Python Programming

Recipe for Users Development

Proposal

Date : 2023.12.18

Name : Park Min E

ID : mine-park

1. Introduction

1) Background

As the number of young people living alone, especially college students, has increased, many of them often struggle to properly prepare their meals due to busy schedules or lack of time and ingredients. Furthermore, with the advancement of food delivery apps, there is a growing preference for convenient delivery meals over cooking at home. Consuming only stimulating and limited food options can easily expose young people to nutritional imbalances and various diseases such as adult illnesses. To address this problem, a program is needed that recommends recipes that can be easily and conveniently prepared using ingredients at home, allowing young people living alone to have healthy and balanced meals.

2) Project goal

The goal is to create a program that analyzes the ingredients customers have on hand and recommends easy recipes that can be prepared with limited ingredients, allowing them to make healthy meals.

3) Differences from existing programs

Our approach is different from the traditional recipe recommendation methods seen on the internet or various TV programs, as they provide absolute recipes without considering the customer's ingredient availability. Many customers find it challenging to follow these recipes because they lack the necessary ingredients. We, on the other hand, aim to accurately assess the customer's ingredient inventory and recommend recipes that align with those ingredients. This way, we prevent customers from being unable to cook due to ingredient unavailability, setting us apart from existing recipe recommendation methods.

2. Functional Requirement

1) Analyzing customer Analyzing the customer's ingredient inventory.

- Functionality to determine the ingredients customers have on hand.

2) Matching recipes Matching recipes that can be prepared
with the customer's available ingredients.

- A feature that matches the customer's gathered ingredient inventory with the ingredients in recipes and outputs only the recipes that have matching ingredients.

3) No matching recipes In case there are no matching recipes.

- If there are no recipes that exactly match the ingredients entered by the user, the code will now calculate the similarity between the user's ingredients and the ingredients of existing recipes. It will rank these recipes based on their similarity and suggest them to the user. Additionally, it will provide alternative solutions.

3. Implementation

1) functionality implementation

(1) Analyzing customer

- Input / Output

user_input : The variable that receives the list of ingredients in the user's refrigerator.

user_ingredients : The variable that splits and stores user_input based on spaces is user_ingredients.

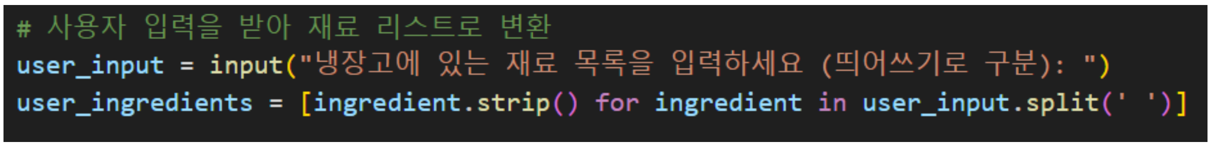
- Explanation

Functionality to determine the ingredients customers have on hand.

- The applied learned concepts include

Input / Output, Word Splitting(split, strip)

- Code Screenshot



```python
# 사용자 입력을 받아 재료 리스트로 변환
user_input = input("냉장고에 있는 재료 목록을 입력하세요 (띄어쓰기로 구분): ")
user_ingredients = [ingredient.strip() for ingredient in user_input.split(' ')]
```

(2) Matching recipes

- Input / Output

user_ingredients : Receiving information about the ingredients the user has.

matched_recipes : Comparing user_ingredients with recipe.txt and outputting matching recipes.

