SW Engineering CSC648/848

Project name: Zesty

Section 04, Team 05

Name	**Role**
Xu Gu	Team Leader
Pasang Sherpa	Back-End Lead
Raymond Liu	Git Master
Dante Vercelli	Back-End Lead
Yonatan Leake	Scrum master
Junghyun Song (Katie)	Front-End Lead
Ruxue Jin	Front-End Lead

Milestone 4

Due day: 5/8/2024

History Table		
Version	Date	feedback
1.0	May 4th	

1. QA Testing

Unit Test

Description of 5 P1 features and Github URL of test cases directory

Feature 1: Search recipes
Descritions
1.1. Develop a search bar that allows users to input the recipe features including recipes name, ingredients, region, difficulty level and returns related recipes.
1.2. Ensure the search results are returned quickly, optimizing the database queries for performance.
test cases directory
Frontend: application/frontend/src/__test__/
Backend:
application/backend/tests/common/search_engine_test.py
application/backend/tests/daos/test_recipe_dao.py
application/backend/tests/models/test_models.py
application/backend/tests/services/test_recipe_service.py

Feature 2: Detailed recipe information
2.1. Ensure each recipe entry includes a list of required ingredients with type and amount.
2.2. Provide clear cooking steps and estimated preparation and cooking time for each recipe.
test cases directory
Frontend: application/frontend/src/__test__/
Backend:
application/backend/tests/daos/test_recipe_dao.py
application/backend/tests/models/test_models.py
application/backend/tests/services/test_recipe_service.py

Feature 3: User Pantry
3.1. Enable users to edit pantry, adding or deleting the ingredients they have at their home.
3.2 Scan and recognize the ingredients they have by uploading photos or taking instant photos
test cases directory
Frontend: application/frontend/src/__test__/
Backend:
application/backend/tests/daos/test_ingredient_dao.py
application/backend/tests/daos/test_useringredient_dao.py
application/backend/tests/models/test_models.py
application/backend/tests/services/test_user_ingredients.py

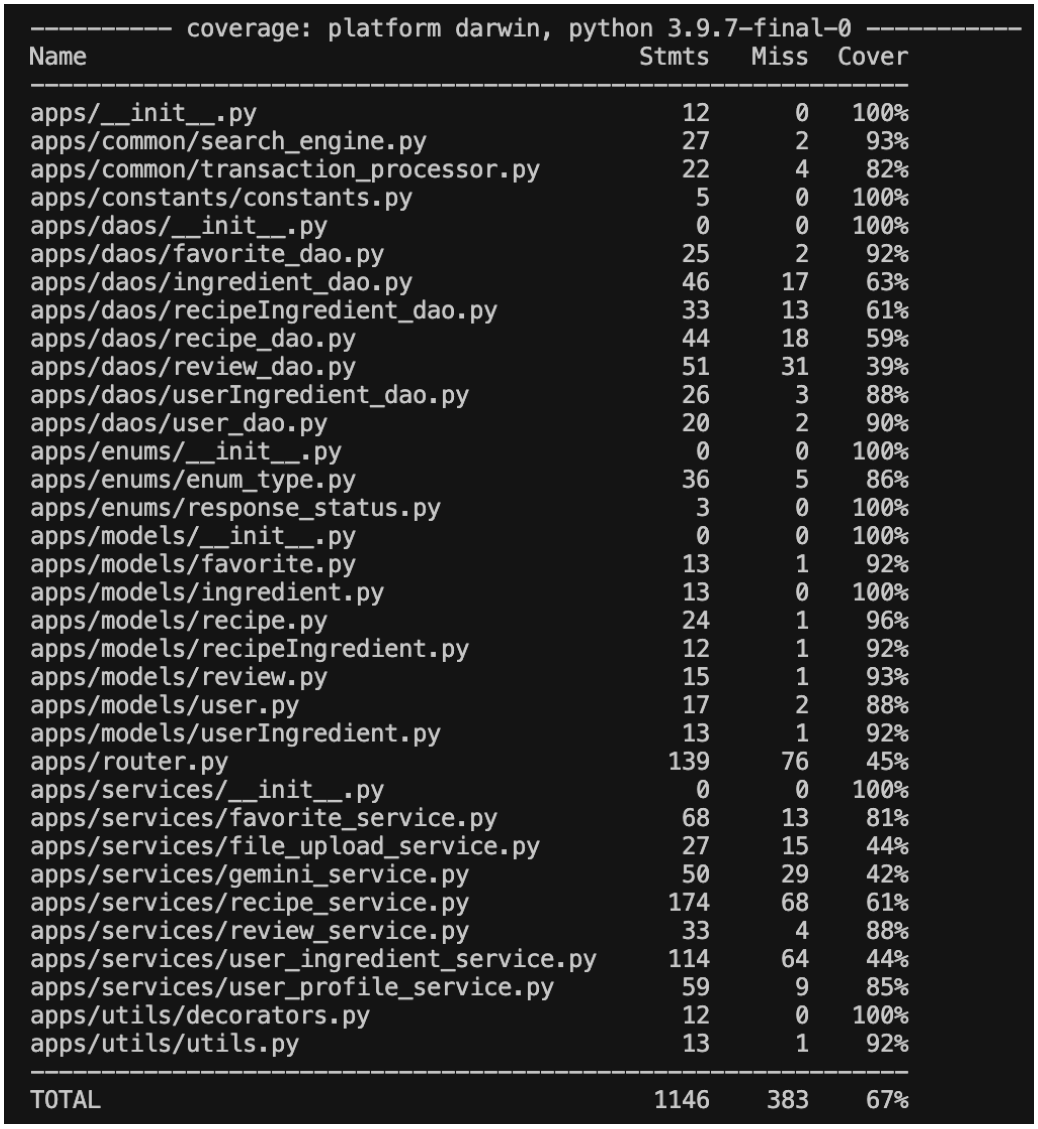
Feature 4: Recommendations based on user pantry
4.1. Give the users recipe recommendations based on their ingredients.

