Design Document for Machine Learning for March Madness

Alex Hoffer, Chongxian Chen, and Jacob Smith
Team Name: Stat Champs

TABLE OF CONTENTS

I. Introduction

Stat Champs
December 2, 2016

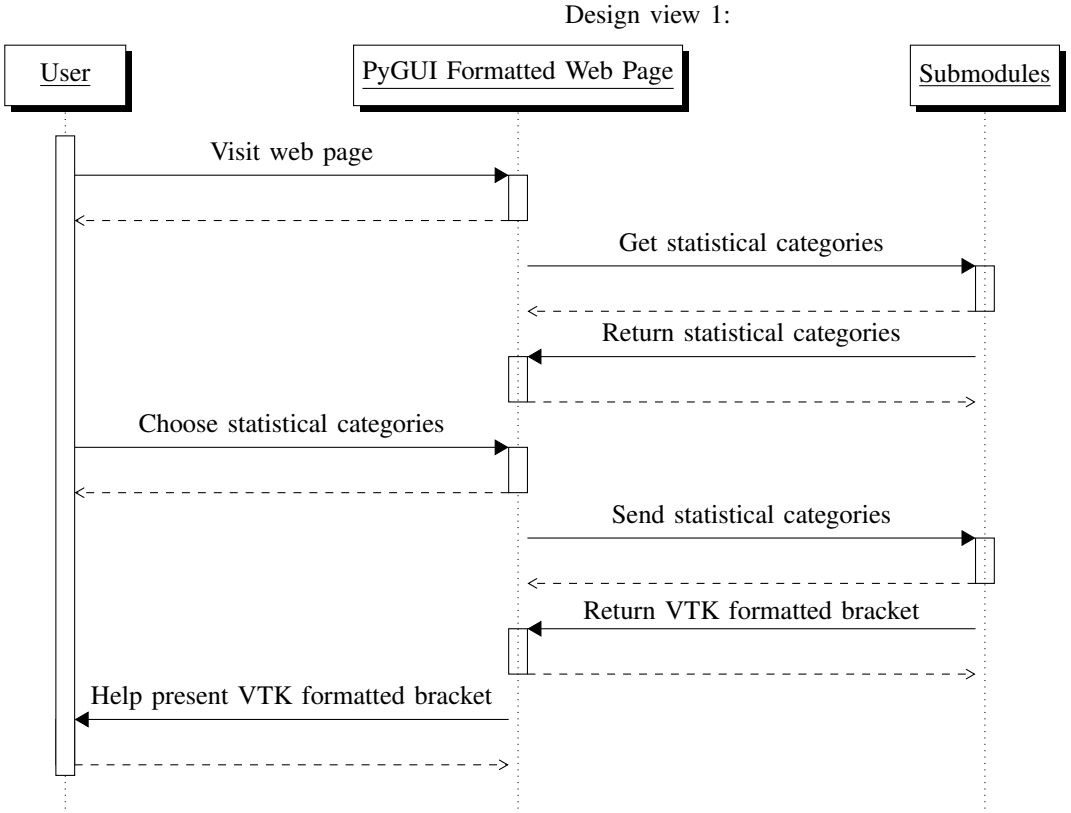
AFTER making reasoned decisions about which technologies to use to implement this module, we must now consider how these technologies will work together. This design document intends to explain how the three technologies each of the Stat Champs have selected to use will complete their assigned task. It will do so through the use of diagrams followed by paragraphs describing what the technology must do and why it must do it. The first three designs were produced by Alex Hoffer. These first three designs are chiefly concerned with the usability of the service, specifically the design of the graphical user interface, the instructions that will be presented to the user, and the display of the machine learning bracket that is generated. In technologies 4 and 5 Jake Smith will talk about how we will obtain and store the sports statistics for easy use by our machine learning algorithms. It is our belief that in order for the tool to be effective in educating students it must be well designed and easy to understand. Chongxian Chen will be responsible for designing and implementing machine learning algorithm. Technologies 6, 7 and 8 will be discussing technology involved in designing machine learning algorithm, namely algorithm library, statistics model and cloud server for hosting the project

II. Glossary

PyGUI: Graphical user interface API designed specifically for use with the Python programming language.
VTK: Data visualization API that will be used with the Python programming language.
Client-server architecture: Website format that consists of a user sending and receiving data to a server which also sends and receives data.
Webix: JavaScript API that is designed to support graphical user interfaces.

