SOCIAL MEDIA CONTENT SYSTEM USING HASH TABLES

By: Brandon Everett

FEBRUARY 16, 2025

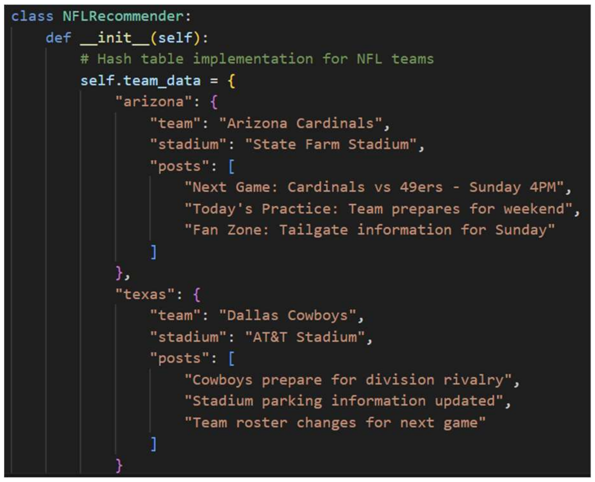
CSC-506

Module: 5

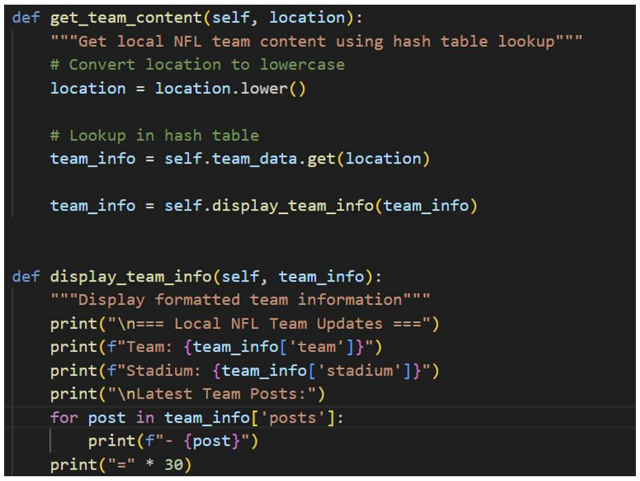
Social Media Content System Using Hash Tables

For this assignment, I built a program that uses hash tables to show NFL team updates based on location. I picked this project since I'm a football fan and wanted to see how social media platforms could show different content to fans depending on where they live. While it's not as complex as Twitter's system, it helped me understand how hash tables actually work.

Implementation: I used Python dictionaries to build the hash table structure. Following the practices from Weinberger et al. (2010), each team entry stores basic info like team names, stadiums, and recent posts. The program takes a location input and finds the matching team's content in the hash table. I originally wanted it to automatically detect user location like websites do, but that turned out to be too complicated for this project. Instead, users just type in their location manually.



```python
class NFLRecommender:
    def __init__(self):
        # Hash table implementation for NFL teams
        self.team_data = {
            "arizona": {
                "team": "Arizona Cardinals",
                "stadium": "State Farm Stadium",
                "posts": [
                    "Next Game: Cardinals vs 49ers - Sunday 4PM",
                    "Today's Practice: Team prepares for weekend",
                    "Fan Zone: Tailgate information for Sunday"
                ]
            },
            "texas": {
                "team": "Dallas Cowboys",
                "stadium": "AT&T Stadium",
                "posts": [
                    "Cowboys prepare for division rivalry",
                    "Stadium parking information updated",
                    "Team roster changes for next game"
                ]
            }
```