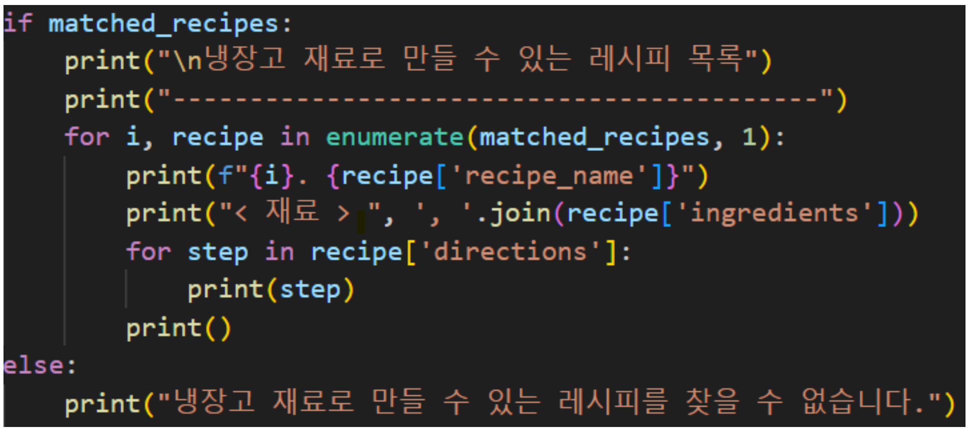
- Explanation

A feature that matches the customer's gathered ingredient inventory with the ingredients in recipes and outputs only the recipes that have matching ingredients.

- The applied learned concepts include

Looping, Conditional Statements, List Append(append)

- Code Screenshot



```python
if matched_recipes:
    print("\n냉장고 재료로 만들 수 있는 레시피 목록")
    print("----------------------------------------")
    for i, recipe in enumerate(matched_recipes, 1):
        print(f"{i}. {recipe['recipe_name']}")
        print("< 재료 > ", ', '.join(recipe['ingredients']))
        for step in recipe['directions']:
            print(step)
        print()
else:
    print("냉장고 재료로 만들 수 있는 레시피를 찾을 수 없습니다.")
```

(3) No matching recipes

- Input / Output

user_ingredient / recipes : Receiving information about both the user's ingredient and existing recipe information.

similar_recipes : When there are no matched_recipes, outputting the calculated similarity values between the user's ingredients and the recipes.

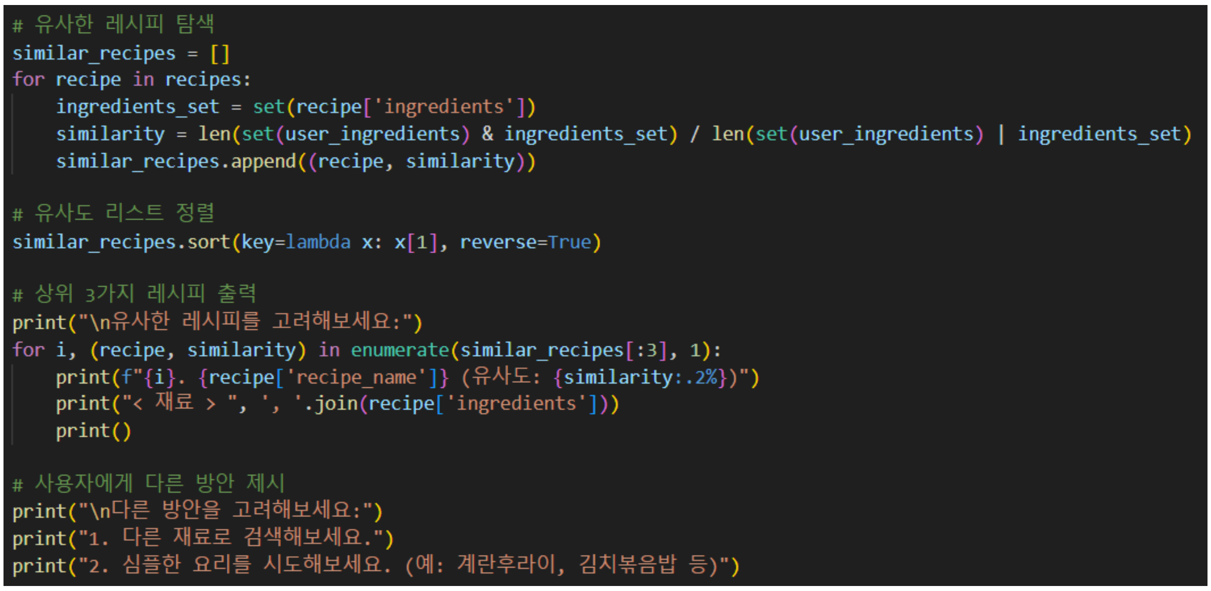
- Explanation

If there are no recipes that exactly match the ingredients entered by the user, the code will now calculate the similarity between the user's ingredients and the ingredients of existing recipes. It will rank these recipes based on their similarity and suggest them to the user. Additionally, it will provide alternative solutions.

