Detailed Design Specification TMMS

Team: TK Force

Team Members:
Richard Sherrill
David Odhiambo
Huadong Feng
Robert Castillo
Table of Contents

1 Introduction .......................................................................................................................... 8
  1.1 Document Overview .................................................................................................... 8
  1.2 Product Overview ........................................................................................................ 8
  1.3 Project Scope ................................................................................................................ 8

2 Architecture Overview ....................................................................................................... 9
  2.1 Input layer ..................................................................................................................... 10
  2.2 Data Controller layer .................................................................................................... 11
  2.3 Database layer: ............................................................................................................. 11
  2.4 Match Event Processor: .............................................................................................. 11
  2.5 Presentation layer: ...................................................................................................... 11

3 Input Layer ....................................................................................................................... 12
  3.1 Description .................................................................................................................. 12
  3.2 Admin Input: Keyboard/Mouse Input .......................................................................... 12
  3.3 Equipment Input: Filter ............................................................................................... 13
  3.4 Equipment Input: Equipment Input ............................................................................ 14
  3.5 Admin Package ........................................................................................................... 14
  3.6 Force Data Package ..................................................................................................... 15

4 Data Controller Layer ...................................................................................................... 17
  4.1 Description .................................................................................................................. 18
  4.2 Input Controller: Object Identifier module ............................................................... 18
  4.3 Database Controller: DB Store .................................................................................. 19
  4.4 Database Controller: Log Creator ............................................................................. 20
  4.5 Database Controller: Request Log ............................................................................ 21
  4.6 Match Event Controller: Timestamp ......................................................................... 22
  4.7 Match Event Controller: Time Module ....................................................................... 23
4.8 Match Event Controller: Match Log ........................................................... 24
4.9 Game Output Controller: Game Output Creator ........................................ 25

5 Database Layer .......................................................................................... 27
5.1 Description ............................................................................................... 27
5.2 Database Manager: Store Contestant ....................................................... 27
5.3 Database Manager: Store Log ................................................................. 28
5.4 Database Manager: Retrieve Match Log .................................................. 29

6 Match Event Processor Layer ....................................................................... 31
6.1 Description ............................................................................................... 31
6.2 Match Event Processor: Admin Match Data ............................................. 31
6.3 Match Event Processor: Force Data ......................................................... 32
6.4 Match Event Processor: Match Event ....................................................... 33
6.5 Calibration: Determine Adjustment Factor ................................................. 34
6.6 Force Calculations: Score Modifiers ......................................................... 35

7 Presentation Layer ....................................................................................... 37
7.1 Description ............................................................................................... 37
7.2 Output Handler: Determine Display Command ......................................... 37
7.3 Admin Display: Update Admin Display .................................................... 38
7.4 Audience Display: Update Audience Display .......................................... 39

8 Quality Assurance ......................................................................................... 41
8.1 Test Plan and Procedures ......................................................................... 41
8.2 Module/Unit Test ...................................................................................... 41
8.3 Component Testing ................................................................................. 45
8.4 Integration Testing .................................................................................... 46
8.5 System Verification Testing ...................................................................... 46
8.6 Test Cases .................................................................................................. 46

9 Requirements Mapping ................................................................................. 48
9.1 Purpose ...................................................................................................... 48
9.2 Layer Requirements Traceability .............................................................. 48
9.3 Module Requirements Traceability ............................................................ 50

10 Acceptance Plan .......................................................................................... 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Overview</td>
<td>52</td>
</tr>
<tr>
<td>10.2</td>
<td>Packaging and Installation</td>
<td>52</td>
</tr>
<tr>
<td>10.3</td>
<td>Acceptance Testing</td>
<td>52</td>
</tr>
<tr>
<td>10.4</td>
<td>Acceptance Criteria</td>
<td>53</td>
</tr>
<tr>
<td>11</td>
<td>Hardware Components</td>
<td>58</td>
</tr>
<tr>
<td>11.1</td>
<td>Overview</td>
<td>58</td>
</tr>
<tr>
<td>11.2</td>
<td>EPRO 2910 Daedo E-Hogu w/out transmitter</td>
<td>58</td>
</tr>
<tr>
<td>11.3</td>
<td>EPRO 2913 Daedo E-Headgear w/out transmitter</td>
<td>59</td>
</tr>
<tr>
<td>11.4</td>
<td>EPRO 2903 Daedo E-Footgear</td>
<td>60</td>
</tr>
<tr>
<td>11.5</td>
<td>EPRO 2909 Daedo E-Hogu &amp; E-Headgear Wireless Transmitter</td>
<td>61</td>
</tr>
<tr>
<td>11.6</td>
<td>EPRO 2906 Daedo USB Wireless Receiver</td>
<td>62</td>
</tr>
</tbody>
</table>
## Document Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision Date</th>
<th>Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>9/23/13</td>
<td>DDS Draft</td>
<td>Compilation of team DDS sections</td>
</tr>
<tr>
<td>2.0</td>
<td>10/4/13</td>
<td>Baseline DDS</td>
<td>Corrections and formatting</td>
</tr>
</tbody>
</table>
List of Tables

Table 3-1: Input Layer Data Flows ................................................................. 12
Table 4-1: Data Controller Data Flows .......................................................... 17
Table 5-1: Database Data Flows ................................................................. 27
Table 6-1: Match Event Processor Data Flows ........................................... 31
Table 7-1: Presentation Data Flows ............................................................. 37
Table 8-1: Test Cases .................................................................................. 46
Table 9-1: Customer Requirements Mapping .............................................. 48
Table 9-2: Performance Requirements Mapping ......................................... 49
Table 9-3: Other Requirements Mapping ..................................................... 49
Table 9-4: Customer Requirements Mapping ............................................. 50
Table 9-5: Performance and Other Requirements Mapping ............................. 51
Table 10-1: Audience Screen Verification .................................................... 53
Table 10-2: Real Time Verification ................................................................ 54
Table 10-3: Save Log Data Verification ....................................................... 55
Table 10-4: User Friendliness Verification .................................................... 55
Table 10-5: Force Accuracy Verification ....................................................... 57
List of Figures

Figure 2-1: Architecture Layout Diagram ................................................................. 9
Figure 2-2: Detail Design Layout Diagram ................................................................. 10
Figure 3-1: Input Layer Module Diagram ................................................................. 12
Figure 4-1: Data Controller Layer Module Diagram ............................................... 17
Figure 5-1: Database Layer Module Diagram .......................................................... 27
Figure 6-1: Match Event Processor Layer Module Diagram ..................................... 31
Figure 7-1: Presentation Layer Module Diagram ...................................................... 37
Figure 11-1: EPRO 2910 Daedo E-Hogu w/out transmitter ..................................... 58
Figure 11-2: EPRO 2913 Daedo E-Headgear w/out transmitter .............................. 59
Figure 11-3: EPRO 2903 Daedo E-Footgear ............................................................. 60
Figure 11-4: EPRO 2909 Daedo E-Hogu & E-Headgear Wireless Transmitter .......... 61
Figure 11-5: EPRO 2906 Daedo USB Wireless Receiver ........................................... 62
1 Introduction

1.1 Document Overview

The Detail Design Specification (DDS) is intended to provide a detailed breakdown of the individual modules in each subsystem defined in the Architecture Design Specification (ADS). Also, this document will provide data flow definitions between the modules, requirements traceability matrix, and testing considerations for the system.

1.2 Product Overview

The Taekwondo Match Management System (TMMS) will be used to manage a match between two players competing in a Taekwondo competition and display parts of this information to an audience. The TMMS will allow administrators to input competitor information and store it in an internal database as well as display this information back to the administrator and an audience. The TMMS will be started from a computer and will communicate with Daedo equipment that two competitors wear for protection. The Daedo equipment is outside of our system, but captures the force of hits and relays that information to the TMMS. The TMMS then uses this information to apply force deductions to a life bar which is displayed to the administrators and an audience. The TMMS also uses the force information to test the equipment using a test mode built into the TMMS which displays the amount of force to the administrators. The system will also allow administrators to apply penalties, set score modifier deductions, start medical timeouts for the system, and pull up match event information for an active competition.

