Software Requirements Specification

For

Chariot

Submitted by

Chariot Dev

|  |  |
| --- | --- |
| **Instructor:** | Dr. Gregory Hislop |
| **Team Members:** | Medard Azandegbe, Kamalludin Colaire, Ryan Hassing, John Ho, Juan Garcia Lopez, Christopher Mak, Enioluwa Segun |
| **Cycle:** | 3 |
| **Date Submitted:** | 4/11/2020 |

Grading Rubric - Requirements Specification

This rubric outlines the grading criteria for this document. Note that the criteria represent a plan for grading. Change is possible, especially given the dynamic nature of this course. Any change will be applied consistently for the entire class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achievement** | **Minimal** | **Exemplary** | **Pts** | **Score** |
| **Content** | Section(s) missing, not useful, inconsistent, or wrong. | Provides all relevant information correctly and with appropriate detail |  |  |
| Introduction |  |  | 10 |  |
| Overall Description |  |  | 10 |  |
| Functions and Constraints |  |  | 40 |  |
| Logical Database |  |  | 20 |  |
| **Grammar and Spelling** | Many serious mistakes in grammar or spelling | Grammar, punctuation, and spelling all correct | 10 |  |
| **Expression** | Hard to follow or poor word choices | Clear and concise. A pleasure to read | 5 |  |
| **Tone** | Tone not appropriate for technical writing | Tone is consistently professional |  |  |
| **Organization** | Information difficult to locate | All information is easy to find, and important points stand out | 5 |  |
| **Layout** | Layout is inconsistent, visually distracting, or hinders use | Layout is attractive, consistent, and helps guide the reader |  |  |
| **Late Submission** |  |  |  |  |
| **Total** |  |  | 100 |  |

# Contents

[Contents 2](#_Toc37490781)

[1 Introduction 4](#_Toc37490782)

[1.1 Purpose 4](#_Toc37490783)

[1.2 Definitions 4](#_Toc37490784)

[2 Overall Description 4](#_Toc37490785)

[2.1 Product Functions 5](#_Toc37490786)

[2.2 User Characteristics 5](#_Toc37490787)

[2.2.1 Developer 5](#_Toc37490788)

[2.2.2 Technician 5](#_Toc37490789)

[3 Specific Requirements 6](#_Toc37490790)

[3.1 Functional Requirements 6](#_Toc37490791)

[FR1 – Installation 6](#_Toc37490792)

[FR2 – Running Chariot 6](#_Toc37490793)

[FR3 – Authentication 7](#_Toc37490794)

[Network Management 7](#_Toc37490795)

[FR4 – Networks 7](#_Toc37490796)

[FR5 – Manage Networks 7](#_Toc37490797)

[FR6 – Network Deletion 8](#_Toc37490798)

[FR7 – Network Configurations 8](#_Toc37490799)

[IoT Device Management 8](#_Toc37490800)

[FR8 – IoT Device Configuration 8](#_Toc37490801)

[FR9 – Adding IoT Devices 9](#_Toc37490802)

[FR10 – IoT Device Removal 9](#_Toc37490803)

[FR11 – Unintentional IoT Device Removal During a Data Collection Episode 10](#_Toc37490804)

[FR12 – Intentional IoT Device Removal During a Data Collection Episode 10](#_Toc37490805)

[FR13 – IoT Device Status 10](#_Toc37490806)

[Data Collection 10](#_Toc37490807)

[FR14 – Data Collection Episode (DCE) 10](#_Toc37490808)

[FR15 – IoT Device Data Collection 11](#_Toc37490809)

[FR16 – Data Collection Episode Configuration 12](#_Toc37490810)

[FR17 – Concurrent Network Data Collection 12](#_Toc37490811)

[Data Access 12](#_Toc37490812)

[FR18 – Database Storage 13](#_Toc37490813)

[FR19 – Database Configuration 13](#_Toc37490814)

[FR20 – Timestamped Data 13](#_Toc37490815)

[Data Output 13](#_Toc37490816)

[FR21 – Data Output 13](#_Toc37490817)

[FR22 – Data Output Receiver 14](#_Toc37490818)

[FR23 – Visualizer 14](#_Toc37490819)

[3.2 Nonfunctional Requirements 14](#_Toc37490820)

[NFR1 – Extensibility 14](#_Toc37490821)