4.2 Recommended recipes have a metric indicating the proportion of ingredients they have.
4.3 The detail page of the recommended recipes will show the missing ingredients
test cases directory
Frontend: application/frontend/src/__test__/
Backend:
application/backend/tests/daos/test_useringredient_dao.py
application/backend/tests/models/test_models.py
application/backend/tests/services/test_user_ingredients.py

Feature 5: Recipe rating system
5.1. Implement a rating system where users can rate recipes on a scale (e.g., 1 to 5 stars).
5.2. Allow users to leave reviews or comments along with their ratings.
test cases directory
Frontend: application/frontend/src/__test__/
Backend:
application/backend/tests/daos/test_review_dao.py
application/backend/tests/models/test_models.py
application/backend/tests/services/test_review_service.py

Description of functional and statement coverage (functional coverage >= 90%, statement coverage >=50%).

Functional coverage
We have unit test for all the P1 features, so the functional coverage should be 100%
Statement coverage
67%, as shown in this screenshot



```
---------- coverage: platform darwin, python 3.9.7-final-0 -----------
Name                                        Stmts   Miss  Cover
---------------------------------------------------------------
apps/__init__.py                               12      0   100%
apps/common/search_engine.py                   27      2    93%
apps/common/transaction_processor.py           22      4    82%
apps/constants/constants.py                     5      0   100%
apps/daos/__init__.py                           0      0   100%
apps/daos/favorite_dao.py                      25      2    92%
apps/daos/ingredient_dao.py                    46     17    63%
apps/daos/recipeIngredient_dao.py              33     13    61%
apps/daos/recipe_dao.py                        44     18    59%
apps/daos/review_dao.py                        51     31    39%
apps/daos/userIngredient_dao.py                26      3    88%
apps/daos/user_dao.py                          20      2    90%
apps/enums/__init__.py                          0      0   100%
apps/enums/enum_type.py                        36      5    86%
apps/enums/response_status.py                   3      0   100%
apps/models/__init__.py                         0      0   100%
apps/models/favorite.py                        13      1    92%
apps/models/ingredient.py                      13      0   100%
apps/models/recipe.py                          24      1    96%
apps/models/recipeIngredient.py                12      1    92%
apps/models/review.py                          15      1    93%
apps/models/user.py                            17      2    88%
apps/models/userIngredient.py                  13      1    92%
apps/router.py                                139     76    45%
apps/services/__init__.py                       0      0   100%
apps/services/favorite_service.py              68     13    81%
apps/services/file_upload_service.py           27     15    44%
apps/services/gemini_service.py                50     29    42%
apps/services/recipe_service.py               174     68    61%
apps/services/review_service.py                33      4    88%
apps/services/user_ingredient_service.py      114     64    44%
apps/services/user_profile_service.py          59      9    85%
apps/utils/decorators.py                       12      0   100%
apps/utils/utils.py                            13      1    92%
---------------------------------------------------------------
TOTAL                                        1146    383    67%
```

Integration Test

Tested P1 features:

1. User register
2. User login
3. Search recipes
4. Recipes details: description, image, cooking time, steps
5. Favorite recipes
6. Rate and Comment on recipes
7. Show ingredient list in the pantry

8. Manage user pantry (add/remove/ingredient)
9. Get recipes based on user pantry
10. Missing ingredient hint on the detail page
11. Create recipes / AI-assist

Test cases

Test Case ID: T01
Description: User Register Successfully
Date Tested: May 7th, 2024
Prerequisites:
Email is not in the user table
Data:
userName: test1,
email: test1@test.com
password: test1
Scenario:

- Navigate to http://localhost:5173/user/register / Site should open / PASS
- Type in userName, password and email / Fields can be entered / PASS
- Click Register / navigate to Login page / PASS

Result:
PASS

Test Case ID: T02
Description: User Register Failed, email exists
Date Tested: May 7th, 2024
Prerequisites:
Email is in the user table
Data:
userName: test1,
email: test1@test.com
password: test1
Scenario:

- Navigate to http://localhost:5173/user/register / Site should open / PASS
- Type in userName, password and email / Fields can be entered / PASS
- Click Register / Cannot register, an alteration appears / PASS

Result:
PASS

Test Case ID: T03
Description: User Login Successfully
Date Tested: May 7th, 2024

Prerequisites:
User registered, password is correct
Data:
userName: test1,
email: test1@test.com
password: test1
Scenario:
- Navigate to http://localhost:5173/user/login / Site should open / PASS
- Type in password and email / Fields can be entered / PASS
- Click Login / Navigate to the homepage/ PASS
- Auto-navigate / user name appears on the right-top/ PASS

Result:
PASS

Test Case ID: T04
Description: User Login Failed, wrong password
Date Tested: May 7th, 2024
Prerequisites:
User registered, password is wrong
Data:
userName: test1,
email: test1@test.com
password: test2
Scenario:
- Navigate to http://localhost:5173/user/login / Site should open / PASS
- Type in password and email / Fields can be entered / PASS
- Click Login /Login failed, an alteration appears, no navigation / PASS

Result:
PASS

Test Case ID: T05
Description: Search recipes with multiple ingredients
Date Tested: May 8th, 2024
Prerequisites:
Write 20 recipes into database
Data:
Search with keywords: chicken, beef
Scenario:
- Navigate to http://localhost:5173/ Site should open / PASS
- Type in searching text / PASS
- Click search button, return the recipes with ingredients chicken or beef/ PASS

Result:
PASS

Test Case ID: T06
Description: Recipe details
Date Tested: May 8th, 2024
Prerequisites:
Write 20 recipes into database
Data:
Recipe url: http://127.0.0.1:5173/recipe/21
Scenario:

- Navigate to http://127.0.0.1:5173/recipe/21 Site should open / PASS
- Detail page contains Recipe title, description, image, steps, ingredients / PASS

Result:
PASS

Test Case ID: T07
Description: Favorite recipes
Date Tested: May 8th, 2024
Prerequisites:
Logged in
The recipe wasn't favorited by the tested user before
Data:
Recipe rated: http://127.0.0.1:5173/recipe/21
Scenario:

- Navigate to http://127.0.0.1:5173/recipe/21 Site should open / PASS
- The "like" is not filled / PASS
- Click the "like" and it'll be filled / PASS

Result:
PASS

Test Case ID: T08
Description: Rate and Comment recipes
Date Tested: May 8th, 2024
Prerequisites:
Logged in
Test account: test1@test.com
Data:
Recipe rated: http://127.0.0.1:5173/recipe/21
Scenario:

- Navigate to http://127.0.0.1:5173/recipe/21 Site should open / PASS
- Commit comment without selecting stars, Cannot commit/ PASS
- Commit comment and select stars Commit successfully / PASS

Result:
PASS

Test Case ID: T09
Description: Manage pantry
Date Tested: May 8th, 2024
Prerequisites:
Logged in
Data:
Test account: test1@test.com
Add 3 ingredients into pantry: apple, beef, egg
Remove apple
Scenario:
- Navigate to test account pantry / No ingredients appear/ PASS
- Add three ingredients to pantry / Add ingredients successfully / PASS
- Remove apple / Remove apple successfully / PASS

Result:
PASS

Test Case ID: T10
Description: Ingredient list in the pantry
Date Tested: May 8th, 2024
Prerequisites:
Logged in
Test Case T09 executed
Data:
Test account: test1@test.com
Scenario:
- Navigate to test account pantry/ beef, egg in the pantry/ PASS

Result:
PASS

Test Case ID: T11
Description: Get recipes based on user pantry
Date Tested: May 8th, 2024
Prerequisites:
Logged in
Data:
Test account: test1@test.com
Ingredients in the pantry: apple, beef, egg
Scenario:
- Navigate to home page, then login/ Can access the page and login successfully / PASS
- Click “Get Your Recipes” button on the dropdown / Add ingredients successfully / get the recipes with masks showing percentage, and sorted according to the percentage/ PASS

Result:
PASS

Test Case ID: T12
Description: Missing ingredient hint on the detail page
Date Tested: May 8th, 2024
Prerequisites:
Logged in
Data:
Test account: test1@test.com
Ingredients in the pantry: apple, beef, egg
Scenario:

- Navigate to home page, then login/ Can access the page and login successfully / PASS
- Click “Get Your Recipes” button on the dropdown / Add ingredients successfully / get the recipes with masks showing percentage, and sorted according to the percentage/ PASS
- Click one of the recipe / navigate to detail page / PASS
- Ingredient test1 already has and misses were hinted on the page / PASS

Result:
PASS

Test Case ID: T13
Description: Create recipes / AI-assist
Date Tested: May 8th, 2024
Prerequisites:
Logged in
Data:
Test account: test1@test.com
A picture of a dish
Scenario:

- Navigate to home page, then login/ Can access the page and login successfully / PASS
- Click “Create Recipe” button on the dropdown / navigate to create recipe page/ PASS
- Upload the dish picture and click “generate recipe” / Recipe generated / PASS
- Edit the description and modify the ingredient then submit / The revised recipe submitted successfully / PASS

Result:
PASS

Integration testing coverage analysis

We tested 11 P1 features with 13 test cases. The testing coverage of the selected P1 features is 100%

2. Coding Practices

A coding style

Please describe what coding style you chose (e.g., Google Python Coding Style, refer to slides), and how to enforce it by the tool or plug-in (refer to slides).

Frontend:

For the frontend TypeScript code, we'll follow the **Airbnb JavaScript Style Guide with TypeScript adaptations**. This style guide provides comprehensive conventions for writing clean and maintainable JavaScript code, which can be seamlessly applied to TypeScript.

Key Points:

1. Naming Conventions:
    - Use camelCase for variable names and functions.
    - Use PascalCase for class names and enum types.
    - Use descriptive names that convey the purpose of the variable or function.
2. Formatting:
    - Use 2 spaces for indentation.
    - Use single quotes for string literals.
    - Use trailing commas for multi-line object literals and arrays.
3. Type Annotations:
    - Use TypeScript's type system to add type annotations to variables, parameters, and return types.
    - Prefer explicit types for clarity and maintainability.
4. Imports:
    - Import TypeScript modules using import statements.
    - Group import statements in the following order: libraries, modules from other packages, and local modules.

Enforcement:
To enforce the Airbnb JavaScript Style Guide with TypeScript, we'll use the following tools and plugins:

1. ESLint with TypeScript Support:
    - ESLint is a popular linter that helps identify and fix code style issues.
    - We'll configure ESLint to support TypeScript by installing the necessary plugins (@typescript-eslint/eslint-plugin, @typescript-eslint/parser) and extending the Airbnb style guide with TypeScript-specific rules.

2. Prettier:
    - Prettier is an opinionated code formatter that enforces consistent code style automatically.
    - We'll configure Prettier to format TypeScript code according to the Airbnb JavaScript Style Guide and integrate it with our code editor.

Backend:

We're using Google Python Coding style. As for how we're enforcing it we're going to run Pylint over our code and review it. Also for formatting we're using black formatter.

Please list source files related to 5 P1 features which demonstrate your coding style.

Feature 1: Search recipes
Descriptions
1.1. Develop a search bar that allows users to input the recipe features including recipes name, ingredients, region, difficulty level and returns related recipes.
1.2. Ensure the search results are returned quickly, optimizing the database queries for performance.
Source files:
application/backend/apps/services/recipe_service.py
application/backend/apps/common/search_engine.py
application/backend/apps/daos/recipe_dao.py

Feature 2: Detailed recipe information
2.1. Ensure each recipe entry includes a list of required ingredients with type and amount.
2.2. Provide clear cooking steps and estimated preparation and cooking time for each recipe.
Source files:
application/backend/apps/services/recipe_service.py
application/backend/apps/daos/recipe_dao.py
application/backend/apps/models/recipeIngredient.py
application/backend/apps/models/recipe.py

Feature 3: User Pantry
3.1. Enable users to edit pantry, adding or deleting the ingredients they have at their home.
3.2 Scan and recognize the ingredients they have by uploading photos or taking instant photos
Source files:
application/backend/apps/models/userIngredient.py
application/backend/apps/daos/userIngredient_dao.py

application/backend/apps/models/userIngredient.py
application/backend/apps/services/user_ingredient_service.py

Feature 4: Recommendations based on user pantry
4.1. Give the users recipe recommendations based on their ingredients.
4.2 Recommended recipes have a metric indicating the proportion of ingredients they have.
4.3 The detail page of the recommended recipes will show the missing ingredients
Source files:
application/backend/apps/models/userIngredient.py
application/backend/apps/models/userIngredient.py
application/backend/apps/models/userIngredient.py
application/backend/apps/services/user_ingredient_service.py

Feature 5: Recipe rating system
5.1. Implement a rating system where users can rate recipes on a scale (e.g., 1 to 5 stars).
5.2. Allow users to leave reviews or comments along with their ratings.

Source files:
application/backend/apps/models/review.py
application/backend/apps/services/review_service.py
application/backend/apps/daos/review_dao.py