Technical Manual

SportsPA

Name: Gerard Dobbs

Student Number: C00196843

Course: Bachelor of Science (Hons) Software Development

Supervisor: Paul Barry

Date: 18/04/2018

Table of Contents

1. Introduction

This document will include all the code produced while working on the SportsPA project. The Python code in app.py will be shown first followed by each of the html files.

It will also explain how SportsPA can be installed by the user. It is intended that I will install SportsPA on PC's in the Sports Science department. However, this document will explain how it can be installed by a user

2. app.py

```
# StudentID:        C00196843
# StudentName:      Ger Dobbs
# Date:             18/04/2017
# Title:            Final Year Project - SportsPA

from bokeh.models import Text, ColumnDataSource, DataRange1d, Plot, LinearAxis,
Grid, \
    ResetTool, CrosshairTool, HoverTool, BoxZoomTool, TapTool, \
    CustomJS, ZoomInTool, ZoomOutTool, Div, WheelPanTool, WheelZoomTool,
BoxAnnotation
from bokeh.models.glyphs import HBar
import xml.etree.ElementTree as eTree
from flask import Flask, render_template, redirect, url_for, request, session
from bokeh.embed import components
from bokeh.resources import INLINE
from bokeh.models.widgets import Select, Button
from bokeh.layouts import column, row
from functools import wraps
import os

app = Flask(__name__)

colors = {
    'Black': '#000000',
    'Red': '#FF0000',
    'Green': '#00FF00',
    'Blue': '#0000FF',
}


def getitem(obj, item, default):
    if item not in obj:
        return default
    else:
        return obj[item]


"""For login required decorator"""
def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
```

```
        if 'logged_in' in session:
            return f(*args, **kwargs)
        else:
            return redirect(url_for('login'))
    return wrap


"""Decorator for displaying bokeh page"""
@app.route("/play", methods=['GET', 'POST'])
def play():
    Try:
        #The selected xml file
        file = str(request.form.get('xml_select'))
        #Get the team names from file
        team1 = file[0:3]
        team2 = file[4:7]
        tree = eTree.ElementTree(file=file)
        root = tree.getroot()
        rootInstance = root.find('ALL_INSTANCES')
        codes_set = set()
        codes_list = []
        codes_count = []
        start = []
        end = []
        # Used to store how many times each code appears
        count = 0
        code_list = []
        temp_dict = {}
        # Will include a list of each code x number of times it appears
        each_code = []
        all_codes = []
        # Will include the number assigned to the code x
        # how many times that code appears
        all_counts = []
        colors = []
        overall_code_count = 0
        all_starts = []
        all_ends = []
        plot_start = 200000
        plot_end = 0
        counts_list = []
        counts_list2 = []
        count1 = 0
        position = 0
        home_team = []
```

```
home_start = []
home_end = []
home_label = []
away_team = []
away_start = []
away_end = []
away_label = []
neither_team = []
neither_start = []
neither_end = []
neither_label = []
temp_list = []
team_color = ['Lime', 'Black']
all_list = []
label_list = []
list_of_all_labels = []
# Loop through xml file to find tags & add values to lists
for child in root:
    if child.tag == 'ALL_INSTANCES':
        for child2 in child:  # instance
            # ID,start,end, code, label
            for child3 in child2:
                if child3.tag == 'ID':
                    key = child3.text
                elif child3.tag == 'start':
                    all_list.append(child3.text)
                elif child3.tag == 'end':
                    all_list.append(child3.text)
                elif child3.tag == 'code':
                    all_list.append(child3.text)
                elif child3.tag == 'label':
                    for child4 in child3:  # group,text
                        label_list.append(child4.text)
            list_of_all_labels.append(label_list)
            label_list = []

"""Make a list of the total number of each code"""
def create_list_of_codes(codes, pos, count):
    for code in codes:
        # loop through rest of list for same code
        for code1 in codes[pos:len(codes)]:
            if code1 == code:  # if code matches count it
                count += 1
        pos = pos + count
        counts_list.extend([count] * count)
```

```
            counts_list2.append(count)
            count = 0

    """Function to convert seconds to h:m:s"""
    def convert_time(seconds):
        m, s = divmod(seconds, 60)
        h, m = divmod(m, 60)
        return "%d:%02d:%02d" % (h, m, s)



"""Get the count of how many distinctive codes there are"""
    for i in rootInstance.iterfind('instance'):
        if i.findtext('code') not in codes_set:
            # First time to see code
            codes_set.add(i.findtext('code'))
            overall_code_count += 1
    codes_set = set()
    count_used_for_title_height = overall_code_count
    theCodes = []

    """Extract all relevant data from xml file"""
    for i in rootInstance.iterfind('instance'):
        theCodes.append(i.findtext('code'))
        # First time to see code
        if i.findtext('code') not in codes_set:
            codes_set.add(i.findtext('code'))
            # Add new code to dictionary
            temp_dict['code'] = i.findtext('code')
            codes_list.append(i.findtext('code'))
            # Loop through codes again
            for j in rootInstance.iterfind('instance'):
                # Code == one in outer loop
                if i.findtext('code') == j.findtext('code'):
                    count += 1
                    start.append(
                        # Add start time to list of starts for that code
                        round(float(j.findtext('start'))))
                    if float(j.findtext('start')) < plot_start:
                        plot_start = float(j.findtext('start'))
                    end.append(round(float(j.findtext('end'))))
                    if float(j.findtext('end')) > plot_end:
                        plot_end = float(j.findtext('end'))
                # Make list of the code x number of times it appears
```

```
            each_code = [i.findtext('code')] * count
        # Add the list of each code to list of all codes
        all_codes.extend(each_code)
        temp_dict['count'] = count
        codes_count.append(count)
        # Add list of start times as value to start key in dict
        temp_dict['start'] = start
        temp_dict['end'] = end
        # Add full dict to list
        code_list.append(temp_dict.copy())
        # Clear for next iteration
        temp_dict.clear()
        all_starts.extend(start)
        all_ends.extend(end)
        start = []
        end = []
        all_counts.extend([overall_code_count] * count)
        count = 0
        overall_code_count -= 1
final_labels = []
temp_labels = []
for i in codes_list:
    for j in range(0, len(theCodes)):
        if theCodes[j] == i:
            temp_labels.append(list_of_all_labels[j])
    final_labels.extend(temp_labels)
    temp_labels = []

"""Reorder the data lists to enable
events to be ordered according to teams"""
pos = 0
for team in all_codes:
    if team1 in team:
        home_team.append(team)
        home_start.append(all_starts[pos])
        home_end.append(all_ends[pos])
        home_label.append(final_labels[pos])
    elif team2 in team:
        away_team.append(team)
        away_start.append(all_starts[pos])
        away_end.append(all_ends[pos])
        away_label.append(final_labels[pos])
    else:
        neither_team.append(team)
        neither_start.append(all_starts[pos])
```

```
            neither_end.append(all_ends[pos])
            neither_label.append(final_labels[pos])
        pos += 1
total_codes = []
total_codes.extend(home_team)
total_codes.extend(away_team)
total_codes.extend(neither_team)
total_starts = []
total_starts.extend(home_start)
total_starts.extend(away_start)
total_starts.extend(neither_start)
total_ends = []
total_ends.extend(home_end)
total_ends.extend(away_end)
total_ends.extend(neither_end)
total_labels = []
total_labels.extend(home_label)
total_labels.extend(away_label)
total_labels.extend(neither_label)
exclusive_codes_list = []
for code in total_codes:
    if code not in exclusive_codes_list:
        exclusive_codes_list.append(code)

"""Create lists of the individual
lists of codes, starts & ends"""
total_codes = []
total_codes.extend(home_team)
