure 23 8 by 6 2nd pattern

Figure 24 8 by 6 2nd pattern

Overall steps of this algorithm:

1. Take first valid cell and make it a Waypoint (6,1).
2. (Condition): go to right three cells.
3. (Condition False) go to next viable cell and make it a Waypoint (4,3)

BPR 2 – Drone Survey System

4. (Condition): go to right three cells.
5. (Condition True): Go right 3 cells make Waypoint (7,3).
6. (Condition): go to right three cells.
7. (Condition False) go to next viable cell and make it a Waypoint (2,5)
8. (Condition): go to right three cells.
9. (Condition True): Go right 3 cells make Waypoint (5,5).
10. (Condition): go to right three cells.
11. (Condition True): Go right 3 cells make Waypoint (8,5)

Path is done. This new algorithm follows a more linear pattern and it is possible to scale better than the previous pattern. The new algorithm scales well within a pattern of 14X12.

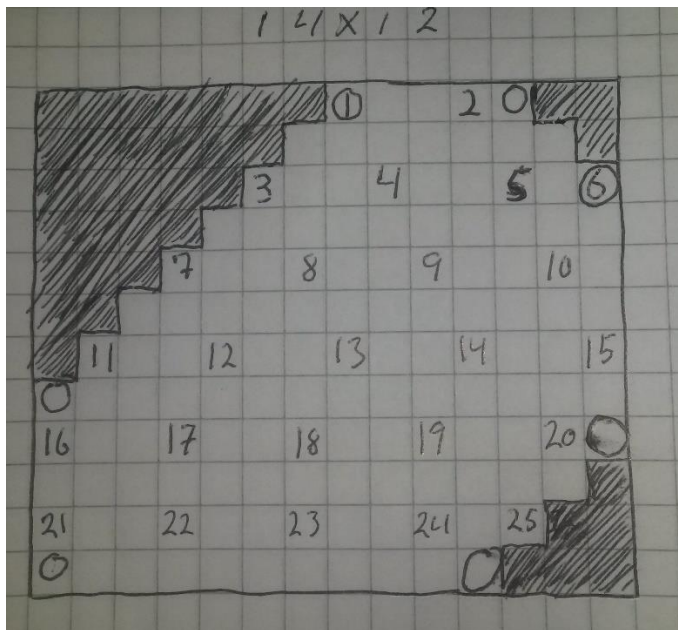


Figure 25 14 by 12 grid

On a grid of 5 by 12, the algorithm could be improved.

BPR 2 – Drone Survey System

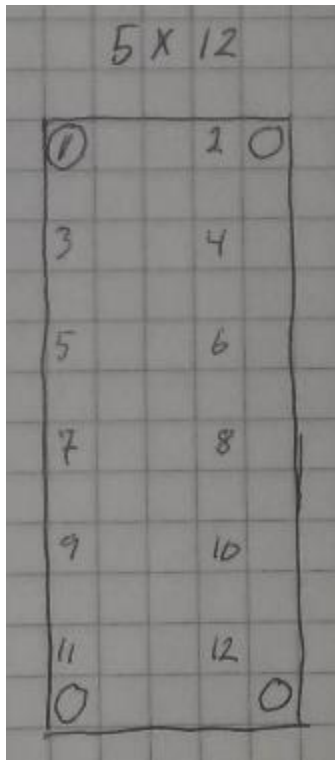


Figure 26 5 by 12 grid

The possibility of improvement is to rotate the grid. To do this, the grid is converted into a matrix and afterwards multiplied by a transformation matrix. To figure out if it is required to rotate and how much, it is needed to mark corners that start and end in a row if they have two edges.

If the corners are in line of two other corners they will not be classified as a corner.

The next process is to measure the distance between two corners, however the focus is only on the distance between the adjacent corners. For example, if there is a rectangle and top left corner is TL and top right is TR and so forth, there is no more focus on the distance between TL and BR nor TR and BL.

Based on the previous statement, each corner would be associated with two other corners, TL would have TR and BL, and so forth. Then it is possible to find the longest distance among these values using simple geometry calculations, and make that one horizontal by matrix multiplication.

Applying this pattern, it is possible to avoid a scenario in which the drone is flying through many short rows in favor of a few long rows. If this technique is applied to the previous grid, it would not be able to optimize it by much because the sides are about the same length.

BPR 2 – Drone Survey System

Mission Planner

After getting the necessary waypoints from the algorithm it is needed to convert them back into GPS coordinates which the drone would need for directions. There are many ways to calculate this. With the curvature of the earth, it is expected to be inaccurate, so it is required to find a solution to prevent this scenario.

To do this, two GPS coordinates is needed from Google's API to "anchor" the Field Image. From this is taken the top pixel (0,0) and look up the corresponding GPS coordinates for them in the API. Same process is done for the bottom right.

The next step is to take the longitude from both and the absolute value of the difference between them. Afterwards, they are divided with the pixel length of the Field Image

Next step is a repetition of the previous one focused on the latitude. The only exception is that the latitude is divide it with the width of the Field Image,

In this way, any point is taken in the image and is multiplied by the x and y coordinates with the corresponding longitude and latitude ratio, the result is the GPS coordinates for that x and y pixel coordinates.

