Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_ Едуард ЖАРІКОВ

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 р.

**ДОДАТОК ДЛЯ СКЛАДАННЯ ЗБАЛАНСОВАНОГО МЕНЮ**

**Програма та методика тестування**

КПІ.ІТ-7304.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

\_\_\_\_\_\_\_\_\_\_\_\_ Тетяна ЛІХОУЗОВА

|  |  |
| --- | --- |
| Нормоконтроль: | Виконавець: |
| \_\_\_\_\_\_\_\_\_\_\_ Катерина ЛІЩУК | \_\_\_\_\_\_\_\_\_\_ Ілля БЄКІШЄВ |

Київ – 2022

зміст

1 Об’єкт випробувань ........................................................................................ 3

2 Мета тестування .............................................................................................. 4

3 Методи тестування.......................................................................................... 6

4 Засоби та порядок тестування........................................................................ 7

5 Вимоги до контрольного прикладу................................................................ 9

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є додаток для складання меню. Додаток створюється для того, щоб надати можливість швидко та зручно створювати меню збалансованого харчування без знань як саме складається меню. Основною ціллю є зменшення часу який потрібен людні для створення меню збалансованого харчування.

Основник критерієм ефективності буде саме виконання основної цілі проекту. Тобто, чим більше буде помітно що люди стали харчуватися краще після публікації веб-додатку, тим більшим будемо вважати показник ефективності функціонування нашого програмного продукту.

Основні базові функції, які має мати наш додаток для досягнення поставлених цілей:

* Реєстрація у додатку.
* Ведення каталогу продуктів та рецептів:
  + додавання продуктів з інформацією про їх поживну цінність;
  + додавання рецептів з існуючих у БД продуктів;
* Створення меню збалансованого харчування з врахуванням обмежень по складовим, калорійності, вартості:
  + списки обраних блюд;
  + списки обраних кухонних приладь;
  + списки обраних продуктів.
* Ведення каталогу магазинів
  + додавання магазинів.

Функціональні вимоги, що є додатковими:

* Інформація про поточний стан здоров’я користувача (перелік обмежень на продукти).
* Список присутніх у користувача кухонних приладь.
* Каталог з цінами на продукти в окремих магазинах.

Основні функції будуть реалізовані в контексті функціоналу сторінок перегляду детальної інформації о продуктах та супермаркетів або у списках супермаркетів та продуктів.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

* перевірка правильності роботи програмного забезпечення у відповідно до функціональних вимог;
* знаходження проблем та недоліків з метою їх усунення;
* перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

* статичне тестування − перевіряється уся документація, яка аналізується на предмет дотримання стандартів програмування;
* динамічне тестування − застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на безлічі тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми в роботі програми;
* тестування «білої скриньки» − об’єктом тестування тут є внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Для тестування проекту було вибрано мануальні та автоматичні тести. Мануальні через економію часу на розробку. В той час як автоматичні тести використовуються або для самих важливих речей, або для тих речей котрі критичні для проекту і де треба виключити людський фактор з тестування.

Автоматизоване тестування чи автоматизація тестування – це метод тестування програмного забезпечення, який виконується з допомогою спеціальних програмних засобів, які, у свою чергу необхідні для виконання набору тестових прикладів. Автоматизоване тестування, найчастіше, вводиться на заміну ручного тестування, яке виконується людиною. Зрозуміло, що виконання ручного тестування займає багато часу та має високу вірогідність допущення помилки людиною, саме через це, наразі, уся індустрія тяжіє до заміщення ручного тестування автоматизованим.

Програмне забезпечення для автоматизації тестування також може вводити тестові дані у тестове середовище, порівнювати очікувані та фактичні результати та створювати докладні звіти про тести. Як правило, автоматизація тестування потребує значних вкладень грошей та ресурсів. Але кінцевий результат, у будь-якому разі принесе більше вигоди.

Для збільшення рентабельності інвестицій в автоматизацію тестування, випадки для автоматизації їх тестування можна вибрати за такими критеріями:

* високі ризики та збої неприпустимі – вкрай актуальні для банківської сфери;
* сценарії дій користувачів, які регулярно повторюються;
* сценарії дій користувачів, які є складними та стомлюючими для виконання вручну;
* сценарії взаємодії компонентів між собою;
* сценарії роботи кожного компоненту окремо.

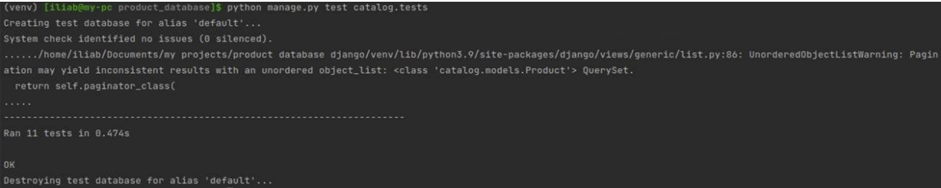


Рисунок 4.1 – Приклад запуску автономних тестів

5 ВИМОГИ ДО КОНТРОЛЬНОГО ПРИКЛАДУ

В таблицях 5.1 – 5.11 детально описані контрольні приклади тестування різних складових додатку.

