Farming Program

Software Requirements Specification

Version 0.2

18 December 2014

1312754 Program Manager

1307211 Lead Designer

Sanjay Lead Programmer

Thomas Tester

1311808 Tester

Prepared for

MOD003263—Software Engineering

Lecturer: Cristina Luca, Ph.D.

Fall 2014

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 12/10/14 | Version 0.1 | 1312754 | Initial documentation |
| 18/12/14 | Version 0.2 | 1312754 | First Draft : Completed Section One of the SRS. |
| 18/12/14 | Version 0.3 | 1312754 | Second Draft: Completed Section 2 of the SRS and renamed the Software Name from Farm Management System to Farming Program. |
|  |  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | <Your Name> | Lead Software Eng. |  |
|  | A. David McKinnon | Instructor, CptS 322 |  |
|  |  |  |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

1.5 Overview 1

2. General Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 <Functional Requirement or Feature #1> 3

3.2.2 <Functional Requirement or Feature #2> 3

3.3 Use Cases 3

3.3.1 Use Case #1 3

3.3.2 Use Case #2 3

3.4 Classes / Objects 3

3.4.1 <Class / Object #1> 3

3.4.2 <Class / Object #2> 3

3.5 Non-Functional Requirements 4

3.5.1 Performance 4

3.5.2 Reliability 4

3.5.3 Availability 4

3.5.4 Security 4

3.5.5 Maintainability 4

3.5.6 Portability 4

3.6 Inverse Requirements 4

3.7 Design Constraints 4

3.8 Logical Database Requirements 4

3.9 Other Requirements 4

4. Analysis Models 4

4.1 Sequence Diagrams 5

4.3 Data Flow Diagrams (DFD) 5

4.2 State-Transition Diagrams (STD) 5

5. Change Management Process 5

A. Appendices 5

A.1 Appendix 1 5

A.2 Appendix 2 5

# 1. Introduction

The Farming Program will aid the day-to-day activities of a farm manager and labourer. The program provides farm information such as the crop detail, harvest times, fertilizer doses and employment patterns, vehicles required for a harvest, storage status, container descriptions, field status, staff asignments and potential buyers. The program has been written in C# with a MS SQL database (T-SQL) on the back end.

## 1.1 Purpose

This is intended to be a live document which is constantly improved to provide the best possible developmental guidelines for the Farm Program. As it stands, the document will guide development teams in developing the Farming Program. Provide an in depth software awareness in order to better support updates to existing code or the addition of new modules to support future expansions.

## 1.2 Scope

The Farming Program is a MS SQL based program which has been designed to aid the farm personel catalog different activities. These activities are all based around the Crop class and the Harvest class. Each class clearly specifies its intended purpose by their miningful names. The current software will not provide dynamic querying capabilities and will not handle well extensive amounts of information or records. However these functionalities can be achieved due to the robust database employed and its architectural modularity.

## 1.3 Definitions, Acronyms, and Abbreviations

This document has five contributors. Each contribution will end with the contributing party’s Student Identification number or SID. Where they have contribute whole sections of this document, the SID will appear at the beginning of the section as opposed to the end. Where no SID appears it should be assumed that all members contributed.

## 1.4 References

The Program Manager <1312754> used:

Dennis, A., Wixom, B., Tegarden, D. and Dennis, A. (2009). *Systems analysis design, UML*

*version 2.0*. Hoboken, NJ: J. Wiley.

## 1.5 Overview

The software was developed to satidfy some of the needs of a farm. The program thrives to provide the farm with a way to catalog their major assets. When reviewing the code and understanding the audience is very impotant since the requirements gathered from them is the engine driving current and future development.

The following sections contain the specific requirements for the Farming Program as well as relevant planning requirements, design, implementation and testing for the current implementation. The sections will detail the different challenges or limitations a development team may encounter while executing the developmental plans outlined below and how to mitigate them with testing.

