**Vision Document**

For a Track & Field Meet Server

Version 1.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Tracy Marshall

CIS 895 – MSE Project

Kansas State University

Table of Contents

[1. Introduction 3](#_Toc454215716)

[1.1. Motivation 3](#_Toc454215717)

[1.2. Terms and Definition 3](#_Toc454215718)

[1.2.1. Server 3](#_Toc454215719)

[1.2.2. Client 3](#_Toc454215720)

[1.2.3. API 3](#_Toc454215721)

[1.2.4. Socket 4](#_Toc454215722)

[1.2.5. Port 4](#_Toc454215723)

[1.2.6. Authentication 4](#_Toc454215724)

[1.2.7. Protocol 4](#_Toc454215725)

[1.3. References 4](#_Toc454215726)

[2. Project Overview 5](#_Toc454215727)

[2.1. Project Goal 5](#_Toc454215728)

[2.2. System Context 5](#_Toc454215729)

[3. Requirements Specification 6](#_Toc454215730)

[3.1. Critical Use Cases 6](#_Toc454215731)

[3.1.1. Use Case 1 6](#_Toc454215732)

[3.2. Assumptions 6](#_Toc454215733)

[3.3. Constraints 6](#_Toc454215734)

[3.4. Environment 6](#_Toc454215735)

# Introduction

The purpose of this document is to give a brief overview of the Track & Field Meet Server (TFMS). The TFMS will provide a centralized location for organizers of a track meet to enhance the execution of the track meet for teams, officials, and spectators. This document will provide some high level goals and context for the system as well as providing critical uses cases and requirements. All of these things will act as a guide for guiding the development of the TFMS project.

## Motivation

Organizing a track meet can be extremely complex with the orchestration of athletes, judges, and spectators partaking in a vast spectrum of events. Many modern track meets have had the fortune to use modern computing technologies to aide in the execution of track meet. The TFMS will change all of this by providing a means for small schools, especially rural high schools and middle schools, as well as nonprofit youth athletic associations to host track meets aided by computer technologies.

The TFMS project aims to be a cheap solution to assisting in the efforts of hosting a track meet by providing an open source system that manages the in’s and out’s of the track meet. The TFMS will include a rich API to allow developers to write mobile applications to communicate with the TFMS and provide a portal to TFMS for athletes, officials, and spectators. This product will create a more efficient operation for track meets and will bring new excitement to the sport in exotic places that didn’t have the means to acquire higher end products to assist with track meets.

## Terms and Definition

### Server

A server is an application that has some sort of functional purpose to serve a client which sends requests to the server. The server sends responses to the client based on the requests it receives and the nature of its utility.

### Client

A client is an application that uses a server in some capacity to provide functionality. Clients will send requests to a server and adhere to the server’s published API.

### API

An API is a defined set of rules and messages that describe the behavior and capability of the communication between a server and a client. API’s are often times developed on top of an existing industry standard protocol.

### Socket

A socket is an abstraction that represents a specific line of communication. This line of communication has a specified endpoint and allows programs to communicate with one another. Typically sockets refer to internet sockets which are built on the internet protocol.

### Port

A port is an enumerated value that is used in conjunction to an IP address to specify a specific process that is to be communicated with at the end point.

### Authentication

Authentication is the process of identifying something or someone. In today’s computer systems this is typically done with the use of cryptographic operations using keys to sign data. This is an important process for subverting malicious entities that intend to manipulate data

### Protocol

A protocol is an agreed upon standard way of doing something. In network systems protocols are used in layers to provide mechanisms for different forms of communications. The most fundamental of these protocols in the Internet Protocol (IP). Most commonly the IP is layered with a transport layer of either the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)

## References

1. Spivak, Ruslan. "Let's Build A Web Server. Part 1." *Ruslans Blog ATOM*. N.p., 09 Mar. 2015. Web. 20 June 2016.
2. Cooksey, Brian. "Chapter 1: Introduction." - An Introduction to APIs. N.p., n.d. Web. 20 June 2016.