Таблиця 5.1 – Перевірка поля name

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного відображення лейбла назви продукту на формі змінення даних продукту |
| Початковий стан | Пуста форма зміни інформації о продукті |
| Схема проведення тесту | Створюємо пусту форму та перевіряємо завдяки assertTrue поле name на формі |
| Очікуваний результат | Застосунок повинен відобразити name як в полі яке належить до поля name на формі |

Таблиця 5.2 – перевірка підказки до поля name

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного відображення підказки до поля назва продукту на формі змінення даних продукту |
| Початковий стан | Пуста форма зміни інформації о продукті |
| Схема проведення тесту | Створюємо пусту форму та перевіряємо завдяки assertEqual допоміжний текст до поля name на формі |
| Очікуваний результат | Застосунок повинен відобразити Enter product name напроти поля name на формі |

Таблиця 5.3 – перевірка поля name

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного відображення лейбла назви продукту на формі для моделі продукт |
| Початковий стан | Створюємо приблизну модель продукту |
| Схема проведення тесту | Отримуємо об’єкт продукту який був попередньо створений, отримуємо лейбл поля name та за допомогою assertEquals порівнюємо з очікуваним результатом |
| Очікуваний результат | Застосунок повинен відобразити лейбл поля name який дорівнює name |

Таблиця 5.4 – перевірка поля weight

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного відображення лейбла ваги продукту на формі для моделі продукт |
| Початковий стан | Створюємо приблизну модель продукту |
| Схема проведення тесту | Отримуємо об’єкт продукту який був попередньо створений, отримуємо лейбл поля weight та за допомогою assertEquals порівнюємо з очікуваним результатом |
| Очікуваний результат | Застосунок повинен відобразити лейбл поля weight який дорівнює weight |

Таблиця 5.5 – перевірка максимальної довжини поля name

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного показнику довжини поля назва продукту |
| Початковий стан | Створюємо приблизну модель продукту |
| Схема проведення тесту | Отримуємо об’єкт продукту який був попередньо створений, отримуємо максимальну довжину поля name та за допомогою assertEquals порівнюємо з очікуваним результатом |
| Очікуваний результат | Попередньо створена модель для тесту має мати максимальну довжину поля назва продукту рівну 200 |

Таблиця 5.6 – перевірка отримання коректного посилання на сторінку з детальною інформацією

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного створення посилання на сторінку з детальною інформацією о продукті |
| Початковий стан | Створюємо приблизну модель продукту |
| Схема проведення тесту | Отримуємо об’єкт продукту який був попередньо створений, за допомогою assertEquals порівнюємо посилання яке згенерувалося само з очікуваним результатом |
| Очікуваний результат | Застосунок повинен згенерувати посилання на сторінку з детальною інформацією о продукті яка дорівнює /catalog/product/1 |

Таблиця 5.7 – перевірка чи посилання до каталогу продуктів існує

|  |  |
| --- | --- |
| Мета тесту | Перевірка на те чи існує сторінка з списком усіх продуктів |
| Початковий стан | Створюємо 23 приблизні моделі продукту |
| Схема проведення тесту | Отримуємо відповідь від сервера при спробі потрапити на сторінку /catalog/products/, за допомогою assertEquals порівнюємо посилання яке згенерувалося з очікуваним результатом |
| Очікуваний результат | Застосунок повинен згенерувати сторінку з списком продуктів і при спробі отримати результат цього створення відповідь від серверу повинна мати статус коду який дорівнює 200 |

Таблиця 5.8 – перевірка чи можливо отримати посилання до сторінки з детальною інформацією о продукті за допомогою назви каталогу

|  |  |
| --- | --- |
| Мета тесту | Перевірка на те чи існує сторінка з списком усіх продуктів за допомогою короткого посилання |
| Початковий стан | Створюємо 23 приблизні моделі продукту |
| Схема проведення тесту | Отримуємо відповідь від сервера при спробі потрапити на сторінку /catalog/products/ за допомогою короткого посилання reverse('products'), за допомогою assertEquals порівнюємо посилання яке згенерувалося само з очікуваним результатом |
| Очікуваний результат | Застосунок повинен згенерувати сторінку з списком продуктів і при спробі отримати результат цього створення відповідь від серверу повинна мати статус коду який дорівнює 200 |

Таблиця 5.9 – перевірка відображення правильного шаблону веб сторінки

|  |  |
| --- | --- |
| Мета тесту | Перевірка використання коректного шаблону сторінки з детальною інформацією о продукті |
| Початковий стан | Створюємо 23 приблизні моделі продукту |
| Схема проведення тесту | Отримуємо відповідь від сервера при спробі потрапити на сторінку /catalog/products/ за допомогою короткого посилання reverse('products'), за допомогою assertEquals порівнюємо шаблон який обрався самостійно з очікуваним результатом |
| Очікуваний результат | Застосунок повинен згенерувати сторінку з списком продуктів і при спробі отримати результат цього створення шаблон сторінки повинен бути правильним |

Таблиця 5.10 – перевірка правильної роботи пагінації

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректної роботи пагінації на сторінці з списком усіх продуктів |
| Початковий стан | Створюємо 23 приблизні моделі продукту |
| Схема проведення тесту | Отримуємо відповідь від сервера при спробі потрапити на сторінку /catalog/products/ за допомогою посилання reverse('products'), за допомогою assertTrue перевіряємо перевіримо те що пагінація працює і саме 20 продуктів відображаються на одній сторінці |
| Очікуваний результат | Застосунок повинен згенерувати сторінку з списком усіх продуктів та на першій сторінці списку повинно бути лише 20 продуктів |

Таблиця 5.11 – перевірка правильного відображення каталогу усіх продуктів

|  |  |
| --- | --- |
| Мета тесту | Перевірка коректного створення сторінок з списком продуктів |
| Початковий стан | Створюємо 23 приблизні моделі продукту |
| Схема проведення тесту | Отримуємо відповідь від сервера при спробі потрапити на сторінку /catalog/products/?page=2 за допомогою короткого посилання reverse('products') + '?page=2', за допомогою assertTrue перевіряємо перевіримо те що пагінація працює і саме 3 продукти відображаються на другій сторінці списку усіх продуктів |
| Очікуваний результат | Застосунок повинен згенерувати сторінку з списком усіх продуктів та на другій сторінці списку повинно бути лише 3 продукти |