

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СТВОРЕННЯ ПЕРСОНАЛЬНИХ
ВІДГУКІВ ПРО СТРАВИ
SOFTWARE FOR PERSONAL FOOD REVIEWS

Виконав: студент(ка) 4 курсу, групи КНТ-119сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Бережний Д.В.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Бережного Дмитра Вікторовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення створення персональних відгуків про страви
Software for Personal Food Reviews

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання «4» квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис)

Бережний Д.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 83 с., 21 рис., 2 табл., 3 дод., 12 джерел.

СТРАВА, РЕСТОРАН, ПЕРСОНАЛЬНИЙ ВІДГУК, РЕЦЕНЗЕНТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ.

Об'єкт дослідження – процес приготування і споживання страв. Предмет дослідження – програмне забезпечення створення персональних відгуків про страви. Мета дослідження – розробити програмне забезпечення створення персональних відгуків про страви для пошуку варіантів урізноманітнення споживання їжі.

Матеріали, методи та технічні засоби: мова Python, вебфреймворк Django, персональний комп'ютер під керування Windows 10.

Результати. У результаті виконано аналіз аналогів і їх порівняння з розробкою, представлено результати порівняння фреймворків для розробки вебзастосунків мовою Python, обрано фреймворк Django у цій якості, представлено діаграму варіантів використання, схему бази даних, схему функціонування програми, описано створені одиниці програми та загальну логіку роботи з програмою.

Висновки. Програма дозволила в результаті створити інструмент для накопичення власного досвіду, що стосується споживання різних страв у різних закладах і вдома, та вибору нових страв або споживання цих страв у певних закладах, виконуючи ці рішення на основі досвіду інших користувачів. Також програма може використовуватися для збереження власних вражень про страви, які користувач фактично як рецензент спробував або приготував.

Галузь використання. Споживання і приготування їжі в ресторанах і вдома.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 21 figures, 2 tables, 3 appendixes, 12 sources.

DISH, RESTAURANT, PERSONAL REVIEW, REVIEWER, SOFTWARE, DATABASE.

The object of research is the process of cooking and eating. The subject of the research is software for creating personal reviews about dishes. The aim of the work is to develop software for creating personal reviews of dishes to find options for diversifying food consumption.

Materials, methods and technical means: Python language, Django web framework, personal computer managed by Windows 10.

Results. As a result the analysis of analogues and its comparison with the development is performed, the results of comparison of frameworks for webapplication development using Python programming language are presented. Django framework is chosen for the purpose. Use case diagram, database scheme, program operation scheme are presented together with the description of interaction with the program.

Conclusions. The program allowed to create a tool for gaining personal experience in consuming different dishes in different establishments and at home, and choosing new dishes or consuming these dishes in certain establishments, implementing these decisions based on the experience of other users. The program can also be used to save user's impressions concerning dishes that a user actually tried as a reviewer or prepared by himself.

Field of use. Consumption and cooking in restaurants and at home.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області	9
1.1 Рецензування страв	9
1.2 Огляд програмних аналогів	10
1.3 Висновки за розділом 1 та завдання роботи	13
2 Розробка архітектури програми	15
2.1 Визначення функціональних вимог до програми	15
2.2 Вибір засобів розробки	17
2.3 Проектування бази даних	18
2.4 Висновки за розділом 2	27
3 Розробка програми	28
3.1 Алгоритм функціонування програми	28
3.2 Опис реалізованих класів програми	29
3.3 Висновки за розділом 3	41
4 Експлуатація та тестування програми	42
4.1 Призначення програми	42
4.2 Умови виконання програми	42
4.3 Експлуатація програми	42
4.4 Тестування програми	47
4.5 Висновки за розділом 4	47
Висновки	48
Перелік джерел посилання	49
Додаток А Технічне завдання	51
Додаток Б Текст програми	55
Додаток В Слайди презентації	77

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVT – Model-View-Template;

БД – база даних.

ВСТУП

Звичайно, що пояснювати, на скільки їжа є важливою для життя людини не потрібно нікому. Це одна з базових потреб організму людини, її фізіологічна потреба, яка зокрема знаходиться в основі піраміди потреб Маслоу, відповідно для досягнення потреб більш високого рівня потрібно спочатку задовольнити її серед фізіологічних потреб.

Але окрім того їжа ще й є центральною частиною багатьох культур [1]. Знайомлячись з певною країною, люди часто знайомляться з кухнею цієї країни як однією з найперших її складових. З розвитком суспільства їжа є частиною процесу комунікації людей, коли вони знаходяться в ресторані або іншому громадському закладі.

Задля того щоб їжа приносила не менше задоволення з часом і була не тільки фізіологічною потребою, люди пробують нові страви або пробують їх в різних закладах за різними рецептами, діляться думками щодо своїх вражень, готують самостійно вдома, використовуючи різні, в тому числі нові рецепти.

Зважаючи на розповсюдження даних процесів, слід вказати на те, що процеси, пов'язані зі споживанням нових страв в ресторанах або страв, які клієнт пробував у іншому ресторані, а не в тому, де знаходиться в поточний момент, а також з приготуванням нових страв або страв за новими рецептами, потребують підтримки на рівні створення певної мережі для взаємодії людей між собою. У цій мережі має бути можливість ділитися користувачам думками між собою щодо страв. Розповсюдженням засобом вираження власних думок щодо будь-яких продуктів у сучасному світі є відгуки.

Тому в результаті в дипломній роботі вирішено було розробляти програмне забезпечення створення персональних відгуків про страви.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Рецензування страв

Рецензування ресторанів є загальновідомим напрямком реалізації програмного забезпечення. Однак, коли клієнт приходить в ресторан, перед ним виникає запитання, що саме обрати, яку саме страву скуштувати. Таке ж запитання перед ним в принципі виникає і тоді, коли він не є клієнтом, а є звичайною людиною, що вирішила приготувати якусь страву, яку вона раніше не куштувала, вдома.

Загалом рецензування страв важливе через наступні причини:

- зменшення ризику: споживачі використовують відгуки як засоби пом'якшення ризику, або зменшуючи ймовірність того, що покупка призведе до розчарування, або, принаймні, зменшуючи серйозність потенційного розчарування;
- скорочення часу пошуку: споживачі мають серйозні труднощі з обробкою інформації та варіантів, коли стикаються з ринками, де представлено велику кількість продуктів, таким чином, створення платформи оглядів зменшує витрати на пошук і допомагає споживачам легко й швидко сортувати інформацію про багато страв і не загубитися серед всього масиву страв з різних кухонь та всіх варіантів рецептів цих страв, які існують у світі;
- докори сумління покупця: звернення до відгуків зменшує жаль покупців, які можуть виникнути після того, як вони витратили нетривіальну суму грошей на свій обід у випадку, коли людина споживає страви в ресторані, простий доступ до інформації на платформах оглядів робить їх відповідним джерелом інформації, щоб зменшити ці докори сумління після використання корисного досвіду;
- груповий вплив: груповий вплив описує спосіб, у який люди погоджуються з групою однолітків або однодумців, це важливо, оскільки загальні відчуття, озвучені у відгуку, можуть вплинути на сприйняття та поведінку споживача, навіть якщо людина ще не споживала цю страву,

платформа для огляду може створити або зміцнити дух клієнта ресторану і, отже, так чи інакше впливати на процес прийняття рішень [2].

1.2 Огляд програмних аналогів

Програма Yelp (рис. 1.1) [3] є одним з розповсюджених засобів створення відгуків, які зокрема використовуються і в ресторанній галузі.

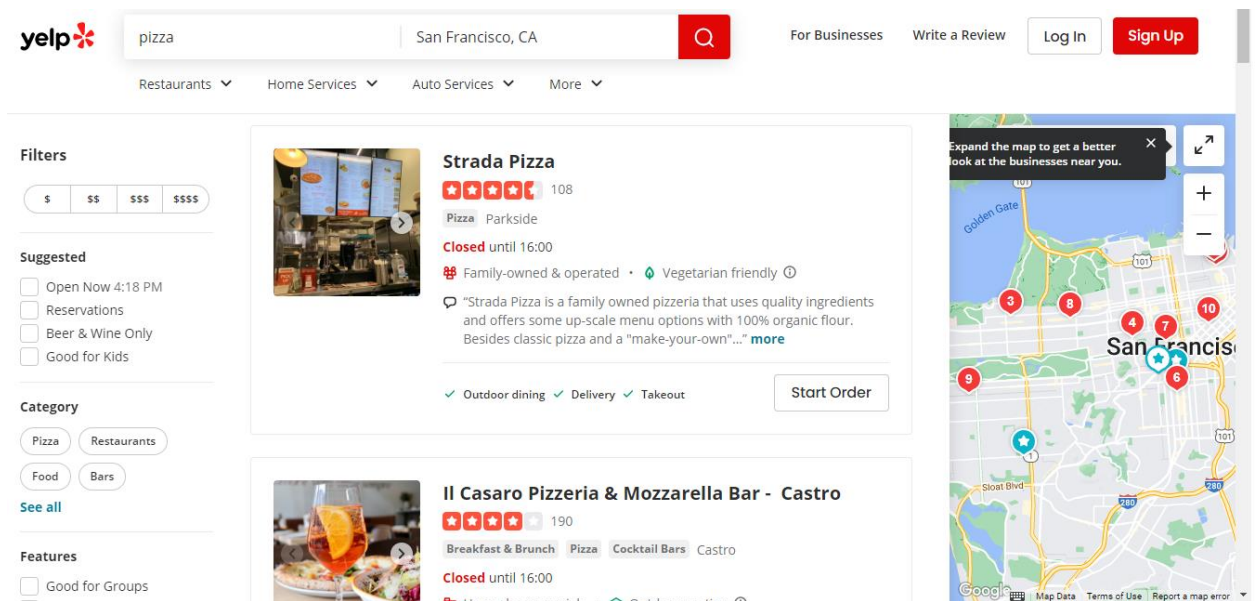


Рисунок 1.1 – Програма Yelp

Yelp дозволяє виконувати пошук сервісів, які надаються на ринку певного міста, до яких належать зокрема і ресторани. Звичайно, що це не єдиний сервіс, де надається можливість рецензування і пошуку відповідно ресторанів. У програмі можна також шукати і страви, але за результатом фактично надається перелік ресторанів, де ця страва готується, а самі відгуки стосуються саме сервісу загалом, тобто конкретного ресторану.

Другим засобом є програма Burpple (рис. 1.2) [4]. Вона надає можливість шукати за закладами, переглядати відгуки за цими відгуками, хоча в якості відгуків часто описується фактично споживання конкретної страви в конкретному ресторані. Це дещо зближує цю програму з

запланованою розробкою, проте це не є ідеєю програми, відповідно засобів шукати за конкретними стравами або за конкретними ресторанами в конкретних ресторанах немає.

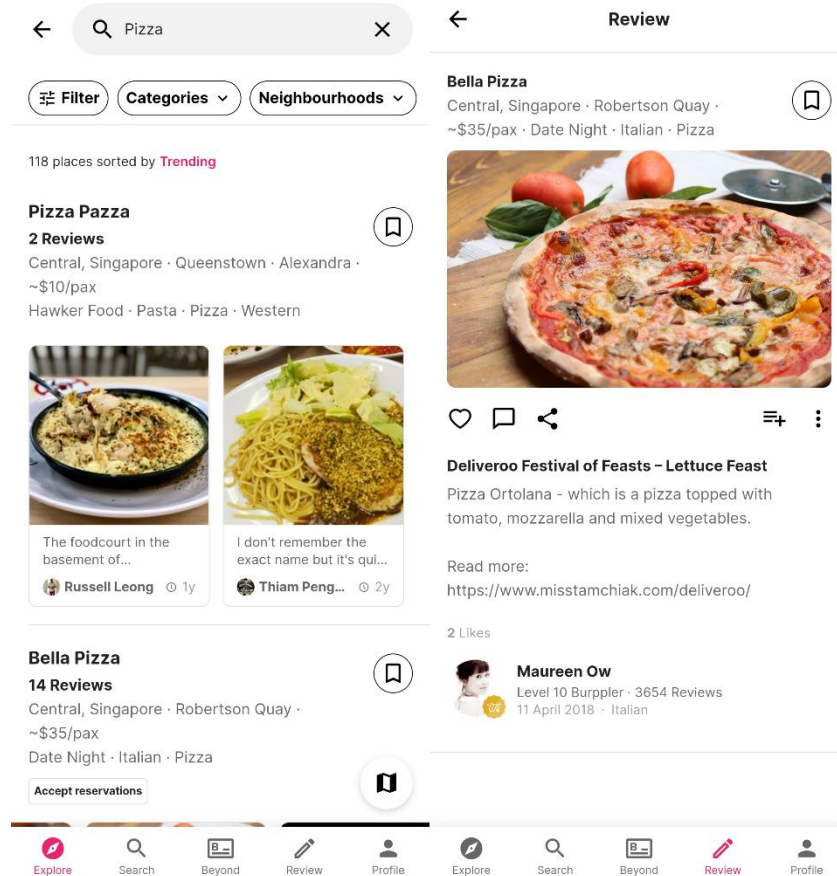


Рисунок 1.2 – Програма Burpple

Burpple надає угоди, які мають форму пропозицій «1-на-1» і доступні в більш ніж 500 ресторанах-партнерах і цей простір далі збільшується: як учасник, користувач має право на певну кількість пропозицій на ресторан – дві для тарифного плану Lite або чотири для тарифного плану Premium, залежно від того, який тариф було придбано, членство дійсне протягом 12 місяців, окрім заощаджень на обідах, фактично вже другорядною перевагою Burpple є можливість відкривати нові заклади харчування завдяки активній спільноті користувачів, які регулярно публікують детальні огляди їжі [5].