- The applied learned concepts include

Looping, Conditional Statements, List Append(append)

- Code Screenshot

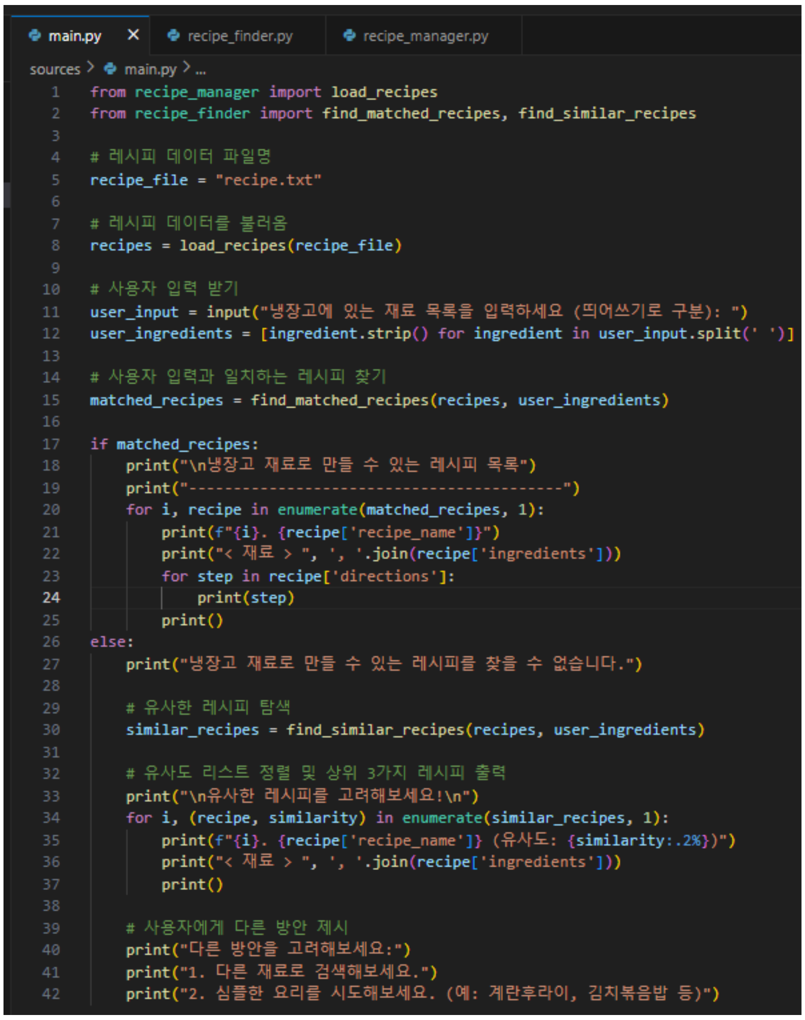


```python
# 유사한 레시피 탐색
similar_recipes = []
for recipe in recipes:
    ingredients_set = set(recipe['ingredients'])
    similarity = len(set(user_ingredients) & ingredients_set) / len(set(user_ingredients) | ingredients_set)
    similar_recipes.append((recipe, similarity))

# 유사도 리스트 정렬
similar_recipes.sort(key=lambda x: x[1], reverse=True)

# 상위 3가지 레시피 출력
print("\n유사한 레시피를 고려해보세요:")
for i, (recipe, similarity) in enumerate(similar_recipes[:3], 1):
    print(f"{i}. {recipe['recipe_name']} (유사도: {similarity:.2%})")
    print("< 재료 > ", ', '.join(recipe['ingredients']))
    print()

# 사용자에게 다른 방안 제시
print("\n다른 방안을 고려해보세요:")
print("1. 다른 재료로 검색해보세요.")
print("2. 심플한 요리를 시도해보세요. (예: 계란후라이, 김치볶음밥 등)")
```