total_codes.extend(away_team)
total_codes.extend(neither_team)
total_starts = []
total_starts.extend(home_start)
total_starts.extend(away_start)
total_starts.extend(neither_start)
total_ends = []
total_ends.extend(home_end)
total_ends.extend(away_end)
total_ends.extend(neither_end)
exclusive_codes_list = []
count_each_code = 0
temp_list1 = []
for code in total_codes:
    if code not in exclusive_codes_list:
        exclusive_codes_list.append(code)
list_index_of_each_code = []
```

```
code_number = len(exclusive_codes_list)
for code in exclusive_codes_list:
    for code1 in total_codes:
        if code1 == code:
            count_each_code += 1
            list_index_of_each_code.append(count_each_code)
    temp_list1.extend([code_number] * count_each_code)
    count_each_code = 0
    code_number -= 1
colors.extend(["white"] * len(total_codes))

"""Get the details of where the team colour bands change"""
pos_count = 0
for code in total_codes:
    if team1 in code:
        pos_count += 1
top_lower = int(temp_list1[pos_count])
pos_count = 0
for code in total_codes:
    if team2 in code or team1 in code:
        pos_count += 1
bottom_higher = int(temp_list1[pos_count])

""" Make a list of the total number of each code"""
create_list_of_codes(total_codes, position, count1)

"""Convert start and end times from seconds to h:m:s"""
converted_start_times = []
converted_end_times = []
time_length = []
converted_time_length = []
for diff in range(0, len(total_ends)):
    time_length.append(int(total_ends[diff])
                       - int(total_starts[diff]))
for t in total_starts:
    converted_start_times.append(convert_time(t))
for t in total_ends:
    converted_end_times.append(convert_time(t))
for t in time_length:
    converted_time_length.append(convert_time(t))

"""Get a new count list of the codes
to match the rearranged codes per team"""
counts_list3 = []
```

```
"""Get rid of the 0's from previous list"""
for c in counts_list2:
    if c is not 0:
        counts_list3.append(c)

"""Get a list of the Starts for home"""
tempList = []
tempList1 = []
tempList2 = []
tempList3 = []
tempLabel = []
homeStart = []
homeLabels = []
homeLabels2 = []
homeEnd = []
eventDuration = []
pos3 = 0
for x in range(0, len(counts_list3)):
    pos2 = counts_list3[x]
    for y in range(pos3, pos3 + pos2):
        tempList.append(convert_time(total_starts[y]))
        tempList1.append(convert_time(total_ends[y]))
        tempList2.append(convert_time(total_ends[y]
                                      - total_starts[y]))
        tempList3.append(list_of_all_labels[y])
        tempLabel.append(total_labels[y])
    homeStart.append(tempList)
    homeEnd.append(tempList1)
    eventDuration.append(tempList2)
    homeLabels.append(tempList3)
    homeLabels2.append(tempLabel)
    pos3 = pos3 + pos2
    tempList = []
    tempList1 = []
    tempList2 = []
    tempList3 = []
    tempLabel = []

"""Create list of duration of each event &
Combine Start & End times into strings for Select options"""

startAndEndTimes = []
counti = 0
for i in homeStart:
    countj = 0
```

```
        for j in i:
            temp_list.append(homeStart[counti][countj]
                             + " - " + homeEnd[counti][countj] +
                             '    (' + eventDuration[counti][countj] + ') '
                             + str(countj + 1))
            countj += 1
        startAndEndTimes.append(temp_list)
        counti += 1
        temp_list = []

    """Create html divs to contain details
    of the event on screen as video plays"""
    divStart = Div(id='divStart')
    divEnd = Div()
    divEvent = Div()
    divEventNum = Div()
    divLabel = Div()
    divBreak = Div(width=20)
    divButton = Div()
```

```
"""The event that will be called
    to play the video on hbar click
    Function to build a suitable CustomJS
    to display the current event in the div model."""
    data = {'x': total_ends, 'y': temp_list1,
            'left': total_starts, 'colors': colors,
            'all_codes': total_codes,
            'counts_list': counts_list,
            'list_index_of_each_code': list_index_of_each_code,
            'list_of_all_labels': total_labels,
            'converted_start_times': converted_start_times,
            'converted_end_times': converted_end_times,
            'converted_time_length': converted_time_length}
    source = ColumnDataSource(data)

"""Function called when bar in Graph is"""

    def display_event(div, divStart, divEnd,
                      divEvent, divEventNum, divLabel,
```

```
                        attributes=None):
            if attributes == None:
                attributes = []
            style = 'float:left;clear:left;font_size=32.5pt'
            return CustomJS(args=dict(div=div,
                                      divStart=divStart, divEnd=divEnd,
                                      divEvent=divEvent,
                                      divEventNum=divEventNum,
divLabel=divLabel,
                                      source=source), code="""
                var data = source.data,
                //gets id of selected point
                selected = source.selected['1d']['indices'],
                select_inds = [selected[0]];
                if(selected.length == 1){
                    selected_x = data['x'][selected[0]]
                    selected_y = data['y'][selected[0]]
                    selected_left = data['left'][selected[0]]
                    selected_color = data['colors'][selected[0]]
                    selected_code = data['all_codes'][selected[0]]
                    selected_list_index_of_each_code =
                    data['list_index_of_each_code'][selected[0]]
                    selected_counts_list = data['counts_list'][selected[0]]
                    selected_converted_start_times =
                    data['converted_start_times'][selected[0]]
                    selected_converted_end_times =
                    data['converted_end_times'][selected[0]]
                    selected_converted_time_length =
                    data['converted_time_length'][selected[0]]
                    if(data['list_of_all_labels'][selected[0]]=="")
                        selected_list_of_all_labels = "NO LABELS";
                    else
                        selected_list_of_all_labels =
                        data['list_of_all_labels'][selected[0]];
                    //Get the times in seconds
                    var date = new Date(null);
                    date.setSeconds(selected_left);
                    start = date.toISOString().substr(11, 8);
                    var date = new Date(null);
                    date.setSeconds(selected_x);
                    end = date.toISOString().substr(11, 8);
                    var date = new Date(null);
                    date.setSeconds(selected_converted_time_length);
                    //duration = date.toISOString().substr(11, 8);
                    divStart.text = "Start = "+start+"    ";
```

```
                    divEnd.text = "End = "+end+"    ";
                    divEvent.text = selected_code + "\\n\\n";
                    divEventNum.text = "Event No."+
                    selected_list_index_of_each_code+" of "
                    + selected_counts_list+"  "
                    divLabel.text = " Labels: "  + "\\n"
                    + selected_list_of_all_labels;
                }
                var attrs = %s;
                var args = [];
                for (var i=0; i<attrs.length; i++ ) {
                    val = JSON.stringify(cb_obj[attrs[i]],
                     function(key, val) {
                        return val.toFixed ? Number(val.toFixed(2)) : val;
                    })
                    args.push(attrs[i] + '=' + val)
                }
                var line = "<span style=%r><b>" +
                cb_obj.event_name + "</b>(" +
                args.join(", ") + ")</span>\\n";
                var text = divStart.text.concat(line);
                //Get start & end times of video from left & right values of
hbar
                var startInt = parseInt(selected_left);
                var start = ((parseInt(selected_left))).toString();
                var duration = (parseInt(selected_x)
                    -parseInt(selected_left));
                var end = ((parseInt(selected_x))).toString();
                var vid = document.getElementById("myVideo");
                vid.currentTime = start;
                vid.play();
                vid.addEventListener("timeupdate", function() {