Третьою програмою, що було окремо виділено, є програма Tasty (рис. 1.3) [6]. Відмінною особливістю програм для рецептів є можливість

обміну знаннями між користувачами, і розробники Tasty вдосконалили це у своїй програмі: є розділи, де можна оцінювати, коментувати та пропонувати зміни рецептів, при цьому всі коментарі слід сприймати з певною недовірою, оскільки вони надходять переважно від кухарів-аматорів, окрім того можна редагувати свої особисті смаки в програмі та вносити в чорний список рецепти, які користувача не цікавлять [7].

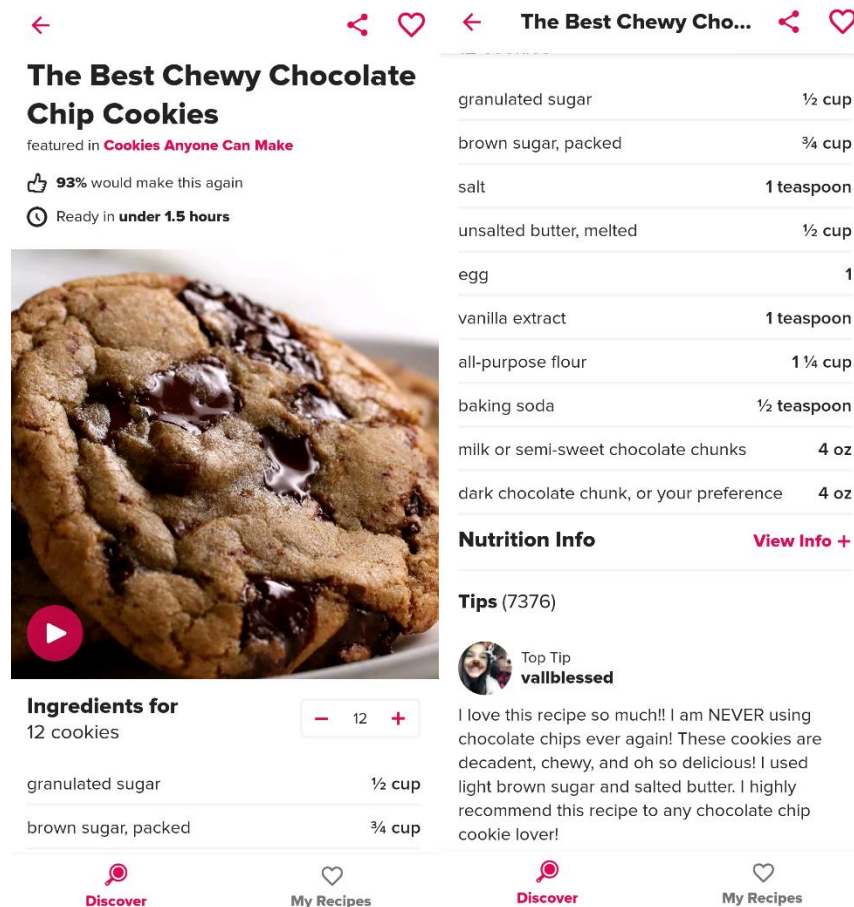


Рисунок 1.3 – Програма Tasty

Ці аналоги виконуваної розробки було порівняно між собою і з цією розробкою. Результати представлені в табл. 1.1.

Виконувана розробка повинна мати можливості, що стосуються як ресторанного споживання страв, так і готування страв, тобто фактично зв'язувати між собою страви і рецепти, надаючи гнучкі можливості співвіднесення відгуків зі стравами, з рецептами, з приготуванням

конкретних страв у конкретних ресторанах. Саме це фактично і є її головною концепцією, що дозволяє привернути увагу різних прошарків користувачів, особливо відділяючи тих, хто хоче знайти цікаву страву і передбачає перед її самостійним приготуванням похід у заклад харчування.

Таблиця 1.1 – Порівняння аналогів з виконуваною розробкою

Показник	Yelp	Burpple	Tasty	Розробка
Пошук за стравами	Тільки якщо страви вказані в тегах закладів	Ні	Так	Так
Фотографії страв	Так	Так	Так	Так
Накопичення відгуків за стравами	Ні	Ні	Ні	Так
Накопичення відгуків за рецептами	Ні	Ні	Так	Так
Накопичення відгуків про ресторани	Так	Так	Ні	Ні
Накопичення відгуків за конкретними стравами в конкретному закладі	Ні	Ні	Ні	Так
Рецепти	Ні	Ні	Так	Так

1.3 Висновки за розділом 1 та завдання роботи

Завдання роботи:

- дослідження проблеми створення персональних відгуків про страви;
- визначення засобів розробки;
- визначення вимог до програми;

- проєктування програми створення персональних відгуків про страви;
- програмна реалізація.

Розглянуто актуальність і важливість рецензування страв з визначенням причин цієї важливості, що охоплюють зменшення ризику, скорочення часу пошуку та вплив групової думки. Виконано огляд програмних засобів, що включають програми Yelp, Burpple, Tasty, визначено особливості розроблюваної програми порівняно з цими аналогами.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Визначення функціональних вимог до програми

Основною направленістю програми є робота з відгуками, що стосуються страв і певних варіантів цих страв. Для того щоб цей процес був зручним і результативним в плані отримання корисної інформації для подальших рішень, було визначено такі функціональні вимоги до програми:

- створення відгуку про споживання страви у певному закладі;
- ознайомлення з відгуками заданого рецензента;
- перегляд власних відгуків;
- перегляд відгуків за стравою;
- перегляд відгуків про споживання певної страви в певному ресторані;
- перегляд відгуків за конкретним рецептом;
- перегляд відгуків уподобаних рецензентів;
- уточнення відгуку;
- ознайомлення зі стравою;
- перегляд страв за країною;
- перегляд рейтингу страв;
- перегляд рейтингу ресторанів щодо приготування певної страви;
- перегляд даних рецензента;
- авторизація рецензента, його реєстрація і оновлення даних;
- оновлення даних про приготування страви в ресторані;
- ознайомлення з рестораном;
- уподобання рецензента і зняття уподобання;
- ознайомлення з типом страв;
- ознайомлення з країною та місцем;
- створення рецепту;
- ознайомлення з рецептом.

Усі ці вимоги було перетворено на набір сценаріїв, за якими рецензент, тобто користувач, який взаємодії з відгуками і створює їх, працює з програмою (рис. 2.1).



Рисунок 2.1 – Діаграма варіантів використання

До сценаріїв роботи рецензента належать:

- створення відгуку;
- перегляд відгуків, що є сценарієм, який узагальнює набір спеціальних сценаріїв;
- ознайомлення зі стравою;
- ознайомлення з рецептом;
- перегляд страв за країною;
- перегляд рейтингу страв;

- перегляд рейтингу ресторанів;
- ознайомлення з ресторанами;
- уподобання рецензента;
- створення рецепту;
- перегляд даних рецензента;
- ознайомлення з відгуками заданого рецензента.

2.2 Вибір засобів розробки

Розробку вебзастосунку заплановано на основі мови Python [8]-[10] через її придатність для вирішення широкого кола задач і зокрема створення вебзастосунків, які за рахунок використання вебфреймворку придатні для використання у бізнесах різного рівня, хоча на рівень впливає і вибір фреймворка. Тому тепер потрібно порівняти (табл. 2.1) два найпопулярніші фреймворки веброзробки для Python: Django [11]-[12] і Flask.

Таблиця 2.1 – Вибір вебфреймворку розробки мовою Python

Показник	Django	Flask
Створення структури бази даних на основі коду	+	–
Швидкий вибір даних з бази даних	+	–
Автоматичне створення засобів адміністрування	+	–
Формування динамічних вебсторінок	+	–
Використовувана мова програмування	Python	Python

Вибір у підсумку фреймворку Django обґрунтований його основними перевагами, що включають можливість дуже зручної роботи з базами даних

(БД) за рахунок уникнення необхідності подвійного визначення структури даних через код і структуру БД, наявність виразів для створення кодів без необхідності використання повних запитів, що традиційно для БД, а також за рахунок того, що створення засобів внесення і редагування базових даних через адміністративні засоби відбувається в даному випадку дуже швидко і не потребує додаткової роботи. Враховуючи вибір цього фреймворку, шаблон проєктування Model-View-Template (MVT) був покладений у основу визначення структури проєкту.

2.3 Проєктування бази даних

Проєктування БД є важливою частиною проєктування застосунку загалом, оскільки виділена структура лягає в основу моделей, які забезпечуються для роботи з даними у відповідності з фреймворком Django.

Збереження страв виконується таблицею Meal за наступними полями:

- id – ідентифікатор страви;
- mealtype – тип страви;
- named – назва страви;
- about – детальний опис страви;
- photo – фотографія страви;
- rating – середній рейтинг страви за всіма відгуками людей, що її вже їли;
- restaurantrating – середній рейтинг страви за всіма відгуками людей, що її їли саме в ресторані;
- homerating – середній рейтинг страви за всіма відгуками людей, що її їли вдома;
- country – країна, з кухні якої походить страва.

Поле mealtype є зовнішнім ключем з таблиці MealType. Таким чином реалізується зв'язок між таблицями Meal і MealType, що визначений як один до багатьох: один тип страви зазвичай має багато страв, що його

представляють, але не може бути такого, що одна страва представляє різні типи.

Збереження рецептів страв виконується таблицею `Recipe` за наступними полями:

- `id` – ідентифікатор рецепту;
- `meal` – страва, рецепт якої визначено;
- `instruction` – інструкція рецепту;
- `about` – опис особливостей репету;
- `photo` – фотографія рецепту;
- `rating` – середній рейтинг даного рецепту за всіма відгуками людей, що її вже їли;
- `vegetarian` – вегетаріанська їжа чи ні;
- `named` – назва рецепту.

Поле `meal` є зовнішнім ключем з таблиці `Meal`. Таким чином реалізується зв'язок між таблицями `Meal` і `Recipe`, що визначений як один до багатьох: одна страва може мати один або багато рецептів для її приготування, але не може бути такого, що рецепт визначає приготування різних страв.

Збереження продуктів виконується таблицею `Product` за наступними полями:

- `id` – ідентифікатор продукту;
- `productgroup` – група продуктів, до якої належить цей продукт;
- `named` – назва продукту;
- `about` – опис продукту;
- `photo` – фотографія продукту.

Поле `productgroup` є зовнішнім ключем з таблиці `ProductGroup`. Таким чином реалізується зв'язок між таблицями `ProductGroup` і `Product`, що визначений як один до багатьох: одна група продуктів може мати багато продуктів, але не може бути такого, що один продукт представляє різні групи.

Збереження груп продуктів виконується таблицею ProductGroup за наступними полями:

- id – ідентифікатор групи продуктів;
- named – назва групи продуктів;
- photo – фотографія групи продуктів.

Збереження відгуків виконується таблицею Review за наступними полями:

- id – ідентифікатор відгуку;
- created – дата публікації відгуку;
- about – враження користувача у відгуку;
- restaurantmeal – ресторан, де було приготовано страву, що спробував користувач;
- reviewer – користувач, який створив відгук;
- rating – загальний рейтинг, виставлений цьому вживанню страви;
- delivery – служба доставки, яка привезла страву користувачу;
- photo – фотографія страви.

Поле restaurantmeal є зовнішнім ключем з таблиці RestaurantMeal. Таким чином реалізується зв'язок між таблицями RestaurantMeal і Review, що визначений як один до багатьох: одна страва в ресторані може мати багато відгуків, але не може бути такого, що один відгук стосується різних страв в ресторані або ресторанах.

Поле reviewer є зовнішнім ключем з таблиці Reviewer. Таким чином реалізується зв'язок між таблицями Reviewer і Review, що визначений як один до багатьох: один рецензент може не мати жодного, мати один або мати багато відгуків, але не може бути такого, що один відгук створений різними рецензентами.

Збереження показника, за яким оцінюється страва користувачем, виконується таблицею Measure за наступними полями:

- id – ідентифікатор показника оцінювання;
- named – показник оцінювання (його назва).

Збереження ресторанів виконується таблицею Restaurant за наступними полями:

- id – ідентифікатор ресторану;
- named – назва ресторану;
- place – місце, де знаходиться ресторан;
- about – опис ресторану;
- photo – фотографія ресторану;
- point – місце, де знаходиться ресторан;
- rating – загальний рейтинг страв, виставлених ресторану;
- phone – номер телефону ресторану.

Поле place є зовнішнім ключем з таблиці Place. Таким чином реалізується зв'язок між таблицями Place і Restaurant, що визначений як один до багатьох: одне місце на мапі може мати багато ресторанів (жодного, один або багато), але не може бути такого, що один ресторан знаходиться в декількох місцях.

Збереження приготування страви в ресторані виконується таблицею RestaurantMeal за наступними полями:

- id – ідентифікатор приготування страви в ресторані;
- restaurant – ресторан, де відбувається приготування;
- meal – страва, яка виготовляється в ресторані;
- price – вартість цієї страви в цьому ресторані;
- recipe – рецепт, за яким готується страва в цьому ресторані (якщо це приблизно відомо);
- rating – середній рейтинговий бал, виставлений за всіма відгуками споживанню цієї страви в цьому ресторані.

Фактично таблиця RestaurantMeal реалізує зв'язок між таблицями Restaurant і Meal багато до багатьох, розширюючи його додатковим полем.

Поле restaurant є зовнішнім ключем з таблиці Restaurant. Таким чином реалізується зв'язок між таблицями Restaurant і RestaurantMeal, що визначений як один до багатьох: один ресторан може мати багато страв, які

готуються в ресторані, але не може бути такого, що одна подана страва приготовлена і в одному ресторані, і в іншому.

Поле `meal` є зовнішнім ключем з таблиці `Meal`. Таким чином реалізується зв'язок між таблицями `Meal` і `RestaurantMeal`, що визначений як один до багатьох: одна страва може бути приготовлена в багатьох ресторанах (не одночасно, а в принципі готуватися), але не може бути такого, що одне приготування реалізовано спільними зусиллями різних ресторанів.