1.3 Project Scope

The TMMS is a software management system that will receive data from Daedo equipment worn by two competitors engaging in a Taekwondo match. The system is a pure software system and has no external parts that make up the system, but must receive data from Daedo equipment in order to execute all of its functions. The main functions of the system are keeping life bars of two competitors based off of force deductions and ensuring that the system can keep track of both competitors in a reliable manner.
2 Architecture Overview

The architecture contains five layers each layer being independent of any other layer in their functionality. Each layer must contain a clear interface and must exchange data in a consistent manner between layers. The five layers are the Input layer, the Data Controller layer, the Database layer, the Match Event Processor layer, and the output layer. The architectural layers are displayed below.

Figure 2-1: Architecture Layout Diagram
2.1 **Input layer**

The Input layer will receive input from an administrator’s screen and the Daedo equipment. The administrator’s screen will enter data such as competitor information, setup of the system, and altering the system. The Daedo equipment will send raw data that represents the force of impact on each competitor’s equipment and the TMMS will receive this data. This layer will then determine the type of data it received and package the data for transfer to the Data Controller layer. The packaged data is data that is relevant to the next layer.
2.2 Data Controller layer

The Data Controller layer will be responsible for routing packaged data received from the administrator’s screen, from the Daedo equipment, and any received data from the database to its correct destination. The Data Controller layer is also responsible for holding data that has been received from the Input layer or the database, and packaging information received from the Database layer or the Match Event Processor layer. The Data Controller layer will then make sure that all the relevant information is packaged together to send to the output layer.

2.3 Database layer:

The Database layer will be responsible for receiving input from the database handler. The database manager inside the Database layer will then decide if this information needs to be stored or information needs to be retrieved based off of the data that has been sent to it from the database handler. The database manager will then create these queries, execute the queries, receive the result of the queries, and then package and send the data back to the Input layer.

2.4 Match Event Processor:

The match event processor will be responsible for changing the state of a match. Any raw force data that has been received from the Daedo equipment will pass through this layer in order to be converted to force data that can be read by the game output packager in the Data Controller layer. This layer will also add any score modifiers to that force data in order to correctly represent the state of each competitor’s life bar.

2.5 Presentation layer:

The Presentation layer will be responsible for correctly displaying the data that has come from the Data Controller layer. This data will be force deductions to the life bar, administration input data such as information about a competitor, and different modes of the TMMS such as test mode. The Presentation layer will also be responsible for making sure that the appropriate information is sent to the administrator screen and audience screen.
3 Input Layer

![Input Layer Module Diagram](image)

Figure 3-1: Input Layer Module Diagram

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>Keyboard and mouse entries from the Admin PC</td>
</tr>
<tr>
<td>I2</td>
<td>Force data and Equipment ID</td>
</tr>
<tr>
<td>I3</td>
<td>Text fields from UI (Contestant and State settings), Admin commands</td>
</tr>
<tr>
<td>I4</td>
<td>Equipment ID</td>
</tr>
<tr>
<td>I5</td>
<td>Force data and Equipment ID</td>
</tr>
<tr>
<td>I6</td>
<td>Force Object</td>
</tr>
<tr>
<td>I7</td>
<td>Penalty string</td>
</tr>
<tr>
<td>I8</td>
<td>Admin Event Object, Contestant Object, State Object</td>
</tr>
<tr>
<td>I9</td>
<td>Force Object</td>
</tr>
</tbody>
</table>

3.1 Description

The Input Layer will be responsible for interacting with the Daedo Defense Equipment and the Admin PC. It will also create and send objects to the Data Controller Layer.

3.2 Admin Input: Keyboard/Mouse Input

3.2.1 Prologue

This module receives all keyboard and mouse input from the Admin PC. This will create an Object with attributes representing the command or text field from the GUI.
3.2.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin PC (physical device)</td>
<td>Keyboard/Mouse Input</td>
<td>Keyboard and Mouse events with commands or text objects</td>
</tr>
<tr>
<td>Keyboard/Mouse Input (Admin Input subsystem)</td>
<td>Admin Package (Input Packager subsystem)</td>
<td>Text objects or Admin command</td>
</tr>
</tbody>
</table>

3.2.3 External Data Dependencies
This module requires a keyboard and mouse attached to the Admin PC.

3.2.4 Internal Data Descriptors
This module will send text objects or a command to the next module.

3.2.5 Pseudo-Code

```csharp
private void button_Click(object sender, EventArgs e) {
    // calls user created method to identify the click
    objectIdentifier(button.Value);
}
```

3.3 Equipment Input: Filter

3.3.1 Prologue
This module acts as a filter for the equipment IDs. It will receive an equipment ID from the Admin Package module. This will allow our system to only focus on receiving inputs from the desired equipment.

3.3.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin Package</td>
<td>Filter</td>
<td>Equipment ID string</td>
</tr>
<tr>
<td>Defense Equipment (physical device)</td>
<td>Filter</td>
<td>Force data string and equipment ID</td>
</tr>
<tr>
<td>Filter</td>
<td>Equipment Input</td>
<td>Force data string and equipment ID</td>
</tr>
</tbody>
</table>

3.3.3 External Data Dependencies
This module requires the Daedo Defense Equipment.

3.3.4 Internal Data Descriptors
Force data string and equipment ID string will be sent to the next module.
3.3.5 Pseudo-Code

```java
setEquipmentFilter(string id);
```

3.4 Equipment Input: Equipment Input

3.4.1 Prologue
This module will receive the equipment ID and force data from the Filter module and create a Force Data Object.

3.4.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>Equipment Input</td>
<td>Sends Equipment Object</td>
</tr>
<tr>
<td>Equipment Input</td>
<td>Force Data Packager</td>
<td>Sends Equipment Object</td>
</tr>
</tbody>
</table>

3.4.3 External Data Dependencies
This module requires an Equipment Object that contains the equipment ID and type.

3.4.4 Internal Data Descriptors
This module will send an Equipment Object to the next layer

3.4.5 Pseudo-Code

```java
private void setEquipment(equipmentID, equipmentType) {
    ForceData forceData = new ForceData();
    forceData.equipmentID = equipmentID;
    forceData.equipmentType = equipmentType;
}
```

3.5 Admin Package

3.5.1 Prologue
This module will receive text data and admin commands that will be used to create an Admin Event object (which will contain the Match State Object), send equipment IDs, and send a penalty command to the Force Data Package.
3.5.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard/Mouse Input</td>
<td>Admin Package</td>
<td>Text data and admin commands</td>
</tr>
<tr>
<td>Admin Package</td>
<td>Force Data Package</td>
<td>Penalty type as a string</td>
</tr>
<tr>
<td>Admin Package</td>
<td>Filter</td>
<td>Equipment Object</td>
</tr>
<tr>
<td>Admin Package</td>
<td>Object Identifier</td>
<td>Admin Event Object, State Object, Contestant Object</td>
</tr>
</tbody>
</table>

3.5.3 External Data Dependencies

Text data object.

3.5.4 Internal Data Descriptors

Admin Event Object, State Object, and Contestant Object

3.5.5 Pseudo-Code

```
AdminEvent adminEvent = new AdminEvent();
```

3.6 Force Data Package

3.6.1 Prologue

This module creates a Force Data Object from the Equipment Input module data, and the Admin Event Object (if applicable).

3.6.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin Package</td>
<td>Force Data Package</td>
<td>Penalty type as a string</td>
</tr>
<tr>
<td>Equipment Input</td>
<td>Force Data Package</td>
<td>Equipment Object</td>
</tr>
<tr>
<td>Force Data Package</td>
<td>Object Identifier</td>
<td>Force Data Object</td>
</tr>
</tbody>
</table>

3.6.3 External Data Dependencies

Penalty string and Equipment Object.

