CS Capstone Software Requirements Document

November 2, 2017

Linking Seasonal Weather Data to AgBizClimate™

Prepared for

Oregon State University

Clark Seavert	Signature	Date

Prepared by

Group26

Thomas Noelcke	Signature	Date
Shane Barrantes	Signature	Date
Shengpei Yuan	Signature	Date

Abstract

Contents

1 Introduction

1.1 Purpose

This SRS describes the requirements and specifications of the AgBizClimate^{TM}web application. This document will explain the functional features of this web application. This includes the interface details, design constraints and considerations such as performance characteristics. This SRS is intended out outline how we will proceed with the development of the *AgBizClimate* system.

1.2 Scope

This project is a part of a much larger AgBiz Logic^{TM}program. However, the purpose of this project is to add a short term climate tool to the *AgBizClimate* module. This limits the scope of the project to the *AgBizClimate* Module. Additionally, we will only be adding the short term climate data tool as the long term climate data tool already exists.

1.3 Definitions, Acronyms and Abbreviations

REST - Representaional State Transfer, This is a type of architecture that manages a the state of the program. This is esspecially populare in web development.
API- Application Programming Interface. This is a peice of softwared that allows a connection to another peice of software providing some sort of service.
NWCTB - Northwest Climate Toolbox. This is the database we will be connecting to that will provide the short term climate data we plan to use.
Climate Scenario - This is a theoretical calculation of yields, inputs and of the overall budget for one situation based on the climate data.
SQL Database - This is a relational database that makes storing and accessing data easy.

1.4 References

[1] C. F. Seavert, "Negotiating new lease arrangements with the transition to direct seed intensive cropping systems," 2017.

[2] S. Y. Thomas Noelcke, Shane Barrantes, "Problem statement," 2017.

1.5 Overview

Seasonal climate is one of the essential factors that affects agricultural production. As a module of *AgBiz Logic*, *AgBizClimate* delivers essential information about climate change to farmers, and help professionals to develop management pathways that best fit their operations under a changing climate. This project aims to link the crucial seasonal climate data from the Northwest Climate Toolbox database to *AgBiz Logic* so that it can provide changes in net returns of crop and livestock enterprises through powerful graphics and tables.
Currently *AgBizClimate* has a long-term climate tool but no such tool exists for short term climate data. We will implement a tool to extract short-term climate data from the Northwest Climate Toolbox database, display it to the user and allow the user to adjust crop and livestock yields or quality of products sold and, production inputs. Moreover, a landing tool will be developed to allow users to switch between short-term seasonal tool and long-term climate data tool.

2 Overall Description

2.1 Product Functions

AgBizClimate is a web based decision tool that will allow users to gain specific insight on how localized climate data for the next seven months will affect their crop and livestock yields or quality of products sold and production inputs. The *AgBizClimate* tool will allow users to input their location (state, county) and a budget for the specific crop or livestock enterprise. *AgBizClimate* will select climate data for the next seven months for that location and provide graphical data showing temperature and precipitation. Users will then be able to change yields or quality of product sold by a percentage they think these factors will affect and modify production inputs. Finally the tool will calculate the net returns.

2.2 User Characteristics

AgBizClimate users can be split into two subgroups, agricultural producers and climate researchers. The first subgroup, agricultural users who use this product tend to be between fifty and sixty years old of mixed gender. Their educational background ranges from high school to the completion of college. The primary language this group uses is English, but there are some Spanish users as well. Most of the users in this group tend to have novice computational skills. The primary domain for these users is agricultural and business management. Most agricultural producers who use this product are motivated by the potential profit that the decision tool *AgBizClimate* could potentially offer. The second subgroup, climate researchers range from ages twenty to forty and are of mixed gender. The educational background for most climate researchers exceed the postgraduate level with their primary language being English. These users generally have advanced computational skills and are motivated by the easily accessible climate and weather data.

2.3 Constraints

There are several key constraints that this product has to work within. The first constraint is that we only have access to two data parameters from the North West Climate Tool box, precipitation and temperature. Secondly, we only have access to their data via the NWCTB API which could have additional restrictions such as limited usage per day, mislabeled data, or poor documentation. Thirdly, we dont have access to any of the hardware that *AgBizClimate* is exists on as it is being managed by a third party. This will prevent us from improving the hardware or cause roadblocks if their servers are having issues. Lastly, we are limited to using the languages Python and JavaScript since we are integrating our product into an already existing project.