Поле `receipt` є зовнішнім ключем з таблиці `Receipt`. Таким чином реалізується зв'язок між таблицями `Receipt` і `RestaurantMeal`, що визначений як один до багатьох: один рецепт може використовуватись різними ресторанами для приготування деякої страви, але не може бути такого, що приготування страви в ресторані реалізується за різними рецептами.

Збереження оцінювань у відгуку страви за показником виконується таблицею `ReviewMeasure` за наступними полями:

- `id` – ідентифікатор оцінювання страви за показником;
- `review` – відгук, до якого відноситься це оцінювання за показником;
- `measure` – показник, за яким виконується оцінювання страви;
- `rating` – оцінка страви.

Поле `review` є зовнішнім ключем з таблиці `Review`. Таким чином реалізується зв'язок між таблицями `Review` і `ReviewMeasure`, що визначений як один до багатьох: один відгук може мати багато оцінювань за різними показниками, але не може бути такого, що одне оцінювання за показником стосується різних відгуків.

Поле `measure` є зовнішнім ключем з таблиці `Measure`. Таким чином реалізується зв'язок між таблицями `Measure` і `ReviewMeasure`, що визначений як один до багатьох: один показник, за яким виконується оцінювання у відгуку, може мати багато оцінювань у різних відгуках стосовно різних страв, але не може бути такого, що одне оцінювання стосується різних показників.

Збереження продуктів у страві виконується таблицею `ReceiptProduct` за наступними полями:

- `id` – ідентифікатор продукту в страві;
- `receipt` – рецепт, до якого належить цей продукт;
- `product` – продукт, який входить в рецепт;
- `quant` – кількість продукту;
- `quantunit` – одиниця виміру кількості продукту.

Фактично таблиця `ReceiptProduct` реалізує зв'язок між таблицями `Receipt` і `Product` багато до багатьох, розширюючи його додатковими полями.

Поле `receipt` є зовнішнім ключем з таблиці `Receipt`. Таким чином реалізується зв'язок між таблицями `Receipt` і `ReceiptProduct`, що визначений як один до багатьох: один рецепт може вимагати багатьох продуктів для приготування страви за цим рецептом, але не може бути такого, що один продукт визначений для різних рецептів, адже чітко регламентується застосування продукту (його кількість), відповідно всі застосування продуктів у різних рецептах різні.

Поле `product` є зовнішнім ключем з таблиці `Product`. Таким чином реалізується зв'язок між таблицями `Product` і `ReceiptProduct`, що визначений як один до багатьох: один продукт може входити до багатьох рецептів приготування страв, але не може бути такого, що одне застосування продукту для приготування стосується різних рецептів.

Збереження рецензентів виконується таблицею `Reviewer` за наступними полями:

- `id` – ідентифікатор рецензента;
- `username` – логін рецензента;
- `password` – пароль рецензента;
- `reviews` – кількість відгуків, створених рецензентом;
- `meals` – кількість різних страв, оцінених рецензентом;
- `rating` – середній рейтинг всіх страв, які скуштував рецензент;
- `likes` – кількість рецензентів, яких уподобав даний рецензент;

– liked – кількість рецензентів, які уподобали даного рецензента.

Збереження місць, де можуть розташовуватися ресторани, виконується таблицею Place за наступними полями:

- id – ідентифікатор місця;
- named – назва місця;
- placetype – тип місця (місто, селище);
- country – країна, де знаходиться це місце.

Поле country є зовнішнім ключем з таблиці Country. Таким чином реалізується зв'язок між таблицями Country і Place, що визначений як один до багатьох: одна країна може мати багато міст і селищ, але не може бути такого, що одне місто або селище одночасно знаходиться в різних країнах.

Збереження типів страв виконується таблицею MealType за наступними полями:

- id – ідентифікатор типу страви;
- named – назва типу страви;
- photo – фотографія цього типу страви.

Збереження даних країни виконується таблицею Country за наступними полями:

- id – ідентифікатор країни;
- named – назва країни;
- cuisinenamed – назва кухні країни (для того щоб отримати форму слова, придатну для виведення перед словом кухня в інтерфейсі програми).

Збереження служби доставки виконується таблицею Delivery за наступними полями:

- id – ідентифікатор служби доставки;
- named – назва служби доставки;
- about – опис служби доставки;
- web – вебадреса служби доставки.

Збереження уподобань рецензентів виконується таблицею Likedreviewer за наступними полями:

- id – ідентифікатор уподобання;
- reviewer – рецензент, який виконав уподобання;
- likereviewer – рецензент, якого було уподобано;
- created – час уподобання.

Поле reviewer є зовнішнім ключем з таблиці Reviewer. Таким чином реалізується зв'язок між таблицями Reviewer і Likedreviewer, що визначений як один до багатьох: один рецензент може виконати багато уподобань, але не може бути такого, що одне уподобання створене різними рецензентами.

Поле likereviewer є зовнішнім ключем з таблиці Reviewer. Таким чином реалізується зв'язок між таблицями Reviewer і Likedreviewer, що визначений як один до багатьох: один рецензент може бути уподобаний багатьма іншими рецензентами, але не може бути такого, що одне уподобання направлене на різних рецензентів.

Збереження середніх оцінок за різними показниками страви виконується таблицею MealMeasure за наступними полями:

- id – ідентифікатор оцінки страви за показником;
- rating – оцінка страви за даним показником;
- meal – страва, якій було виставлено оцінку;
- measure – показник, за яким було виконано оцінювання страви.

Фактично таблиця MealMeasure реалізує зв'язок між таблицями Meal і Measure багато до багатьох, розширюючи його додатковим полем.

Поле meal є зовнішнім ключем з таблиці Meal. Таким чином реалізується зв'язок між таблицями Meal і MealMeasure, що визначений як один до багатьох: одна страва може мати багато оцінок за різними показниками, але не може бути такого, що одна така конкретна оцінка виставлена різним стравам.

Поле measure є зовнішнім ключем з таблиці Measure. Таким чином реалізується зв'язок між таблицями Measure і MealMeasure, що визначений як один до багатьох: один показник може бути використаний в багатьох

оцінюваннях різних страв, але не може бути такого, що одне оцінювання стосується різних показників.

Створена схема структури БД показана на рис. 2.2.

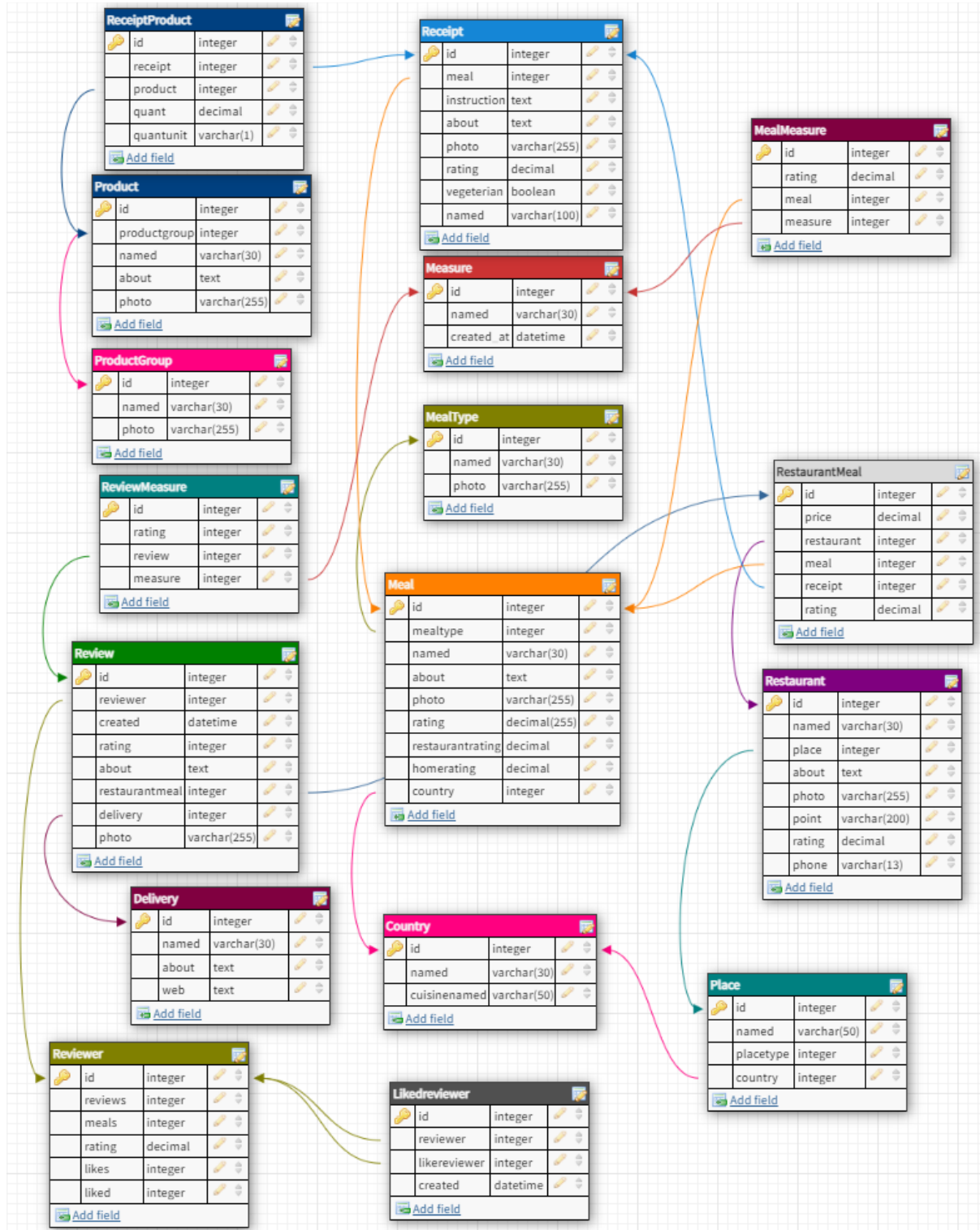


Рисунок 2.2 – Схема структури БД

2.4 Висновки за розділом 2

За результатами проєктування представлено результати порівняння фреймворків для розробки вебзастосунків мовою Python, обрано фреймворк Django у цій якості, що визначило структуру програми на основі використання шаблону проєктування MVT, представлено функціональні вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв'язки між ними.

З РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Процес роботи рецензента з програмою було уніфіковано для покращення орієнтування в програмі, оскільки вона має достатньо багато функціональних можливостей, які прості у використанні, проте потребують забезпечення доступу до них. Головним засобом керування доступом було вирішено визначити меню (рис. 3.1).

Будь-який користувач у випадку традиційного звернення до вебсервера, тобто вказуючи загальний шлях до вебсервера, розпочинає свою роботу з процесу відображення страв. Це окремий достатньо розгалуджений процес, який включає і рейтинги страв, і пошук страв, і відображення окремих страв, які було обрано з цих сформованих списків. Далі в будь-який момент користувач може змінювати і вибирати інший напрямок дій.

Окремим розділом є напрямок дій, який пов'язаний з відображенням рецептів. Хоча до рецептів можна отримати доступ і, наприклад, через сторінку страви. Але вони також і окремо виділені, щоб якщо користувача цікавлять тільки рецепти, він міг отримати швидко до них доступ.

Відображення ресторанів з меню дозволяє отримати список ресторанів, обрати потрібний, а звідти перейти до списку страв, які готуються в ресторані, і ознайомитись з відгуками за конкретними приготуваннями.

Відображення стрічки є ще одним варіантом, який можна обрати з меню. Він дозволяє працювати з відгуками від рецензентів, які уподобав рецензент. При цьому можна зокрема обирати інтервал балів для відображення відгуків у заданому діапазоні.

Відображення відгуків є окремим напрямком діяльності, адже тут фактично може реалізовуватися не тільки відображення відгуків загалом, але і відображення власних відгуків, а за необхідності перехід до створення

власного нового відгуку. Відгуки також можна створювати і з інших розділів меню.

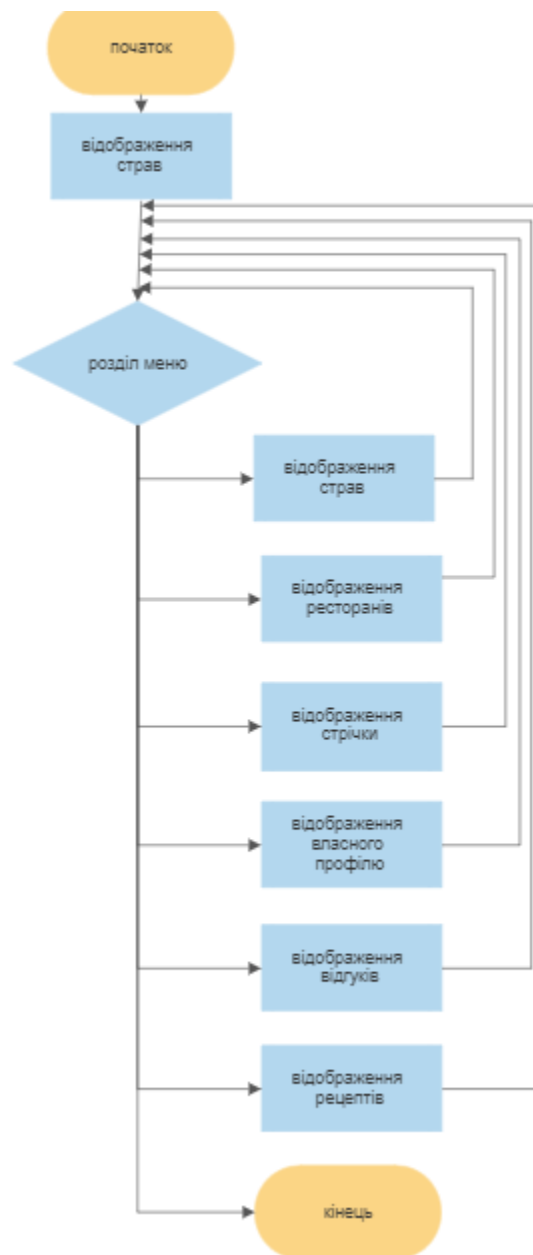


Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів програми

При реалізації програми за рахунок того, що було застосовано фреймворк Django, вдалось зменшити час реалізації через уникнення подвійної розробки, коли створюється окремо БД і засоби роботи з класами,

подібними до таблиць БД, в програмі. Цього дозволило уникнути створення класів моделей і подальша міграція БД на основі цих класів з автоматичним створенням таблиць БД.

