Micheal Willard
932480626
CS 496
Assignment 2: A Review of Dynamic Pages
URL: https://lecture-145722.appspot.com

Description

This site is designed to work as a scheduling app for a baseball team. The admin can enter and edit scheduled games. The admin can also add players to the roster in the database. When players are in the roster database, they can be selected during the creation or edit of a game object. Selecting a player for a game object translates to them being able to make it to the game.

There was thought put into how this web app would function later on. From the game object data being stored, an API can be created to produce formatted data. That API data can be used in other app features, as the crucial data for managing an adult league sports team is whether or not you have enough players for each game and who those players are. This is the critical data being stored and managed by the web app.

Test Cases:

1. Add a game with all input fields filled - PASSED
2. Add a player name - PASSED
3. Add a game with a missing input - PASSED (all fields should be required in DB)
4. Add a game with a player - PASSED
5. View the games on the viewer - PASSED
6. Open a game in edit mode
    1. Change the opponent and save, changes reflected in view/admin table - PASSED
    2. Change the field name and save - PASSED
    3. Change the date and save - PASSED
    4. Change the Time and save - PASSED
    5. Change the attending players and save - PASSED
    6. Change Home/Away and Save - PASSED
7. View a game in Edit mode:
    1. The Home/Away buttons are correct - PASSED
    2. The Opponent Name is auto populated correctly - PASSED
    3. The Field Name is auto populated correctly - PASSED
    4. The Date is auto populated correctly - PASSED
    5. The Time is auto populated correctly - PASSED
    6. The Players are auto populated correctly - PASSED
8. Exit the app, revise the site. Data should all remain - PASSED

The test were are fairly simple to run. No input controls were implemented at this time on text. The Date and Time controls were implement through the use of form inputs. Blank inputs were allowed. In future projects, the use of " if blank" statements will be implemented in the python backend code. It will stop the input and return an error message. At this time, blank inputs were allowed to offer more flexibility in testing the edit mode.

Templating

The jinja2 template use was implemented based on the lectures from class. Each of the 3 pages (View, Edit, Admin) utilized it's own HTML template with some CSS styling. The python

backend handled redirection as well as the template variables and values. All dynamic content was returned as variables and values from the python backend pages. The jina2 tempting allowed embedded code in the html files. If/else and for statements were utilized on the html pages to display the variables sent over by the python backend pages. The tempting also allows for the base html page to remain static, while the POST and GET requests are handled on the server backend in the python code.

Changes & Improvements

There are a few changes that would be good to implement, given more time. The text input fields for teams and fields could be controlled more from user input. One idea would be to have these as their own databases. The input forms could populated option dropdown for team, further control user input from error.

The layout and flow of the page could use some improvements. It took a while to get the CSS up and running, so visually the app is ok with room for improvement. The flow of the app isn't ideal. More time could be spent mapping out user cases and designing the flow around that.