For this, the whole waypoint list is taken. The waypoint list is the one received from the waypoint generator. This list is converted into a dictionary containing all GPS coordinates. These coordinates are saved in a json file, which will be uploaded to the website. Their final usage is by the drone for surveys.

Surveys

Regards surveys, this topic covers an explanation of colors applied on the image instead of the multispectral analysis.

When the drone does a survey, it will take an image on every GPS coordinate and upload it to the docking station. When it is finished taking the images, the docking station will go over every image and sample the colors on it.

The color values are used to give a representative color overlay on the Field Image, on where the waypoint is. After going through all images, it will composite all the results with the Field Image and use the image as the survey result. Afterwards, the result is uploaded and previewed on the website.

6 Implementation

As presented in the last section, the design of the system represents a blueprint for the Implementation phase. Here, it is presented a realization of the flows illustrated by the Sequence Diagrams and the Web-Application is shaped according to the class diagram. Finally, it is deployed on a remote server. As for the Docking Station, details about the implementation of the algorithms are presented.

6.6.1 Login Feature implementation

As stated in the sequence diagram for this use case, once the “Login” button on the “Home” page is clicked, the user is asked to enter the credentials. One interesting thing used in more cases in this project is the fact that pressing the “Login” button creates a GET http request for “localhost:5000/Home/Login”.

However, handling the form requires either using another path (controller function) or sending a POST request to the same path. This is illustrated in the following code snippet.

```
[HttpPost]
0 references | Alex, 35 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public IActionResult Login(String email, String password)
{
    var query = from e in db.Users
                where e.email == email
                where e.password == password
                orderby e.email
                select e;
    var myusers = query.ToList();

    if (myusers.Count != 0)
    {
        var token = tokenBuilder.Build(myusers[0].name, email, myusers[0].is_admin);
        Response.Cookies.Append("token", token);
        RedirectToActionResult redirectResult = new RedirectToActionResult("Index", "Home", new { });
        return redirectResult;
    }
    else
    {
        ViewData["Error"] = "User or password are not correct!";
        return View();
    }
}

[Authorize(Policy = "NotLogged")]
0 references | Alex, 35 days ago | 1 author, 1 change | 0 requests | 0 exceptions
public IActionResult Login()
{
    return View();
}
```

Figure 27 Login implementation

BPR 2 – Drone Survey System

As it can be seen, there are 2 functions that handle the request on “localhost:5000/Home/Login”. However, the first one has an attribute that specifies that it will be used only for POST requests. The second one handles GET by default and is responsible for returning the template of the page. In order to pass parameters to the POST function, it is necessary to have the name attribute of the inputs in the form the same as the parameter name in the function declaration. This correlation can be seen by analyzing the template for the form:

```
@ViewData["Error"]
<div class="container">
  <form id="login_form" method="post" class="row" asp-action="Login">
    <fieldset class="col-md-4 col-md-offset-4">
      <div class="form-input row">
        <label for="email" class="col-md-4">Email:</label>
        <input id="email" type="text" name="email" class="col-md-8" value="" />
      </div>

      <div class="form-input row">
        <label for="password" class="col-md-4">Password:</label>
        <input id="password" type="password" name="password" class="col-md-8" value="" />
      </div>
      <input type="submit" name="submit" value="Log in" />
    </fieldset>
  </form>
</div>
```

Figure 28 Login request handling

Furthermore, this functionality is tightly related to the authorization feature. As it can be seen in the first code snippet, the first function handles the form data. Once a user with the inputted email is found in the system, it is necessary to create a token for the current user. This process is comprised in the following code section:

BPR 2 – Drone Survey System

```
public class JwtTokenBuilder
{
    1 reference | Alex, 35 days ago | 1 author, 1 change | 0 exceptions
    public string Build(string name, string email, int admin)
    {
        //Key used for encoding the token. It is known only by the server.
        var secretKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("verysecretkeylinuxftw"));
        //Create claims associated with the token.
        var claims = new Claim[] {
            new Claim(ClaimTypes.Name, name),
            new Claim(JwtRegisteredClaimNames.Email, email)
        };
        //From the database the 'admin' field indicates the user's privileges.
        string role = admin == 1 ? "Admin" : "User";
        //Building the token.
        var token = new JwtSecurityToken(
            issuer: "DroneSystem",
            audience: role,
            claims: claims,
            notBefore: DateTime.Now,
            expires: DateTime.Now.AddDays(1),
            signingCredentials: new SigningCredentials(secretKey, SecurityAlgorithms.HmacSha256)
        );
        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

Figure 29 Authorization

One important aspect is the fact that this authentication scheme is claims-based, which means that it is making use of a concept where one system acknowledges a user or another system to possess a certain attribute. In other words, a system can claim that a user's email is "[user@mail.com](#)", for instance. In this way, it is possible to validate information regarding entities in a system, taken the fact that claims are always asserted to be true. In this way, in this case, by adding claims to the token, it is possible to know which user is requesting which resource at any time, which is a big reason for this implementation choice.

Once the token is created, it registered in the user's browser's cookies. In this way, the user can be authenticated for each request he makes. However, as stated previously in the design phase, it is not possible to attach the token to each request made with ASP.NET Core and therefore other tools or an additional step before processing the request are necessary. To solve this problem, a middleware class was added to the system, having the responsibility of adding the token to the header of each incoming request.