Клас Meal має таку програмно визначену структуру:

- mealtype – зовнішній ключ з моделі MealType, який визначає тип страви;
- named – символічне поле, що визначає назву страви;
- about – текстове поле, що визначає детальний опис страви;
- photo – файлове поле, що визначає фотографію страви;
- rating – числове поле з обмеженою до 1 знаку після коми точністю, що представляє середній рейтинг страви за всіма відгуками людей, що її вже їли;
- restaurantrating – числове поле з обмеженою до 1 знаку після коми точністю, що представляє середній рейтинг страви за всіма відгуками людей, що її їли саме в ресторані;
- homerating – числове поле з обмеженою до 1 знаку після коми точністю, що представляє середній рейтинг страви за всіма відгуками людей, що її їли вдома;
- country – зовнішній ключ з моделі Countrym що представляє країну, з кухні якої походить страва.

Як відомо з описаної логіки в розділі 2, страва має декілька показників, якими вона характеризується, а значення яких розраховуються автоматично на основі відповідних відгуків рецензентів. Тому при створенні страви одразу потрібно внести записи про значення показників цієї страви. Для цього визначення функція `addmealmeasures`, яка за кожним існуючим показником створює відповідні об'єкти, які стосуються середніх оцінок для даної страви:

```
meas = Measure.objects.all()
pmeal = instance.id
for pmes in meas:
```

```

mm = MealMeasure.objects.create(meal = pmeal, measure =
pmes)
mm.save()

```

Клас Receipt має таку програмно визначену структуру:

- meal – зовнішній ключ з моделі Meal, який визначає страву, рецепт якої визначено;

- instruction – текстове поле, яке представляє інструкцію рецепту;
- about – текстове поле, яке представляє опис особливостей репету;
- photo – файлове поле, яке представляє фотографію рецепту;
- rating – числове поле з обмеженою до 1 знаку після коми точністю, що представляє середній рейтинг даного рецепту за всіма відгуками людей, що її вже їли.

Клас Product має таку програмно визначену структуру:

- productgroup – зовнішній ключ моделі ProductGroup, що представляє групу продуктів, до якої належить цей продукт;

- named – символічне поле, яке представляє назву продукту;
- about – текстове поле, яке представляє опис продукту;
- photo – файлове поле, яке представляє фотографія продукту.

Клас ProductGroup має таку програмно визначену структуру:

- named – символічне поле, що представляє назву групи продуктів;
- photo – файлове поле, що представляє фотографію групи продуктів.

Клас Review має таку програмно визначену структуру:

- created – поле дати і часу, що представляє дату публікації відгуку;
- about – тестове поле, що представляє враження рецензента у відгуку;
- restaurantmeal – зовнішній ключ з моделі RestaurantMeal, що представляє ресторан, де було приготовано страву, що спробував рецензент;

- reviewer – зовнішній ключ з моделі Reviewer, що представляє рецензента, який створив відгук;

- rating – числове поле з обмеженою до 1 знаку після коми точністю, що представляє загальний рейтинг, виставлений цьому вживанню страви;

– delivery – зовнішній ключ з моделі Delivery, що представляє службу доставки, яка привезла страву рецензенту.

Цей клас має достатньо особливий метод save, адже він автоматично розраховує нові значення рейтингів інших об'єктів на основі результатів, отриманих в рецензії.

Під час роботи методу спочатку виконується обчислення середнього рейтингу самої страви, до якої належала створена рецензія. При цьому обчислюється і загальний рейтинг, і рейтинг вживання страви в ресторані, і рейтинг домашнього споживання.

```

        reviews = Review.objects.filter(restaurantmeal__meal =
self.restaurantmeal.meal)
        rat = 0
        ratrest = 0
        rathome = 0
        for rev in reviews:
            rat = rat + rev.rating / reviews.count()
            if rev.restaurantmeal.restaurant.id == 0:
                ratrest = ratrest + rev.rating / reviews.count()
            else:
                rathome = rathome + rev.rating / reviews.count()
        meal.rating = rat
        meal.restaurantrating = ratrest
        meal.homerating = rathome

```

Після цього виконується обчислення рейтингу рецензента, який створив цей відгук, шляхом усереднення виставлених оцінок різним блюдам, а також кількості страв, яку він оцінив за весь час. Тобто якщо рецензент споживав одну й ту саму страву в різних ресторанах, то страва для другого показника враховується тільки один раз. Також обчислюється середній рейтинг виставлений всім стравам ресторану, а також середній рейтинг виставлений за цим рецептом загалом і середній рейтинг приготування цієї страви саме в цьому ресторані, що визначені у рецензії.

Клас Measure має тільки одне поле named, що є символьним полем, яке представляє показник оцінювання (його назву).

Клас Restaurant має таку програмно визначену структуру:

- named – символічне поле, яке представляє назву ресторану;
- place – зовнішній ключ з моделі Place, що представляє місце, де знаходиться ресторан;
- about – текстове поле, яке представляє опис ресторану;
- photo – файлове поле, яке представляє фотографію ресторану;
- point – символічне поле, яке представляє місце, де знаходиться ресторан;
- rating – числове поле з обмеженою до 1 знаку після коми точністю, що представляє загальний рейтинг страв, виставлених ресторану;
- phone – символічне поле, яке представляє номер телефону ресторану.

Клас RestaurantMeal має таку програмно визначену структуру:

- restaurant – зовнішній ключ моделі Restaurant, що представляє ресторан, де відбувається приготування;
- meal – зовнішній ключ моделі Meal, що представляє страву, яка виготовляється в ресторані;
- price – числове поле з обмеженою до 2 знаків після коми точністю, що представляє вартість цієї страви в цьому ресторані;
- recipe – зовнішній ключ з моделі Recipe, який визначає рецепт, за яким готується страву в цьому ресторані;
- rating – числове поле з обмеженою точністю до 1 знаку після коми, що представляє середній рейтинговий бал, виставлений за всіма відгуками споживанню цієї страви в цьому ресторані.

Клас ReviewMeasure має таку програмно визначену структуру:

- review – зовнішній ключ з моделі Review, що представляє відгук, до якого відноситься це оцінювання за показником;
- measure – зовнішній ключ з моделі Measure, що представляє показник, за яким виконується оцінювання страви;
- rating – цілочисельне поле, що представляє оцінку страви.

Метод `save` даного класу реалізує переобчислення середнього рейтингу за цим показником для даної страви серед всіх таких відгуків.

Клас `ReceiptProduct` має таку програмно визначену структуру:

- `receipt` – зовнішній ключ з моделі `Receipt`, що представляє рецепт, до якого належить цей продукт;
- `product` – зовнішній ключ з моделі `Product`, що представляє продукт, який входить в рецепт;
- `quant` – числове поле з обмеженою до 2 знаків після коми точністю, що представляє кількість продукту;
- `quantunit` – символічне поле зі значенням з обмеженого набору, що визначає одиницю виміру кількості продукту.

Клас `Reviewer` має таку програмно визначену структуру, будучи створеним наслідуванням стандартного класу `Django AbstractUser`:

- `reviews` – цілочисельне поле, що представляє кількість відгуків, створених рецензентом;
- `meals` – цілочисельне поле, що представляє кількість різних страв, оцінених рецензентом;
- `rating` – числове поле з обмеженою до 1 знаку після коми точністю, що представляє середній рейтинг всіх страв, які скуштував рецензент;
- `likes` – цілочисельне поле, що представляє кількість рецензентів, яких уподобав даний рецензент;
- `liked` – цілочисельне поле, що представляє кількість рецензентів, які уподобали даного рецензента.

Клас `Place` має таку програмно визначену структуру:

- `named` – символічне поле, що представляє назву місця;
- `placetype` – символічне поле з обмеженим набором значень, що представляє тип місця;
- `country` – зовнішній ключ з моделі `Country`, що представляє країну, де знаходиться це місце.

Клас `MealType` має таку програмно визначену структуру:

- `named` – символічне поле, що представляє назву типу страви;
- `photo` – файлове поле, що представляє фотографію цього типу страви.

Клас `Country` має таку програмно визначену структуру:

- `named` – символічне поле, що представляє назву країни;
- `cuisinenamed` – символічне поле, що представляє назву кухні країни.

Клас `Delivery` має таку програмно визначену структуру:

- `named` – символічне поле, що представляє назву служби доставки;
- `about` – текстове поле, що представляє опис служби доставки;
- `web` – текстове поле, що представляє веб адресу служби доставки.

Клас `Likedreviewer` має таку програмно визначену структуру:

- `reviewer` – зовнішній ключ з моделі `Reviewer`, що представляє рецензента, який виконав уподобання;
- `likereviewer` – зовнішній ключ з моделі `Reviewer`, що представляє рецензента, якого було уподобано;
- `created` – поле дати і часу, що представляє час уподобання.

Клас `MealMeasure` має таку програмно визначену структуру:

- `rating` – числове поле з обмеженою до 1 знаку після коми точністю, що представляє оцінку страви за даним показником;
- `meal` – зовнішній ключ з моделі `Meal`, що представляє страву, якій було виставлено оцінку;
- `measure` – зовнішній ключ з моделі `Measure`, що представляє показник, за яким було виконано оцінювання страви.

Уявлення `ViewParticularReview` виконує підготовку даних для деталізованого перегляду відгуку рецензента, включаючи до них оцінки за окремими показниками у вигляді окремого списку.

Уявлення `ViewReviewerReviews` виконує підготовку даних для відображення списку всіх відгуків певного рецензента. Для формування списку використовуються параметри GET-запиту, що включають номер самого рецензента в параметрі `author`, а також номер категорії відгуків, які

потрібно відобразити, в параметрі `revgroup`. Значення останнього параметра може бути наступним:

- значення 0 відповідає відображенню тільки відмінних відгуків (відгуків з оцінкою не нижче 4,5);
- значення 1 відповідає відображенню добрих і відмінних відгуків (з оцінкою не нижче 4);
- значення 2 відповідає відображенню тільки добрих відгуків (з оцінкою в інтервалі від 4 до 4,5);
- значення 3 відповідає відображенню задовільних, добрих і відмінних відгуків (з оцінкою не нижче 3,5);
- значення 4 відповідає відображенню тільки задовільних відгуків (з оцінкою в інтервалі від 3,5 до 4,0);
- значення 5 відповідає відображенню тільки поганих відгуків (з оцінкою не вище 3,5);
- усі інші значення дозволяють відобразити всі відгуки цього рецензента.

Уявлення `ViewMyReviews` виконує підготовку даних для відображення списку всіх відгуків авторизованого в даний момент рецензента. Для формування списку використовується параметр GET-запиту `revgroup`, що визначає номер категорії відгуків, які потрібно відобразити, абсолютно аналогічний описаному вище.

Уявлення `ViewMealReviews` виконує підготовку даних для відображення списку всіх відгуків за певною стравою. Для формування списку використовується параметр GET-запиту `meal`, який визначає номер страви, відгуки за якою потрібно відобразити, а також параметр `revgroup`, що визначає номер категорії відгуків, які потрібно відобразити, абсолютно аналогічний однойменному параметру, описаному вище.

Уявлення `ViewRestaurantMealReviews` виконує підготовку даних для відображення списку всіх відгуків за приготуванням певної страви у певному ресторані. Для формування списку використовується параметр GET-запиту

restmeal, який визначає номер приготування страви в ресторані, відгуки за яким потрібно відобразити, а також параметр revgroup, що визначає номер категорії відгуків, які потрібно відобразити, абсолютно аналогічний однойменному параметру, описаному вище.

Уявлення ViewReceiptReviews виконує підготовку даних для відображення списку всіх відгуків за певним рецептом. Для формування списку використовується параметр GET-запиту receipt, який визначає номер рецепту, відгуки за яким потрібно відобразити, а також параметр revgroup, що визначає номер категорії відгуків, які потрібно відобразити, абсолютно аналогічний однойменному параметру, описаному вище.

Уявлення ViewLikedReviews виконує підготовку даних для відображення списку всіх відгуків уподобаних рецензентів авторизованим користувачем. Для формування списку використовується параметр GET-запиту revgroup, що визначає номер категорії відгуків, які потрібно відобразити, абсолютно аналогічний однойменному параметру, описаному вище.

Уявлення ViewNewReview виконує роботу з даними стосовно створення нового відгуку. Внесення автора відгуку реалізується програмно за авторизованим рецензентом у поточний момент. Додатково використовуються два параметри GET-запиту: rest, який визначає номер ресторану, і meal, який визначає номер страви. За значеннями цих параметрів визначається номер приготування страви в ресторані, який і вноситься у відповідне поле моделі відгуку.

Уявлення ViewRenewReview виконує роботу з даними стосовно редагування даних відгуку. Додатково використовуються два параметри GET-запиту: rest, який визначає номер ресторану, і meal, який визначає номер страви. За значеннями цих параметрів визначається номер приготування страви в ресторані, який і вноситься у відповідне поле моделі відгуку.

Уявлення `ViewParticularMeal` виконує підготовку даних для деталізованого перегляду страви, включаючи до них рецепти приготування цієї страви.

Уявлення `ViewCountryMeals` виконує підготовку даних для відображення списку всіх страв за заданою країною. Для формування списку використовуються параметри GET-запиту `country`, що визначає номер країни, до якої належать страви, та `reggroup`, що визначає категорії відгуків, які потрібно відобразити, абсолютно аналогічний однойменному параметру, описаному вище.