2) file modularization

(1) File 1

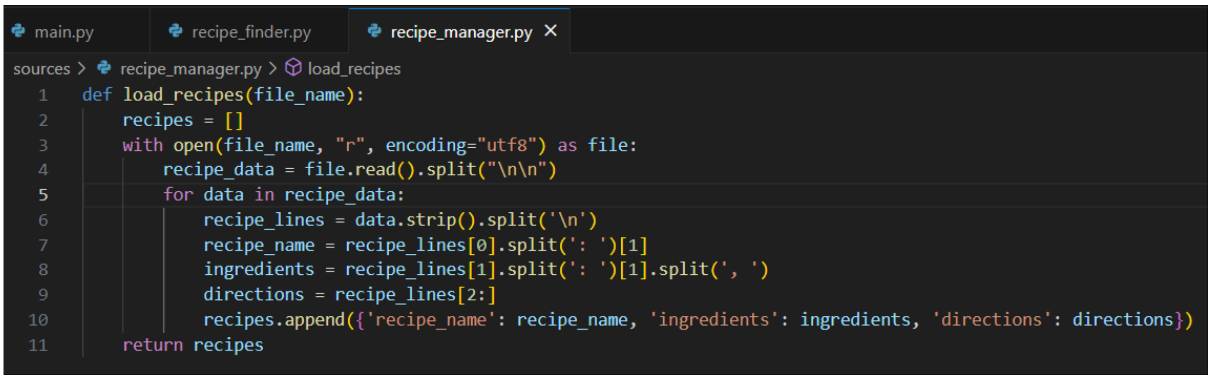
- main.py



```python
from recipe_manager import load_recipes
from recipe_finder import find_matched_recipes, find_similar_recipes

# 레시피 데이터 파일명
recipe_file = "recipe.txt"

# 레시피 데이터를 불러옴
recipes = load_recipes(recipe_file)

# 사용자 입력 받기
user_input = input("냉장고에 있는 재료 목록을 입력하세요 (띄어쓰기로 구분): ")
user_ingredients = [ingredient.strip() for ingredient in user_input.split(' ')]

# 사용자 입력과 일치하는 레시피 찾기
matched_recipes = find_matched_recipes(recipes, user_ingredients)

if matched_recipes:
    print("\n냉장고 재료로 만들 수 있는 레시피 목록")
    print("----------------------------------------")
    for i, recipe in enumerate(matched_recipes, 1):
        print(f"{i}. {recipe['recipe_name']}")
        print("< 재료 > ", ', '.join(recipe['ingredients']))
        for step in recipe['directions']:
            print(step)
        print()
else:
    print("냉장고 재료로 만들 수 있는 레시피를 찾을 수 없습니다.")

    # 유사한 레시피 탐색
    similar_recipes = find_similar_recipes(recipes, user_ingredients)

    # 유사도 리스트 정렬 및 상위 3가지 레시피 출력
    print("\n유사한 레시피를 고려해보세요!\n")
    for i, (recipe, similarity) in enumerate(similar_recipes, 1):
        print(f"{i}. {recipe['recipe_name']} (유사도: {similarity:.2%})")
        print("< 재료 > ", ', '.join(recipe['ingredients']))
        print()

    # 사용자에게 다른 방안 제시
    print("다른 방안을 고려해보세요:")
    print("1. 다른 재료로 검색해보세요.")
    print("2. 심플한 요리를 시도해보세요. (예: 계란후라이, 김치볶음밥 등)")
```