                    if (vid.currentTime >= end) {
                        vid.pause();
                        start = 0;
                        end = 900000;
                    }
                }, false);
            """ % (attributes, style))

        """Callback to update details in Time
        Select when Event is Selected"""
        dataSelect = {'y': startAndEndTimes}
        sourceSelect = ColumnDataSource(dataSelect)
        callbackCodes = CustomJS(args=dict(source=sourceSelect,
```

```
                                        divEnd=divEnd,
                                        divEvent=divEvent), code="""
            var data = source.data;
            var codeList = document.getElementById("code");
            var starts = data['y'][codeList.selectedIndex];
            s2.set('options', starts);
            s2.trigger('change');
        """)

        """Callback to play Video & display
        details when Time Selected"""
        dataSelectTime = {'homeLabels': homeLabels2}
        sourceSelectTime = ColumnDataSource(dataSelectTime)
        callbackTimes = CustomJS(args=dict(source=sourceSelectTime,
                                        divStart=divStart, divEnd=divEnd,
                                        divEventNum=divEventNum,
                                        divEvent=divEvent,
divLabel=divLabel), code="""
            var data = source.data;
            selected = source.selected['1d']['indices'],
            select_inds = [selected[0]];
            var codeList = document.getElementById("code");
            var timeList = document.getElementById("time");
            var selectedTime = timeList.options[timeList.selectedIndex].value;
            var selectedStart = selectedTime.substring(0,7);
            var selectedEnd = selectedTime.substring(10,18)
            var selectedCode = codeList.options[0].value;
            var selectedEventNum = selectedTime.substring(30)
            var startSplit = selectedStart.split(':');
            var endSplit = selectedEnd.split(':');
            var start = (+startSplit[0]) * 60 * 60 +
                (+startSplit[1]) * 60 + (+startSplit[2]);
            var end = (+endSplit[0]) * 60 * 60
                + (+endSplit[1]) * 60 + (+endSplit[2]);

if(data['homeLabels'][codeList.selectedIndex][timeList.selectedIndex]=="")
                        selected_list_of_all_labels = "NO LABELS";
                    else
                        selected_list_of_all_labels =

data['homeLabels'][codeList.selectedIndex][timeList.selectedIndex];
            divStart.text = "Start = "+selectedStart+"   ";
        divEnd.text = "End = "+selectedEnd+"   ";
        divEvent.text = "Event = "+selectedCode;
        divEventNum.text = "Event No. "+selectedEventNum+" of "+
```

```
    timeList.options.length;
divLabel.text = "Label: "+selected_list_of_all_labels;
    var vid = document.getElementById("myVideo");
    vid.currentTime = start;
    vid.play();
    vid.addEventListener("timeupdate", function() {
        if (vid.currentTime >= end) {
            vid.pause();
            start = 0;
            end = 90000;
        }
    }, false);
""")

callbackButton = CustomJS(args=dict(source=sourceSelectTime,
                                    div=divButton,
                                    divStart=divStart, divEnd=divEnd),
                          code="""
    var divStartTime = divStart.text.substring(8,16);
    var divEndTime = divEnd.text.substring(6,14);
    var startSplit = divStartTime.split(':');
    var endSplit = divEndTime.split(':');
    var start = (+startSplit[0]) * 60 * 60
        + (+startSplit[1]) * 60 + (+startSplit[2]);
    var end = (+endSplit[0]) * 60 * 60
        + (+endSplit[1]) * 60 + (+endSplit[2]);
    var vid = document.getElementById("myVideo");
    vid.currentTime = start;
    vid.play();
    vid.addEventListener("timeupdate", function() {
        if (vid.currentTime >= end) {
            vid.pause();
            start = 0;
            end = 90000;
        }
    }, false);
""")

"""Create the dropdown Select boxes"""
Times = Select(title="Times", id="time",
               value=startAndEndTimes[0][0],
               options=startAndEndTimes[0],
               callback=callbackTimes)
Codes = Select(title="Codes", id="code",
               value=exclusive_codes_list[0],
```

```
                        options=exclusive_codes_list,
                        callback=callbackCodes)

        """Send times dropdown to callback for codes"""
        callbackCodes.args["s2"] = Times

        Replay = Button(id="Button", label="Replay",
                        button_type="success",
                        callback=callbackButton)

        hover = HoverTool(tooltips="""
                <style>.zoom:hover{transform:scale(1.5);}</style>
                <div bgcolor="#E6E6FA" class = "zoom">
                    <span style="font-size: 10px;
                        font-weight: bold;">Code = @all_codes</span></br>
                    <span style="font-size: 10px; font-weight: bold;">
                    Total number =
@list_index_of_each_code/@counts_list</span></br>
                    <span style="font-size: 8px; color: #966;">
                        Start = @converted_start_times</span></br>
                    <span style="font-size: 8px; color: #966;">
                        End = @converted_end_times</span></br>
                    <span style="font-size: 8px; color: #966;">
                    Duration: = @converted_time_length</span></br>
                </div> """)

        xdr = DataRange1d()
        ydr = DataRange1d()
        plot_height = 550
        plot = Plot(
            title=None, x_range=xdr, y_range=ydr,
            plot_width=1190, plot_height=plot_height - 200,
            h_symmetry=False, v_symmetry=False, min_border=0,
            toolbar_location="right",
            background_fill_color="green")

        """Get the locations on plot for colour bands"""
        low_box = BoxAnnotation(top=bottom_higher,
                                fill_alpha=0.3,
                                fill_color='Green')
        mid_box = BoxAnnotation(bottom=bottom_higher,
                                top=top_lower, fill_alpha=0.5,
                                fill_color='#00CC00')
        high_box = BoxAnnotation(bottom=top_lower, fill_alpha=0.4,
                                 fill_color='Lime')
```

```
"""Add tools to plot"""
plot.add_layout(low_box)
plot.add_layout(mid_box)
plot.add_layout(high_box)
plot.add_tools(ResetTool())
plot.add_tools(hover)
plot.add_tools(WheelPanTool())
plot.add_tools(BoxZoomTool())
plot.add_tools(CrosshairTool())
plot.add_tools(ZoomOutTool())
plot.add_tools(ZoomInTool())
plot.add_tools(WheelZoomTool())
glyph = HBar(y="y", right="x",
             left="left", height=0.5,
             fill_color="colors")
plot.add_glyph(source, glyph)

"""Set the plot ticks"""
xaxis = LinearAxis()
yaxis = LinearAxis()
plot.add_layout(Grid(dimension=0,
                     ticker=xaxis.ticker))
plot.add_layout(Grid(dimension=1,
                     ticker=yaxis.ticker))

yaxis_offset = 0.5
code_index = len(codes_list) - 1
for code in exclusive_codes_list:
    text = Text(x=-470, y=yaxis_offset,
                text=[exclusive_codes_list[code_index][0:30]],
                text_font_size="5pt",
                text_font_style="normal",
                text_color="White", text_baseline="alphabetic")
    plot.add_glyph(text)
    yaxis_offset += 1.01
    code_index -= 1

"""Set the title from the file names"""
home_title = Text(x=-470, y=count_used_for_title_height + 2,
                  text=[team1], text_font_size="8pt",
                  text_font_style="bold",
                  text_alpha=1.0, text_font='helvetica',
                  text_color=team_color[0],
                  text_baseline="alphabetic")
```

```
versus = Text(x=-330, y=count_used_for_title_height + 2,
              text=['v'], text_font_size="8pt",
              text_font_style="bold",
              text_alpha=1.0, text_font='helvetica',
              text_color='White', text_baseline="alphabetic")
away_title = Text(x=-280, y=count_used_for_title_height + 2,
                  text=[team2], text_font_size="8pt",
                  text_font_style="bold",
                  text_alpha=1.0, text_font='helvetica',
                  text_color=team_color[1],
                  text_baseline="alphabetic")

"""Add the glyphs ro the plot"""
plot.add_glyph(home_title)
plot.add_glyph(versus)
plot.add_glyph(away_title)
plot.ygrid.grid_line_color = "navy"
plot.ygrid.minor_grid_line_color = 'navy'