# Project Overview

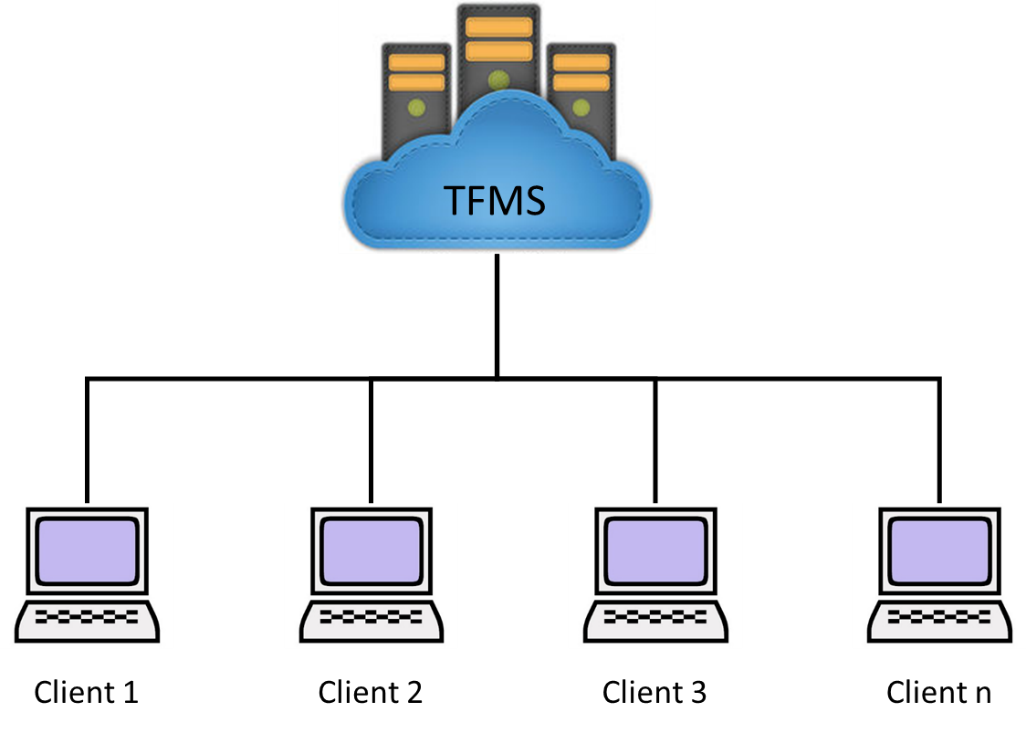


Figure 2.0-1 - Track & Field Meet Server Overview

The Track & Field Meet server will be capable of necessary functionality for hosting a track meet. The TFMS will be a repository for all data that is collected at the track meet. A rich API will be developed to provide developers to develop client side applications to submit, collect, and manipulate data. An important piece of the TFMS will be the capability to authenticate users and prevent malicious users from tampering with and intercepting data that is in transport to or from the server.

## Project Goal

The goal of this project to have a fully functional TFMS that clients can connect to for the purposes of being a participant, host, or spectator at a track meet. The TFMS will provide a rich and easy to use API to allow for easy client side development while maintaining data integrity and security. The TFMS will be able to have multiple clients simultaneously communicating with it and be capable of providing responses in a timely manner.

