**User Manual**

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

[Table of Contents 2](#_Toc468910762)

[1. Introduction 3](#_Toc468910763)

[2. Installation 3](#_Toc468910764)

[2.1. Prerequisites 3](#_Toc468910765)

[2.2. Required Files 3](#_Toc468910766)

[2.3. Setup 3](#_Toc468910767)

[3. Running the TFMS Demo 4](#_Toc468910768)

[4. API 4](#_Toc468910769)

[4.1. Command Header 4](#_Toc468910770)

[4.2. Add User 4](#_Toc468910771)

[4.2.1. Command 4](#_Toc468910772)

[4.2.2. Response 4](#_Toc468910773)

[4.3. Get User 4](#_Toc468910774)

[4.3.1. Command 4](#_Toc468910775)

[4.3.2. Response 4](#_Toc468910776)

[4.4. Modify User 4](#_Toc468910777)

[4.4.1. Command 4](#_Toc468910778)

[4.4.2. Response 5](#_Toc468910779)

[4.5. Add Results 5](#_Toc468910780)

[4.5.1. Command 5](#_Toc468910781)

[4.5.2. Response 5](#_Toc468910782)

[4.6. Update Status 5](#_Toc468910783)

[4.6.1. Command 5](#_Toc468910784)

[4.6.2. Response 5](#_Toc468910785)

[4.7. Other Failures 5](#_Toc468910786)

# Introduction

The purpose of this document is to provide guidance to users on how to use the Track & Field Meet Server. Information contained in this document deal with acquiring the source code and running the application. There are also details about necessary setup steps to allow the program to run correctly.

# Installation

## Prerequisites

To run the TFMS, the system running the application must have Python 2.7 installed. If not installed then go to the following website and download Python 2.7.0.

<https://www.python.org/download/releases/2.7/>

Follow the steps provided on the website to finish installation.

## Required Files

To download the source files for the TFMS go to the following GitHub repository:

<https://github.com/ttmarshall12/KSUMSE>

In the Phase 3 directory users will want to download the following list of files:

* TCPServer.py
* Database.py
* DataManager.py
* Users.py

## Setup

To be able to use the TCP server, users must obtain a private-public key pair. This can be done without using a for profit Certificate Authority. Details on minting your own certificate to sign your keys the follow the instructions provided at the following link:

<http://www.linux.org/threads/creating-a-self-signed-certificate-with-python.4591/>

The TCPServer.py must then be updated to point to the new certificate and key pair. Users will also need to update the TCPServer.py set the Host IP to the IP of the platform they are running the TFMS on.

# Running the TFMS Demo

Running the TFMS is as simple as running the TCPServer.py in a Python 2.7 Interpreter such as IDLE. At that time the server will set up the database and begin listening for user requests.

Using a SQL database editor, you can bypass the protocol with the server to add an administrator so that you can begin to send test requests.

To demonstrate the functionality of the server users can use the TCPClient.py file that can be found in the same GitHub repository as mentioned previously. Example requests are already included in this file and users can update the requests per the API below to test out functionality of the server.

# API

The following are the commands that can be sent by the user to the TFMS. The requests are formatted arrays of the command and parameters that are serialized using pickle. Literal strings are identified with quotes and parameters are listed in italics and will all be provided as strings. Command response pairs will be provided for each interface.

## Command Header

The command header is the start of each of the command messages. The command itself will be replace the string literal defined in each of the interfaces.

[*CommandString, CommandingUserName, CommandingUserID*]

## Add User

## Command

[CommandHeader(‘adduser’), *usertype, username, userid, userteam, usergender*]

## Response

[‘AddSuccess’]

## Get User

## Command

[CommandHeader(‘getuser’)*, username, userid*]

## Response

[UserType, UserName, UserID, UserTeam, UserGender] OR

[‘NoneFound’]

## Modify User

## Command

[CommandHeader(‘modifyuser’)*, originalusername, originaluserid, newusertype, newusername, newuserid, newuserteam, newusergender*]

## Response

[‘ModifySuccess] OR

[‘ModifyFail’]

## Add Results

## Command

[CommandHeader(‘addresults’)*, eventid, athletename, athleteid, athleteteam, attempt, value*]

## Response

[‘AddSuccess’] or

[‘AddFail’]

## Update Status

## Command

athleteid will be ignored when a user of type athlete sends this command. A coach will only be allowed to send this command for athlete on their team. Allowable values for status are: At Event, At Another Event, Did Not Show,

[CommandHeader(‘updatestatus’)*, eventid, athletename, athleteid, status*]

## Response

[‘StatusUpdateFail’] or

[‘StatusUpdateSuccess’]

## Other Failures

If the user sending the command can’t be authenticated with the TFMS then the following error response will occur.

[‘AuthenticationFail’]

If the user does not have the correct permissions for the command they issued, then the following error response will occur.

[‘PermissionFail’]

When a command is not recognized then the following error response will occur.

[‘UnknownFail’]