Уявлення `ViewRatingMeals` виконує підготовку даних для відображення списку, що представляє рейтинг страв, до якого віднесено 100 перших за загальною оцінкою. Для формування списку використовується параметр GET-запиту `mealtype`, що визначає тип страви. Якщо тип страви не заданий, то подаються до рейтингу всі страви.

Уявлення `ViewRestRatingMeals` виконує підготовку даних для відображення списку, що представляє рейтинг страв, до якого віднесено 100 перших за оцінкою серед варіантів, коли споживання відбувалось у ресторані. Для формування списку використовується параметр GET-запиту `mealtype`, що визначає тип страви. Якщо тип страви не заданий, то подаються до рейтингу всі страви.

Уявлення `ViewHomeRatingMeals` виконує підготовку даних для відображення списку, що представляє рейтинг страв, до якого віднесено 100 перших за оцінкою серед варіантів, коли споживання відбувалось удома. Для формування списку використовується параметр GET-запиту `mealtype`, що визначає тип страви. Якщо тип страви не заданий, то подаються до рейтингу всі страви.

Уявлення `ViewRestaurantMealRating` виконує підготовку даних для відображення списку найкращих ресторанів, де готується певна страва. Для формування списку використовується параметр GET-запиту `meal`, що визначає страву.

Уявлення ViewNewRestaurantMeal виконує роботу з даними стосовно створення нового запису про приготування певної страви в певному ресторані.

Уявлення ViewParticularReviewer виконує підготовку даних для деталізованого перегляду даних рецензента.

Уявлення ViewNewReviewer виконує роботу з даними стосовно створення нового запису про рецензента.

Уявлення ViewRenewReviewer виконує роботу з даними стосовно оновлення існуючого запису про рецензента.

Уявлення ViewRenewRestaurantMeal виконує роботу з даними стосовно оновлення приготування страви в ресторані.

Уявлення ViewParticularRestaurant виконує підготовку даних для деталізованого перегляду даних ресторану.

Уявлення ViewRenewReviewMeasure виконує роботу з даними стосовно оновлення оцінки за показником у відгуку.

Уявлення ViewNewLikedreviewer виконує роботу з даними стосовно уподобання авторизованим рецензентом профілю іншого рецензента. Дані того рецензента, який виконав уподобання, вносяться автоматично на основі авторизованого профілю.

Уявлення ViewDeleteLikedreviewer виконує роботу з даними стосовно вилучення уподобання рецензента рецензентом.

Уявлення ViewParticularMealType виконує підготовку даних для деталізованого перегляду типу страви. При деталізованому перегляді до даних додається список страв з відповідним рейтингом, що належать до даного типу страви.

Уявлення ViewParticularCountry виконує підготовку даних для деталізованого перегляду країни. При деталізованому перегляді до даних додається список страв з кухні відповідної країни, відсортованих за рейтингом, та список ресторанів з цієї країни, відсортованих за рейтингом.

Уявлення `ViewParticularPlace` виконує підготовку даних для деталізованого перегляду місця. При деталізованому перегляді до даних додається список ресторанів з цього місця, відсортованих за рейтингом.

Уявлення `ViewParticularReceipt` виконує підготовку даних для деталізованого перегляду рецепту приготування певної страви, включаючи до них продукти, які необхідні для приготування страви.

Уявлення `ViewNewReceipt` виконує роботу з даними стосовно створення нового рецепту.

Уявлення `ViewRenewReceipt` виконує роботу з даними стосовно редагування даних рецепту.

Уявлення `ViewSearchReceipt` виконує підготовку даних для відображення списку результатів пошуку рецептів. Для визначення параметрів пошуку використовується параметр GET-запиту `meal`, який визначає номер страви, за якою потрібно відобразити рецепти, `mealtype`, який визначає тип страви, рецепти за яким потрібно відобразити, `vegetarian`, який визначає, чи потрібно відображати тільки вегетаріанські рецепти, чи ні, `products`, який визначає конкретні продукти, що мають входити в рецепт.

Уявлення `ViewRatingReceipt` виконує підготовку даних для відображення рейтингу рецептів. Для формування списку використовується параметр GET-запиту `meal`, який визначає номер страви, за якою потрібно відобразити рецепти, `mealtype`, який визначає тип страви, рецепти за яким потрібно відобразити, `vegetarian`, який визначає, чи потрібно відображати тільки вегетаріанські рецепти, чи ні. Відповідні значення формуються через елементи керування в користувацькому інтерфейсі.

Уявлення `ViewNewReceiptProduct` виконує роботу з даними стосовно внесення нового продукту до рецепту.

Уявлення `ViewRenewReceiptProduct` виконує роботу з даними стосовно редагування даних продукту в рецепті.

3.3 Висновки за розділом 3

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми, включаючи класи моделей і уявлень, принцип їх роботи і організації, та загальну логіку роботи користувача, який виступає фактично рецензентом, з програмним забезпеченням.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Приготування страв є одним з дуже популярних напрямків для сучасного блогерства і загалом частиною життя багатьох людей. Сучасна людина багато страв споживає приготовлених у різних закладах, хоча може їх і готувати. Усе це разом складається в певний персональний досвід. Відомими програмами є програми, які дозволяють оцінювати заклади і орієнтуватися таким чином у цьому світі. Однак цікавим є і процес вибору і пошуку нових страв, орієнтування між ними в різних закладах. Саме на це і направлена створена програма.

Загалом призначенням програми є створення інструменту для накопичення власного досвіду, що стосується споживання різних страв у різних закладах і вдома, та вибору нових страв або споживання цих страв у певних закладах, виконуючи ці рішення на основі досвіду інших користувачів.

4.2 Умови виконання програми

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 4 ядра і частотою 2,9 ГГц, оперативну пам'ять у 8 Гб, жорсткий диск у 1 Гб.

4.3 Експлуатація програми

Робота користувача з програмою є достатньо простою. Кожен користувач має абсолютно однаковий інтерфейс, проте не всі функції йому можуть бути доступними. Для того щоб стати повноцінним рецензентом в системі, йому потрібно авторизуватися. Проте у будь-якому випадку всі

пункти меню будуть доступні і без авторизації, але скористатися функціями, як наприклад, створення відгуку, не можна буде.

Основним і першим пунктом, з яким можна працювати в програмі, є звичайно страви, оскільки саме страви і є визначальною сутністю цієї предметної області. Цей розділ меню має цілий набір засобів, кожен з яких призводить до відкриття дещо іншої сторінки, що стосується страв. Додавати самі страви може тільки адміністратор через адмінпанель Django. А ось переглядати і групувати їх певним чином може кожен користувач.

Першою кнопкою, яка дозволяє керувати відображенням страв, є кнопка «Рейтинг», що дозволяє відобразити рейтинг найкращих страв (рис. 4.1).

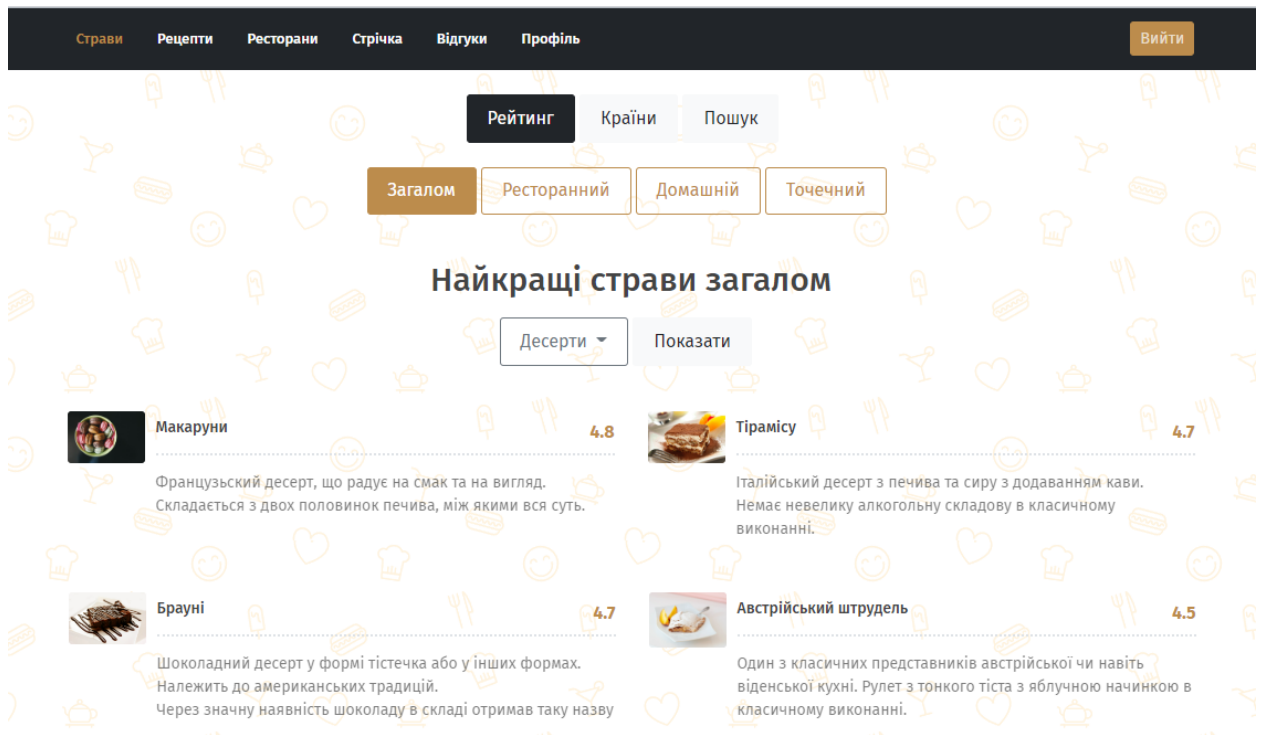


Рисунок 4.1 – Рейтинг страв

Рейтинг найкращих страв може визначатися одним з чотирьох варіантів, яким відповідають чотири кнопки в нижньому рядку:

– загалом, що дозволяє ознайомитись з рейтингом за загальною оцінкою, виставленою стравам, при цьому потрібно задати один з типів

страв, а якщо він не буде обраний, то відобразиться рейтинг без обмеження типу страв;

- ресторанний, що формує рейтинг страв за оцінками, які були виставлені тільки під час споживання в закладах харчування;

- домашній, що формує рейтинг страв за оцінками, які були виставлені після власного приготування і споживання страв вдома;

- точечний, що складає рейтинг зі всіх споживань страв у різних ресторанах, тобто в такому випадку одна страва у виконаннях різних ресторанів може потрапити до рейтингу декілька разів.

На будь-яку страву, відображену в рейтингу, можна натиснути, тоді відкриється окрема сторінка з повною інформацією про страву (рис. 4.2).

За кожною стравою відображається фотографія, короткий опис та середній загальний бал, виставлений за всіма відгуками цієї страви, а також звичайно відображається назва.

Окрім рейтингів можна натиснути на кнопку «Країни» для того, щоб ознайомитися фактично зі стравами різних кухонь світу. Тобто задається країна кухні і за нею формується список страв. При цьому на цей список можна вплинути, вибираючи певний інтервал середнього балу, як це було описано в розділі 3.

При натисканні на кнопку «Пошук» відкривається сторінка пошуку страв, де можна підібрати до вподоби одну зі страв.

Всі ці процеси по суті мають бути в подальшому направлені на сторінку перегляду страви. На цій сторінці за стравою відображається її назва, а над нею – тип страви, під нею – назва кухні. На будь-який з цих параметрів можна натиснути і перейти до перегляду відповідних страв. Ще нижче розташований опис страви, а ще нижче – середня оцінка, виставлена страві, переведена до п'ятибальної системи з відображенням у вигляді серцець.

Страви

Рецепти

Ресторани

Стрічка

Відгуки

Профіль

Вийти

Сніданки

Сирники

Кухня: українська

Одна з найвідоміших страв в українській кухні.

Виготовляється із домашнього сиру.

Подається зазвичай як десерт.

Оцінка

♡

♡

♡

♡

♡

Залишити власний відгук

Відгуки

Підсумок :

Загальна: 84%

Ресторани: 82%

Вдома: 92%

Показники :

Ситність: 92%

Оригінальність: 71%

Здоров'я: 68%

Смак: 86%

Рецепти :

Протеїнові сирники

Рецепт сирників без використання борошна - з протеїном. Сирники будуть дуже м'якими, рум'яними.

Класичні сирники

Рецепт сирників, історично досліджений в Україні.

Рисунок 4.2 – Ознайомлення зі стравою

На сторінці страви також відображається список рецептів цієї страви, натискаючи на будь-який з них, можна перейти до сторінки рецепту.

Окрема кнопка дозволяє перейти до списку відгуків за цією стравою. Вона називається «Відгуки». У свою чергу кнопка «Залишити власний відгук» дозволяє створити новий відгук за цією стравою. Потрібно буде також задати ресторан або вказати, що це було домашнє виконання страви.

Окремий розділ передбачений для перегляду і пошуку рецептів. Він в меню має назву «Рецепти». При цьому мова йде вже про самі рецепти, а відкриваючи відповідну сторінку, можна переглянути інструкцію з приготування страви, список продуктів, що знадобляться для неї.

Пункт «Ресторани» дозволяє шукати потрібний ресторан, визначаючи країну і місто, переглядати список страв, доступних у цьому ресторані, а також переходити до відгуків про конкретні приготування страв у ресторані. При цьому якщо необхідної страви в ресторані не визначено, то користувач, створюючи відгук, може задати інформацію про її приготування саме в цьому ресторані.

Окремий пункт меню передбачений для роботи з відгуками. Звідки за допомогою відповідних кнопок можна працювати з власними відгуками, з відгуками за всіма стравами загалом та за рецептами, тобто в даному випадку відображаються списки саме відгуків, а натискаючи на потрібний можна перейти на сторінку відгуку (рис. 4.3).