3.6.4 Internal Data Descriptors

Force Data Object that contains the force of a strike, the equipment ID, and the Penalty enumerated type.
3.6.5 Pseudo-Code

class ForceData {
    string force;
    string equipmentID;
    string equipmentLocation;
    enum penaltyType;
}
4 Data Controller Layer

![Diagram of Data Controller Layer Module Diagram](image)

**Figure 4-1: Data Controller Layer Module Diagram**

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC1</td>
<td>Contestant object containing contestants first and last name and equipment ID</td>
</tr>
<tr>
<td>DC2</td>
<td>Match Event object containing all the events that occurred during the match</td>
</tr>
<tr>
<td>DC3</td>
<td>State object containing all information for setting up a match</td>
</tr>
<tr>
<td>DC4</td>
<td>Force object containing integer force data and equipment IDs</td>
</tr>
<tr>
<td>DC5</td>
<td>Force object containing integer force data and equipment IDs</td>
</tr>
<tr>
<td>DC6</td>
<td>Timestamp as a string which represents the current time</td>
</tr>
<tr>
<td>DC7</td>
<td>Stack object containing Match Event objects</td>
</tr>
<tr>
<td>DC8</td>
<td>Timestamp to be displayed in the presentations log</td>
</tr>
<tr>
<td>DC9</td>
<td>Match Event to be displayed in the presentations log form</td>
</tr>
<tr>
<td>DC10</td>
<td>Contestant Object containing contestants first and last name and equipment ID</td>
</tr>
<tr>
<td>DC11</td>
<td>Game Output object containing data that must be updated in the Presentation layer</td>
</tr>
<tr>
<td>DC12</td>
<td>Admin Event object containing commands that will start new forms</td>
</tr>
<tr>
<td>DC13</td>
<td>Admin Event object containing a request for the match log with the match ID</td>
</tr>
<tr>
<td>DC14</td>
<td>Match ID as an integer to be retrieved from the database</td>
</tr>
<tr>
<td>DC15</td>
<td>Contestant object containing contestants first and last name and equipment ID</td>
</tr>
<tr>
<td>DC16</td>
<td>Match Log object containing all of the match events that occurred during a specific match</td>
</tr>
<tr>
<td>DC17</td>
<td>Time object containing the updated time to be displayed in the presentation layer</td>
</tr>
</tbody>
</table>

**Table 4-1: Data Controller Data Flows**
4.1 Description
The Data Controller Layer controls the flow of data and routes it to the proper subsystem.

4.2 Input Controller: Object Identifier module

4.2.1 Prologue
This module will take in a Contestant object, State object, Admin event object, and a Force object it will then determine what kind of object it has and send that object to the appropriate module for further processing.

4.2.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin Package (Input Packager Subsystem)</td>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>Contestant object containing names, and country</td>
</tr>
<tr>
<td>Admin Package (Input Packager Subsystem)</td>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>State object containing all information for setting up a match.</td>
</tr>
<tr>
<td>Admin Package (Input Packager Subsystem)</td>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>Admin Event object containing a command for altering the GUI.</td>
</tr>
<tr>
<td>Force Data Package (Input Packager Subsystem)</td>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>Force object containing the amount of force and the equipment ID that force was applied to</td>
</tr>
<tr>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>Db Connector (Database Controller Subsystem)</td>
<td>Contestant and Admin Event objects</td>
</tr>
<tr>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>Time Controls (Match Event Processor Subsystem)</td>
<td>State, Force, Admin Event objects</td>
</tr>
<tr>
<td>Object Identifier (Input Controller Subsystem)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Contestant and Admin Event objects</td>
</tr>
</tbody>
</table>

4.2.3 External Data Dependencies:
Contestant, State, Admin Event, and Force objects
4.2.4 Internal Data Descriptors:
Contestant, State, Admin Event, and Force objects

4.2.5 Pseudo-code:

```java
public void objectIdentifier(Object object)
{
    if(object.typeOf(Contestant))
    {
        dbStore((Contestant)object);
        goCreator((Contestant)object);
    }
    else if(object.typeOf(State))
        timeModule((State)object);
    else if(object.typeOf(Force))
        timeStamp((Force)object);
    else if (object.typeOf(AdminEvent))
    {
        if (RequestLog)
            requestLog((AdminEvent)object);
        else
        {
            timeStamp((AdminEvent)object);
        }
    }
}
```

4.3 Database Controller: DB Store

4.3.1 Prologue
This module will receive the Contestant object from the input controller. After receiving the Contestant information from the input controller this module will prepare the information to be input into the database.

4.3.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
</table>

DB Store Interfaces
4.3.3 External Data Dependencies

Contestant object

4.3.4 Internal Data Descriptors

Contestant object

4.3.5 Pseudo-code

```java
public void dbStore(Contestant contestant)
{
    if(!(contestant == null))
        storeCompetitor(contestant);
    else
        Console.error("Contestant object was not created");
}
```

4.4 Database Controller: Log Creator

4.4.1 Prologue

The function it will perform is receive the stack containing the logs of events that occurred during the match. After receiving the stack this module will pop off each Match Event object from the stack to send to the Database layer to be stored in the Database.

4.4.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match Log (Match Event Controller)</td>
<td>Log Creator (Database Controller)</td>
<td>Stack object which contains all Match Events that have occurred during the match</td>
</tr>
<tr>
<td>Log Creator (Database Controller)</td>
<td>Store Log (Database Manager)</td>
<td>Match Event object which contains information for an event that occurred during a match</td>
</tr>
</tbody>
</table>
4.4.3 External Data Dependencies
Stack of Match Event objects

4.4.4 Internal Data Descriptors
Match Events

4.4.5 Pseudo-code

```java
public void logCreator(Stack matchEventStack)
{
    if(!matchEventStack.isEmpty())
    {
        //Will get Match Event objects
        storeLog(matchEventStack.pop());
    }
    else
    {
        Console.error("Match Event Stack was not created");
    }
}
```

4.5 Database Controller: Request Log

4.5.1 Prologue
The function this module will perform is to make a request to the database layer that will retrieve the match log for a specified match and then send this information to the GoCreator.

4.5.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Identifier (Input Controller)</td>
<td>Request Log (Database Controller)</td>
<td>A match ID that specifies the match log to get from the database</td>
</tr>
<tr>
<td>Request Log (Database Controller)</td>
<td>Retrieve Match Log (Database Manager)</td>
<td>A match ID that specifies the match log to get from the database</td>
</tr>
<tr>
<td>Retrieve Match Log (Database Manager)</td>
<td>Request Log (Database Controller)</td>
<td>Logged data from previous matches, received from the database</td>
</tr>
<tr>
<td>Request Log (Database Controller)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Logged information for a match</td>
</tr>
</tbody>
</table>
4.5.3 External Data Dependencies
Match ID

4.5.4 Internal Data Descriptors
Match ID

4.5.5 Pseudo-code

```java
public void requestLog(AdminEvent command)
{
    int matchID;
    Log matchLog;

    matchID = command.getMatchID();
    matchLog = retrieveMatchLog(matchID);

    gameOutputCreator(matchLog);
}
```

4.6 Match Event Controller: Timestamp

4.6.1 Prologue:
This module will get a timestamp from the timer and place a timestamp on each event that must be stored in the database.

4.6.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Identifier (Database Controller)</td>
<td>Timestamp (Match Event Controller)</td>
<td>Force object which will contain value of force and equipment IDs</td>
</tr>
<tr>
<td>Object Identifier (Database Controller)</td>
<td>Timestamp (Match Event Controller)</td>
<td>Admin commands which can contain admin stops and starts</td>
</tr>
<tr>
<td>Timestamp (Match Event Controller)</td>
<td>Admin Match Data (Match Event Processor)</td>
<td>Match State object for setting a match</td>
</tr>
<tr>
<td>Timestamp (Match Event Controller)</td>
<td>Time Module (Match Event Processor)</td>
<td>Request time from round clock</td>
</tr>
<tr>
<td>Time Module (Match Event Controller)</td>
<td>Timestamp (Match Event Controller)</td>
<td>String value of the current time</td>
</tr>
</tbody>
</table>
4.6.3 External Data Dependencies
Force object, Admin commands, and string value of time

4.6.4 Internal Data Descriptors
String timestamps

4.6.5 Pseudo-code

```java
public void timeStamp(Object object)
{
    String timestamp;
    timestamp = timeModule.getTimeStamp();
    if(object.type == TimeClock)
        forceDate(object, timestamp);
    else
        adminMatchData(object, timestamp);
}
```

4.7 Match Event Controller: Time Module

4.7.1 Prologue
This module will create a new timer for each request from the Admin such as the round clock, medical clock, and break clock. This module will also create a new thread in order to update the clock.