```

var cookie = context.Request.Cookies["token"];
if (cookie != null)
{
    //check if it is not a logout request
    if (!context.Request.Path.ToString().ToLower().Contains("/home/logout"))
    {
        //if token exists in cookie
        if (!string.IsNullOrEmpty(cookie))
        {
            var token = cookie;
            if (token != null)
            {
                var headerValue = "Bearer " + token;
                //Create Authorization Header which can be used by the Authentication Middleware
                if (context.Request.Headers.ContainsKey("Authorization"))
                {
                    context.Request.Headers["Authorization"] = headerValue;
                }
                else
                {
                    context.Request.Headers.Append("Authorization", headerValue);
                }
            }
        }
        await _next.Invoke(context);
    }
}

```

Figure 30 Authorization continued

This snippet is from the “AuthorizationHeader” class and it shows the way an authorization header containing the token is appended just before the request is being processed by the next middleware, the “JwtAuthorization” middleware.

Both are added as configuration options in the Startup class. As a result, it is possible to indicate which code block should be executed depending on the authorization the user has. This is stressed in the first code snippet, where there is an “Authorize” attribute in square brackets, that specified that only not logged users can access the route.

6.1.2 Add Field Implementation

Whereas the previous Use Case is mainly implemented by making use of the tools provided by the ASP.NET Core framework, the AddField feature required the help of jQuery. This happens at 2 points in this flow: when the free docking stations` ids are requested dynamically and when submitting the form. In the second case, this is needed in order to send the coordinates on the map

to the server. In the next section, there will be a more thorough presentation of the usage of Google Maps.

```

$('#adddrone_form #email').on('focusout', function () {
    let email = $(this).val();

    $.ajax({
        type: "GET",
        url: "/Admin/GetDS",
        async: true,
        success: function (msg) {

            msg = JSON.parse(msg);
            let filtered = msg.filter(ms => {
                return ms !== null;
            })
            let options = "";
            filtered.forEach(elem => {
                options += '<option value="' + elem + '">' + elem + '</option>';
            })
            let def = $('#dsid').html();
            $('#dsid').empty().append(def + options);

        },
        error: function () {
            return "error";
        }
    });
});

```

Figure 31 add drone form

Here, an AJAX call is performed in order retrieve all the IDs of all the docking stations that are not associated with any Field entity at the moment of the request. The “Success” parameter of the AJAX call specifies a function that can filter the IDs and append them to the form as options for a dropdown input.

In the other case, when trying to submit the form, ASP.NET Core will receive the data from the values of the form’s inputs. However, the map itself is not an input, and therefore it is needed that before the submission, a hidden field has assigned as value the coordinates of the map.

```

$('#addfield_form').on('submit', function (e) {
    let coordinates;
    //retrieve coordinates from the map.
    map.data.toGeoJson(function (o) {
        if (o.features[0] !== undefined) {
            coordinates = o.features[0].geometry.coordinates[0];
        }
    });
    //field was not selected on the map.
    if (coordinates === undefined) {
        $('#em').empty().append('Please select the field on the map!');
        e.preventDefault();
    }
    //add coordinates to the value of a hidden field in the form.
    else {
        $('#addfield_form').find('[type="hidden"]').val(JSON.stringify(coordinates));
    }
});

```

Figure 32 add field form

6.1.3 View Field Implementation

The ViewField feature is interesting from an implementation point of view, because it integrates and makes use of the Google Maps Developer API. In this sense, a map is integrated within the page and is configured. It is important to note that this process involves ASP.NET Core only to the point of loading specific JavaScript scripts which do the work and provide coordinates for the map to display.

```

function initMap() {
    map = new google.maps.Map(document.getElementById('map'), settings);

    map.data.setStyle(map_style);
    map.data.setControls(['Polygon']);
    map.data.addGeoJson(geoJson);
}

```

Figure 33 view google maps

Once the google maps script is loaded, the above function is called with the purpose of initializing the map. All the settings variables are declared with some initial values, however because the map is shared with the “AddField” feature, the configurations vary depending on

BPR 2 – Drone Survey System

which one is making use of it. In this way, when viewing a field, the “geoJson” variable contains the coordinates of the field and is initialized if any coordinates are provided from the server-side. Aside from that, the map is disabled for editing.

6.1.4 Database E-R diagram implementation

There are various ways of working with a database when developing .NET. For this system, the Entity Framework is used for managing the persistence layer. As presented in the class diagram, the “Model” namespace contains the entities of this application. In the implementation phase, these were translated into code. The “DbContext” class, which is mandatory when working with EF, is defined in the following way:

```
public class DbContext : DbContext
{
    0 references | Alex, 36 days ago | 1 author, 1 change | 0 exceptions
    public DbContext(DbContextOptions<DbContext> options) : base(options)
    {}
    6 references | Alex, 36 days ago | 1 author, 1 change | 0 exceptions
    public DbSet<User> Users { get; set; }
    11 references | Alex, 30 days ago | 1 author, 1 change | 0 exceptions
    public DbSet<Field> Field { get; set; }
    5 references | Alex, 7 days ago | 1 author, 1 change | 0 exceptions
    public DbSet<Drone> Drone { get; set; }
    7 references | Alex, 7 days ago | 1 author, 1 change | 0 exceptions
    public DbSet<DockingStation> DockingStation { get; set; }
    3 references | Alex, 7 days ago | 1 author, 1 change | 0 exceptions
    public DbSet<Survey> Survey { get; set; }