plot.ygrid.minor_grid_line_alpha = 0.1
plot.ygrid.grid_line_alpha = 0.1
plot.xgrid.grid_line_color = None
plot.axis.major_tick_out = 3
plot.axis.minor_tick_in = -3
plot.axis.minor_tick_out = 2

# Grab the inputs arguments from the URL
args = request.args
# Get all the form arguments in the url with defaults
color = getitem(args, 'color', 'Black')

"""Get the resources for embedding the plot in html page"""
js_resources = INLINE.render_js()
css_resources = INLINE.render_css()

"""Get the video that was selected
and put it into a Div"""
video_select = str(request.form.get('video_select'))
div = Div(text=f"""<video id="myVideo"
width="540" height="310" controls>
            <source id="mySrc"
            src= '/static/video/{video_select}'
            type="video/mp4">
            </video>""", width=540, height=310)

"""Create and add the tap tool which
```

```
        will play video when clicked"""
        taptool = TapTool(callback=display_event(div,
            divStart, divEnd, divEvent, divEventNum, divLabel))
        plot.add_tools(taptool)

        """Set the layout for all the components
        and embed in html page"""
        layout = column(
            row(div, column(divBreak),
                column(Codes, Times, divButton, Replay),
                column(divStart, divEnd, divEvent,
                       divEventNum, divLabel)), plot)
        script, div1 = components(layout)
        return render_template('embed.html',
                               plot_script=script,
                               plot_div=div1,
                               js_resources=js_resources,
                               css_resources=css_resources,
                               color=color
                               )

    except:
        """If an error occurs repopulate
        the dropdowns and send error message"""
        error = None
        """Read XML & Video files from static folder"""
        video_path = 'static/video'
        video_list = []
        xml_path = 'static/xml'
        xml_list = []
        video_listing = os.listdir(video_path)
        xml_listing = os.listdir(xml_path)
        for infile in video_listing:
            video_list.append({'name': infile})
        for infile in xml_listing:
            xml_list.append({'name': infile})
        return render_template('login1.html',
                               error=error,
                               video_data=video_list,
                               xml_data=xml_list)


"""Route for handling the login page """
@app.route("/", methods=['GET', 'POST'])
def login():
```

```
    error = None
    if request.method == 'POST':
        if request.form['username'] != 'admin' or \
                        request.form['password'] != 'admin':
            error = 'Invalid Credentials. Please try again.'
        else:
            session['logged_in'] = True
            return redirect(url_for('upload'))
    """Read XML & Video files from static folder,
    put into lists & send to html page"""
    video_path = 'static/video'
    video_list = []
    xml_path = 'static/xml'
    xml_list = []
    video_listing = os.listdir(video_path)
    xml_listing = os.listdir(xml_path)
    for infile in video_listing:
        video_list.append({'name': infile})
    for infile in xml_listing:
        xml_list.append({'name': infile})
    return render_template('login1.html',
                           error=error,
                           video_data=video_list,
                           xml_data=xml_list)


"""Route for handling logout"""
@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    session.pop('logged_in', None)
    return redirect(url_for('login'))


@app.route('/login', methods=['GET', 'POST'])
def logins():
    error = ""
    if request.method == 'POST':
        if request.form['username'] != 'JBradley' or \
                        request.form['password'] != 'JBradleyPassword':
            error = "Login Unsuccessful!"
            pass
        else:
            session['logged_in'] = True
            return redirect(url_for('upload'))
```

```
    return render_template('login.html', error=error)


"""Route for uploading files"""
@app.route("/upload", methods=['GET', 'POST'])
@login_required
def upload():
    error = ""
    try:
        """Read XML & Video files from static folder"""
        video_path = 'static/video'
        video_list = []
        xml_path = 'static/xml'
        xml_list = []
        video_listing = os.listdir(video_path)
        xml_listing = os.listdir(xml_path)
        for infile in video_listing:
            video_list.append({'name': infile})
        for infile in xml_listing:
            xml_list.append({'name': infile})
        """Get path to project"""
        app_root = os.path.dirname(os.path.abspath(__file__))
        """Get a list of the files from form to be uploaded"""
        for file in request.files.getlist("file"):
            """Get file name from file object"""
            filename = file.filename
            name, ext = os.path.splitext(filename)
            if ext == '.mp4':
                target = os.path.join(app_root, 'static/video/')
            else:
                target = os.path.join(app_root, 'static/xml/')
                destination2 = "/".join([app_root, filename])
                file.save(destination2)
            destination = "/".join([target, filename])
            file.save(destination)
            error = "Upload Successful"
        return render_template('upload.html', error=error)
    except:
        error = "Upload Unsuccessful"
        return render_template('upload.html', error=error)


"""Route for deleting files"""
@app.route("/delete", methods=['GET', 'POST'])
@login_required
```

```
def delete():
    """Read XML & Video files from static folder"""
    file_select = str(request.form.get('file_select'))
    error = ""
    video_path = 'static/video'
    video_list = []
    xml_path = 'static/xml'
    xml_list = []
    video_listing = os.listdir(video_path)
    xml_listing = os.listdir(xml_path)
    for infile in video_listing:
        video_list.append({'name': infile})
    for infile in xml_listing:
        xml_list.append({'name': infile})
    app_root = os.path.dirname(os.path.abspath(__file__))
    name, ext = os.path.splitext(file_select)

    if ext == '.mp4':
        delete_path = video_path
    else:
        delete_path = xml_path
    return render_template('delete.html', error=error,
                           video_data=video_list,
                           xml_data=xml_list)


@app.route("/delete1", methods=['GET', 'POST'])
@login_required
def delete1():
    video_path = 'static/video'
    video_list = []
    xml_path = 'static/xml'
    xml_list = []
    video_listing = os.listdir(video_path)
    xml_listing = os.listdir(xml_path)
    for infile in video_listing:
        video_list.append({'name': infile})
    for infile in xml_listing:
        xml_list.append({'name': infile})
    app_root = os.path.dirname(os.path.abspath(__file__))
    try:
        file_select = str(request.form.get('file_select'))
        name, ext = os.path.splitext(file_select)

        if ext == '.mp4':
```

```
            delete_path = video_path
        else:
            delete_path = xml_path
        if os.path.isfile(delete_path + "/" + file_select):
            os.remove(delete_path + "/" + file_select)
            error = "Deletion Successful"
            video_path = 'static/video'
            video_list = []
            xml_path = 'static/xml'
            xml_list = []
            video_listing = os.listdir(video_path)
            xml_listing = os.listdir(xml_path)
            for infile in video_listing:
                video_list.append({'name': infile})
            for infile in xml_listing:
                xml_list.append({'name': infile})
        else:
            error = "No File found for Deletion "
            return render_template('delete.html',
                                   error=error,
                                   video_data=video_list,
                                   xml_data=xml_list)
        return render_template('delete.html', error=error,
                               video_data=video_list,
                               xml_data=xml_list)
    except:
        error = "No File found for Deletion"
        return render_template('delete.html',
                               error=error,
                               video_data=video_list,
                               xml_data=xml_list)


if __name__ == "__main__":
    app.run(port=5000, debug=True)
```