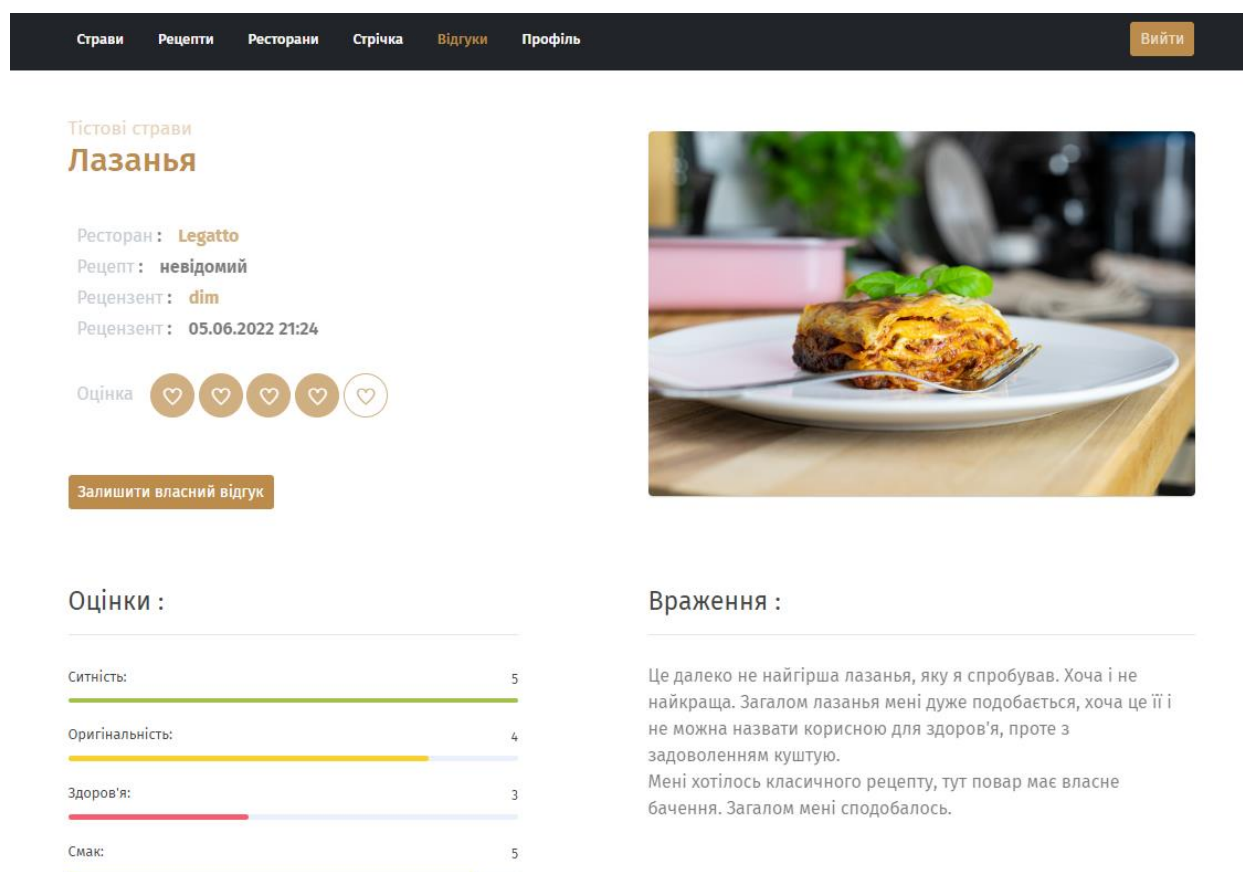


Рисунок 4.3 – Ознайомлення з відгуком

Сторінка відгуку містить тип страви, назву страви, ресторан, де було приготовлено страву під час рецензування. Якщо страва була домашнього

приготування, то без посилання відображається відповідний запис про це. Також відображається логін рецензента. Якщо натиснути на нього, то відкриється сторінка з інформацією про рецензента, звідки можна підписатися або відписатися від рецензента. Тут також можна перейти до створення власного відгуку за цією стравою в цьому ресторані.

На сторінці також відображається інформація про виставлені бали за всіма наявними показниками, а також безпосередньо сам текст відгуку. Ще нижче розташовано інформацію про використану службу доставки для тих випадків, коли вона використовувалась.

Пункт меню «Профіль» дозволяє продивитися дані про себе, а пункт «Стрічка» дозволяє переглядати останні відгуки рецензентів, на яких підписався авторизований користувач.

4.4 Тестування програми

Тестування програми дозволило перевірити роботу програми щодо виконання функціональних можливостей, які передбачались у процесі розробки. Тестування спочатку виконувалось під час реалізації цих можливостей і дозволяло виявляти проблеми і вирішувати їх. У подальшому, після того, як всі такі проблеми було усунуто, було проведено окремий етап тестування з перевіркою виконання всіх функцій, тому успішність реалізації була підтверджена за результатами тестування на персональному комп'ютері під керуванням операційної системи Windows 10.

4.5 Висновки за розділом 4

Описано процес експлуатації програми користувачем, надано порядок і засоби доступу до його основних функцій. Реалізовано тестування, що підтвердило успішність реалізації програми загалом.

ВИСНОВКИ

Метою роботи було розробити програмне забезпечення створення персональних відгуків про страви для пошуку варіантів урізноманітнення споживання їжі, що було досягнуто в результаті. Для цього було досягнуто наступні результати за відповідними, встановленими спочатку, завданнями.

За результатами аналізу проблеми виконано аналіз аналогів і їх порівняння з розробкою, визначено її особливості, що стосуються загалом ідеї концентрації саме на стравах.

За результатами проєктування представлено результати порівняння фреймворків для розробки вебзастосунків мовою Python, обрано фреймворк Django у цій якості, що визначило структуру програми на основі використання шаблону проєктування MVT, представлено функціональні вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв'язки між ними.

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми, включаючи класи моделей і уявлень, та загальну логіку роботи користувача, який виступає фактично рецензентом, з програмою. Під час виконання розробки та за результатами її виконання реалізовано тестування програми, що дозволило отримати результат, який відповідає запланованим для реалізації у технічному завданні вимогам.

Розроблена програма дозволила в результаті створити інструмент для накопичення власного досвіду, що стосується споживання різних страв у різних закладах і вдома, та вибору нових страв або споживання цих страв у певних закладах, виконуючи ці рішення на основі досвіду інших користувачів. Також програма може використовуватися для збереження власних вражень про страви, які користувач фактично як рецензент спробував або приготував.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Schutte, L. A Survey of Food Blogs and Videos [Text] : An Explorative : Senior Theses / L. Schutte. – Columbia : University of South Carolina, 2018. – 21 p.
2. Weisskopf, J.-P. Restaurant Reviews [Electronic resource] : Their Use and Their Impact on Business / J.-P. Weisskopf, P. Masset. – Access mode : <https://www.hospitalitynet.org/opinion/4089762.html>.
3. Restaurants, Dentints, Bars, Beauty Salons, Doctors – Yelp [Electronic resource]. – Access mode : <https://www.yelp.com/>.
4. Burpple [Electronic resource] : Discover Good Food & Best Restaurants in Singapore. – Access mode : <https://www.burpple.com/sg>.
5. Chan, A. Burpple vs Entertainer vs Chope – Which Offers the Best Dining Deals? [Electronic resource] / A. Chan. – Access mode : <https://www.singsaver.com.sg/blog/burpple-chope-entertainer-compare>.
6. Tasty – Food Videos And Recipes [Electronic resource]. – Access mode : <https://tasty.co/>.
7. Johnson, S. The 9 Best Recipe Apps To Channel Your Inner Chef [Electronic resource] / S. Johnson. – Access mode : <https://www.themanual.com/food-and-drink/best-recipe-apps/>.
8. Ramalho, L, Fluent Python [Text] : Clear, Concise, and Effective Programming / L. Ramalho. – Newton, Massachusetts : O'Reilly Media, 2022. – 1012 p.
9. Elder, J. Intro To Python Programming [Text] : Beginners Guide Series / J. Elder. – Paris : Codemy, 2022. – 157 p.
10. Sweigart, A. Automate the Boring Stuff with Python [Text] : Practical Programming for Total Beginners / A. Sweigart. – San Francisco : No Starch Press, 2019. – 592 p.
11. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.

12. Vincent, W. Django for Professionals [Text] : Production websites with Python & Django (Welcome to Django) / W. Vincent. – New York : WelcomeToCode, 2020. – 288 p.

ДОДАТОК А
Технічне завдання

Вступ

Приготування страв є одним з дуже популярних напрямків для сучасного блогерства і загалом частиною життя багатьох людей. Сучасна людина багато страв споживає приготовлених у різних закладах, хоча може їх і готувати. Усе це разом складається в певний персональний досвід. Відомими програмами є програми, які дозволяють оцінювати заклади і орієнтуватися таким чином у цьому світі. Однак цікавим є і процес вибору і пошуку нових страв, орієнтування між ними в різних закладах. Саме на це і має бути направлено створення програми в даній роботі.

А.1 Підстави для розробки

Розробка в межах дипломної кваліфікаційної роботи реалізується на підставі наказу № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка» за темою «Програмне забезпечення створення персональних відгуків про страви».

А.2 Призначення розробки

Призначенням розробки є створення інструменту для накопичення власного досвіду, що стосується споживання різних страв у різних закладах і вдома, та вибору нових страв або споживання цих страв у певних закладах, виконуючи ці рішення на основі досвіду інших користувачів.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, які повинна мати програма, охоплюють:

- створення відгуку про споживання страви у певному закладі;
- ознайомлення з відгуками заданого рецензента;
- перегляд власних відгуків;
- перегляд відгуків за стравою;
- перегляд відгуків про споживання певної страви в певному ресторані;
- перегляд відгуків за конкретним рецептом;
- перегляд відгуків уподобаних рецензентів;
- уточнення відгуку;
- ознайомлення зі стравою;
- перегляд страв за країною;
- перегляд рейтингу страв;
- перегляд рейтингу ресторанів щодо приготування певної страви;
- перегляд даних рецензента;
- авторизація рецензента, його реєстрація і оновлення даних;
- оновлення даних про приготування страви в ресторані;
- ознайомлення з рестораном;
- уподобання рецензента і зняття уподобання;
- ознайомлення з типом страв;
- ознайомлення з країною та місцем;
- створення рецепту;
- ознайомлення з рецептом.

A.3.2 Умови експлуатації

Експлуатація програми має забезпечуватися з одного боку звичайними користувачами, які працюють з даними, наповнюють їх, а з іншого боку – адміністраторами системи та БД.

А.3.3 Вимоги до складу та параметрів технічних засобів

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 4 ядра і частотою 2,9 ГГц, оперативну пам'ять у 8 Гб, жорсткий диск у 1 Тб.

А.4 Порядок контролю та приймання

Для контролю виконання роботи використовується календарний план, а приймання роботи виконується під час проведення процедури захисту роботи перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла mainviews.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.urls import reverse_lazy
from django.views.generic.edit import CreateView
from django.views.generic import ListView
from django.views.generic import UpdateView
from django.views.generic.detail import DetailView
from django.views.generic.edit import DeleteView
from .models import Review, ReviewMeasure, Likedreviewer,
Reviewer, Meal, Receipt, Restaurant, RestaurantMeal, Place, Country,
MealType

class ViewParticularReview (DetailView):
    template_name = 'review.html'
    model = Review

    def get_context_data(self, **kwargs):
        reviewing = super().get_context_data(**kwargs)
        measures = ReviewMeasure.objects.filter(review =
self.object.pk)
        measures = measures.order_by('measure__id')
        reviewing ['measures'] = measures
        return reviewing

class ViewReviewerReviews (ListView):
    model = Review
    template_name = 'revreviews.html'

    def get_queryset(self):
        author = self.request.GET.get('author')
        author = Reviewer.objects.filter(id = author)
        userreviews = Review.objects.filter(reviewer = author)
        revgroup = self.request.GET.get('revgroup')
        if revgroup == 0:
            userreviews = userreviews.filter(rating__gte =
4.5)
        elif revgroup == 1:
            userreviews = userreviews.filter(rating__gte =
4.0)
        elif revgroup == 2:
            userreviews = userreviews.filter(Q(rating__gte =
4.0) & Q(rating__lte = 4.5))
        elif revgroup == 3:

```

```

        userreviews = userreviews.filter(rating__gte =
3.5)
        elif revgroup == 4:
            userreviews = userreviews.filter(Q(rating__gte =
3.5) & Q(rating__lte = 4.0))
        elif revgroup == 5:
            userreviews = userreviews.filter(rating__lte =
3.5)

        userreviews = userreviews.order_by('-created')
        return userreviews

    def get_context_data(self, **kwargs):
        reviewing = super(ViewReviewerReviews,
self).get_context_data(**kwargs)
        author = self.request.GET.get('author')
        reviewing ['author'] = author
        revgroup = self.request.GET.get('revgroup')
        reviewing ['revgroup'] = revgroup
        return reviewing

class ViewMyReviews (ListView):
    model = Review
    template_name = 'myreviews.html'

    def get_queryset(self):
        author = self.request.user
        author = Reviewer.objects.filter(id = author)
        userreviews = Review.objects.filter(reviewer = author)
        revgroup = self.request.GET.get('revgroup')
        if revgroup == 0:
            userreviews = userreviews.filter(rating__gte =
4.5)

            elif revgroup == 1:
                userreviews = userreviews.filter(rating__gte =
4.0)

            elif revgroup == 2:
                userreviews = userreviews.filter(Q(rating__gte =
4.0) & Q(rating__lte = 4.5))
            elif revgroup == 3:
                userreviews = userreviews.filter(rating__gte =
3.5)

            elif revgroup == 4:
                userreviews = userreviews.filter(Q(rating__gte =
3.5) & Q(rating__lte = 4.0))
            elif revgroup == 5:

```