III. Designs

Design viewpoint 1: Using PyGUI for GUI of webpage

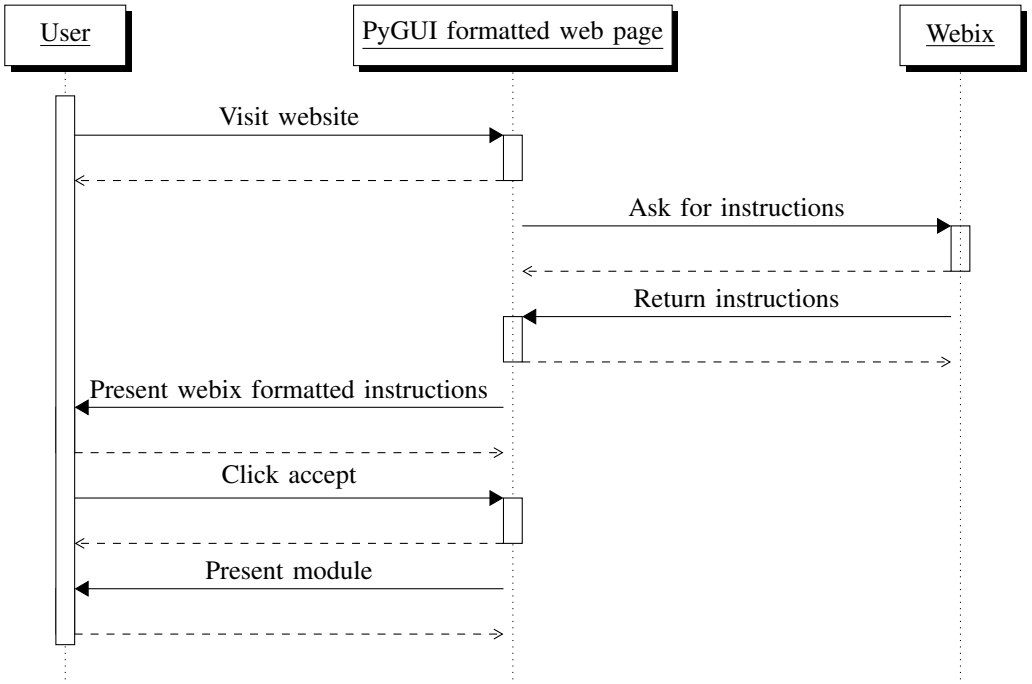


Design Rationale: The module needs a web page to operate. This web page needs to be presented in a way that is pleasing to the eye and easy to use, because our intended users are biochemists, not computer scientists. This means the page needs to be as non-esoteric as possible. To achieve this, we will be using PyGUI to format our web page. The precondition of this sequence diagram is that the user has a browser. The postcondition of this diagram are that the user is presented with a

bracket that resulted from collaboration between PyGUI and VTK. The flow of events from the perspective of PyGUI is quite simplistic. While a typical client-server architecture will allow us to transfer data back and forth between our submodules, PyGUI must make this data transmission appear as convenient as possible to the user.