3. templates/base.html

```
<!DOCTYPE html>
<html>
<head>
   <title>SportsPA</title>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head><!--<body style="background:url(/static/css/hockey.png); ">-->

{% block content %}{% endblock %}

</html>
```

4. templates/delete.html

```
{% extends "base.html" %}
{% block content %}

<body style="background:url(/static/css/hockey.png); ">
<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#">SportsPA</a>
        </div>
        <ul class="nav navbar-nav">
            <li class=""><a href="/logout">Logout</a></li>
            <li class=""><a href="/upload">Upload</a></li>
            <li class="active"><a href="/delete">Delete</a></li>
        </ul>
    </div>
</nav>
<div class="container">
    <div class="row">
        <div class="col-lg-8 col-lg-offset-2">
            <h1>Welcome to SportsPA</h1>
            <p class="lead"></p>
            <form method="POST" action="{{ url_for('delete1') }}">
                <div class="controls">
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                            <!--    <label><b>Uploaded Files</b></label>-->
                            </div>
                        </div>
                    </div>
                </div>
                <div class="controls">
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="file_select">Select a Video file:</label>
                                <select name="file_select" id="file_select" class="selectpicker
```

```
form-control">
                        {% for o in video_data %}
                        <option value="{{ o.name }}">{{ o.name }}</option>
                        {% endfor %}
                        {% for o in xml_data %}
                        <option value="{{ o.name }}">{{ o.name }}</option>
                        {% endfor %}
                    </select>
                    <div class="help-block with-errors"></div>
                </div>
            </div>
        </div>
        <div class="row">

            <div class="col-md-12">
                <input type="submit" class="btn btn-success btn-send" value="DELETE">
            </div>
        </div>
        <div class="row">

            <div class="col-md-12">
                <p></p>
                <h2>{{ error }}</h2>
            </div>
        </div>
    </div>
  </form>
 </div>
</div>
</body>

{% endblock %}
```