4.7.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Identifier (Input Controller)</td>
<td>Time Module (Match Event Processor)</td>
<td>State object that will set the round clock timer</td>
</tr>
<tr>
<td>Timestamp (Match Event Processor)</td>
<td>Time Module (Match Event Processor)</td>
<td>A request to get the current time from the round clock</td>
</tr>
<tr>
<td>Time Module (Match Event Processor)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>The updated time</td>
</tr>
</tbody>
</table>

4.7.3 External Data Dependencies
State object

4.7.4 Internal Data Descriptors
Time objects
4.7.5 Pseudo-code

```java
class TimeModule{
    Time roundClock;
    Time breakClock;
    Time medicalClock;

    public TimeModule(State clock){
        roundClock = clock.roundClock();
        medicalClock = clock.medicalClock();
        breakClock = clock.breakClock();
    }

    public String getTimeStamp()
    {
        return roundClock.getTime().toString();
    }

    public Time updateTime()
    {
        return roundClock;
    }

    public Time startMedical()
    {
        roundClock.pause();
        medicalClock.start();
        return medicalClock;
    }
}
```

4.8 Match Event Controller: Match Log

4.8.1 Prologue

This module will receive the match events from the Match Event subsystem and will create a stack of these objects to be sent to the database controller.

4.8.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match Event (Match Event Processor)</td>
<td>Match Log (Match Event Controller)</td>
<td>Match Event objects which contains a match event with its timestamp</td>
</tr>
<tr>
<td>Match Log (Match Event Controller)</td>
<td>Log creator (Database Controller)</td>
<td>Stack which contains match event objects</td>
</tr>
<tr>
<td>Match Log (Match Event Controller)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Match logs to be displayed</td>
</tr>
</tbody>
</table>
4.8.3 External Data Dependencies
Match Events object

4.8.4 Internal Data Descriptors
Stack object containing Match Event objects

4.8.5 Pseudo-code

```java
public void matchLog(MatchEvent matchEvent)
{
    Stack<MatchEvent> matchStack;
    matchStack.push(matchEvent);
    logCreator(matchStack);
    gameOutputCreator(matchEvent);
}
```

4.9 Game Output Controller: Game Output Creator

4.9.1 Prologue
This module will create an output object which will contain all of the new information to be updated in the presentation layer.

4.9.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Identifier (Input Controller)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Contestant object to send to be presented</td>
</tr>
<tr>
<td>Match Log (Match Event Processor)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Match Events to be updated in the presentation layer</td>
</tr>
<tr>
<td>Request Log (Database Controller)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Match Logs to be viewed by the presentation layer</td>
</tr>
<tr>
<td>Time Module (Match Event Processor)</td>
<td>Game Output Creator (Game Output Controller)</td>
<td>Time object to update the round clock, break clock, and medical clock</td>
</tr>
<tr>
<td>Game Output Creator (Game Output Controller)</td>
<td>Determine Display Command (Output Handler)</td>
<td>Admin Event object which changes the forms</td>
</tr>
<tr>
<td>Game Output Creator (Game Output Controller)</td>
<td>Determine Display Command (Output Handler)</td>
<td>GameOutput Object which contains the data to be updated in the form</td>
</tr>
</tbody>
</table>
4.9.3  External Data Dependencies
Match Event object, Contestant object, Match Log object, and Time object

4.9.4  Internal Data Descriptors
Game Output Object which contains data to be updated and Admin Event object

4.9.5  Pseudo-code

```java
public void goCreator(Object object)
{
    if(object instanceof AdminEvent)
        determineDisplayCommand(object);
    else
    {
        GameOutput goOutput;
        if(object instanceof Time)
            goOutput.setTime(object);
        determineDisplayCommand(object);
        else if(object instanceof Contestant)
            goOutput.setContestant(object);
        determineDisplayCommand(goOutput);
        else if(object instanceof MatchLog)
            goOutput.setMatchLog(object);
        determineDisplayCommand(goOutput);
        else if(object instanceof MatchEvent)
            goOutput.setMatchEvent(object);
        determineDisplayCommand(goOutput);
    }
}
```
5 Database Layer

![Database Layer Module Diagram](image)

Table 5-1: Database Data Flows

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB1</td>
<td>SELECT query that will retrieve a specific match’s logged data</td>
</tr>
<tr>
<td>DB2</td>
<td>Results from SELECT query will contain match log data</td>
</tr>
<tr>
<td>DB3</td>
<td>Match Log object containing Match Event information for a specific match</td>
</tr>
<tr>
<td>DB4</td>
<td>INPUT query that will insert contestants name and country information</td>
</tr>
<tr>
<td>DB5</td>
<td>INPUT query that will insert match event data into a match log</td>
</tr>
</tbody>
</table>

5.1 Description

5.2 Database Manager: Store Contestant

5.2.1 Prologue

This module will create a query that will store contestant information into the database.

5.2.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Store (Database Controller)</td>
<td>Store Contestant (Database Manager)</td>
<td>Contestants first and last name along with their</td>
</tr>
</tbody>
</table>
5.2.3 External Data Dependencies

Contestant Object

5.2.4 Internal Data Descriptors

MySQL queries

5.2.5 Pseudo-code

```java
public void storeContestant(Contestant contestant)
{
    String sql1 = "INSERT INTO Contestant VALUES (" + contestant.getBlueFirst() + ", " + contestant.getBlueLast() + ", " + contestant.getBlueCountry() + "");
    String sql2 = "INSERT INTO Contestant VALUES (" + contestant.getRedFirst() + ", " + contestant.getRedLast() + ", " + contestant.getRedCountry() + ")";
    MySQL db = new MySQL("Database");
    db.execute(sql1);
    db.execute(sql2);
}
```

5.3 Database Manager: Store Log

5.3.1 Prologue:

This module will create queries to store match events into the database.

5.3.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Creator (Database Controller)</td>
<td>Store Log (Database Manager)</td>
<td>Match Event objects that contain information about events to be store for one each match.</td>
</tr>
<tr>
<td>Store Log (Database Manager)</td>
<td>MySQL</td>
<td>Queries to store match events into the database</td>
</tr>
</tbody>
</table>
5.3.3 External Data Dependencies
Match Event objects

5.3.4 Internal Data Descriptors
MySQL Queries

5.3.5 Pseudo-code

```java
public void storeLog(MatchEvent log)
{
    String sql = "INSERT INTO MatchLog VALUES (" + log.event() + ", " + log.timestamp() + ");
    MySQL db = new MySQL("Database");
    db.execute(sql);
}
```

5.4 Database Manager: Retrieve Match Log

5.4.1 Prologue:
This module will create a query to request a match log from the database.

5.4.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Log (Database Controller)</td>
<td>Retrieve Match Log (Database Manager)</td>
<td>Match ID which identifies a unique match in the database</td>
</tr>
<tr>
<td>Retrieve Match Log (Database Manager)</td>
<td>MySQL</td>
<td>Queries that request a specific match log from the database</td>
</tr>
<tr>
<td>MySQL</td>
<td>Retrieve Match Log (Database Manager)</td>
<td>Match Log information from the database</td>
</tr>
</tbody>
</table>

5.4.3 External Data Dependencies
Match ID

5.4.4 Internal Data Descriptors
MySQL Queries
5.4.5 Pseudo-code

```java
public Log retrieveMatchLog(int matchID)
{
    string retrieveSQL = "SELECT * FROM MatchLog WHERE matchID = " + matchID + ";";
    MySql db = new MySql("Database");
    Log returnedLog;
    returnedLog = db.execute(sql);
    return returnedLog;
}
```
6 Match Event Processor Layer

Figure 6-1: Match Event Processor Layer Module Diagram

Table 6-1: Match Event Processor Data Flows

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Force Data for calibration</td>
</tr>
<tr>
<td>M2</td>
<td>Force Data Object to be evaluated</td>
</tr>
<tr>
<td>M3</td>
<td>State Object to set the initial match state</td>
</tr>
<tr>
<td>M4</td>
<td>Adjustment factor float value</td>
</tr>
<tr>
<td>M5</td>
<td>Match Event Object sent for modification</td>
</tr>
<tr>
<td>M6</td>
<td>Modified Match Event Object</td>
</tr>
<tr>
<td>M7</td>
<td>Match Event Object</td>
</tr>
</tbody>
</table>

6.1 Description

This layer’s purpose is to determine the amount of force to deduct from a competitor through strikes, combination strikes, or penalties.