(2) File 2

- recipe_manager.py

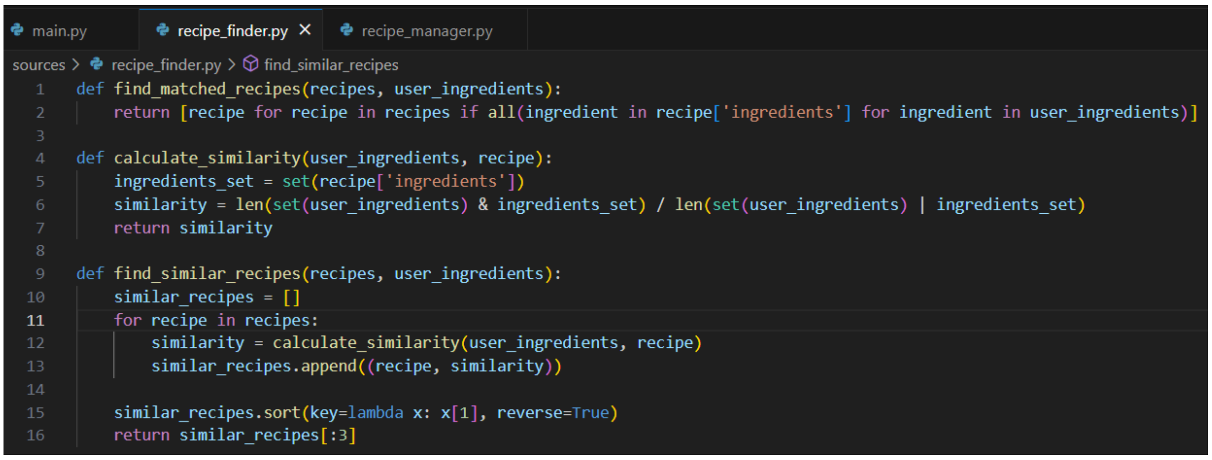

main.py | recipe_finder.py | recipe_manager.py ×

sources > recipe_manager.py > load_recipes

```python
def load_recipes(file_name):
    recipes = []
    with open(file_name, "r", encoding="utf8") as file:
        recipe_data = file.read().split("\n\n")
        for data in recipe_data:
            recipe_lines = data.strip().split('\n')
            recipe_name = recipe_lines[0].split(': ')[1]
            ingredients = recipe_lines[1].split(': ')[1].split(', ')
            directions = recipe_lines[2:]
            recipes.append({'recipe_name': recipe_name, 'ingredients': ingredients, 'directions': directions})
    return recipes
```

(3) File 3

- recipe_finder.py


main.py | recipe_finder.py × | recipe_manager.py

sources > recipe_finder.py > find_similar_recipes

```python
def find_matched_recipes(recipes, user_ingredients):
    return [recipe for recipe in recipes if all(ingredient in recipe['ingredients'] for ingredient in user_ingredients)]

def calculate_similarity(user_ingredients, recipe):
    ingredients_set = set(recipe['ingredients'])
    similarity = len(set(user_ingredients) & ingredients_set) / len(set(user_ingredients) | ingredients_set)
    return similarity

def find_similar_recipes(recipes, user_ingredients):
    similar_recipes = []
    for recipe in recipes:
        similarity = calculate_similarity(user_ingredients, recipe)
        similar_recipes.append((recipe, similarity))

    similar_recipes.sort(key=lambda x: x[1], reverse=True)
    return similar_recipes[:3]
```

3. Test Result

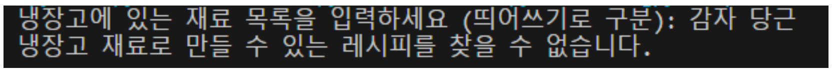
1) result of the test

(1) Analyzing customer

- Explanation

This code segment receives a list of ingredients from the user for items in their refrigerator. It then splits the input based on spaces, creating a list. The final list, named user_ingredients, is obtained by removing leading and trailing spaces from each ingredient in the previously created list comprehension.

- Test result screenshot



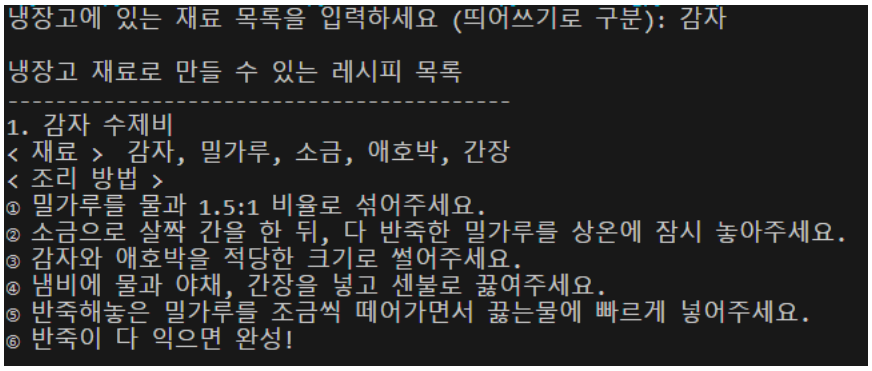
```
냉장고에 있는 재료 목록을 입력하세요 (띄어쓰기로 구분): 감자 당근
냉장고 재료로 만들 수 있는 레시피를 찾을 수 없습니다.
```

(2) Matching recipes

- Explanation

Initialize the matched_recipes list. Check each recipe for a match with the user's ingredients. Add matching recipes to the matched_recipes list. If matching recipes exist, print their details. Display the recipe number, name, ingredients used, and cooking steps. If no matching recipes are found, print a message: "Unable to find recipes with the provided ingredients."