    0 references | Alex, 7 days ago | 1 author, 1 change | 0 exceptions
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Survey>()
            .HasKey(c => new { c.date, c.field_id });
    }
}
```

Figure 34 Database E-R Diagram

As it can be seen, this class contains DbSet for each type of entity. The protected function is used for explicitly specifying, in this case, that the “Survey” entity has a composite primary key. Usually, it is enough to add an annotation on top of the field desired as primary key in the class definition of the entity. However, this is not possible for composite keys.

BPR 2 – Drone Survey System

In this situation, the model-first EF is used, which indicates that the database is not autogenerated, but rather created by executing manually written queries.

```
CREATE TABLE [dbo].[Survey] (
    [date]      DATETIME      NOT NULL,
    [field_id]  INT           NOT NULL,
    [data]      VARCHAR (MAX) NULL,
    [status]    VARCHAR (25)  NULL,
    PRIMARY KEY CLUSTERED ([date] ASC, [field_id] ASC),
    CONSTRAINT [FK_Survey_ToTable] FOREIGN KEY ([field_id]) REFERENCES [dbo].[Field] ([field_id]) ON DELETE CASCADE
);
```

Figure 35 Database Table

This example illustrates the query for defining the “Survey” database table. It is worth noting that when deleting an entry in this table, all related fields in other table will also be deleted. This is defined by the “ON DELETE CASCADE” clause.

This system's database is hosted on the Azure platform and the connection to it is made through a connection string provided by the platform, which is specified as a value for the “DefaultConnection” parameter in the “appsetting.json” configuration file.

6.1.5 Algorithm implementation

As mentioned in the design part, the first step is generating the waypoints for the drone. In the code snippet below, the Field image is opened and convert it into an array. Besides that, a list of the area is made where the field is in the Field image.

The X, Y coordinates are added manually for every field, in a list which is needed later. It is required of them to be in a list of tuples to make the field path properly. Afterwards, the field path is needed to check where to look at the image. This is done using matplotlib’s Path which makes it easier to check points in the image, if any specific coordinate is inside of the polygon or not.

BPR 2 – Drone Survey System

```

1  from PIL import Image,ImageDraw, ImageFont
2  import time
3  import matplotlib.path as mplPath
4  import numpy as np
5
6  class WaypointGenerator:
7      class Waypoint:
8          def __init__(self,x, y):
9              self.X = x
10             self.Y = y
11
12     def __init__(self, im):
13         self.Coordinates = []
14
15         print ("\nLoading Image")
16         background = Image.open("field.jpg")
17         fieldimage = np.array(background)
18         imout = np.array(fieldimage)
19         background.show()
20
21         #defining the polygon and adding it to matplotlib path
22         poly = [125,51,796,176,787,305,814,433,872,527,921,7
23         length = len(poly)/2
24         new_array = []
25
26         p,i=0,0
27         while True :
28             new_array.append([poly[p],poly[p+1]])
29             p = p + 2
30             i = i + 1
31             if i == length:
32                 break
33
34         field_path = mplPath.Path(new_array)
35

```

Figure 36 Waypoint generator

There are distinct phases to this process. First it is needed to know that the waypoints generated are inside the field or not (instead of random numbers) and how well they cover the area. For this, a graphical illustration is made for both developers for testing and users, who will need to see the end result of the survey. To that end, a few things need to be setup, which is shown in the code snippet below.

BPR 2 – Drone Survey System

```

36 #setup the image draw/composite
37 print("Starting")
38 t0 = time.time()
39 size = width, height = background.size
40 background = background.convert("RGBA")
41
42 fill_red = (255,0,0,50)
43 fill_green = (0,255,0,50)
44 color_layer_red = Image.new('RGBA', background.size, fill_red)
45 color_layer_green = Image.new('RGBA', background.size, fill_green)
46
47
48 alpha_mask0 = Image.new('L', background.size, 0)
49 alpha_mask_draw0 = ImageDraw.Draw(alpha_mask0).polygon(poly,fill=50,outline="black")
50
51 alpha_mask1 = Image.new('L', background.size, 0)
52 alpha_mask_draw1 = ImageDraw.Draw(alpha_mask1)
53

```

Figure 37 Image composition

The algorithm discussed in design is then implemented. It searches through the “field_path” using the same pattern as designed, then saves the waypoints and draws the next layer.

BPR 2 – Drone Survey System

```

55 #going through image and marking waypoints and drawing rectangles
56 x,y,i,w = 0,0,0,0
57 boo = False
58 print("Drawing")
59 while True :
60     x0 = x - 10;x1 = x + 10;y0 = y - 10;y1 = y + 10;
61     if field_path.contains_point((x,y)):
62         alpha_mask_draw1.rectangle([(x0,y0),(x1,y1)] , fill=50)
63         text = str(w)
64         wp = WaypointGenerator.Waypoint(x,y)
65         self.Coordinates.append(wp)
66         w = w + 1
67         alpha_mask_draw1.text((x0+8,y0+5),text, fill="ffffff", font=None)
68         boo = True
69     elif field_path.contains_point((x-10,y)):
70         alpha_mask_draw1.rectangle([(x0-5,y0),(x1-5,y1)] , fill=50)
71         text = str(w)
72         wp = WaypointGenerator.Waypoint(x,y)
73         self.Coordinates.append(wp)
74         w = w + 1
75         alpha_mask_draw1.text((x0+8,y0+5),text, fill="ffffff", font=None)
76         boo = True
77     if x > width :
78         x = 0
79         if boo :
80             y = y+20
81             boo = False
82         else:
83             y = y+1
84     if y > height :
85         break
86     if boo :
87         x = x +20
88     else:
89         x = x+1
90     i = i +1

```

Figure 38 image and waypoints marking

