Pytock ReadMe

HSE AI & CV Program
Object Oriented Programming course - Final Python Project
Professor Alex Dorian

Pytock implements a restaurant booking system according to stated specifications of the assignment.

submitted by Richard Zulch

Note: tock is a popular restaurant management system in the US, so this is named **pytock** *in keeping with pythonic naming conventions.*

Usage

To install the Python environment and prerequisites:

```
python3.13 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

Note: This uses 3.13 above to avoid an install error with streamlit dependency *pyarrow* which is evidently not yet updated to work with Python 3.14. Some online sources state that using Conda instead of venv avoids this problem. Despite the voluminous *requirements.txt*, the only direct pytock dependency is streamlit so a virtual environment wouldn't even be necessary if you already have streamlit installed globally.

To execute pytock:

```
streamlit run pytock.py
```

Important: Real Time Updating

Please do not miss that **pytock** supports multiple browsers and tabs with real-time updating. If you open additional pages to the same site you will see status displays and controls update as

bookings are made on other pages. The project description states:

> The data that is transmitted to the page is calculated anew each time - this provides the ability for several users (or several browser tabs) to work independently with one server.

but also

> Without special tricks, the interaction between these components occurs only at the command of the browser or the user.

I was curious what it would take to synchronize shared backend access with real-time updates, and the **pytock** architecture is oriented towards making it feasible. In particular, the `Tables` and `Bookings` classes make use of the streamlit shared resource cache with change detection and propagation. It was a bit tricky to implement but the result is gratifying.

A Note on Walk-In (ad hoc) Bookings

The assignment description states:

> There must also be a function of taking a table if guests came to the restaurant without a booking.

Because our date/time model is "timeless" and we don't know when the bookings will actually become active in real life, I chose to make walk-in bookings a separate "layer" above the reservations. Because they are separate you can take a table for a walk-in even if it has a reservation, and you can make a reservation for a table that currently has a walk-in. The UI shows both for table status.