- Test result screenshot



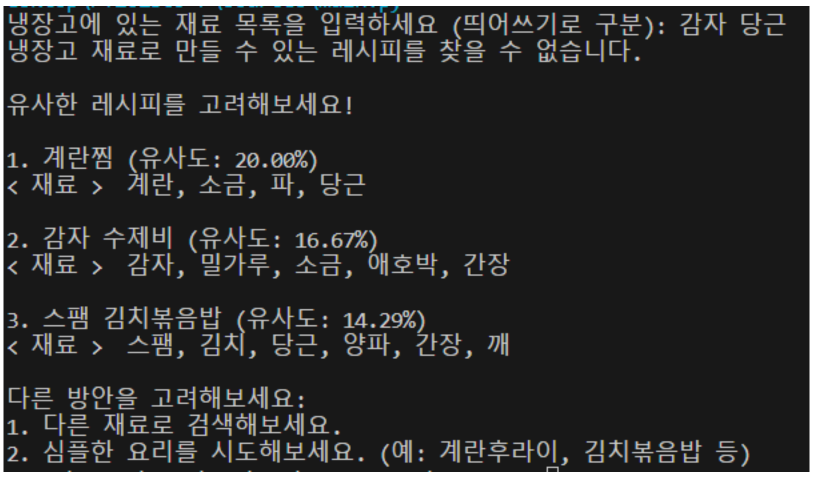
```
냉장고에 있는 재료 목록을 입력하세요 (띄어쓰기로 구분): 감자

냉장고 재료로 만들 수 있는 레시피 목록
----------------------------------------
1. 감자 수제비
< 재료 >  감자, 밀가루, 소금, 애호박, 간장
< 조리 방법 >
① 밀가루를 물과 1.5:1 비율로 섞어주세요.
② 소금으로 살짝 간을 한 뒤, 다 반죽한 밀가루를 상온에 잠시 놓아주세요.
③ 감자와 애호박을 적당한 크기로 썰어주세요.
④ 냄비에 물과 야채, 간장을 넣고 센불로 끓여주세요.
⑤ 반죽해놓은 밀가루를 조금씩 떼어가면서 끓는물에 빠르게 넣어주세요.
⑥ 반죽이 다 익으면 완성!
```

(3) No matching recipes

- Explanation

The code calculates the similarity between the user's input ingredients and each recipe in the dataset using the Jaccard similarity coefficient. It then sorts the recipes in descending order based on their similarity. The top 3 similar recipes are displayed, showing the recipe name, similarity percentage, and the list of ingredients used. Additionally, the code provides various suggestions to the user, such as trying a different set of ingredients or attempting simpler recipes.

- Test result screenshot



```
냉장고에 있는 재료 목록을 입력하세요 (띄어쓰기로 구분): 감자 당근
냉장고 재료로 만들 수 있는 레시피를 찾을 수 없습니다.

유사한 레시피를 고려해보세요!

1. 계란찜 (유사도: 20.00%)
< 재료 >  계란, 소금, 파, 당근

2. 감자 수제비 (유사도: 16.67%)
< 재료 >  감자, 밀가루, 소금, 애호박, 간장

3. 스팸 김치볶음밥 (유사도: 14.29%)
< 재료 >  스팸, 김치, 당근, 양파, 간장, 깨

다른 방안을 고려해보세요:
1. 다른 재료로 검색해보세요.
2. 심플한 요리를 시도해보세요. (예: 계란후라이, 김치볶음밥 등)
```

4. Lessons Learned & Feedback

I had previously learned C and C++, but this semester I started learning Python. I initially thought Python was the easiest and simplest programming language, but I found the course progressing faster than I expected. While I grasped the concepts, I struggled with solving examples, falling slightly behind in class. Despite the seemingly easy concepts in the 'number & operation' section, solving problems proved challenging.

However, as I continued with the Python course, my understanding improved, and problem-solving became much easier. The professor's consideration, such as adjusting the pace and difficulty of examples based on student feedback, significantly aided my comprehension.

Through the project, I had the opportunity to explore potential future tasks and challenges, attempting to create a program from a consumer's perspective. Tasks like reading and writing files and modularizing code enhanced my coding skills. Initially considering it a non-major subject, I gradually found Python fascinating and got immersed in creating various programs. As I conclude this project, I look forward to future opportunities to use various programming languages like C, C++, and Java to create more professional and diverse programs than this experience.