2.4 Assumptions and Dependencies

We are assuming that the Northwest Climate Toolbox is a functional API that will allow us to pull location based temperature and precipitation data. This data will most likely come in the form of a text body of which we will then format into a JSON object and store in a Mongo database for future use. Due to the fact that we are writing an addition to an existing project we do not need to interact with the user budgets as these have already been defined. This fact extends to the calculations portion of the *AgBizClimate* product. Our team will simply be accessing data via the NWCT API, then format the data, store the data, and hand the data over to the tool while will provide some additional front end support.

3 Specific Requirements

This section contains all of the functional and quality requirements of the *AgBizClimate* System. We will give detailed description of what's being added to the *AgBiz Logic* system along with the features we will implement.

3.1 External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the short term climate tool for the *AgBizClimate* system. This section will also provide descriptions for the hardware, software and communication interfaces. Additionally we will provide detailed prototypes for the user interface for the short term climate data tool.

3.1.1 User Interface

When the user first navigates to *AgBizClimate* from the *AgBiz Logic* main page the user will be directed to a landing page. This landing page will allow the user to either select the existing long term climate tool or the short term climate tool that we will be developing. On this page there will be a brief description of the tool and what does. This description will also describe the difference between the long term climate data tool and the short term data climate tool. Below this description will be two buttons one to run the long term climate data tool and one to run the short term climate data tool. Clicking the long term climate data tool button will take you to the long term climate data tool page. Clicking the short term climate data tool button will take you to the short term climate data tool page. A prototype of this page is shown below in figure 3.1 (Not yet available)

After clicking on the short term climate data tool page you will be directed to a page that will allow you to create a new climate scenario. This page will allow the user to chose which budgets they would like to adjust and make notes about this scenario. This page also allows them to cancel or delete the scenario they are currently working on. A prototype for this page is shown below in figure 3.2 (Not yet available)

Once the user Selects their budgets, makes notes on the scenario and clicks continue the will be redirected to a page where they will be prompted to enter their location. This page will take a county and a sate as input. This page will have a drop down menu for the state and the county. Before the user can enter a County they will be forced to enter a state. Once a state is entered the county drop down menu will auto populate and the user will be allowed to enter a county. A prototype of this page is shown below in figure 3.3 (Not yet available)

After clicking continue the user will be taken to the plots for the location they selected. This page will initially have the 7 month average precipitation plot selected as the default. However, the user can select from four options via a drop down menu. Changing what is selected in the drop down menu will change which plot is displayed. Once the user has reviewed the plots they can then enter in how much they think their yield will be effected based on the climate data. A prototype of this page is shown below in figure 3.4 (Not yet available)

Next the user will be directed to the budget review page. This page will display a summary of the budget. The summary of the budget will include Income, General Cash Costs and A total summary of the budget. This page will allow the user to adjust the cost per unit, input costs, and quantity sold and will adjust their budget in near real time. This page will also allow the user to remove or add inputs to their general costs. This page will also allow the user to save their budget and output it as a PDF. A prototype of this page is shown below in figure 3.5 (Not yet available)

3.1.2 Hardware Interfaces

This project requires no designed hardware and the hardware this system is running on is managed by the Operating system as a result no hardware interfaces are necessary. Additionally the connection the the NWCTB is managed over the network and requires no hardware interface.

3.1.3 Software Interface

There are several software interfaces in this application, one between the back end and the front end, one between the back end and the Database and finally a software interface between our RESTFUl API and the NWCTB. The majority of the connections between the front end and the back end are managed for us the the current *AgBiz Logic* system. However, we will still need to connect our API to the back end to provide the data from the NWCTB. To make this connection we will use a RESTFULL API. This will allow for the back end code to make a direct call to our API to extract the data.
We will also need a connection between the NWCTB and our API. At the time of writing this document we have not received any information from the NWCTB regarding the API they have promised access too. However, We do know that we will need to authenticate the connection with the data base, send a request and then receive the result. Authenticating the connection will involve some sort of hand shake where we pass the NWCTB a key. Once we authenticate the connection we can then send a request. A request will contain a variety of information including a location for which we want the data. Once we have sent the request we will then wait for the response and accept the data. We may also need to store this data is a data base. If this is necessary we will need an additional software interface between the database and the API.