Design viewpoint 2: Using Webix to present instructions for the module

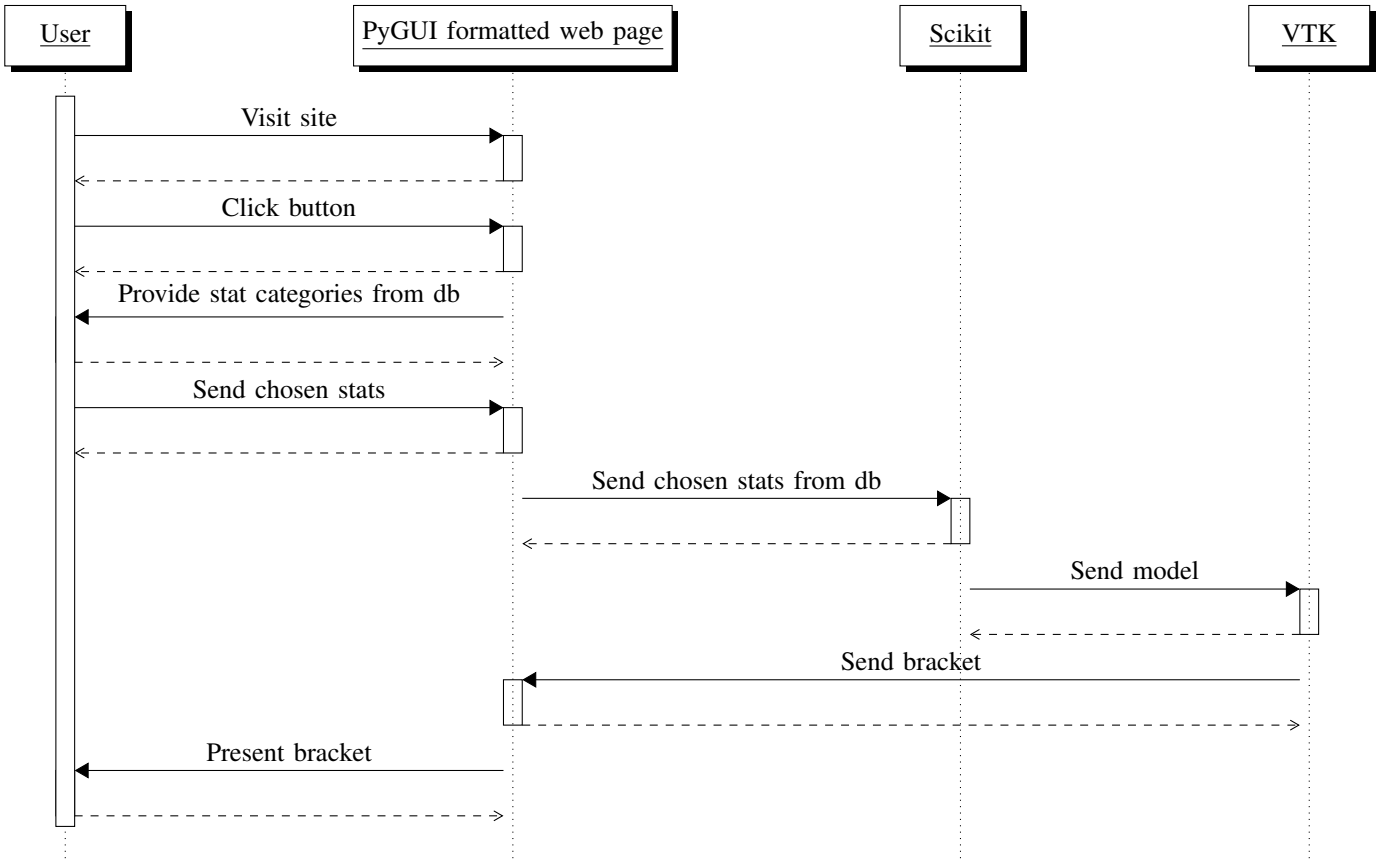
Design view 2:



Design Rationale: The module needs to present instructions to the user which informs them how to use the service. These instructions should appear seamlessly and should be visually inoffensive. Our main aim is reducing the chance of these instructions appearing unclear and thus maximizing utility. To achieve this, we will be using Webix, which will interact with our client/server code as well as with PyGUI. The precondition of this sequence diagram is that the user has a browser and has loaded the web page. The postconditions are that the user has seen the instructions (hopefully having read them) and has clicked accept to begin the service. The flow of events of the average use case are as follows: 1) the user visits the web page 2) the user is presented with instructions on properly using the module 3) the user reads the instructions 4) the user clicks accept 5) the module is presented. Note that the PyGUI formatted page in this diagram is a bit vague. PyGUI, of course, cannot handle the entire module. The web page will be calling a number of other technologies to present the module, but these technologies are not necessary when only considering how Webix will be used.

Design viewpoint 3: Generating bracket using VTK

Design view 3:



Design Rationale: The most essential function of the entire module is presenting a machine learned March Madness bracket. However, the process of machine learning is completely separate from the design of the bracket which will be displayed. Once Scikit generates a machine learned model that reflects the user's chosen statistics, it must communicate to VTK what this data is, and VTK will handle the bracket generation. In order for VTK to produce a bracket for display, a number of preconditions must occur. First, the user must have read and accepted the instructions. Then, the user must have selected data produced by a database. This data must be passed to Scikit, which generates a learned model. The postconditions of this sequence diagram is that the user will have seen the March Madness bracket that results from their chosen statistics. The flow of events for the average use case is as follows: 1) The user reads and accepts the instructions 2) The user selects statistics 3) The server sends these statistics to a machine learning submodule 4) The submodule generates a model 5) The submodule passes the model to VTK 6) VTK transforms the model into a bracket 7) A collaboration of VTK, PyGUI, and the server produce this bracket for the user to view.