5. templates/embed.html

```
{% extends "base.html" %}
{% block content %}


<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#">SportsPA</a>
        </div>
        <ul class="nav navbar-nav">
            <li class=""><a href="/">Home</a></li>
            <li><a href="/login">Login</a></li>
        </ul>
    </div>
</nav>

{{ js_resources|indent(4)|safe }}
{{ css_resources|indent(4)|safe }}
{{ plot_script|indent(4)|safe }}
{{ plot_div|indent(4)|safe }}

{% endblock %}
```

6. templates/login.html

```
{% extends "base.html" %}
{% block content %}

<body style="background:url(/static/css/hockey.png) center center fixed;
background-repeat:no-repeat; background-size:cover; ">
<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#">SportsPA</a>
        </div>
        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li class="active"><a href="/login">Login</a></li>
        </ul>
    </div>
</nav>
<div class="container">
    <div class="row">
        <div class="col-lg-8 col-lg-offset-2">
            <h1>Welcome to SportsPA</h1>
            <p class="lead"></p>
            <form method="POST" action="/login">
                <div class="messages"></div>
                <div class="controls">
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="username">Username</label>
                                <input type="text" id="username" class="form-control" placeholder="Enter
Username"
                                       name="username" value="{{ request.form.username }}" required
autofocus >
                                <div class="help-block with-errors"></div>
                            </div>
                        </div>
                    </div>
                    <div class="row">
```

```
                    <div class="col-md-6">
                        <div class="form-group">
                            <label for="password">Password</label>
                            <input id="password" type="password" class="form-control"
autocomplete="off"
                                   placeholder="Enter Password" name="password" value="{{
request.form.password }}"
                                   required>
                            <div class="help-block with-errors"></div>
                        </div>
                    </div>
                    <div class="col-md-12">
                        <input type="submit" class="btn btn-success btn-send" value="LOGIN">
                    </div>
                    <div class="row">

                    <div class="col-md-12">
                        <p></p>
                        <h2>{{ error }}</h2>
                    </div>
                </div>
                </div>
            </div>
        </form>
    </div>
  </div>
</div>
</body>

{% endblock %}
```