```

92 print("Total waypoints: ", w)
93 print("Compositing Image")
94 background = Image.composite(color_layer_red, background, alpha_mask0)
95 background.show()
96 background.save("field.png")
97 background = Image.composite(color_layer_green, background, alpha_mask1)
98
99 t1 = time.time()
100 totaltime = t1-t0
101 print("Time elapsed: ",totaltime, "\n")
102 background.show()
103
104 background.save("result.png")
105

```

Figure 39 image and waypoints marking 2

BPR 2 – Drone Survey System

Compositing the layers, it shows the process.

This is the initial image which is used to perform the image scan.



Figure 40 Initial top-view of the field

Here the polygon layer is applied and should cover the actual field that needs surveying.



Figure 41 polygon layer of the field

In the next step, the waypoint layer is applied and ideally the green should cover the red as much as possible.

BPR 2 – Drone Survey System

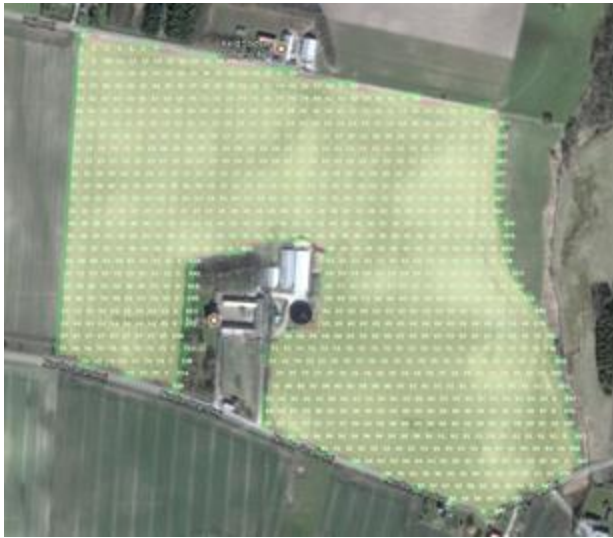


Figure 42 Waypoints Layered Field

After conducting a survey, all the images from every waypoint are sampled. The next step is to composite them, all together along with the Field Image which is the final result. Since the drone does not meet the requirements to perform the survey, testing the drone is done with randomly generated images. Even though the result is not very appealing, it demonstrates that it functions properly and meets all the requirements of the project.

BPR 2 – Drone Survey System



Figure 43 Final Result of Field

The above image is the survey result, uploaded and available for users to view.

6.1.6 Web-Application-docking station communication

The communication between the 2 parts of the system is implemented under the form of APIs, where there is one at each end. In this context, the APIs are represented as a list of paths to which http requests can be made and which are expected to have a predefined behavior. For the Web-Application parts, the “DroneController” class has routes defined so that the Docking Station can request any needed data and this is very similar to the other controller classes of the application. On the other side, the Docking Station, being implemented in Node.js, makes use of a middleware

BPR 2 – Drone Survey System

named “Express” which provides an easy configurable server and a very flexible way of defining API paths. Although, in the scope of this project there is no real need of having an API on the Docking Station side, one was setup up while taking in consideration possible improvements and add-ons to the system.

For accessing the “DroneController” API, the Docking Station makes use of an external package named “Request”. This is added as a dependency to the application and allows to create http requests in an intuitive manner.

All in all, the list of paths that the “DroneController” has defined is:

- <http://serverpath:port/Drone/getID> - handles GET requests. Creates a new entry in the “DockingStation” database table and returns the its ID.