```

        userreviews = userreviews.filter(rating__lte =
3.5)
        userreviews = userreviews.order_by('-created')
        return userreviews

class ViewMealReviews (ListView):
    model = Review
    template_name = 'mealreviews.html'

    def get_queryset(self):
        meal = self.request.GET.get('meal')
        meal = Meal.objects.filter(id = meal)
        mealreviews =
Review.objects.filter(restaurantmeal__meal = meal)
        revgroup = self.request.GET.get('revgroup')
        if revgroup == 0:
            mealreviews = mealreviews.filter(rating__gte =
4.5)
            elif revgroup == 1:
                mealreviews = mealreviews.filter(rating__gte =
4.0)
            elif revgroup == 2:
                mealreviews = mealreviews.filter(Q(rating__gte =
4.0) & Q(rating__lte = 4.5))
            elif revgroup == 3:
                mealreviews = mealreviews.filter(rating__gte =
3.5)
            elif revgroup == 4:
                mealreviews = mealreviews.filter(Q(rating__gte =
3.5) & Q(rating__lte = 4.0))
            elif revgroup == 5:
                mealreviews = mealreviews.filter(rating__lte =
3.5)

        mealreviews = mealreviews.order_by('-created')
        return mealreviews

    def get_context_data(self, **kwargs):
        reviewing = super(ViewMealReviews,
self).get_context_data(**kwargs)
        meal = self.request.GET.get('meal')
        reviewing ['meal'] = meal
        revgroup = self.request.GET.get('revgroup')
        reviewing ['revgroup'] = revgroup
        return reviewing

```

```

class ViewRestaurantMealReviews (ListView):
    model = Review
    template_name = 'restmealreviews.html'

    def get_queryset(self):
        restmeal = self.request.GET.get('restmeal')
        restmeal = RestaurantMealReviews.objects.filter(id =
restmeal)

        retmealreviews = Review.objects.filter(restaurantmeal
= restmeal)

        revgroup = self.request.GET.get('revgroup')
        if revgroup == 0:
            retmealreviews =
retmealreviews.filter(rating__gte = 4.5)
        elif revgroup == 1:
            retmealreviews =
retmealreviews.filter(rating__gte = 4.0)
        elif revgroup == 2:
            retmealreviews =
retmealreviews.filter(Q(rating__gte = 4.0) & Q(rating__lte = 4.5))
        elif revgroup == 3:
            retmealreviews =
retmealreviews.filter(rating__gte = 3.5)
        elif revgroup == 4:
            retmealreviews =
retmealreviews.filter(Q(rating__gte = 3.5) & Q(rating__lte = 4.0))
        elif revgroup == 5:
            retmealreviews =
retmealreviews.filter(rating__lte = 3.5)
        retmealreviews = retmealreviews.order_by('-created')
        return retmealreviews

    def get_context_data(self, **kwargs):
        reviewing = super(ViewRestaurantMealReviews,
self).get_context_data(**kwargs)
        restmeal = self.request.GET.get('restmeal')
        reviewing ['restmeal'] = restmeal
        revgroup = self.request.GET.get('revgroup')
        reviewing ['revgroup'] = revgroup
        return reviewing

class ViewReceiptReviews (ListView):
    model = Review
    template_name = 'receiptreviews.html'

```

```

def get_queryset(self):
    receipt = self.request.GET.get('receipt')
    receipt = Receipt.objects.filter(id = receipt)
    recreviews =
Review.objects.filter(restaurantmeal__receipt = receipt)
    revgroup = self.request.GET.get('revgroup')
    if revgroup == 0:
        recreviews = recreviews.filter(rating__gte = 4.5)
    elif revgroup == 1:
        recreviews = recreviews.filter(rating__gte = 4.0)
    elif revgroup == 2:
        recreviews = recreviews.filter(Q(rating__gte =
4.0) & Q(rating__lte = 4.5))
    elif revgroup == 3:
        recreviews = recreviews.filter(rating__gte = 3.5)
    elif revgroup == 4:
        recreviews = recreviews.filter(Q(rating__gte =
3.5) & Q(rating__lte = 4.0))
    elif revgroup == 5:
        recreviews = recreviews.filter(rating__lte = 3.5)
    recreviews = recreviews.order_by('-created')
    return recreviews

def get_context_data(self, **kwargs):
    reviewing = super(ViewReceiptReviews,
self).get_context_data(**kwargs)
    receipt = self.request.GET.get('receipt')
    reviewing ['receipt'] = receipt
    revgroup = self.request.GET.get('revgroup')
    reviewing ['revgroup'] = revgroup
    return reviewing

class ViewLikedReviews (ListView):
    model = Review
    template_name = 'likedreviews.html'

def get_queryset(self):
    sourceuser = self.request.user
    likes = Likedreviewer.objects.filter(likereviewer =
sourceuser)

    users = []
    for l in likes:
        if not (l.likereviewer in users):
            users.append(l.likereviewer)

```

```

        userreviews = Review.objects.filter(reviewer__in =
users)
        revgroup = self.request.GET.get('revgroup')
        if revgroup == 0:
            userreviews = userreviews.filter(rating__gte =
4.5)
        elif revgroup == 1:
            userreviews = userreviews.filter(rating__gte =
4.0)
        elif revgroup == 2:
            userreviews = userreviews.filter(Q(rating__gte =
4.0) & Q(rating__lte = 4.5))
        elif revgroup == 3:
            userreviews = userreviews.filter(rating__gte =
3.5)
        elif revgroup == 4:
            userreviews = userreviews.filter(Q(rating__gte =
3.5) & Q(rating__lte = 4.0))
        elif revgroup == 5:
            userreviews = userreviews.filter(rating__lte =
3.5)

        userreviews = userreviews.order_by('-created')
        return userreviews

    def get_context_data(self, **kwargs):
        reviewing = super(ViewLikedReviews,
self).get_context_data(**kwargs)
        revgroup = self.request.GET.get('revgroup')
        reviewing['revgroup'] = revgroup
        return reviewing

class ViewNewReview (LoginRequiredMixin, CreateView):
    model = Review
    fields = ['rating', 'about', 'delivery']
    template_name = 'newreview.html'

    def get_success_url(self):
        return reverse_lazy('newreview', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        restaurant = get_object_or_404(Restaurant,
id=self.kwargs['rest'])
        kwargs['rest'] = restaurant
        meal = get_object_or_404(Meal, id=self.kwargs['meal'])

```

```

        kwargs['meal'] = meal
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.reviewer = self.request.user
        rest = get_object_or_404(Restaurant,
id=self.kwargs['rest'])
        m = get_object_or_404(Meal, id=self.kwargs['meal'])
        restmeal = RestaurantMeal.objects.get(restaurant =
rest, meal = m)
        self.restaurantmeal = restmeal
        return super().form_valid(form)

class ViewRenewReview (LoginRequiredMixin, UpdateView):
    model = Review
    fields = ['rating', 'about', 'delivery']
    template_name = 'renewreview.html'

    def get_success_url(self):
        return reverse_lazy('renewreview', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        restaurant = get_object_or_404(Restaurant,
id=self.kwargs['rest'])
        kwargs['rest'] = restaurant
        meal = get_object_or_404(Meal, id=self.kwargs['meal'])
        kwargs['meal'] = meal
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        rest = get_object_or_404(Restaurant,
id=self.kwargs['rest'])
        m = get_object_or_404(Meal, id=self.kwargs['meal'])
        restmeal = RestaurantMeal.objects.get(restaurant =
rest, meal = m)
        self.restaurantmeal = restmeal
        return super().form_valid(form)

class ViewParticularMeal (DetailView):
    template_name = 'meal.html'
    model = Meal

    def get_context_data(self, **kwargs):

```

```

        mealing = super().get_context_data(**kwargs)
        receipts = Receipt.objects.filter(meal =
self.object.pk)
        receipts = receipts.order_by('id')
        mealing ['receipts'] = receipts
        return mealing

class ViewCountryMeals (ListView):
    model = Meal
    template_name = 'countrymeals.html'

    def get_queryset(self):
        countr = self.request.GET.get('country')
        countr = Country.objects.filter(id = countr)
        meals = Meal.objects.filter(country = countr)
        revgroup = self.request.GET.get('revgroup')
        if revgroup == 0:
            meals = meals.filter(rating__gte = 4.5)
        elif revgroup == 1:
            meals = meals.filter(rating__gte = 4.0)
        elif revgroup == 2:
            meals = meals.filter(Q(rating__gte = 4.0) &
Q(rating__lte = 4.5))
        elif revgroup == 3:
            meals = meals.filter(rating__gte = 3.5)
        elif revgroup == 4:
            meals = meals.filter(Q(rating__gte = 3.5) &
Q(rating__lte = 4.0))
        elif revgroup == 5:
            meals = meals.filter(rating__lte = 3.5)
        meals = meals.order_by('-rating')
        return meals

    def get_context_data(self, **kwargs):
        mealing = super(ViewCountryMeals,
self).get_context_data(**kwargs)
        countr = self.request.GET.get('country')
        mealing ['country'] = countr
        revgroup = self.request.GET.get('revgroup')
        mealing ['revgroup'] = revgroup
        return mealing

class ViewSearchMeals (ListView):
    model = Meal
    template_name = 'searchmeals.html'

```

```

def get_queryset(self):
    mealt = self.request.GET.get('mealtype')
    countr = Mealtype.objects.filter(id = mealt)
    meals = Meal.objects.filter(country = countr)
    mealname = self.request.GET.get('mealname')
    meals = meals.filter(named__contains = mealname)
    revgroup = self.request.GET.get('revgroup')
    if revgroup == 0:
        meals = meals.filter(rating__gte = 4.5)
    elif revgroup == 1:
        meals = meals.filter(rating__gte = 4.0)
    elif revgroup == 2:
        meals = meals.filter(Q(rating__gte = 4.0) &
Q(rating__lte = 4.5))
    elif revgroup == 3:
        meals = meals.filter(rating__gte = 3.5)
    elif revgroup == 4:
        meals = meals.filter(Q(rating__gte = 3.5) &
Q(rating__lte = 4.0))
    elif revgroup == 5:
        meals = meals.filter(rating__lte = 3.5)
    meals = meals.order_by('-rating')
    return meals

def get_context_data(self, **kwargs):
    mealing = super(ViewSearchMeals,
self).get_context_data(**kwargs)
    mealt = self.request.GET.get('mealtype')
    mealing ['mealtype'] = mealt
    mealname = self.request.GET.get('mealname')
    mealing ['mealname'] = mealname
    revgroup = self.request.GET.get('revgroup')
    mealing ['revgroup'] = revgroup
    return mealing

class ViewRatingMeals (ListView):
    model = Meal
    template_name = 'ratingmeals.html'

def get_queryset(self):
    mealt = self.request.GET.get('mealtype')
    meals = Meal.objects.all()
    if mealt:
        meals = Meal.objects.filter(mealtype = mealt)

```

```

        meals = meals.order_by('-rating')[:100]
        return meals

    def get_context_data(self, **kwargs):
        mealing = super(ViewRatingMeals,
self).get_context_data(**kwargs)
        mealtype = self.request.GET.get('mealtype')
        mealing['mealtype'] = mealtype
        return mealing

class ViewRestRatingMeals (ListView):
    model = Meal
    template_name = 'restratingmeals.html'

    def get_queryset(self):
        meal = self.request.GET.get('mealtype')
        meals = Meal.objects.all()
        if meal:
            meals = Meal.objects.filter(mealtype = meal)
        meals = meals.order_by('-restauranrating')
[:100]

        return meals

    def get_context_data(self, **kwargs):
        mealing = super(ViewRestRatingMeals,
self).get_context_data(**kwargs)
        mealtype = self.request.GET.get('mealtype')
        mealing['mealtype'] = mealtype
        return mealing

class ViewHomeRatingMeals (ListView):
    model = Meal
    template_name = 'homeratingmeals.html'

    def get_queryset(self):
        meal = self.request.GET.get('mealtype')
        meals = Meal.objects.all()
        if meal:
            meals = Meal.objects.filter(mealtype = meal)
        meals = meals.order_by('-homerating')[:100]
        return meals

    def get_context_data(self, **kwargs):
        mealing = super(ViewHomeRatingMeals,
self).get_context_data(**kwargs)

```

```

        mealtype = self.request.GET.get('mealtype')
        mealing ['mealtype'] = mealtype
        return mealing

class ViewRestaurantMealRating (ListView):
    model = RestaurantMeal
    template_name = 'restmealrating.html'

    def get_queryset(self):
        meal = self.request.GET.get('meal')
        rm = RestaurantMeal.objects.filter(meal = meal)
        rm = rm.order_by('-rating')
        return rm

    def get_context_data(self, **kwargs):
        mealing = super(ViewRestaurantMealRating,
self).get_context_data(**kwargs)
        meal = self.request.GET.get('meal')
        mealing ['meal'] = meal
        return mealing

class ViewNewRestaurantMeal (LoginRequiredMixin, CreateView):
    model = RestaurantMeal
    fields = ['restaurant', 'meal', 'receipt', 'price']
    template_name = 'newrestmeal.html'

    def get_success_url(self):
        return reverse_lazy('newrestmeal', kwargs={'pk':
self.object.pk})

class ViewParticularReviewer (DetailView):
    template_name = 'reviewer.html'
    model = Reviewer

class ViewNewReviewer (LoginRequiredMixin, CreateView):
    model = Reviewer
    fields = ['username', 'password']
    template_name = 'newreviewer.html'

    def get_success_url(self):
        return reverse_lazy('newreviewer', kwargs={'pk':
self.object.pk})

class ViewRenewReviewer (LoginRequiredMixin, UpdateView):
    model = Reviewer

```