7. templates/index.html

```
{% extends "base.html" %}
{% block content %}

<body style="background:url(/static/css/hockey.png); ">
<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#">SportsPA</a>
        </div>
        <ul class="nav navbar-nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="/login">Login</a></li>
        </ul>
    </div>
</nav>
<div class="container">
    <div class="row">
        <div class="col-lg-8 col-lg-offset-2">
            <h1>Welcome to SportsPA</h1>
            <p class="lead"></p>
            <form method="POST" action="{{ url_for('play') }}">
                <div class="messages"></div>
                <div class="controls">
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="video_select">Select a Video file:</label>
                                <select name="video_select" id="video_select" class="selectpicker
form-control">
                                    {% for o in video_data %}
                                    <option value="{{ o.name }}">{{ o.name }}</option>
                                    {% endfor %}
                                </select>
                                <div class="help-block with-errors"></div>
                            </div>
                        </div>
                    </div>
```

```
                <div class="row">
                    <div class="col-md-6">
                        <div class="form-group">
                            <label for="xml_select">Select an xml file:</label>
                            <select name="xml_select" id="xml_select" class="selectpicker
form-control">
                                {% for o in xml_data %}
                                <option value="{{ o.name }}">{{ o.name }}</option>
                                {% endfor %}
                            </select>
                            <div class="help-block with-errors"></div>
                        </div>
                    </div>
                    <div class="col-md-12">
                        <input type="submit" class="btn btn-success btn-send" value="PLAY">
                    </div>
                </div>
            </div>
        </form>
    </div>
  </div>
</div>
</body>

{% endblock %}
```