## System Context

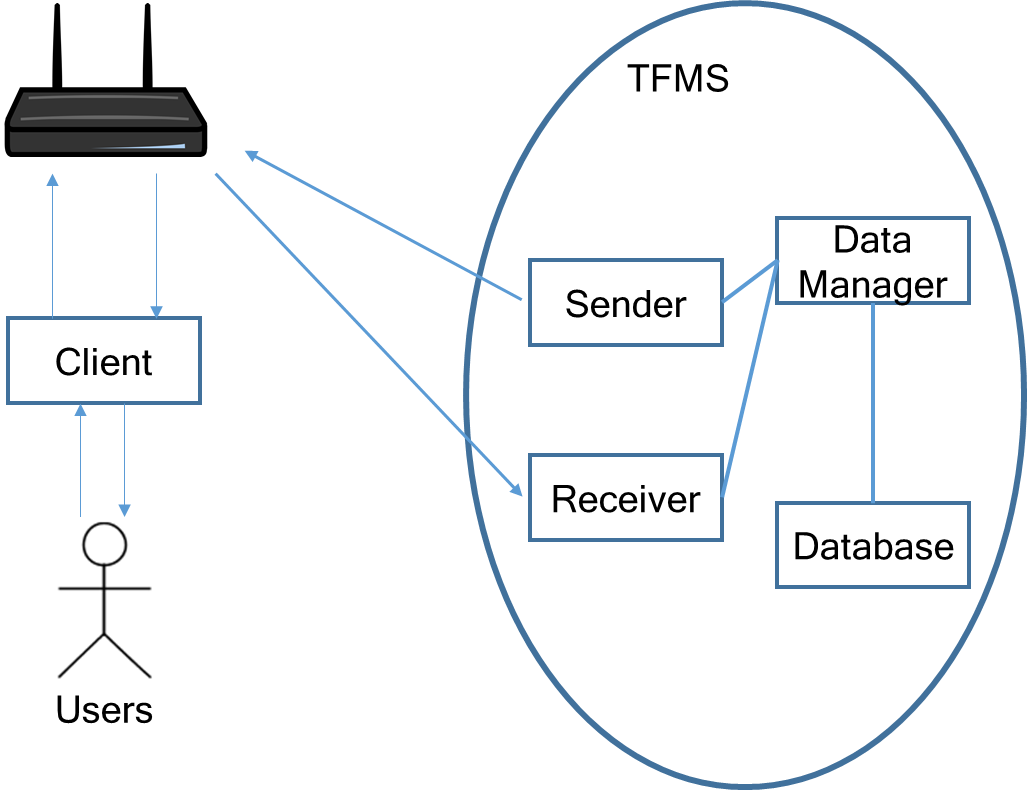


Figure 2.2-1 – TFMS System Context Diagram

The TFMS is made up of fairly few high level components. The system will network connected via a common NAT router. The client will also be on the same network as the TFMS via the NAT router. This network connection will provide the capability for the users to interact with the client to send/receive data to/from the server. The server will have dedicated components for sending and receiving data. That data will be managed, manipulated, and processed by the data manager which will store data in a database.

# Requirements Specification

## Critical Use Cases

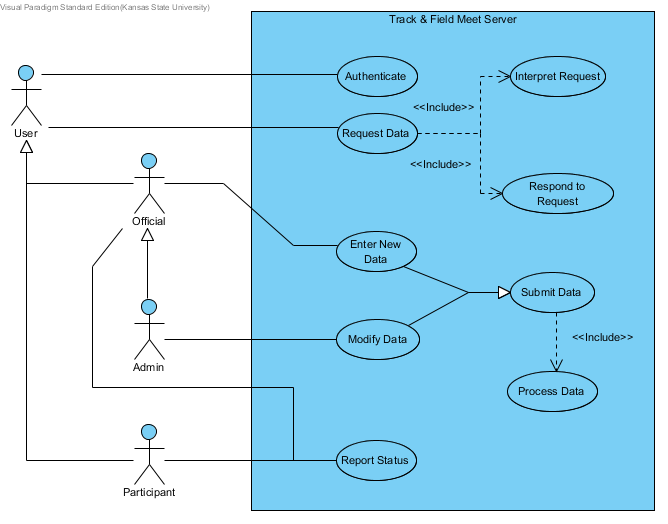


Figure 3.1-1 Critical Use Cases

### Use Case 1: Authenticate

**Description**: This use case describes the ability for any user to authenticate with the system and prove their identity.

**Includes:** N/A

**Pre-Conditions:** The system must know something about the user that it can authenticate the user with.

**Details:** The system will authenticate users as they interact with the system to verify the integrity of the requests and commands of the users and to make sure that user has privileges to perform their desired actions.

**Post Conditions:** User will be authenticated

**Specific Requirements:**

#### SR1.1 [Critical Requirement]

The system SHALL authenticate users with everyone interaction they have with the system.

### Use Case 2: Request Data

**Description**: This use case describes the ability for users to use the API to request data from the server.

**Includes:** Interpret Request, Respond to Request

**Pre-Conditions:** The user must be authenticated.

**Details:** The system will have a published API that will allow users to send different requests for data about the track meet.

**Post Conditions:** The system will have a queued request.

**Specific Requirements:**

#### SR2.1 [Critical Requirement]

The system SHALL receive data requests from concurrent users.