6.2 Match Event Processor: Admin Match Data

6.2.1 Prologue

This module will set the initial match state.
6.2.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Controls</td>
<td>Admin Match Data</td>
<td>Object with timestamp</td>
</tr>
<tr>
<td>Admin Match Data</td>
<td>Match Event</td>
<td>Match Event Object</td>
</tr>
</tbody>
</table>

6.2.3 External Data Dependencies

This module will receive an Object with a timestamp.

6.2.4 Internal Data Descriptors

This module will send a Match Event Object.

6.2.5 Pseudo-Code

```java
public void adminMatchData(Object object, String timestamp) {
    if(object.typeof(State))
        matchEvent(object, timestamp);
    else if(object.typeof(MatchEvent))
        matchEvent(object, timestamp);
}
```

6.3 Match Event Processor: Force Data

6.3.1 Prologue

This module will check to see if the force is for calibration or for a match.

6.3.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Stamp</td>
<td>Force Data</td>
<td>Force Data Object with timestamp</td>
</tr>
<tr>
<td>Force Data</td>
<td>Determine Adjustment Factor</td>
<td>Force Data Object with calibration flag</td>
</tr>
<tr>
<td>Force Data</td>
<td>Match Event</td>
<td>Force Data Object</td>
</tr>
</tbody>
</table>

6.3.3 External Data Dependencies

This module receives a Force Data Object with a timestamp.

6.3.4 Internal Data Descriptors

This module sends a Force Data Object.
6.3.5 Pseudo-Code

```java
public void forceData(Object object, String timestamp)
{
  if(object.calibrate == true)
    matchEvent(determineAdjustmentFactor(object), timestamp);
  else
    matchEvent(object, timestamp);
}
```

6.4 Match Event Processor: Match Event

6.4.1 Prologue

This module determines the legitimacy of a Match Event.

6.4.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force Data</td>
<td>Match Event</td>
<td>Force Object</td>
</tr>
<tr>
<td>Determine Adjustment Factor</td>
<td>Match Event</td>
<td>Adjustment Factor float for calibration</td>
</tr>
<tr>
<td>Admin Match Data</td>
<td>Match Event</td>
<td>Initial Match State</td>
</tr>
<tr>
<td>Match Event</td>
<td>Score Modifiers</td>
<td>Send Match Event Object</td>
</tr>
<tr>
<td>Score Modifiers</td>
<td>Match Event</td>
<td>Receive Match Event Object</td>
</tr>
<tr>
<td>Match Event</td>
<td>Match Log</td>
<td>Send Match Event</td>
</tr>
</tbody>
</table>

6.4.3 External Data Dependencies

Must receive Force Object or Match State Object.

6.4.4 Internal Data Descriptors

Match Event Object.
6.4.5 Pseudo-Code

```java
public void matchEvent(Object object, timestamp)
{
    if(object instanceof Force)
    {
        if(object.getForce() >= THRESHOLD)
        {
            Force modifiedForce = new modifiedForce();
            MatchEvent matchStackEvent = new MatchEvent();
            modifiedForce = scoreModifiers(object);
            matchStackEvent.setEvent(modifiedForce);
            matchStackEvent.setTimestamp(timestamp);
            matchLog(matchStackEvent);
        }
        else
        {
            MatchEvent matchStackEvent = new MatchEvent();
            matchStackEvent.setEvent(object);
            matchStackEvent.setTimestamp(timestamp);
            matchLog(matchStackEvent);
        }
    }
}
```

6.5 Calibration: Determine Adjustment Factor

6.5.1 Prologue

This module returns a float value for calibration.

6.5.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force Data</td>
<td>Determine Adjustment Factor</td>
<td>Force Data Object</td>
</tr>
<tr>
<td>Determine Adjustment Factor</td>
<td>Match Event</td>
<td>Adjustment Factor float value for calibration</td>
</tr>
</tbody>
</table>

6.5.3 External Data Dependencies

Must receive a Force Data Object.

6.5.4 Internal Data Descriptors

Will determine the adjustment factor for force values.
6.5.5 Pseudo-Code

```java
public void determineAdjustmentFactor(Object object)
{
    const FORCE_CALIBRATION = 25;
    if (object.getForce() != FORCE_CALIBRATION)
    {
        modifier = FORCE_CALIBRATION/object.getForce();
        object.setModifier(modifier);
        return object;
    }
    else
    {
        object.setModifier(1);
        return object;
    }
}
```

6.6 Force Calculations: Score Modifiers

6.6.1 Prologue
This module will determine any score modifications that can be applied to the Match Event Object.

6.6.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match Event</td>
<td>Score Modifiers</td>
<td>Match Event Object</td>
</tr>
<tr>
<td>Score Modifiers</td>
<td>Match Event</td>
<td>Adjusted Match Event Object</td>
</tr>
</tbody>
</table>

6.6.3 External Data Dependencies
Must receive a Match Event Object.

6.6.4 Internal Data Descriptors
Will return an adjusted Match Event Object.
6.6.5 Pseudo-Code

```java
public MatchEvent scoreModifiers(MatchEvent event)
{
    if (event.timestamp - previousEvent.timestamp) <= COMBINATION_CLOCK)
    {
        event.Force = SCOREMULTIPLIER * event.Force;
        return event;
    }
    else if (event.equipment() == HEAD)
    {
        event.Force = HEADMULTIPLIER * event.Force;
        return event;
    }
    else
    {
        return event;
    }
}
```
7 Presentation Layer

Figure 7-1: Presentation Layer Module Diagram

Table 7-1: Presentation Data Flows

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Object which will contain data that must be updated in the Admin display</td>
</tr>
<tr>
<td>P2</td>
<td>Object which will contain data that must be updated in the Audience display</td>
</tr>
<tr>
<td>P3</td>
<td>Force, round, and time data to update the audience form</td>
</tr>
<tr>
<td>P4</td>
<td>Commands to start new forms and force, round, and time data to update Administrator forms</td>
</tr>
</tbody>
</table>

7.1 Description

This layer will handle output of data to the Admin display and the Audience display

7.2 Output Handler: Determine Display Command

7.2.1 Prologue

This module will take in a GoObject which will contain all of the information that needs to be updated and will also take in an Admin Event to change the forms that can be seen.