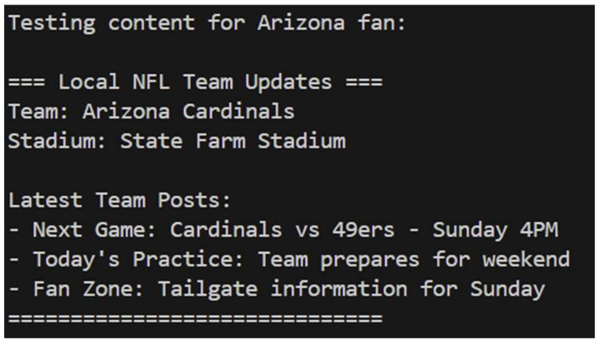
```python
def get_team_content(self, location):
    """Get local NFL team content using hash table lookup"""
    # Convert location to lowercase
    location = location.lower()

    # Lookup in hash table
    team_info = self.team_data.get(location)

    team_info = self.display_team_info(team_info)


def display_team_info(self, team_info):
    """Display formatted team information"""
    print("\n=== Local NFL Team Updates ===")
    print(f"Team: {team_info['team']}")
    print(f"Stadium: {team_info['stadium']}")
    print("\nLatest Team Posts:")
    for post in team_info['posts']:
        print(f"- {post}")
    print("=" * 30)
```

Here's what happens when the program runs and searches for "arizona":

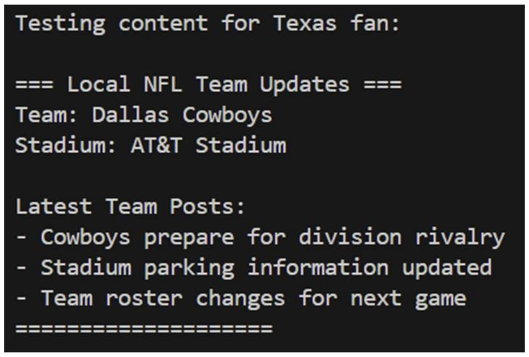


```
Testing content for Arizona fan:

=== Local NFL Team Updates ===
Team: Arizona Cardinals
Stadium: State Farm Stadium

Latest Team Posts:
- Next Game: Cardinals vs 49ers - Sunday 4PM
- Today's Practice: Team prepares for weekend
- Fan Zone: Tailgate information for Sunday
==============================
```

Here's what happens when the program runs and searches for "texas":



```
Testing content for Texas fan:

=== Local NFL Team Updates ===
Team: Dallas Cowboys
Stadium: AT&T Stadium

Latest Team Posts:
- Cowboys prepare for division rivalry
- Stadium parking information updated
- Team roster changes for next game
====================
```

Time Complexity: Based on what I learned from the HackerEarth tutorial, hash tables can find data really quickly - it takes the same amount of time whether you're looking through 2 teams or all 32 NFL teams. I saw this firsthand when testing my code.

Challenges: The biggest issue was trying to do too much at first. I wanted to add automatic location detection like real websites have, but after looking into it, I realized that would be way too complex for this assignment. I had to scale back and just focus on making the basic hash table work properly.

Results: The program works as expected - when you enter a location, it finds the matching team's info. I kept it basic to focus on the hash table part. While actual social media platforms are more complex, this shows how hash tables can be used to retrieve content.

Skills Learned: Through this project, I learned about implementing hash tables and using Python dictionaries effectively. I gained some insight into how larger social media systems would need to handle more teams and users. While I kept the program basic without features like browser location, it helped me understand how to build programs that need to retrieve information quickly.

Even though this is a simple version, it showed me how hash tables work for finding and showing content quickly. Looking at real-world examples like TikTok's system (Skovorodnikov, 2023), I can see why the fast lookup time would matter when you're dealing with millions of users.

Conclusion: This project showed me how hash tables work in an actual program. I created a basic content delivery system that helped me understand hash tables and what goes into building these kinds of systems. While it's simpler than real social platforms, it demonstrates how to store and retrieve data efficiently.

References: "Basics of Hash Tables Tutorials & Notes." HackerEarth, www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/. Accessed 16 Feb. 2025.

Skovorodnikov, Henry. "The Secret Sauce of TikTok's Recommendations." Shaped Blog, 17 Feb. 2023, www.shaped.ai/blog/the-secret-sauce-of-tik-toks-recommendations. Accessed 16 Feb. 2025.

Weinberger, Kilian, et al. "Feature Hashing for Large Scale Multitask Learning." arXiv, 12 Feb. 2010, arxiv.org/abs/0902.2206. Accessed 16 Feb. 2025.