# 2. General Description

This section is not intended to provide specific requrirements, but a general overview of the basic requirements needed during the requirements gathering phase and other phases leading up to development, testing and deployment.

## 2.1 Product Perspective

As started in previous sections, the product tracks day-to-day administrative tasks done by the farm manager and the labourer of a farm. The program provides a database where both the manager and labourers can update or get updates for their daily activities. Especilfically the manager can check what crops are at what field and who is the staff assigned to take care of such crop. Additionally the labourer can see their field assignment and fertilizer requirements for the crop planted in such field.

## 2.2 Product Functions

The program is designed to store the following information in its database:

* The different types of crops cultivated with its field assignment.
* Harvest times and equipment required.
* Fertilisers to be used depending on the crop's requirements and the periods when the crop has to be treated.
* Provide a note section for more information about special one off treatments.
* Storage type and min/max temperature needed.
* The labour for sawing/planting, treating the crops and harvesting
* The Vehicles necessary for harvesting
* The Containers for storing the harvest

## 2.3 User Characteristics

Taking under consideration that the product is going to be used by farmers with little to no technologic experience, the program uses miningful classes with an easily understandable and intuitive design. There are no complicated queries and all of the information is presented as soon as the application loads.

## 2.4 General Constraints

The main constraint in this application will be the dependency implementation. The application is built with an embedded database as opposed to a dedicated SQL connection to a local/external server. This is to minimize the application footprint and maintenance for the customer.

This is based on the assumption that most of the farmers using the system will not have a dedicated Information Technology (IT) team. With this deployment method, the application becomes more portable and does not require the need of additional software installations (in this case MS SQL Server). During the implementation phase the developer team is not to use the built in wizards for creating SQL connections or DataSets. This is to keep the application’s code dynamic, reusable and undertandible to other team members.

## 2.5 Assumptions and Dependencies

There are to main assumptions for the dependencies of this software that must be addressed in this section.

* The system which will be running the application has .Net Framework 4.0 and above and is running a windows Oprationg System compatible with Visual Studio 2012 as a minimum.
* The system used for the development of this application is running Visual Studio 2012 or above with a combination of the following especifications:
  + Operating systems:
    - Windows 7 SP1 (x86 and x64)
    - Windows 8 (x86 and x64)
    - Windows Server 2008 R2 SP1 (x64)
    - Windows Server 2012 (x64)
  + Depending on the version Visual Studio, the version will install .Net 4.0 or 4.5 on the developer’s machine.
  + Architectures
    - 32-bit (x86)
    - 64-bit (x64)
  + Hardware Requirements
    - 1.6 GHz or faster processor.
    - 1 GB of RAM (1.5 GB if running on a virtual machine)
    - 10 GB (NTFS) of available hard disk space.
    - 5400 RPM hard drive.
    - DirectX 9-capable video card running at 1024 x 768 or higher display resolution.

# 3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.

Each requirement in this section should be:

* Correct
* Traceable (both forward and backward to prior/future artifacts)
* Unambiguous
* Verifiable (i.e., testable)
* Prioritized (with respect to importance and/or stability)
* Complete
* Consistent
* Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

### 3.1.4 Communications Interfaces

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 <Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

### 3.2.2 <Functional Requirement or Feature #2>

…

## 3.3 Use Cases

### 3.3.1 Use Case #1

### 3.3.2 Use Case #2

…

## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

### 3.4.2 <Class / Object #2>

…

## 3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

### 3.5.1 Maintainability

### 3.5.2 Portability

## 3.6 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

## 3.7 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

## 3.8 Other Requirements

Catchall section for any additional requirements.

# 4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

## 4.1 Sequence Diagrams

## 4.3 Data Flow Diagrams (DFD)

## 4.2 State-Transition Diagrams (STD)

# 5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

# 8. Testing

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS’s overall set of requirements.

*Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.*

# 7. Program Management