8. templates/upload.html

```
        {% extends "base.html" %}
{% block content %}

<body style="background:url(/static/css/hockey.png); ">
<nav class="navbar navbar-default">
   <div class="container-fluid">
      <div class="navbar-header">
         <a class="navbar-brand" href="#">SportsPA</a>
      </div>
      <ul class="nav navbar-nav">
         <li class=""><a href="/logout">Logout</a></li>
         <li class="active"><a href="/upload">Upload</a></li>
         <li class=""><a href="/delete">Delete</a></li>
      </ul>
   </div>
</nav>
<div class="container">
   <div class="row">
      <div class="col-lg-8 col-lg-offset-2">
         <h1>Welcome to SportsPA</h1>
         <p class="lead"></p>
         <form id=upload-form" action="{{ url_for('upload') }}" method="POST"
enctype="multipart/form-data">
            <div class="controls">
               <div class="row">
                  <div class="col-md-6">
                     <div class="form-group">
                     <!--    <label><b>Uploaded Files</b></label>-->
                     </div>
                  </div>
               </div>
            </div>
            <div class="controls">
               <div class="row">
                  <div class="col-md-6">
                     <div class="form-group">
                      <!--   {% for o in video_data %}
```

```
                        {{ o.name }}
                        {% endfor %}

            ;
                        {% for o in xml_data %}
                        {{ o.name }}
                        {% endfor %}
                        <p></p>-->
                        <label for="upload">File(s) to upload</label>
                        <input type="file" id="upload" class="form-control" name="file"
                             accept="video/mp4,text/xml " multiple>
                        <div class="help-block with-errors"></div>
                      </div>
                    </div>
                  </div>
                  <div class="row">
                    <div class="col-md-12">
                      <input type="submit" class="btn btn-success btn-send"
value="UPLOAD">
                    </div>
                  </div>
                  <div class="row">

                    <div class="col-md-12">
                      <p></p>
                      <h2>{{ error }}</h2>
                    </div>
                  </div>
                </div>
              </form>
            </div>
          </div>
        </div>
        </body>

        {% endblock %}
```

9. Installation Instructions

SportsPA can be installed onto a PC by following these instructions.

1. Download Python 3.6. from https://www.python.org/downloads/release/python-360/

2. In Command Prompt run the following commands:

    pip3 install flask
    Pip3 install bokeh

3. Copy SportsPA from usb into location of your choice on PC.

4. Open up command prompt in location where SportsPA is located.

5. Run the following command:

    python app.py

6. Go to browser and type the following in the address bar:

    localhost:5000

7. Browser will now display the home page.