[NFR2 – Deployment 15](#_Toc37490822)

[NFR3 – Data Integrity 15](#_Toc37490823)

[NFR4 – Real-Time Data Collection 15](#_Toc37490824)

[NFR5 – Error Handling 15](#_Toc37490825)

[NFR6 – Security 15](#_Toc37490826)

[NFR7 – Documentation 16](#_Toc37490827)

[3.3 Data Requirements 17](#_Toc37490828)

[3.4 Design Constraints 18](#_Toc37490829)

[DC1 – Graphic User Interface 18](#_Toc37490830)

[DC2 – External Library Usage 18](#_Toc37490831)

[DC3 – LGPL Licensing 18](#_Toc37490832)

[DC4 – Networking Standard Compliance 18](#_Toc37490833)

[DC5 – Adaptation of Drexel Wireless’ IoT Sensor Framework 18](#_Toc37490834)

[4 References 19](#_Toc37490835)

# Introduction

## Purpose

This document shall provide a description and the requirement specifications for Chariot, an IoT sensor framework for the development and operation of heterogeneous sensor networks. Chariot is built upon the foundations provided by Dr. Mongan’s IoT Sensor Framework, but Chariot will expand and adapt from it and other IoT Sensor Frameworks. This document shall outline the requirements that were specified by stakeholders and the product development team for the development and review of the final product.

## Definitions

This section describes technical terms commonly used in the Software Requirements Specification.

1. **Internet of Things (IoT)** - the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data.
2. **Core Component** – components of Chariot that provide core functionality for Chariot to perform its tasks.
3. **IoT device** – a hardware device that records data about its environment and has the capability to connect to a network and communicate with other devices.
4. **Network** - a collection of configured, connected IoT devices
5. **Initialized Network –** a network that exists within the system with connected IoT devices that is ready for use.
6. **Data Collection Episode** **(DCE)** – the timeframe during which Chariot is actively collecting and storing data received from IoT devices.
7. **Received Data** – data transmitted, or in transmission from an IoT device that has not yet been stored.
8. **Collected Data** – data received from an IoT device that has been saved to a storage unit
9. **Database Writer** – an adapter that connects Chariot to a database management system
10. **Data Output Receiver** – an external program or service that connects to Chariot and receives outputted data.
11. **Database Management System (DBMS)\_-** a database program and its associated APIs

# Overall Description

The Internet of Things (IoT) is a system of connected devices capable of collecting and transferring data. Different users of IoT devices may be interested in different types of data. A gardener might want to track the pH value of their soil and the air humidity. A city planner may look for a way to track pedestrian traffic in common spaces. A factory manager might look for a way to track the status of machinery and production rates.

IoT infrastructure has exploded in the past decade, so there are devices on the market to collect data for a wide variety of applications. But the wide variety of devices requires special considerations. A Wi-Fi 4K camera connects to a network much differently than a home thermometer. A motion detector may collect and transfer data in a different form than a pressure sensor.

So, despite progress in the IoT industry, it is still difficult to integrate different types of devices into a unified system. Chariot solves this problem. Chariot is a platform for IoT device management, data storage, and analysis. Users can add, remove, and configure a variety of devices on a single network. With Chariot, a gardener could purchase a pH and humidity IoT devices off the shelf and gather data from his soil and the air from one convenient interface.

## Product Functions

Chariot will provide the following functions:

* Easy addition and removal of IoT devices on a network
* Configuration of devices
* Simultaneous usage of heterogeneous devices on a single network
* Real-time data collection
* Device status reporting
* Flexible data export, including a web API
* Data analysis tools for data visualization and processing
* Modular expansion

## User Characteristics

This section identifies and describes the user types Chariot is designed for.

### Developer – A user within depth familiarity with the system and its underlying code. They will have the knowledge to develop new IoT device adapters, database writers, and data output receiver. They can also provide improvements to existing Chariot Modules.

### Technician– This user does not need to interact with the system on a deep technical level or have knowledge of all aspects of the system. But they will need to have some knowledge with the devices they use and be able to troubleshoot hardware issues.

