**Kong: Requirement Specification**

### Bader Albader, Charles Mezhir, Garrett Senor, Jacob Tran, Tyler Valentine

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Revision By** | **Changes Made** |
| 06/11/18 | 1.0 | Bader Albader | Document created |
| 06/18/18 | 2.0 | Bader Albader | Wrote down requirements discussed in meeting into this document. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## **Table of Contents**

### **1. Introduction**

1.1 Purpose

1.2 Scope

**2. Overall Description**

2.1 Product Perspective

2.2 Product Functions

**3. User Stories (Lab 1, Part 3)**

3.1 User Story 1

3.2 User Story 2

3.3 User Story 3

**1. Introduction**

1.1 Purpose

We are trying to produce a product to monitor and communicate useful data within a greenhouse environment in order better maintain ideal growing conditions. The product will use a webpage to display real-time and historical data in order to facilitate a healthy growing environment and better manage resources. The program will utilize hardware to monitor current conditions within the greenhouse which the user can view from the webpage. Additionally, historic data will be stored in a database so that trends within the growing environment can be analyzed. The hardware will communicate with the database through Wifi or bluetooth connectivity.

1.2 Scope

KONG-GROW will be the premiere temperature and humidity monitoring software accessible to amateur and professional growers. Its web-based design will provide an affordable and user friendly approach to maintaining and optimizing greenhouse environments as well as maximizing profits through efficient resource management.

**2. Overall Description**

2.1 Product perspective

**System Interfaces**:

|  |  |  |  |
| --- | --- | --- | --- |
| System Requirement | Input | Software Transformation | Output |
| Hardware connectivity to wifi | Microcontroller. | C, Python | Wifi connectivity. |
| Data transmitted to database. | Sensor data | SQL, NodeJS | Data stored in database. |
| Data displayed on front end website. | Database information. | HTML, CSS | Data shown in frontend. |

**User interfaces** :

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Display Layout | Description | Constraints |
| Front end | Website | Will contain a login system and data from temperature and humidity sensor. | Locations to test project. |
| Backend + Middle Layer | SQL, NodeJS | Will store data information of login information and temperature + humidity information. | Size of database and speed to data acquisition. |
| Hardware | No display | Will track and manage data and send the data over wifi to the database. | Testing and accuracy of data acquisition. |

2.2 Product functions

|  |  |
| --- | --- |
| Priority Level | Function |
| **HIGH** | Hardware connectivity to wifi. |
|  | Hardware data acquisition. |
|  | Database |
| **MEDIUM** | Front end website functionalities. |
| **LOW** | Website looking pretty. |

**3. User Stories (Lab 1, Part 3)**

3.1 User Story 1

Farmer wants to monitor the temperature of his greenhouse in real time to minimize losses due to under-managed temperatures

Ranking among stories: 1 (for first round) changed to 2 after debate

3.2 User Story 2:

Systems admin who wants a way to encrypt a password database

Ranking among stories: 2 (for first round) changed to 1 after debate

* + - Reached a consensus after it was pointed out that public algorithms can be used.

3.3 User Story 3:

Computer science professor who wants a way to check to see if students are submitting plagiarized code.

Ranking among stories: 3