- <http://serverpath:port/Drone/Survey> - supports GET requests. Checks in the system if there are any pending surveys for the field which is associated with the requesting docking station. Once an occurrence is found, the Docking Station receives the coordinates for the field.
- <http://serverpath:port/Drone/PostResult> - supports POST requests. Receives the result data from the Docking Station and updates the “data” column of the performed survey in the “Survey” database table. Additionally, the “status” column is set to “Finished”.

It is important to note that handling the data exchange between the two parts of the system in such a way is very flexible and intuitive. The biggest asset that this solution provides is interoperability in case the need of third-party component integration.

6.1.7 Drone-Docking Station Communication

While working with a prototype drone during this project, some limitations were imposed, and this led to designing the system in a drone-type-independent manner. In this way, the Docking Station would be able to control different type of drones if needed. However, in the case of the prototype, Parrot AR 2.0 Drone, controlling it required the use of a Node.js package - “node-ar-drone”. Since the only way to control the drone is by connecting to its WIFI network, a separate module had to be created with the purpose of switching between the internet and the drone’s WIFI network.

BPR 2 – Drone Survey System

6.1.8 Deployment implementation

As talked previously in the Design phase, the system is planned to be deployed in a production environment. This was mainly done by running the Web-Application part on an external server. It is possible to observe how all the parts fit together when the system is deployed in the following diagram:

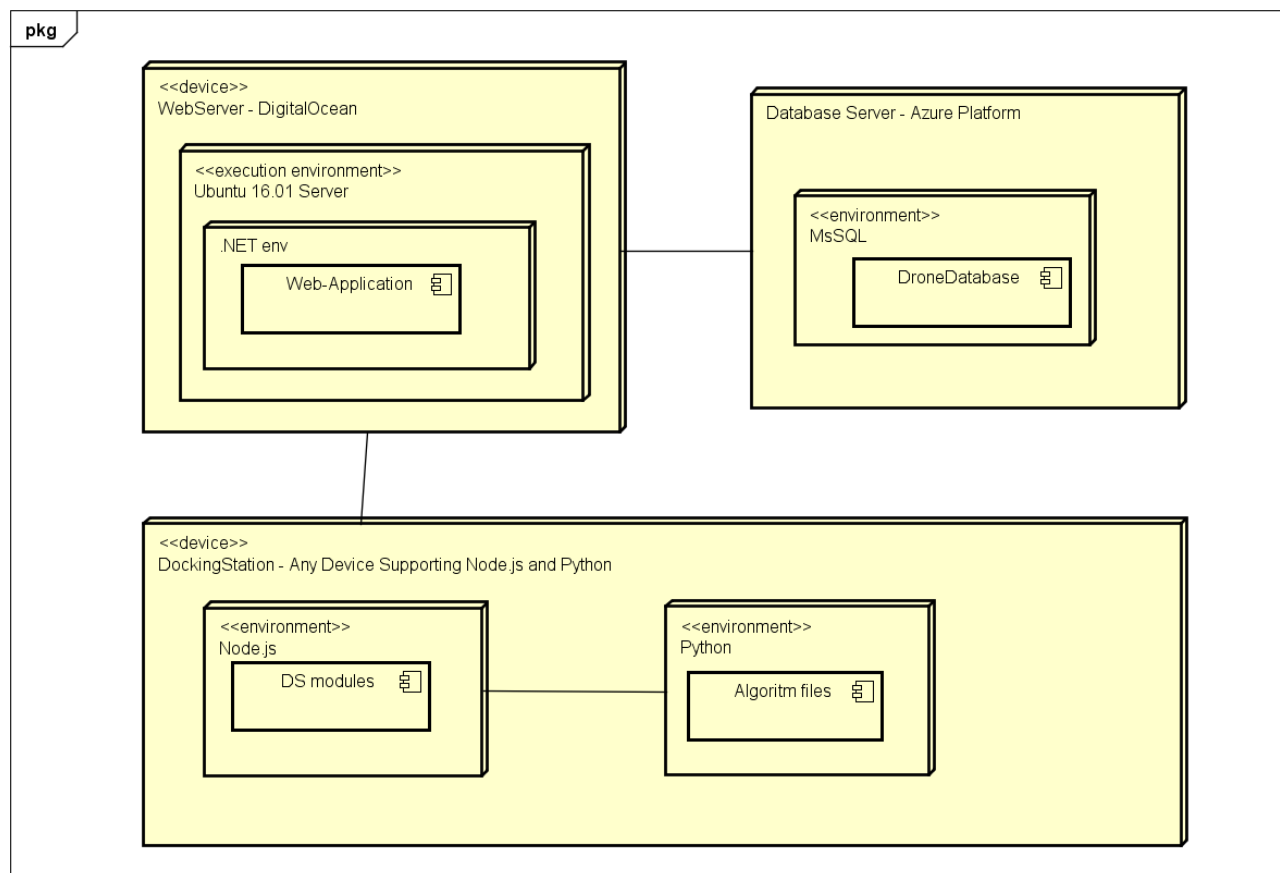


Figure 44 Deployment Diagram

First, the Web-Application is running in a “droplet” on Digital Ocean’s servers. Digital Ocean is a PaaS solution provider and it offers a Linux Ubuntu environment which can be used as a web-server. The .NET runtime is installed on the droplet to make it possible to run the application. On the other side, the database is running on the Azure platform and therefore it is easily accessed

BPR 2 – Drone Survey System

from any type of device when needed. The same would also be possible if it was running the web-server, except with the cost of having to manage and configure a web-server as well as the database on it manually.

For the Docking Station part, the application would run from any device with Node.js and Python installed on it, as previously mentioned: laptop, beaglebone or even mobile phone.

7. Testing

In order to test the required functionality of the system, it was decided to sacrifice comprehensive testing for a more rapid development. For this purpose, unit testing is not included because reliability is out of scope. Because of the scale of the project, it was decided to use smoke testing and limited system integration testing along with regression testing throughout the testing phase.