7.2.2 Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Output Creator</td>
<td>Determine Display Command</td>
<td>Game Output object containing data to be updated</td>
</tr>
<tr>
<td>(Game Output Controller)</td>
<td>(Output Handler)</td>
<td></td>
</tr>
<tr>
<td>Game Output Creator</td>
<td>Determine Display Command</td>
<td>Admin Event that will change the appropriate forms based on these administrator events</td>
</tr>
<tr>
<td>(Game Output Controller)</td>
<td>(Output Handler)</td>
<td></td>
</tr>
</tbody>
</table>
Determine Display Command (Output Handler)  Update Admin Display (Admin Display)  Administrator data that needs to be updated on the Admin form

Determine Display Command (Output Handler)  Update Audience Display (Audience Display)  Audience data that needs to be updated in the Audience form

7.2.3 External Data Dependencies
Game Output object and Admin Events objects

7.2.4 Internal Data Dependencies
Admin object and Audience object

7.2.5 Pseudo-code

```java
public void determineDisplayCommand(Object object) {
    if (object.command == ACCEPT)
        updateAdminDisplay(object);
    else if (object.command == TEST)
        updateAdminDisplay(object);
    else if (object.command == LOG)
        updateAdminDisplay(object);
    else if (object.command == ENU)
        updateAdminDisplay(object);
    else if (object.command == FIGHT)
        updateAdminDisplay(object);
    else if (object.command == FORCE)
        updateAdminDisplay(object);
    else if (object.command == TIME)
        updateAdminDisplay(object);
    else if (object.command == ROUND)
        updateAdminDisplay(object);
}
```

7.3 Admin Display: Update Admin Display

7.3.1 Prologue
This module will take in updated information and will use this updated information to update the controls on the form.
7.3.2 Interfaces

Update Admin Display Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine Display Command (Output Handler)</td>
<td>Update Admin Display (Admin Display)</td>
<td>Admin object which contains data that needs to be updated in the administrator form</td>
</tr>
<tr>
<td>Update Admin Display (Admin Display)</td>
<td>Administrator Screen</td>
<td>Updated views</td>
</tr>
</tbody>
</table>

7.3.3 External Data Dependencies

Admin object

7.3.4 Internal Data Descriptors

None

7.3.5 Pseudo-code

```java
public void updateAdminDisplay(Object object) {
    if (object.command == ACCEPT) {
        adminForm.createNewForm("ControlPanel");
    } else if (object.command == TEST) {
        adminForm.createNewForm("TestMode");
    } else if (object.command == LOG) {
        adminForm.createNewForm("Log");
    } else if (object.command == END) {
        adminForm.closeForm("AudienceDisplay");
    } else if (object.command == MEDICAL) {
        adminForm.createNewForm("Medical");
        adminForm.updateMedicalTimer(object.medicalTimer());
    } else if (object.command == FIGHT) {
        adminForm.updateStartTimer(object.roundClock());
    } else if (object.command == FORCE) {
        adminForm.updateLifeBar(object.force());
    } else if (object.command == TIME) {
        adminForm.updateTimer(object.roundClock());
    } else if (object.command == ROUND) {
        adminForm.updateRound(object.round());
    }
}
```

7.4 Audience Display: Update Audience Display

7.4.1 Prologue

This module will take in updated information such as the updated force, round, and time to update the views on the Audience form that will be displayed on the Audience display.
7.4.2 Interfaces

Update Audience Display Interfaces

<table>
<thead>
<tr>
<th>Source Module</th>
<th>Destination Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine Display Command (Output Handler)</td>
<td>Update Admin Display (Admin Display)</td>
<td>Admin object which contains data that needs to be updated in the administrator form</td>
</tr>
<tr>
<td>Update Admin Display (Admin Display)</td>
<td>Administrator Screen</td>
<td>Updated views</td>
</tr>
</tbody>
</table>

7.4.3 External Data Dependencies

Admin object

7.4.4 Internal Data Descriptors

None

7.4.5 Pseudo-code

```java
public void updateAudienceDisplay(Object object)
{
    if(object.command == FIGHT)
        audienceForm.startForm("AudienceDisplay");
        audienceForm.updateTimer(object.roundClock());
    else if(object.command == FORCE)
        audienceForm.updateLifeBar(object.force());
        audienceForm.updateTimer(object.roundClock());
    else if(object.command == TIME)
        audienceForm.updateLifeBar(object.time());
        audienceForm.updateRound(object.round());
    else if(object.command == ROUND)
        audienceForm.updateRound(object.round());
}
```
8 Quality Assurance

8.1 Test Plan and Procedures

The system architecture will be tested by team TK Force to ensure that all the functionalities and system components are reviewed and tested based on the requirements defined in the System Requirements Specification, Architecture Design Specification, and Detailed Design Specification. Changes will be made to all the documents to ensure that every document have been updated correctly. Each component will be tested individually and the system will also be tested as a whole to verify that the integration of each individual component is properly made.

8.2 Module/Unit Test

8.2.1 Input Layer

8.2.1.1 Admin Input Subsystem: Keyboard/Mouse Input Module

This module shall be tested to ensure it correctly receives all keyboard and mouse input from the Admin PC.

8.2.1.2 Input Packager Subsystem: Admin Package Module

This module shall be tested to ensure it will receive the text data and admin commands, and use the input to correctly create the Admin Event object, send the equipment IDs, and send a penalty command to the Force Data Package Module.

8.2.1.3 Input Packager Subsystem: Force Data Package Module

This module shall be tested to ensure it correctly creates a Force Data object from the Equipment Input Module.
8.2.1.4 Equipment Input Subsystem: Filter Module

This module shall be tested to ensure that it will receive the equipment ID from the Admin Package Module and match the IDs to only focusing on receiving inputs from the desired equipment.

8.2.1.5 Equipment Input Subsystem: Equipment Input Module

This module shall be tested to ensure it will receive the equipment ID and force data from the Filer Module and it will correctly create a Force Data object.

8.2.2 Data Controller Layer

8.2.2.1 Input Controller Subsystem: Object Identifier Module

This module shall be tested to ensure it will take in Contestant objects, State objects, Admin Event objects and Force objects. Then it shall be able to determine correctly what kind of object it has and send that object to the appropriate module for further processing.

8.2.2.2 Database Controller Subsystem: Request Log Module

This module shall be tested to ensure the function will make a request to the Database Layer that will retrieve the match log for a specified match and then send this information to the GoCreator.

8.2.2.3 Database Controller Subsystem: DB store Module

This module shall be tested to ensure it will receive the Contestant objects from the Input Controller, and this module will prepare the information to be input into the database.

8.2.2.4 Database Controller Subsystem: Log Creator Module

This module shall be tested to ensure the function will receive the stack containing the logs of events that occurred during the match, and this
module will pop off each Match Event object from the stack to send to the Database Layer to be stored in the database.

8.2.2.5 Game Output Controller Subsystem: Game Output Creator Module

This module shall be tested to ensure it will create an output object correctly which will contain all of the new information to be updated in the Presentation Layer.

8.2.2.6 Match Event Controller Subsystem: Time Stamp Module

This module shall be tested to ensure it will get a timestamp correctly from the timer and place a timestamp on each event that much be stored in the database.

8.2.2.7 Match Event Controller Subsystem: Time Module

This module shall be tested to ensure it will create a new timer correctly for each request from the Admin such as the round clock, medical clock, and the break clock. It will also be tested to ensure it will create a new thread correctly in order to update the clock.

8.2.2.8 Match Event Controller Subsystem: Match Log Module

This module shall be tested to ensure it will receive the match events from the Match Event subsystem and it will create a stack of these objects correctly to be sent to the Database Controller Layer.

8.2.3 Database Layer

8.2.3.1 Database Manager Subsystem: Store Competitor Module

This module shall be tested to ensure it will create a query correctly that will store competitor information into the database.
8.2.3.2 Database Manager Subsystem: Store Log Module

This module shall be tested to ensure it will create queries correctly to store match events into the database.

8.2.3.3 Database Manager Subsystem: Retrieve Match Log Module

This module shall be tested to ensure it will create a query correctly to request a match log from the database.

8.2.3.4 Data Storage Subsystem

8.2.4 Match Event Processor Layer

8.2.4.1 Match Event Processor Subsystem: Admin Match Data

This module shall be tested to ensure it set the initial match state correctly.

8.2.4.2 Match Event Processor Subsystem: Match Event

This module shall be tested to ensure it will correctly determine the legitimacy of a Match Event.

8.2.4.3 Match Event Processor Subsystem: Force Data

This module shall be tested to ensure it will check to see if the force is for calibration or for a match, and determine them correctly.

8.2.4.4 Match Calculations Subsystem: Score Modifiers

This module shall be tested to ensure it will correctly determine any score modifications that can be applied to the Match Event objects.
8.2.4.5 Calibration Subsystem: Determine Adjustment Factor

This module shall be tested to ensure it will return a float value correctly for the Calibration subsystem.

