Iteration 2

Completing the Daily Forecast Classes

In this phase, you'll build the core classes responsible for handling daily forecast data in your Weather Forecast Application. These classes will allow the system to represent, convert, load, and manage forecast data that was previously written to CSV by the ForecastWorker.

You will complete two Python files:

- `daily_forecast_class.py`
- `daily_forecast_manager_class.py`

Each contains `TODO` comments to guide your work.

Your Task

DailyForecast Class

File: `daily_forecast_class.py`

This class represents one period of daily weather data (e.g., Monday Night).

You will:

- Write two helper functions for converting temperature between Fahrenheit and Celsius.
- Create a daily forecast class
- Write the `__init__` method
- Implement a function to create a daily forecast object from a dictionary that will:
    - Handle cases where temperature or unit is missing.
    - Strip whitespace and standardize the temperature unit.
    - Convert the temperature to the other unit.
    - Format the precipitation string using 💧 (e.g., 💧 20%).
    - Return a daily forecast object using the cleaned and converted data.

DailyForecastManager Class

File: `daily_forecast_manager_class.py`

This class loads the CSV file that contains daily forecasts and turns each row into a daily forecast object.

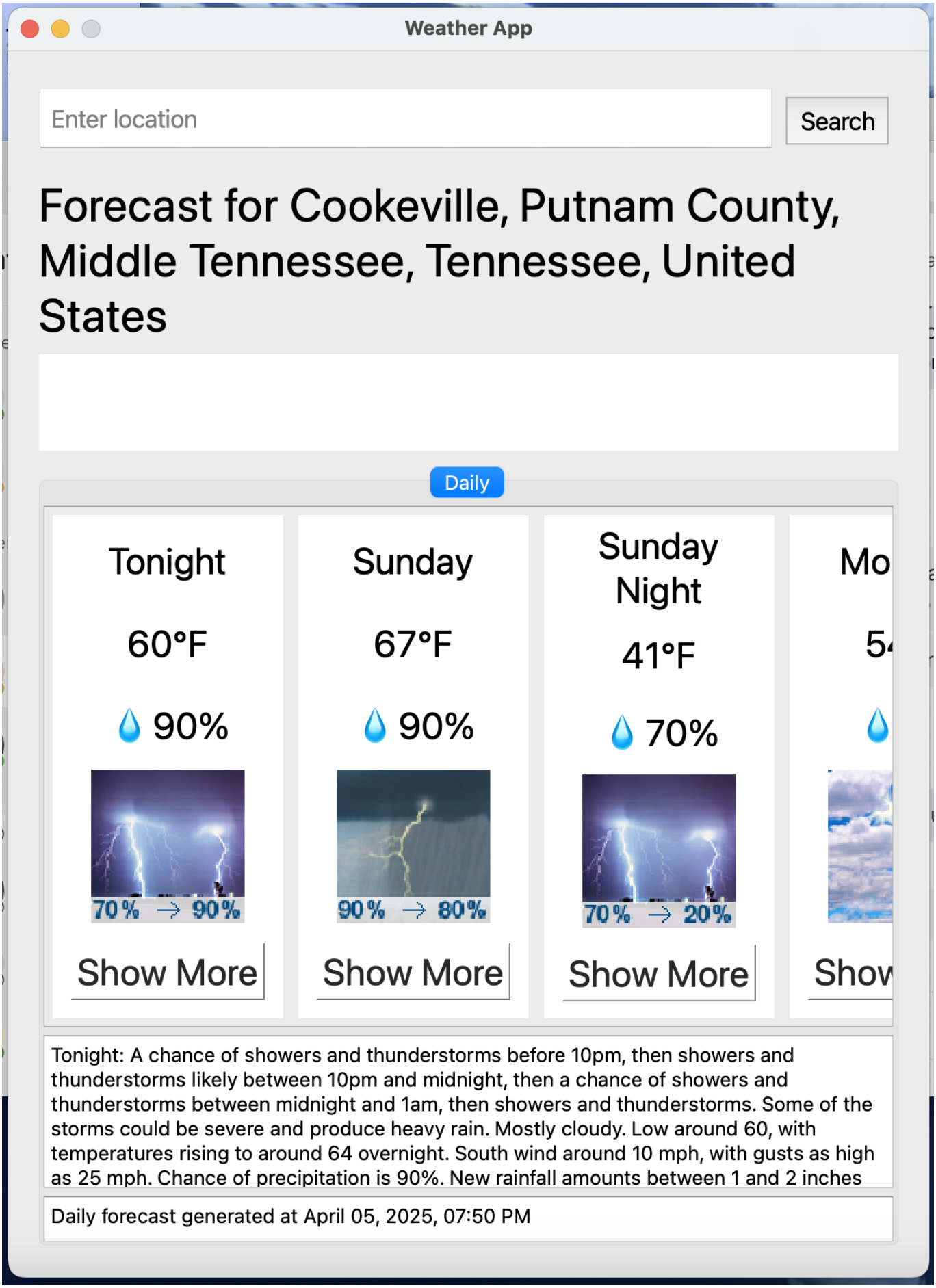
You will:

- Implement the daily forecast manager class with these responsibilities:
    - Store the CSV filename, the forecast generation time, and the source URL.
    - Load the forecast data from the CSV
    - Convert each row into a daily forecast object
    - Handle errors during file loading and return `True` or `False` accordingly.

- Return the list of forecasts via a getter method.
- Override `__str__()` to provide a nicely formatted summary including the generation time and count of forecast entries.

Testing Your Code

You can test your code by running the main script. You should see something similar to the following:



What to Submit

Push your completed versions of `skeleton_daily_forecast_class.py` and `skeleton_daily_forecast_manager_class.py` to your GitHub repository.

Make sure to:

- Replace all TODO comments by completing the code.
- Use good syntax and style.
- Add meaningful commit messages.

Example commit message

```
git commit -m "Implement DailyForecast and DailyForecastManager for
Iteration 2"
```

Now breathe deep — you just built the backbone of your app's forecast handling. One step closer to your final product.