7.1 In Scope

Website functionalities are tested through deployment and demonstrations. Covered examples:

7.1.1 Website testing:

Table 8 Website Requirements Testing

Requirement	Pass/Fail
User must be able to log in based on given account.	Pass
User must be able to start the survey.	Pass
User must be able to view the analysis of the survey.	Pass
User must be able to log out.	Pass
User must be able to edit profile.	Pass
The drone should be able survey the field.	Pass
The docking station should perform the image analysis.	Pass
The drone would not be able to record the survey.	Pass
The drone must be able to forward the images to the docking station.	Pass
The “docking station” (laptop) must be able to store the result of the survey in a database.	Pass
Administrator must be able to login.	Pass
Administrator must be able to logout.	Pass
Administrator must be able to view Fields.	Pass
Administrator must be able to view Drones.	Pass

BPR 2 – Drone Survey System

Administrator must be able to register new user.	Pass
Administrator must be able to register drone.	Pass
Administrator could be able to delete docking station.	Pass
Administrator could be able to delete drone information.	Pass
Administrator would not be able to edit users.	Pass
Administrator must be able to associate a drone id with a user id.	Pass
The website would not be able to notify the user when the task is done.	Pass

7.1.2 Docking station testing:

1. Query website for new survey requests
2. Sending drone commands
3. Receiving images from drone
4. Processing images
5. Uploading survey result

7.2 Out of scope

Performance testing for the website represents one of the issues that is out of scope. Besides performance, deployment of the docking station and verification of connectivity (with third party systems) are not considered for testing.

Verification of the website uptime is an issue that is not tested. This is due to the fact that proper testing requires more time and focus without giving any benefits. Exception handling is not tested because the type of the project is proof of concept.

7.3 Types of testing performed on website

Smoke testing was done whenever a build was uploaded to Bitbucket. This action assured that the major functionalities are working properly.

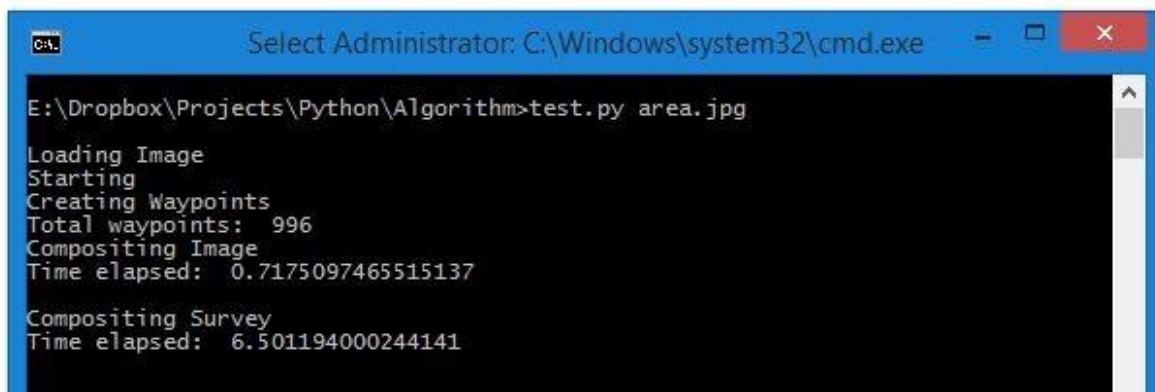
System Integration Testing was performed on the website. This is done to verify that the entire application works as per the requirements.

BPR 2 – Drone Survey System

Regression Testing was performed each time a new build contained bugs or new enhancements, and then later it was deployed, where we did system integration testing.

7.3 Types of testing performed on Docking Station

Development was done module by module, for the docking station. Both system integration testing and regression testing were done for every iteration. For the algorithm, the system integration testing was done with a testing module.



