Software Design Specification

For

Chariot

Submitted by

Medard Azandegbe, Kamalludin Colaire,   
Ryan Hassing, John Ho, Juan Garcia Lopez, Christopher Mak, Enioluwa Segun

|  |  |
| --- | --- |
| **Instructor:** | Dr. Gregory Hislop |
| **Team Members:** | 7 |
| **Cycle:** | 2 |
| **Date Submitted:** |  |

Document template copyright 2005-2019, Gregory W. Hislop. Version 2.3. Use permitted under Creative Commons license CC-BY-NC-SA. See http://creativecommons.org/licenses/by-nc-sa/3.0/.

Grading Rubric - Design 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 |  |
| Architectural Description |  |  | 10 |  |
| Interface Description |  |  | 15 |  |
| Detailed Design |  |  | 50 |  |
| **Grammar and Spelling** | Many serious mistakes in grammar or spelling | Grammar, punctuation, and spelling all correct | 5 |  |
| **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 |  |

Table of Contents

[1 Introduction 5](#_Toc402170391)

[1.1 Scope 5](#_Toc402170392)

[1.2 Definitions, Acronyms, and Abbreviations 5](#_Toc402170393)

[1.3 Requirements Traceability Matrix 5](#_Toc402170394)

[2 Architectural Description 6](#_Toc402170395)

[2.1 Component <ID 1> Decomposition 6](#_Toc402170396)

[2.2 Component <ID 2> Decomposition 6](#_Toc402170397)

[3 Interface Description 6](#_Toc402170398)

[3.1 User Interface 6](#_Toc402170399)

[3.2 Data Interface 6](#_Toc402170400)

[3.3 Programming Interface 6](#_Toc402170401)

[4 Detailed Design 6](#_Toc402170402)

[4.1 <Entity ID> – <Entity Name> (<writer>) 6](#_Toc402170403)

Introduction

## Scope

This document describes the design and implementation details of Chariot, an IoT device framework. The framework is largely based on Dr. Bill Mongan’s IoT sensor framework, with several key improvements to allow for more general use cases. The design of Chariot is meant to satisfy the requirements outlined in the Software Requirements Specification.

## Definitions, Acronyms, and Abbreviations

The following definitions, acronyms, and abbreviations appear multiple times throughout this document.

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. 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.
3. Data Collection Episode – the timeframe during which Chariot is actively collecting and storing data received from IoT devices
4. Received Data – data transmitted, or in transmission from an IoT device that has not yet been stored
5. Collected Data – data received from an IoT device that has been saved to a storage unit
6. Storage Unit – where collected data is saved, be it a database, csv file, or other format or data structure
7. Network - a collection of configured, connected IoT devices
8. Module – an addon to Chariot that adds functionality or compatibility tools to the system
9. UI – User Interface, a means for a user to interact with a system to interact with one another.

## Requirements Traceability Matrix

This section maps the relationship between requirement statements and detailed design entities. As such it shows how requirements are covered by the design, and demonstrates the purpose for which design entity exists.

The values in the cells of the table show which requirements provide the purpose for each entity. The cell values are:

* **Blank** – the design entity does not implement any of that requirement
* **P** for Primary - the design entity implements all or most of the requirement
* **S** for Secondary – the design entity implements a smaller but essential part of the requirement

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **<DE1>** | **<DE2>** | **Etc.** |  |  |  |  |  |
| **<Req ID1>** |  |  |  |  |  |  |  |  |
| **<Req ID2>** |  |  |  |  |  |  |  |  |
| **<Req ID3>** |  |  |  |  |  |  |  |  |
| **Etc** |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Figure 1 – Traceability between Requirements and Design Entities

# Architectural Description

## Component <ID 1> Decomposition

## Component <ID 2> Decomposition

# Interface Description

## User Interface

## Data Interface

This section defines the data transactions used between Chariot components, and with external systems.

### Device data stream

Type: stream

Description:

An external IoT device may output data in any number of ways. But Chariot needs a unified model for data to synchronize different types of devices. This is the motivation for the following data format.

A single data point is represented by a JSON object with three fields: deviceID, time, and data. data is itself a JSON object containing key-value pairs with data labels and values. For example, a single data point from a temperature-humidity sensor would be represented as follows:

{

“deviceID”: “1”,

“time”: “14.05”,

“data”: {

“temp”: “24.5”,

“humidity”: “.34”

}

}

A data stream, then, is simply a sequence of data points with the same deviceID and ascending time values.

### Network data stream

Type: stream

Description:

A network data stream associates data from multiple devices by time. Consider a network composed of the temperature-humidity sensor described above, as well as a pH sensor with the following representation:

{

“deviceID”: “2”,

“time”: “14.05”,

“data”: {

“pH”: “8.00”

}

}

Then the network data stream would associate these data points in the following format:

{

“time”: “14.05”,

“data”: {

“1”: {

“temp”: “24.5”,

“humidity”: “.34”

},

“2”: {

“pH”: “8.00”

}

}

}

A network data stream is a sequence of such objects with ascending time values.

### Device configuration

Type: Config file

Description:

Different devices initialize and terminate data collection in unique ways. Incoming data may also be formatted differently. To handle these disparities, each IoT device will have a device configuration associated with it. The device configuration will have the following format:

{

“deviceID”: “1”,

“start”: “devicescripts/device1/start.sh”,

“stop”: “devicescripts/device1/stop.sh”,

“dataFormat”: “regex”,

“socket”: “50.60.70.80:8888”,

“activeKeys”: {

“key1”: “1”,

“key2”: “0”,

...

}

}

start and stop are scripts that initialize and terminate data collection. dataFormat is a regular expression that transforms the device’s default data output to the device data format described above. socket is the socket address of the device. activeKeys is a list of boolean values indicating whether to record each data element.

## Programming Interface

# Detailed Design

## <Entity ID> – <Entity Name>

**Type**: <Type Name>

**Description**:

<Description text here>