3.1.4 Communications Interfaces

This project will require communication between its various parts. One key lane of communication is between the front end and back end of the application. Another key lane of communication will be between the NWCTB and our API. Most of the communication between components will be carried out through HTTP requests. These are managed by the operating system and will not effect the way our application works.

3.2 Functional Requirements

In this Section we will list and discuss the functional requirements for this project. Each functional requirement shall have an ID, Title, Description and Dependencies. The ID shall be unique for each requirement. The description will give a detailed expiation of the requirement. The dependencies section will list the requirements that will need to be complete before starting work on that requirement.

3.2.1 Functional Requirements For API

Functional Requirement 1.1 ID: FR1.1
Title: Request for Users Location Data
Description: The API shall take an HTTP Post request with the users State and County as parameters. Upon receiving the HTTP Post request the API will strip the parameters off the Request and store the values in variables for later use.
Dependencies: FR2.1

Functional Requirement 1.2 ID: FR1.2
Title: Transforming Users Location data.
Description: The API shall take the users Sate and county information and transform it into latitude and longitude. It shall then store the results in a variable for later use.
Dependencies: FR1.1 and FR2.1

Functional Requirement 1.3 ID: FR1.3
Title: NWCTB Authentication Description: The API shall authenticate the connection with the NWCTB by sending a request to connect to the NWCTB. This request will include a validation key to ensure that the connection is valid.
Dependencies: None.

Functional Requirement 1.4 ID: FR1.4
Title: Requesting data
Description: After the correct user information has been gathered and the connection with the NWCTB has been authenticated the API shall send a request to the NWCTB API for the information the user requested. This request shall be made via an HTTP Get request.
Dependencies: FR1.1, FR1.2, FR1.3 and FR2.1

Functional Requirement 1.5 ID: FR1.5
Title: Receive Response from NWCTB
Description: After sending the request for the data to the NWCTB API the API shall wait for a response to the request. Once a response has been send the API will receive this response through an HTTP Response. If the request for the user data results in an error the API shall send the appropriate response code along with a useful error message. This shall be done through an HTTP Post response.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4 and FR1.5.

Functional Requirement 1.6 ID: FR1.6
Title: Processing the Data to JSON
Description: Once the request has been successfully received the API shall process the raw data in the response body of the HTTP response sent by the NWCTB API. The raw data shall be placed into a JSON object and useful labels shall be applied to the JSON object so it can be easily displayed later.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5 and FR2.1

Functional Requirement 1.7 ID: FR1.7
Title: Send the Data to the Front End
Description: Once the data has been place into a JSON object and formatted the API shall send the resulting data to the front end application so it can be displayed. This shall be done via an HTTP Post response. The JSON object that contains the formatted data shall be placed in the response body. The appropriate headers for the Response shall be set to indicate that the response contains a JSON object.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR1.6 and FR2.1

3.2.2 Functional Requirements For Front End

Functional Requirement 2.1 ID: FR2.1
Title: Landing Page.
Description: Upon loading the *AgBizClimate* application the user will be directed to a loading page this page will allow the user to select between the short term climate tool and the long term climate tool.

Dependencies: None.

Functional Requirement 2.2 ID: FR2.2
Title: Request data from API
Description: After selecting the short term data tool the user will be directed to a page where they will be required to enter the location they would like to analyze. This page will require the user to first enter a state via a drop down menu and then a county via a drop down menu. Clicking the continue button will send an HTTP request to the API for the climate data for the location the user requested. The Front end will then wait for a response to the request sent to the API. This request shall be made through an HTTP Post request.
Dependencies: FR2.1

Functional Requirement 2.3 ID: FR2.3
Title: Plot the Data
Description: After the API processes the request for the data for the location the the user specified the front end will receive the response from the API. The front end shall then take the data out of the response body and use a plotting library to plot the data and display it to the user.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR, 1.6, FR1.7, FR2.1 and FR2.2