# Specific Requirements

The below table describes the priorities of each requirement.

|  |  |
| --- | --- |
| **Priority** | **Description** |
| **Essential** | A requirement that is necessary for the final product to meet the stakeholder's expectations. |
| **High** | A requirement that is not required but will significantly improve the quality of the final product. Will be added after Essential requirements, time permitting. |
| **Low** | A requirement that offers minor improvements to the final product. Will be added after High requirements, time permitting. |

## Functional Requirements

### FR1 – Installation

**Priority: Essential**

**Target User: All**

Chariot shall have a single program install all components of the system to a computer.

### FR2 – Running Chariot

**Priority: Essential**

**Target User: All**

A user shall start Chariot from a computer connected to one or more IoT devices.

### FR3 – Authentication

**Priority: High**

**Target User: All**

Chariot shall have an authentication system for the management of user access.

### Network Management

The following section describes the core component of Chariot’s IoT Network Management, and its administration of collection of IoT devices.

### FR4 – Networks

**Priority: Essential**

**Target User: All**

Chariot shall group IoT devices into user-defined heterogeneous collections called networks. Users will initialize these by selecting a subset of connected devices.

#### FR4.1 - Network Initialization

**Priority: Essential**

**Target User: All**

During the initialization of a network, in addition to providing a configuration for the network, the user will also have to provide the configuration for every device added to that network. Configurations for the network and devices can be either created in Chariot or imported from a file.

### FR5 – Manage Networks

**Priority: Essential**

**Target User: All**

Users shall be able to manage initialized networks and the devices within them. This will include the addition and removal of devices, and making changes to their configurations, see **FR9 – FR12**.

### FR6 – Network Deletion

**Priority: Essential**

**Target User: All**

A user shall be able to delete a network from the list of initialized networks, removing its device configurations and data collection parameters from the system.

### FR7 – Network Configurations

**Priority: Essential**

**Target User: All**

Users shall be able to create network configurations from within Chariot. These configurations will contain information about the network itself, as well as the configurations for all devices in the network.

#### FR7.1 – Import/Export Network Configurations

**Priority: Essential**

**Target User: All**

A user shall be able to export their Network Configurations to an external file. A user can then import these files into a Chariot system.

### 

### IoT Device Management

The following section describes the core component of IoT Device Management, and how Chariot will interact with external IoT Devices.

### FR8 – IoT Device Configuration

**Priority: Essential**

**Target User: All**

Users shall be able to create configurations for IoT devices from within Chariot. These configurations will determine how the IoT device will perform, and how Chariot will interact with it.

#### FR8.1 – Export/Import IoT Device Configuration

**Priority: Essential**

**Target User: All**

A user shall be able to export their IoT Device Configurations to an external file. A user can then import these files into a Chariot system.

### FR9 – Adding IoT Devices

**Priority: Essential**

**Target User: All**

A user shall be able to add IoT devices to new or existing networks outside of a Data Collection Episode (DCE), **See FR15**. The user will either provide the configuration details for a new IoT device profile or use an existing device profile.

#### FR10.1 –Addition During a Data Collection Episode

**Priority: Low**

**Target User: All**

A user shall be able to add IoT devices to a network that is currently in a Data Collection Episode (DCE), **See FR15**.

### FR10 – IoT Device Removal

**Priority: Essential**

**Target User: All**

A user shall be able to remove a connected IoT device from an initialized network. Removal will prevent the further reception and collection of data from that IoT device onto the network.

### FR11 – Unintentional IoT Device Removal During a Data Collection Episode

**Priority: High**

**Target User: All**

In the event that Chariot loses connection to an IoT device, Chariot shall attempt to reconnect to the device before finally informing the user of the disconnection and preventing further data collection from that device until the user manually restarts the devices data collection.

### FR12 – Intentional IoT Device Removal During a Data Collection Episode

**Priority: Low**

**Target User: All**

A user can remove an IoT Device from a network during a DCE. Chariot will continue to store any previously received data and remove the IoT device from the network, stopping data collection from the device.

### FR13 – IoT Device Status

**Priority: High**

**Target User: All**

Chariot shall record and make accessible the status of all IoT devices on a network within a device defined polling rate. A device status will include the device ID, uptime, resource utilization, and other information pertinent to the model of device.

### Data Collection

The following section describes the core component of Data Collection and the Data Collection Episode.

### FR14 – Data Collection Episode (DCE)

**Priority: Essential**

**Target User: All**

Chariot shall take received data from a network’s IoT devices and save it as collected data into a database during a DCE.