```
Select Administrator: C:\Windows\system32\cmd.exe

E:\Dropbox\Projects\Python\Algorithm>test.py area.jpg

Loading Image
Starting
Creating Waypoints
Total waypoints: 996
Compositing Image
Time elapsed: 0.7175097465515137

Compositing Survey
Time elapsed: 6.501194000244141
```

Figure 45 Algorithm testing

As shown in the picture above, no errors were found. The waypoint generator managed to finish within a second. The survey managed to complete the sampling of 996 images and compositing them under 7 seconds.

7.4 Issues and Recommendations

A lot of the tests had to be done individually and manually. This is because a lot of the development was done locally by separate developers. A suggestion is to have everything on Gitbucket (not just the website) and have a script to make the testing automatic.

7.5 Exit Criteria

These are the current exit criteria:

1. Completion of testing by fulfilling certain conditions

BPR 2 – Drone Survey System

2. All tests cases should be executed
3. All critical/major/medium defects should be solved, and any trivial defects should have an action plan with an expected deadline

7.6 Conclusion

As the exit criteria is met and satisfied as mentioned in previous section, the website is suggested to ‘Go Live’. The docking station is suggested to go into deployment, given that an appropriate drone and testing field can be acquired for deployment.

8. Results and discussion

The results are compiled from four main sections. These sections are the analysis, design, implementation and testing.

The analysis phase comprises the initial assessment of possible requirements. These requirements are taken from survey results on farmers, one meeting with SEGES and research. The next step of the analysis is transmuting the requirements into a persona scenario, use cases, use case descriptions and activity diagrams.

The analysis phase includes four main website use case descriptions: login, start survey, view field and add field. Based on these use cases, the activity diagrams are built. Besides the website use cases, the analysis includes the docking station modules with its activity diagrams. As an overview of the system, the domain model diagram is presented in the last phase of analysis.

The result of the analysis phase is having a clear view on the “What” of the project. Based on this, it is possible to proceed to the “How”, which is the Design phase.

In the design section, the overall view of the website is done as a mock-up design. The mock-up design follows certain usability guidelines for an easier use. Regards the website, the authorization scheme is created based on tokens. To visualise the MVC structure, a class diagram of the MVC connections is designed. For a better understanding of communication among classes in the MVC, sequence diagrams are used.

A database is required to handle required data. The database E-R diagram is comprised of five main entities: survey, field, user, docking station and drone.

BPR 2 – Drone Survey System

Besides designing the website, one of the most important aspects is designing the docking station algorithm that provides the drone a certain path to cover all necessary fields. The deployment design is taken in consideration as the next action after testing.

The design of the system represents a blueprint for the implementation phase. The implementation phase result includes implementation of previous use cases mentioned, the E-R diagram and all the relevant diagrams mentioned before.

Next phase that follows the implementation is testing. Testing is done on the website and on the docking station as well. The results are satisfactory, based on the exit criteria mentioned in the specified section.

9. Conclusion

This project meets all requirements stated in the analysis phase. The result of the analysis phase is having a clear view on the “What” of the project through proper assessment of necessary requirements.

The design phase required both technical expertise and creativity. It is meant to provide a deeper understanding of connection between concept and actualizing the project.

During the implementation phase, many technical difficulties are experienced due to lack of version documentation on the MVC .NET Core. Despite that fact, all the necessary functionality is implemented.

With the testing phase, all requirements underwent a verification process that deemed them satisfactory based on the exit criteria.

10. Project Future Considerations

10.1 What Is missing

1. Drone equipped with a flight planner and a multispectral camera.
2. Drone with proper encryption on communication
3. Mobile client support for the website

10.2 What can be improved

1. The docking station can be automated.
2. Automatization of adding the field.

3. Docking station log in (making use of the API on the DS side)
4. The website's design can be more appealing
5. The docking station could have induction charging.
6. The docking station could be fully customized for the drone's needs.

11. References

CIA, 2017. *Denmark*. [Online]

Available at: <https://www.cia.gov/library/publications/the-world-factbook/geos/da.html>
[Accessed 3 December 2017].

Danish Agriculture and Food Council, 2017. [Online]

Available at: www.agricultureandfood.dk/danish-agriculture-and-food/organic-farming
[Accessed 22 March 2017].

Hørfarter, R., 2017. *capigi.eu*. [Online]

Available at:

http://www.capigi.eu/Portals/9/Images/CAPIGI%202016/Speakers/Bio%20en%20Abstract/Rita_H%C3%B8rfarter_Bio_Abstract.pdf?ver=2016-05-19-164333-517

[Accessed 20 March 2017].

Microsoft, 2017. *ASP.NET overview*. [Online]

Available at: <https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx>
[Accessed 4 december 2017].

SEGES, 2017. *SEGES FOR ICT BACHELOR STUDENTS*. [Online]

Available at: [https://studienet.via.dk/Class/IT-BPR1-S17/Session%20Material/CHBM_21022017_welcome%20to%20SEGES_UK%20\(1\).pdf](https://studienet.via.dk/Class/IT-BPR1-S17/Session%20Material/CHBM_21022017_welcome%20to%20SEGES_UK%20(1).pdf)
[Accessed 20 March 2017].

Shneiderman, M. O. L. a. B., 2003. *Research-Based Web Design & Usability Guidelines*. [Online]

Available at: https://www.usability.gov/sites/default/files/documents/guidelines_book.pdf
[Accessed 03 September 2017].

W3schools, 2017. *Bootstrap grid system*. [Online]
Available at: https://www.w3schools.com/bootstrap/bootstrap_grid_system.asp
[Accessed 04 December 2017].

12. Glossary

Drone: an unmanned aircraft or ship that can navigate autonomously, without human control or beyond line of sight. It is the device used for survey of fields.

User: represents a person that operates or interacts to a certain degree with either the website or the drone.

DSS: Drone Survey System

DAAS: Danish Agricultural Advisory Council.

API: Application program interface. It represents a set of protocols, routines and tools for building software application. It specifies how software components should interact.

Spectrum: a band of colors, as seen in a rainbow, produced by separation of the components of light by their different degrees of refraction in accordance to wavelength

SWO-Acronym for Belbin test results that stands for “Strengths “, “allowable weaknesses “, “don’t be surprised to find that “

Google Map API: this is where we get the Field Image and the GPS coordinates that we need for the fields. This is given in geojson

Field Image: this is an image from google maps giving an overview of the field, and will be used by the waypoint generator to calculate from GPS longitude and latitude to pixel x and y coordinates

Pixel: a point in the image displaying the field.

Waypoint: the coordinates that the drone will stop and take a picture in, this will also capture the surrounding area.

Survey image: for every waypoint that the drone will stop at, it’ll take a picture which we call survey image.

BPR 2 – Drone Survey System

Cells: will represent pixels on the Field Image

Grid: to simulate the Field Image we will be using a grid, to troubleshoot our waypoint algorithm, and come up with the algorithm that suits our needs best.

13. List of Appendixes

- 13.1 Appendix A Project Description
- 13.2 Appendix B User Guide
- 13.3 Appendix C Source code
- 13.4 Appendix D Diagrams and use case descriptions
- 13.5 Appendix E Survey Data Sheets