8.2.5 Presentation Layer

8.2.5.1 Output Handler Subsystem: Determine Display Command

This module shall be tested to ensure it will take in a GoObject which will contain all of the information that needs to be updated and will also take in an Admin Event to change the forms that can be seen.

8.2.5.2 Admin Display Subsystem: Update Admin Display

This module shall be tested to ensure it will take in updated information and will use this updated information to update the controls on the form

8.2.5.3 Audience Display Subsystem: Update Audience Display

This module shall be tested to ensure it will take in updated information such as the updated force, round, and time to update the views on the Audience form that will be displayed on the Audience display

8.3 Component Testing

8.3.1 E-Hogu & E-Headgear

The Hogu and Headgear shall be tested to ensure it returns correct data of the hit which contains the force data. The Hogu and Headgear will be tested based on applying different amount of forces on it and check the return data to see if the returned force matches the force applied on the equipment.

8.3.2 Transmitters

The transmitters shall be tested to ensure the connections between the transmitter and receiver is stable, and also, the data that transmitters sent to the receiver are in the
correct format at a correct frequency. The signal strength will also be tested to ensure during a match, every valid hit will be correctly taken into the system. The signal strength will be tested based on the distance between the transmitters and receiver to determine what the appropriate distance for the transmitters and receiver is.

### 8.3.3 Receiver

The receiver shall be tested to ensure it works correctly with the operating system. The driver of the USB receiver will be tested for this. And also the connection between the transmitters and receiver shall also be tested. The way to test this is same as how to test the transmitters.

### 8.4 Integration Testing

Reference to the interfaces tables for every module, all the inputs shall be passed into different modules from a specific module within the system. All the inputs must be in the correct format and range. Every module within the system shall work correctly as long as the inputs are in the correct format and range.

Every module in the system shall work separately from other modules, the only connection between any modules shall only be input/output based. Based on this, the integration shall be ensured as long as the data is in the correct format and range.

### 8.5 System Verification Testing

The system shall be subjected to a series of tests to verify fulfillment of all requirements according to the system specifications. The system will be tested by the customer as a whole in real world cases. Customer will inspect the system by using it to run matches. The feedback of the customer will be the result of the system verification testing.

### 8.6 Test Cases

<table>
<thead>
<tr>
<th>Test case</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit the Hogu and Heargear without the Foodgear.</td>
<td>Nothing should happen at all</td>
</tr>
<tr>
<td>Apply a force that’s lower than the threshold.</td>
<td>Force shall be displayed, but no hit shall be taken to impact the match.</td>
</tr>
<tr>
<td><strong>A valid hit is arrived on one competitor</strong></td>
<td>The life bar on the audience screen shall be deducted; a record shall be made for the log.</td>
</tr>
<tr>
<td><strong>Hits during the Test Mode.</strong></td>
<td>The Test Mode screen shall display the information about the hit, but no hit will be taken into the log or the match.</td>
</tr>
<tr>
<td><strong>Administrator click on the start fight button.</strong></td>
<td>The fight clock shall start to count down, all valid hits shall be recorded and have impacts to the match.</td>
</tr>
<tr>
<td><strong>Administrator click on the pause button.</strong></td>
<td>The fight clock shall stop counting down. No hits will be taken into the system to have impact to the match.</td>
</tr>
<tr>
<td><strong>Administrator click on the medical button</strong></td>
<td>The fight shall pause and medical clock shall start to count down.</td>
</tr>
</tbody>
</table>
9 Requirements Mapping

9.1 Purpose

9.2 Layer Requirements Traceability

9.2.1 Customer requirements

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Input</th>
<th>Data Controller</th>
<th>Database</th>
<th>Match Event Processor</th>
<th>Presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Audience Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.2</td>
<td>Start screen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.3</td>
<td>Control panel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.4</td>
<td>Match Setup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.5</td>
<td>Test Mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.6</td>
<td>Equipment hit detection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.7</td>
<td>Health calculations</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.8</td>
<td>Penalties</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.9</td>
<td>Score modifiers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.10</td>
<td>Equipment parts on / off</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.11</td>
<td>Match log</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3.12</td>
<td>Administrator display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### 9.2.2 Performance requirements

#### Table 9-2: Performance Requirements Mapping

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Input</th>
<th>Data Controller</th>
<th>Database</th>
<th>Match Event Processor</th>
<th>Presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Start Program</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5.2</td>
<td>Life bar deduction response</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5.3</td>
<td>Recording log</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5.4</td>
<td>Response to button clicks</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5.5</td>
<td>Processing force data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

### 9.2.3 Other requirements

#### Table 9-3: Other Requirements Mapping

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Input</th>
<th>Data Controller</th>
<th>Database</th>
<th>Match Event Processor</th>
<th>Presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Store contestant information</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.2</td>
<td>Store match events</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.3</td>
<td>Store country information</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.4</td>
<td>Store match statistics</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.5</td>
<td>Calibration</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### 9.3 Module Requirements Traceability

Table 9-4: Customer Requirements Mapping

<table>
<thead>
<tr>
<th>Req</th>
<th>3.1</th>
<th>3.2</th>
<th>3.3</th>
<th>3.4</th>
<th>3.5</th>
<th>3.6</th>
<th>3.7</th>
<th>3.8</th>
<th>3.9</th>
<th>3.10</th>
<th>3.11</th>
<th>3.12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard/Mouse Input</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Filter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equipment Input</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admin Package</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Force Data Package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Identifier</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Request Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB Store</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log Creator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Stamp</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Time Module</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Match Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game Output Creator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Competitor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieve Match Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admin Match Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force Data</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Match Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine Adjustment Factor</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Score Modifiers</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Determine Display Command</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Update Admin Display</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Update Audience Display</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

10/4/13 TMMS DDS 50
<table>
<thead>
<tr>
<th>Req</th>
<th>5.1</th>
<th>5.2</th>
<th>5.3</th>
<th>5.4</th>
<th>5.5</th>
<th>8.1</th>
<th>8.2</th>
<th>8.3</th>
<th>8.4</th>
<th>8.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard/Mouse Input</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equipment Input</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Admin Package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force Data Package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Object Identifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Request Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB Store</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log Creator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Stamp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Module</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Match Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game Output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Competitor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store Log</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieve Match Log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Admin Match Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force Data</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Match Event</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine Adjustment Factor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Score Modifiers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Determine Display Command</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Admin Display</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Audience Display</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
10 Acceptance Plan

10.1 Overview
This section describes the minimum criteria that must be met by the sponsor in order to be considered acceptable by the customers.

10.2 Packaging and Installation

10.2.1 Packaging Components:

10.2.1.1 Installation Disk
The executable file for installing the software will be putting on a CD to distribute.

10.2.1.2 User Manual
A detailed user manual about the software will be made available on the installation disk. It will address the frequently asked question and troubleshooting tips.

10.2.2 Installation:
The software can be installed with an executable file.

10.3 Acceptance Testing
System testing shall be conducted to ensure that the system meets the acceptance criteria. The system shall go through module, subsystem, layer, integration and overall system testing. The details of the testing shall be provided in the System Test Plan document.
10.4 Acceptance Criteria

10.4.1 Verify that the system displays the audience screen.

10.4.1.1 Requirement(s) addressed

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Audience Display</td>
<td>During a match, the TMMS will provide a display with the competitor information to be viewed by the audience. Critical information to be displayed shall be name of competitors, life-bars, country, round and match victors, and time remaining for a match.</td>
</tr>
<tr>
<td>3.7</td>
<td>Health Calculations</td>
<td>The system shall calculate the remaining health of a competitor which will decrease based on the force of a successful hit (more force means more life deducted from the life bar), the number of penalties assigned to a fighter, and any modifiers that have been applied.</td>
</tr>
<tr>
<td>3.8</td>
<td>Penalties</td>
<td>The health of a fighter in the system can be affected by a penalty issued to a fighter and will contain two types of penalties major penalty (jooyo) and a minor penalty (danjo).</td>
</tr>
<tr>
<td>3.9</td>
<td>Score Modifiers</td>
<td>The score modifiers will alter the amount of life decrease from a successful hit.</td>
</tr>
<tr>
<td>5.2</td>
<td>Life-bar Deduction Response</td>
<td>The program shall reduce the life bar indicator on the screen when there is a successful hit detected by the receiver. The response delay from the hit to the screen will be minimal.</td>
</tr>
</tbody>
</table>

Table 10-1: Audience Screen Verification
10.4.1.2 Verification Procedure

TMMS will be tested to ensure that the system can display the audience screen. After the match has been setup, the audience screen will be displayed on the monitor.