#### FR14.1 – Initialize Data Collection Episode

**Priority: Essential**

**Target User: All**

Users will initialize a DCE to receive data from a network’s connected devices and save it to a data storage unit.

#### FR14.2 – Pause Data Collection Episode

**Priority: Low**

**Target User: All**

A user shall be able to pause an ongoing DCE, suspending the receiving of data from a network’s IoT devices.

#### FR14.3 – Resume Data Collection Episode

**Priority: Low**

**Target User: All**

A user shall be able to resume a paused DCE. This will allow for the further collection of data the network’s IoT devices after it is resumed.

#### FR14.4 – Terminate Data Collection Episode

**Priority: Essential**

**Target User: All**

A user shall be able to terminate a data collection episode. This will immediately cause Chariot to stop receiving data from IoT devices and save all received data as collected data.

### FR15 – IoT Device Data Collection

**Priority: Essential**

**Target User: All**

Chariot shall receive and collect data from multiple heterogeneous IoT devices on a network during a DCE. The data collected from each device shall be uniquely identifiable and not be corrupted or manipulated by the data collection from other devices.

### FR16 – Data Collection Episode Configuration

**Priority: Essential**

**Target User: All**

A user must create a DCE configuration to define what data is to be collected from the network’s IoT devices, the storage unit to use, formatting of the collected data, and duration of the data collection episode.

#### FR16.1 –Import/Export Data Collection Episode Configuration

**Priority: High**

**Target User: All**

A user shall be able to export their DCE configuration to an external file. A user can then import these files into a Chariot system.

### FR17 – Concurrent Network Data Collection

**Priority: Low**

**Target User: All**

Chariot shall be able to receive and collect data from at least two separate networks simultaneously.

#### FR17.1 – Concurrent Network Data Collection of Network Sharing IoT Devices

**Priority: Low**

**Target User: All**

Chariot shall allow for at least two separate networks that share at least one IoT device amongst their collections to perform concurrent data collection episodes if all shared devices have the same IoT device configuration, **See FR9**.

### Data Access

The following requirements indicate how Chariot manages user access to data.

### FR18 – Database Storage

**Priority: Essential**

**Target User: All**

Users shall be able to connect Chariot to externally managed databases for the storage of collected data.

### FR19 – Database Configuration

**Priority: Essential**

**Target User: All**

Users shall be able to create configurations for database storages. These configurations will allow for the connection to a database management system and other DBMS specific configurations.

### FR20 – Timestamped Data

**Priority: Essential**

**Target User: All**

Chariot shall timestamp all collected data for proper data sorting.

### Data Output

The following requirements specify requirements for Chariot’s data output capabilities, and how data will be accessible by other external programs and systems. Examples of a data output tool include visualizers, data processing tools and algorithms.

### FR21 – Data Output

**Priority: Essential**

**Target User: All**

Chariot shall provide an output stream during a data collection episode from which users can access the received data in real time alongside data collection. Data output receivers can be connected to the stream to provide new functionality to Chariot.

### FR22 – Data Output Receiver

**Priority: High**

**Target User: All**

Users shall be able to connect data output receivers that read in data from the Data Output stream, **See FR21**, allowing for new methods of real time data manipulation and visualization.

### FR23 – Visualizer

**Priority: Essential**

**Target User: All**

Chariot shall have a visual graphing tool that will display information on all connected devices during a DCE.

#### FR23.1 – Visualizer Replay

**Priority: Low**

**Target User: All**

A user shall be able to simulate a live visualizer graphing by using existing collected data.

## Nonfunctional Requirements

The statements below define the non-functional requirements for the system.

### NFR1 – Extensibility

**Priority: Essential**

**Target User: Developer**

A developer shall be able to increase Chariot’s functionality by creating and adding IoT devices, databases, and data output receivers to a Chariot system. There shall be documentation and guidelines for what is required for each type to interact with Chariot.

### NFR2 – Deployment

**Priority: Essential**

**Target User: Technician**

Chariot’s deployment process, and usage documentation, shall be streamlined to the point where a technician described in **Section 2.2.2** can create and run a data collection using existing configuration settings, adapters, and other modules provided to them by a developer.