Design viewpoint 4: Collection of data for the database

Design Rationale: The Machine Learning algorithms we want to work with get more accurate the more examples and stats you feed into it. Therefore, for the data collection we need to include as many possible data points as possible. We will do this by collecting stats from every major stat website and also pull more advanced data from the hoop-math which breaks down each players moves into more specialized data points. For example, hoop-math with break down a players shots from just saying he shot a 2 pointer to telling you where the shot was and how close a defender was to him at that moment. The design to grab the stats is to manually download all the csv files I can from Sports-Reference and upload them to the database then once I am caught up with the current season I will create a python script that takes the stats from all the daily games and uploads it into the database.

Design viewpoint 5: Database Design

Design view 4: Couldnt get UML database design to post correctly. Design Rationale:

For designing the database I have decided to have multiple connected databases. The fist will be a running tally for all the players stats, everything from FG percentage to player efficiency and more. The second will be all the team game logs with their opponents stats for that game as well. That way we can do look ups for past matchups and analyze how both teams and players did against each other and apply that for future matchups. Then for the final database that will be connected to the game logs will be the advanced game player stats for example it will have play by play stats like who passed to who, when and who shot the ball, from where, and who was guarding him. The user will be able to choose between either all three databases in there bracket analysis or just two or even one. Each bracket should be able to bring in stats that should change the outcome of the machine learning predictions.

IV. AGREEMENT

Client

Developer

Developer

Developer

REFERENCES

[1] Kivy: Cross-platform Python Framework for NUI, *Kivy*. [Online]. Available: https://kivy.org/. [Accessed: 14-Nov-2016].

[2] D. Bolton, 5 Top Python GUI Frameworks for 2015 - Dice Insights, *Dice*, Feb-2016. [Online]. Available: http://insights.dice.com/2014/11/26/5-top-python-guis-for-2015/. [Accessed: 14-Nov-2016].

[3] JavaScript Framework and HTML5 UI Library for Web App Development-Webix, *Webix*. [Online]. Available: https://webix.com/. [Accessed: 14-Nov-2016].

[4] UKI, *Best Web Frameworks*. [Online]. Available: http://www.bestwebframeworks.com/web-framework-review/javascript/117/uki/. [Accessed: 14-Nov-2016].

[5] J. Shore, An Unconventional Review of AngularJS, *Let's Code Javascript*, 14-Jan-2015. [Online]. Available: http://www.letscodejavascript.com/v3/blog/2015/01/angular_review. [Accessed: 14-Nov-2016].

[6] F. Lungh, Notes on Tkinter Performance, *Effbot*, 14-Jul-2002. [Online]. Available: http://effbot.org/zone/tkinter-performance.htm. [Accessed: 14-Nov-2016].

[7] VTK-Enabled Applications, *VTK*. [Online]. Available: http://www.vtk.org/. [Accessed: 14-Nov-2016].

[8] Wax GUI Toolkit, *Python*. [Online]. Available: https://wiki.python.org/moin/wax. [Accessed: 14-Nov-2016].

[9] Amazon Machine Learning, *Amazon*. [Online]. Available: https://aws.amazon.com/machine-learning/. [Accessed: 14-Nov-2016].

[10] "SciKit Learn,Machine Learning in Python, *scikit-learn*. [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 14-Nov-2016].

[11] "Pylearn2 devdocumentation, *Pylearn2*. [Online]. Available: http://deeplearning.net/software/pylearn2/. [Accessed: 14-Nov-2016].

[12] "ONID - OSU Network ID, *Oregon State University*. [Online]. Available: http://onid.oregonstate.edu. [Accessed: 14-Nov-2016].

[13] "Python statistics Mathematical statistics functions, *Python*. [Online]. Available: https://docs.python.org/3/library/statistics.html. [Accessed: 14-Nov-2016].

[14] "Statistical functions (scipy.stats), *scipy.stats*. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/stats.html. [Accessed: 14-Nov-2016].

[15] "Python Data Analysis Library, *Pandas*. [Online]. Available: http://pandas.pydata.org. [Accessed: 14-Nov-2016].