10.4.2 Verify that the system runs in real time

10.4.2.1 Requirement(s) addressed:

Table 10-2: Real Time Verification

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>Life-bar Deduction Response</td>
<td>The program shall reduce the life bar indicator on the screen when there is a successful hit detected by the receiver. The response delay from the hit to the screen will be minimal.</td>
</tr>
<tr>
<td>5.3</td>
<td>Recording Log</td>
<td>The program shall record the logs of a match immediately.</td>
</tr>
<tr>
<td>5.4</td>
<td>Response to Button Click</td>
<td>The program shall respond to any button clicks immediately.</td>
</tr>
</tbody>
</table>

10.4.2.2 Verification Procedure:

When the system and equipment are online, the pad of each competitor will be struck to verify the life bar response. From the match log, the first event will be selected and applied to ensure the match log reverts to the initial match state.
10.4.3 Verify that the system saves data for Log

10.4.3.1 Requirement(s) addressed:

Table 10-3: Save Log Data Verification

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.11</td>
<td>Match Log</td>
<td>The system shall store the history of all scoreboard changing events and their attributes. If you select an event in the match log and hit accept, the match will be reverted to that point.</td>
</tr>
<tr>
<td>8.2</td>
<td>Store Match Events</td>
<td>This information will be stored in a database and will correspond to events that affect the score in a single match. This information may also be used to return a match to a specific point in time.</td>
</tr>
<tr>
<td>8.3</td>
<td>Store Match Statistics</td>
<td>This is a list of countries represented in the Olympic Games. Must be stored in a database as 3 letter country code, and must store corresponding country flag.</td>
</tr>
</tbody>
</table>

10.4.3.2 Verification Procedure:

TMMS will be tested and verified that the system responds to the match and store all the logs immediately.

10.4.4 Verify that the system is user friendly

10.4.4.1 Requirement(s) addressed:

Table 10-4: User Friendliness Verification

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section</td>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3.2</td>
<td>Start Screen</td>
<td>When the system is started the software shall display a start screen showing the logo and name of the system.</td>
</tr>
<tr>
<td>3.3</td>
<td>Control Panel</td>
<td>The system shall have a control panel that allows a user to view details about a match, apply test mode for a match, apply minor and major penalties, apply medical to a match, and alter details about a match.</td>
</tr>
<tr>
<td>3.4</td>
<td>Match Setup</td>
<td>The system shall have a match setup screen to input contestant information, register equipment to contestant, and specify other match information.</td>
</tr>
<tr>
<td>3.5</td>
<td>Test Mode</td>
<td>The system shall have a mode for users to view the forces that occur during a match and the times the forces occurred.</td>
</tr>
<tr>
<td>3.8</td>
<td>Penalties</td>
<td>The health of a fighter in the system can be affected by a penalty issued to a fighter and will contain two types of penalties: major penalty (jooyo) and a minor penalty (danjo).</td>
</tr>
<tr>
<td>3.9</td>
<td>Score Modifiers</td>
<td>The score modifiers will alter the amount of life decrease from a successful hit.</td>
</tr>
<tr>
<td>3.11</td>
<td>Match Log</td>
<td>The system shall store the history of all scoreboard changing events and their attributes. If you select an event in the match log and hit accept, the match will be reverted to that point.</td>
</tr>
</tbody>
</table>

**10.4.4.2 Verification Procedure:**

TMMS will be tested by the sponsor to verify ease of use.
10.4.5 Verify that the system is accurately recording force

10.4.5.1 Requirement(s) addressed:

Table 10-5: Force Accuracy Verification

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Test Mode</td>
<td>The system shall have a mode for users to view the forces that occur during a match and the times the forces occurred.</td>
</tr>
<tr>
<td>3.7</td>
<td>Health Calculations</td>
<td>The system shall calculate the remaining health of a competitor which will decrease based on the force of a successful hit (more force means more life deducted from the life bar), the number of penalties assigned to a fighter, and any modifiers that have been applied.</td>
</tr>
<tr>
<td>5.3</td>
<td>Processing Force Data</td>
<td>When a hit is registered by the program, the force recorded from the hit will be retrieved from the Daedo equipment.</td>
</tr>
<tr>
<td>9.5</td>
<td>Calibration</td>
<td>This will allow the force of hits on the equipment to be adjusted to ensure accuracy.</td>
</tr>
</tbody>
</table>

10.4.5.2 Verification Procedure:

TMMS will be tested after calibration to ensure the force of the hits is accurate.
11 Hardware Components

11.1 Overview
This section describes all the hardware components will be used in the Taekwondo Match Management System (TMMS). Each components purpose, specifications and interface will be describe in blow.

11.2 EPRO 2910 Daedo E-Hogu w/out transmitter

11.2.1 Purpose
The purpose of having these two chest protector is to capture hit information from chest-hit during the match.

11.2.2 Specifications
Resistant proximity and impact sensors
Sensors only activates when the magnet in the foot-pad in close
Send captured information through Ethernet cable
Reversible red and blue sides

11.2.3 Interfaces
The EPRO 2910 Daedo E-Hogu will interface with the wireless transmitter that connected to the Ethernet cable.
11.3 EPRO 2913 Daedo E-Headgear w/out transmitter

11.3.1 Purpose
The purpose of having these two head protector is to capture hit information from head-hit during the match.

11.3.2 Specifications
Sensor embedded all around protective gear
Power threshold set up independent from E-Hogu
Send captured information through Ethernet cable

11.3.3 Interfaces
The EPRO 2913 Daedo E-Headgear will interface with the wireless transmitter that connected to the Ethernet cable.
11.4 EPRO 2903 Daedo E-Footgear

![EPRO 2903 Daedo E-Footgear](image)

Figure 11-3: EPRO 2903 Daedo E-Footgear

11.4.1 Purpose
The purpose of having E-Footgear is to register hit to other equipment. A hit can only be registered with E-Footgear.

11.4.2 Specifications
Magnet pads built in
Magnet pads activates other sensor to register hit

11.4.3 Interfaces
The EPRO 2903 Daedo E-Footgear interfaces with EPRO 2910 Daedo E-Hogu and EPRO 2913 Daedo E-Headgear to register hit.
11.5 EPRO 2909 Daedo E-Hogu & E-Headgear Wireless Transmitter

![Image](image.png)

**Figure 11-4: EPRO 2909 Daedo E-Hogu & E-Headgear Wireless Transmitter**

11.5.1 Purpose
The purpose of having the wireless transmitter is to receive the information from the E-Hogu and E-Headgear; the transmitter will process the data internally and transmit the packaged data to the wireless receiver.

11.5.2 Specifications
Frequency: 2.4 GHz
Powered by rechargeable batteries
Every transmitter has a unique ID
The transmitter is connected to the gears through Ethernet cable

11.5.3 Interfaces
The EPRO 2909 Daedo E-Hogu & E-Headgear Wireless Transmitter interfaces with EPRO 2910 Daedo E-Hogu and EPRO 2913 Daedo E-Headgear to receive data and transmit the data to EPRO 2906 Daedo USB Wireless Receiver.
11.6 EPRO 2906 Daedo USB Wireless Receiver

11.6.1 Purpose
The purpose of having this USB wireless receiver is to receive the match event information from the wireless transmitter and send the information to the system through USB port.

11.6.2 Specifications
Frequency: 2.4 GHz
Receives data from
Send captured information through Ethernet cable

11.6.3 Interfaces
The EPRO 2906 Daedo USB Wireless Receiver will interface with the wireless transmitter that connected to the Ethernet cable.