### NFR3 – Data Integrity

**Priority: Essential**

**Target User: All**

Chariot shall maintain integrity of collected data in such a way that Chariot does not alter values during the collection process.

### NFR4 – Real-Time Data Collection

**Priority: High**

**Target User: All**

Chariot shall collect and store data in a timely manner to allow for real time accessibility of data.

### NFR5 – Error Handling

**Priority: High**

**Target User: All**

Chariot must not crash due to the unexpected failure of an IoT Device or network. The system shall have error handling measures to ensure the continued, uninterrupted collection of data.

### NFR6 – Security

**Priority: Low**

**Target User: All**

Chariot shall minimize the available attack surface for security attacks on its internal components and allow for the encryption of data that is sent out to external services and databases.

### NFR7 – Documentation

**Priority: High**

**Target User: All**

Chariot shall be packaged with extensive internal and external documentation. Internal documentation will clearly describe how Chariot’s code functions. External documentation will describe how to operate the system.

## Data Requirements

The following table describes the various kinds of data expected to be utilized by Chariot

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Comment** |
| User information | Database | Variable | Username/password information for each user of the system, **See FR3** |
| Networks | List | Variable | List of initialized networks. **See FR4** |
| Devices on network | List | Variable | List of devices connected to a network. **See FR5** |
| Device Configuration | JSON | Variable | Configurable settings of a device and their values. **See FR9** |
| Network Configuration | JSON | Variable | Configurable settings for a network and their values. **See FR7** |
| Database Configuration | JSON | Variable | Configurable settings for connection to a database. **See FR19** |
| Data Collection Configuration | JSON | Variable | Configurable settings for a Data Collection Episode, including the associated network and database. **See FR16** |
| Saved Configurations | Database | Variable | Database of all locally saved or imported configurations. |
| Device status | Stream | Variable | Uptime, resource utilization, and other status information pertinent to device. Different from Device information, as Device status will frequently change. **See FR16** |
| Device data stream | Stream | Variable | Data received from device sensors. |
| Network Configuration | JSON |  | Configurable settings for a network. |
| Data collection episode status | Integer |  | Indicates if data collection episode is in progress, paused, or terminated. Also indicates errors in data collection. |
| Collected data | Various | Variable | Incoming data from data analysis episode is written to a database. Previously collected data can also be loaded from a file. The type is dependent on the storage unit used by the data collection episode. |
| Output Source | Stream | Variable | An output stream that can be connected to by an external service or program to receive data from Chariot. |

## Design Constraints

The following section specifies the design constraints restricting design of the system.

### DC1 – Graphic User Interface

The user shall be able to use the system via a graphic user interface. Stakeholders have expressed frustration using the command line for tasks this system would accomplish, so a graphic user interface is preferred.

### DC2 – External Library Usage

Chariot will make use of external libraries to interface with IoT devices. The system will use these libraries as specified.

### DC3 – LGPL Licensing

Chariot shall be licensed with the GNU Lesser General Public License (LGPL) (Free Software Foundation, Inc., 2007). Due to the potential use of proprietary IoT Devices, Databases, and Data Output Receivers, we cannot guarantee that all components of Chariot are modifiable. But the core components of the system will be made modifiable by end users.

### DC4 – Networking Standard Compliance

Chariot shall comply will need to be able to communicate with IoT devices using various networking standards such as, IEEE 802.3, 802.11 and continue to develop for newer networking technologies such as LoRa. (Institute of Electrical and Electronics Engineers, 2018).

### DC5 – Adaptation of Drexel Wireless’ IoT Sensor Framework

Chariot will use and adapt related work in the field of IoT data collection. The system will use several key features of Dr. Bill Mongan’s IoT Sensor Framework (Mongan, 2019). Chariot will adhere to the conventions of original work and will document where work is used or adapted.

# References

Free Software Foundation, Inc. (2007, June 29). *GNU Lesser General Public License*. Retrieved from GNU Operating System: https://www.gnu.org/licenses/lgpl-3.0.en.html

Institute of Electrical and Electronics Engineers. (2018, November 14). *IEEE Standards Activities in the Internet of Things.* Retrieved from IEEE Standards Association Web site: https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/other/iot.pdf

Mongan, W. (2019). *IoT Sensor Framework*. Retrieved from William Mongan portfolio: http://www.billmongan.com/portfolio/iotframework/