### Use Case 3: Interpret Request

**Description**: This use case describes how the system takes a user’s request and determines what kind of request it is.

**Includes:** N/A

**Pre-Conditions:** A request has been received

**Details:** The system has a published API that will describe different requests that can be made to ask for different information about the meet. These different request will have some sort of identifier and will require different types of responses.

**Post Conditions:** Determines the validity of what the user wants.

**Specific Requirements:**

#### SR3.1 [Critical Requirement]

The system SHALL interpret user requests based on the API.

### Use Case 4: Respond to Request

**Description**: This use case describes how the system responds to the user based on the request they have made.

**Includes:** N/A

**Pre-Conditions:** A valid request has been made.

**Details:** Valid requests can seek to fetch a variety of different data sets from the system and that data will be determined and sent back to the user.

**Post Conditions:** Data response to the user

**Specific Requirements:**

#### SR4.1 [Critical Requirement]

The system SHALL send response messages to the users.

### Use Case 5: Submit Data

**Description**: This use case describes how certain users can control data in the system.

**Includes:** Process Data

**Generalizes:** Enter New Data, Modify Data

**Pre-Conditions:** User is authenticated and has valid permissions.

**Details:** The system needs to be capable of having its data manipulated prior, during, and after the track meet. This will be done through the published API is only available to users who have official or admin permissions.

**Post Conditions:** Submittal is queued.

**Specific Requirements:**

#### SR5.1 [Critical Requirement]

The system SHALL receive data submittals from official and admin users based on the API.

### Use Case 6: Process Data

**Description**: This use case describes how the system handles a data submittal and processes it and puts it into the database.

**Includes:** N/A

**Pre-Conditions:** Submittal queued

**Details:** The system has a lot of data that is stored in it involving the track meet and this data has many users that can look at it and manipulate it. Some data must be processed before being put in the system and all data will have protections against possible data corruption.

**Post Conditions:** System data is updated with submitted data.

**Specific Requirements:**

#### SR6.1 [Critical Requirement]

The system SHALL put submitted data into the system.

### Use Case 7: Enter New Data

**Description**: This use case describes the ability of the system to handle new data entries about data that is described in the API.

**Includes:** N/A

**Pre-Conditions:** User has been authenticated and is an Official.

**Details:** The system will need to add information to the system about the track meet or add data that was missed before the start of the track meet.

**Post Conditions:** Data is added to the system

**Specific Requirements:**

#### SR7.1 [Critical Requirement]

The system SHALL add data to the system that is submitted by officials.

### Use Case 8: Modify Data

**Description**: This use case describes the ability of the system to modify data that is described by the API.

**Includes:** N/A

**Pre-Conditions:** User has been authenticated and is an Admin.

**Details:** The system will have the occasional clerical error or other failures from official errors. These errors need to ability to be correct but keep the ability to modify data limited to administrative users.

**Post Conditions:** Data is modified in the system.

**Specific Requirements:**

#### SR8.1 [Critical Requirement]

The system SHALL modify data in the system that is submitted by administrators.

### Use Case 9: Report Status

**Description**: This use case describes the ability for participants and officials to put a status in the system.

**Includes:** N/A

**Pre-Conditions:** User is authenticated as a participant.

**Details:** The system will have the capability for participants to communicate to officials their status for a given even so to keep the track meet running smoothly.

**Post Conditions:** Status is entered in the system

**Specific Requirements:**

#### SR91 [Critical Requirement]

The system SHALL store a status that is reported by a participant or official.

## Assumptions

* + 1. All users will be on the same local network as the TFMS.
    2. User authentication will be possible without an encrypted tunnel for communication.
    3. Client side applications will be developed by other developers.
    4. The system will not be impervious to advanced man in the middle attacks.

## Constraints

* + 1. The TFMS will be constrained to how many requests and responses it can send and received based on the throughput of the server’s network connection and network bandwidth.
    2. The TFMS will be constrained to users that an administrator has informed the system of and given privileges to.

## Environment

* + 1. The TFMS will be written in Java and compiled using the Java SE 8.
    2. The TFMS will be developed using the Eclipse IDE.
    3. The TFMS will run on a Raspberry Pi 3 using the most recent distribution of Raspbian.