Functional Requirement 2.4 ID: FR2.4
Title: Entering Adjustments to yield
Description: Once the data has been plotted and displayed the the user front end shall allow the user to make adjustments to the expected yield of the crop they are analyzing. This shall be done via a text box. The user input shall be limited to a decimal number with no more than one decimal place.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR, 1.6, FR1.7, FR2.1, FR2.2 and FR2.3

Functional Requirement 2.5 ID: FR2.5
Title: Redirect To Budget Tool.
Description: Once the user has entered the adjustments to the yield for the crop they are analyzing they shall then be redirected to the existing *AgBizClimate* budgeting tool. This tool will be sent the data they entered in the previous step and shall reflect the adjustments to yield that they made.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR, 1.6, FR1.7, FR2.1, FR2.2, FR2.3 and FR2.4

3.2.3 Functional Requirements for Testing

Functional Requirement 3.1 ID: FR3.1
Title: Unit Tests For API Routes. Description: We shall write unit tests to cover 100 percent of our API routes. Each route shall have a test case that is non-trivial and seeks to emulate how the route will be used by the application. These test cases shall test that the correct response is returned from each API route.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR1.6 and FR2.1

Functional Requirement 3.2 ID: FR3.2
Title: Unit Tests for User Actions on Front End
Description: We shall provide unit tests for the various user actions on the front end of the application. The front end tests shall ensure that clickable objects on the page perform the correct action. We shall provide tests for all clickable objects that redirect the application to another page. Our tests shall ensure that if a clickable action is clicked that the correct action is taken for each clickable object.

Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR, 1.6, FR1.7, FR2.1, FR2.2, FR2.3, FR2.4 and FR2.5

Functional Requirement 3.3   ID: FR3.3
Title: Unit Tests for Budget Save
Description: We shall provide unit tests that ensure that the budget is being correctly saved once the user is all done making adjustments to their budget. This test case shall ensure that the values sent to the back-end of the application are posted to the database. The test will also ensure that the values posted in the database are correct.
Dependencies: FR1.1, FR1.2, FR1.3, FR1.4, FR1.5, FR, 1.6, FR1.7, FR2.1, FR2.2, FR2.3, FR2.4 and FR2.5

3.3 Performance Requirements

In this section we will list the performance metrics for our application. This performance metrics will describe acceptable performance for our application. Each requirement will have a title and a description that provides details for how that metric will be measured.

3.3.1 Performance Metric 1

Title: Run Time Performance
Our application shall provide the plotted data in a reasonable amount of time 100 percent of the time. Reasonable being defined as less than 5 seconds. This time shall be measured from the time the user presses the next button after selecting their state and county to the time the web page is loaded with the plot being displayed.

3.3.2 Performance Metric 2

Title: Reliability
Our application shall provide the correct result to the user 100 percent of the time. This result can include an error if the user enters a location that doesn't exist or if the data the user request is unavailable.

3.4 Design Constraints

The constraints of this project can be divided into two parts. Software constraints and hardware constraints. In this section we will discuss these constraints.

3.4.1 Software Constraints

This project shall be implemented using Django, Angular JS and SQL databases. What we add must also be done using Django, Angular JS and SQL.

Additionally, what we create must work with the the existing *AgBiz Logic* system. This means we will be forced to use software interfaces and databases that have already been implemented that we have no control over.

3.4.2 Hardware Constraints

Another factor that will effect how our software works is the hardware that this system is running on. The hardware this system runs on has already been determined and is managed by the university. We don't expect that the hardware will affect our project. However, we have no control over the hardware the *AgBiz Logic* system is running on.

3.5 Other Requirements

This section will cover any additional information pertinent to this project but not covered in other sections.

3.5.1 Stretch Requirements

In this section we will discuss requirements not apart of our contract with our client. These goals will not be required as apart of this project but are objectives that we would like to complete.

Stretch Requirement 1.1 Title: Location Via a Map
Description: In requirement 2.2 instead of using two drop down menus to select the location the user would be able to use a pin on a map of the US to select their location. This would be done through an existing library that would allow us to take a pin on a map and produce a latitude and longitude.
Dependencies: FR2.1 and FR2.2

4 Gantt Chart

Here we will place our Gantt Chart. Will get completed before Friday.