```

        fields = ['username', 'password']
        template_name = 'renewreviewer.html'

        def get_success_url(self):
            return reverse_lazy('renewreviewer', kwargs={'pk':
self.object.pk})

class ViewRenewRestaurantMeal (LoginRequiredMixin, UpdateView):
    model = RestaurantMeal
    fields = ['restaurant', 'meal', 'receipt', 'price']
    template_name = 'renewrestmeal.html'

    def get_success_url(self):
        return reverse_lazy('renewrestmeal', kwargs={'pk':
self.object.pk})

class ViewParticularRestaurant (DetailView):
    template_name = 'restaurant.html'
    model = Restaurant

    def get_context_data(self, **kwargs):
        mealing = super().get_context_data(**kwargs)
        meals = RestaurantMeal.objects.filter(restaurant =
self.object.pk)
        meals = meals.order_by('id')
        mealing ['meals'] = meals
        return mealing

class ViewRenewReviewMeasure (LoginRequiredMixin, UpdateView):
    model = ReviewMeasure
    fields = ['rating']
    template_name = 'renewreviewmeasure.html'

    def get_success_url(self):
        return reverse_lazy('renewreviewmeasure',
kwargs={'pk': self.object.pk})

class ViewNewLikedreviewer (LoginRequiredMixin, CreateView):
    model = Likedreviewer
    fields = ['likereviewer']
    template_name = 'newlikedreviewer.html'

    def get_success_url(self):
        return reverse_lazy('newlikedreviewer', kwargs={'pk':
self.object.pk})

```

```

    def form_valid(self, form):
        self.reviewer = self.request.user
        return super().form_valid(form)

class ViewDeleteLikedreviewer (DeleteView):
    model = Likedreviewer

class ViewParticularMealType (DetailView):
    template_name = 'mealtype.html'
    model = MealType

    def get_context_data(self, **kwargs):
        mealing = super().get_context_data(**kwargs)
        meals = Meal.objects.filter(mealtype = self.object.pk)
        meals = meals.order_by('-rating')
        mealing ['meals'] = meals
        return mealing

class ViewParticularCountry (DetailView):
    template_name = 'country.html'
    model = Country

    def get_context_data(self, **kwargs):
        ctr = super().get_context_data(**kwargs)
        meals = Meal.objects.filter(country = self.object.pk)
        meals = receipts.order_by('-rating')
        ctr ['meals'] = meals
        rests = Restaurant.objects.filter(place__country =
self.object.pk)
        rests = rests.order_by('-rating')
        ctr ['rests'] = rests
        return ctr

class ViewParticularPlace (DetailView):
    template_name = 'place.html'
    model = Place

    def get_context_data(self, **kwargs):
        ctr = super().get_context_data(**kwargs)
        rests = Restaurant.objects.filter(place =
self.object.pk)
        rests = rests.order_by('-rating')
        ctr ['rests'] = rests
        return ctr

```

Б.2 Текст файла models.py

```

from django.db import models
from django.utils import timezone
from django.urls import reverse
from django.contrib.auth.models import AbstractUser
import datetime

PLACETYPES = [
    ('C', 'місто'),
    ('V', 'селище'),
    ('R', 'поза населеним пунктом'),
    ('T', 'селище міського типу')
]

QUNITS = [
    ('G', 'г'),
    ('K', 'кг'),
    ('L', 'л'),
    ('P', 'фунт')
]

class Meal (models.Model):
    mealtype = models.ForeignKey(MealType,
on_delete=models.CASCADE)
    named = models.CharField(max_length=30)
    about = models.TextField(blank=True)
    photo = models.FileField(upload_to='img/meals/', blank=True)
    rating = models.DecimalField(max_digits = 2, decimal_places
= 1, default = 0)
    restauranrating = models.DecimalField(max_digits = 2,
decimal_places = 1, default = 0)
    homerating = models.DecimalField(max_digits = 2,
decimal_places = 1, default = 0)
    country = models.ForeignKey(Country,
on_delete=models.CASCADE)

    def __str__(self):
        return self.named

    def get_absolute_url(self):
        return reverse('mealview', kwargs={'pk': self.pk})

```

```

def addmealmeasures(instance, *args, **kwargs):
    meas = Measure.objects.all()
    pmeal = instance.id
    for pmes in meas:
        mm = MealMeasure.objects.create(meal = pmeal, measure
= pmes)
        mm.save()

post_init.connect(addmealmeasures, Meal)

class Receipt (models.Model):
    meal = models.ForeignKey(Meal, on_delete=models.CASCADE)
    named = models.CharField(max_length=100)
    instruction = models.TextField(blank=True)
    about = models.TextField(blank=True)
    photo = models.FileField(upload_to='img/reseipts/',
blank=True)
    rating = models.DecimalField(max_digits = 2, decimal_places
= 1, default = 0)
    vegetarian = models.BooleanField()

    def get_absolute_url(self):
        return reverse('receiptview', kwargs={'pk': self.pk})

class Product (models.Model):
    productgroup = models.ForeignKey(ProductGroup,
on_delete=models.CASCADE)
    named = models.CharField(max_length=30)
    about = models.TextField(blank=True)
    photo = models.FileField(upload_to='img/products/',
blank=True)

    def __str__(self):
        return self.named

    def get_absolute_url(self):
        return reverse('productview', kwargs={'pk': self.pk})

class ProductGroup (models.Model):
    named = models.CharField(max_length=30)
    photo = models.FileField(upload_to='img/products/',
blank=True)

    def __str__(self):
        return self.named

```

```

    def get_absolute_url(self):
        return reverse('productgroupview', kwargs={'pk':
self.pk})

    class Review (models.Model):
        reviewer = models.ForeignKey(Reviewer,
on_delete=models.CASCADE)
        created =
models.DateTimeField(default=datetime.datetime.now)
        rating = models.SmallIntegerField()
        about = models.TextField()
        restaurantmeal = models.ForeignKey(RestaurantMeal,
on_delete=models.CASCADE)
        delivery = models.ForeignKey(Delivery,
on_delete=models.CASCADE)
        photo = models.FileField(upload_to='img/reviews/',
blank=True)

    def get_absolute_url(self):
        return reverse('reviewview', kwargs={'pk': self.pk})

    def save(self, *args, **kwargs):
        meal = self.restaurantmeal.meal
        reviews = Review.objects.filter(restaurantmeal__meal =
self.restaurantmeal.meal)
        rat = 0
        ratrest = 0
        rathome = 0
        for rev in reviews:
            rat = rat + rev.rating / reviews.count()
            if rev.restaurantmeal.restaurant.id == 0:
                ratrest = ratrest + rev.rating /
reviews.count()
            else:
                rathome = rathome + rev.rating /
reviews.count()
        meal.rating = rat
        meal.restaurantrating = ratrest
        meal.homerating = rathome
        meal.save()
        i = self.user
        reviews = Review.objects.filter(reviewer = i)
        rat = 0
        meals = []
        for rev in reviews:

```

```

        rat = rat + rev.rating / reviews.count()
        if not (rev.restaurantmeal.meal in meals):
            meals.append(rev.restaurantmeal.meal)
    i.rating = rat
    i.reviews = reviews.count()
    i.meals = meals.count()
    i.save()
    reviews =
Review.objects.filter(restaurantmeal__restaurant =
self.restaurantmeal.restaurant)
    rat = 0
    for rev in reviews:
        rat = rat + rev.rating / reviews.count()
    rest = self.restaurantmeal.restaurant
    rest.rating = rat
    rest.save()
    reviews =
Review.objects.filter(restaurantmeal__receipt =
self.restaurantmeal.receipt)
    rat = 0
    for rev in reviews:
        rat = rat + rev.rating / reviews.count()
    rec = self.restaurantmeal.restaurant.receipt
    rec.rating = rat
    rec.save()
    rm = RestaurantMeal.objects.get(id =
self.restaurantmeal)
    reviews = Review.objects.filter(restaurantmeal = rm)
    rat = 0
    for rev in reviews:
        rat = rat + rev.rating / reviews.count()
    rm.rating = rat
    rec.save()
    super(Review, self).save(*args, **kwargs)

def addreviewmeasures(instance, *args, **kwargs):
    meas = Measure.objects.all()
    rev = instance.id
    for m in meas:
        rm = ReviewMeasure.objects.create(review = rev,
measure = m)
        rm.save()

post_init.connect(addreviewmeasures, Review)

```

```

class Measure (models.Model):
    named = models.CharField(max_length=30)

class Restaurant (models.Model):
    named = models.CharField(max_length=30)
    place = models.ForeignKey(Place, on_delete=models.CASCADE)
    about = models.TextField(blank=True)
    photo = models.FileField(upload_to='img/restaurants/',
blank=True)
    point = models.CharField(max_length=200)
    rating = models.DecimalField(max_digits = 2, decimal_places
= 1, default = 0)
    named = models.CharField(max_length=13)

    def __str__(self):
        return self.named

    def get_absolute_url(self):
        return reverse('restaurantview', kwargs={'pk':
self.pk})

class RestaurantMeal (models.Model):
    price = models.DecimalField(max_digits = 10, decimal_places
= 2, default = 0)
    restaurant = models.ForeignKey(Restaurant,
on_delete=models.CASCADE)
    meal = models.ForeignKey(Meal, on_delete=models.CASCADE)
    receipt = models.ForeignKey(Receipt,
on_delete=models.CASCADE)
    rating = models.DecimalField(max_digits = 2, decimal_places
= 1, default = 0)

    def __str__(self):
        return self.price

class ReviewMeasure (models.Model):
    rating = models.SmallIntegerField(default = 0)
    review = models.ForeignKey(Review, on_delete=models.CASCADE)
    measure = models.ForeignKey(Measure,
on_delete=models.CASCADE)

    def __str__(self):
        return self.rating

```

```

def save(self, *args, **kwargs):
    meal = self.review.restaurantmeal.meal
    reviewmeasures = ReviewMeasure.objects.filter(measure
= self.measure)
    reviewmeasures = reviewmeasures
filter(review__restaurantmeal__meal = self.review.restaurantmeal.meal)
    rat = 0
    for rev in reviewmeasures:
        rat = rat + rev.rating / reviewmeasures.count()
    mealmeas = MealMeasure.objects.filter(meal =
self.review.restaurantmeal.meal, measure = self.measure)
    mealmeas.rating = rat
    mealmeas.save()
    super(ReviewMeasure, self).save(*args, **kwargs)

```

```

class MealMeasure (models.Model):
    rating = models.DecimalField(max_digits = 2, decimal_places
= 1, default = 0)
    meal = models.ForeignKey(Meal, on_delete=models.CASCADE)
    measure = models.ForeignKey(Measure,
on_delete=models.CASCADE)

```

```

def __str__(self):
    return self.rating

```

```

class ReceiptProduct (models.Model):
    receipt = models.ForeignKey(Receipt,
on_delete=models.CASCADE)
    product = models.ForeignKey(Product,
on_delete=models.CASCADE)
    quant = models.DecimalField(max_digits = 10, decimal_places
= 2, default = 0)
    quantunit = models.CharField(max_length=1, choices=QUNITS,
default='G')

```

```

class Reviewer (AbstractUser):
    reviews = models.IntegerField()
    meals = models.IntegerField()
    rating = models.DecimalField(max_digits = 2, decimal_places
= 1, default = 0)
    likes = models.IntegerField()
    liked = models.IntegerField()

def get_absolute_url(self):
    return reverse('reviewerview', kwargs={'pk': self.pk})

```

```

class Place (models.Model):
    named = models.CharField(max_length=50)
    placetype = models.CharField(max_length=1,
choices=PLACETYPES, default='C')
    country = models.ForeignKey(Country,
on_delete=models.CASCADE)

    def __str__(self):
        return self.named

class MealType (models.Model):
    named = models.CharField(max_length=30)
    photo = models.FileField(upload_to='img/mealtypes/',
blank=True)

    def __str__(self):
        return self.named

class Country (models.Model):
    named = models.CharField(max_length=30)
    cuisinenamed = models.CharField(max_length=50)

    def __str__(self):
        return self.named

class Delivery (models.Model):
    named = models.CharField(max_length=30)
    about = models.TextField()
    web = models.TextField()

    def __str__(self):
        return self.named

    def get_absolute_url(self):
        return reverse('deliveryview', kwargs={'pk': self.pk})

class Likedreviewer (models.Model):
    reviewer = models.ForeignKey(Reviewer,
on_delete=models.CASCADE)
    likereviewer = models.ForeignKey(Reviewer,
on_delete=models.CASCADE)
    created =
models.DateTimeField(default=datetime.datetime.now)

```

```

def save(self, *args, **kwargs):
    self.reviewer = self.request.user
    i = self.request.user
    he = self.likereviewer
    likes = Likedreviewer.objects.filter(reviewer = i)
    i.likes = likes.count()
    i.save()
    likes = Likedreviewer.objects.filter(likereviewer =
he)

    he.likes = likes.count()
    he.save()
    super(Likedreviewer, self).save(*args, **kwargs)

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

Дипломна кваліфікаційна робота

**Програмне забезпечення
створення персональних
відгуків про страви**

Виконав:

студент групи КНТ-119сп

Д. В. Бережний

Керівник:

к.т.н, доцент

С. Ю. Скрупський

2022

Рисунок В.1 – Слайд № 1

2

Об'єкт дослідження – процес приготування і споживання страв.

Предмет дослідження – програмне забезпечення створення персональних відгуків про страви.

Мета дослідження – розробити програмне забезпечення створення персональних відгуків про страви для пошуку варіантів урізноманітнення споживання їжі.

Рисунок В.2 – Слайд № 2

Постановка завдань

3

- дослідження проблеми створення персональних відгуків про страви;
- визначення засобів розробки;
- визначення вимог до програми;
- проектування програми створення персональних відгуків про страви;
- програмна реалізація.

Рисунок В.3 – Слайд № 3

Порівняння аналогів з виконуваною розробкою

4

Показник	Yelp	Burpple	Tasty	Розробка
Пошук за стравами	Тільки якщо страви вказані в тегах закладів	Ні	Так	Так
Фотографії страв	Так	Так	Так	Так
Накопичення відгуків за стравами	Ні	Ні	Ні	Так
Накопичення відгуків за рецептами	Ні	Ні	Так	Так
Накопичення відгуків про ресторани	Так	Так	Ні	Ні
Накопичення відгуків за конкретними стравами в конкретному закладі	Ні	Ні	Ні	Так
Рецепти	Ні	Ні	Так	Так

Рисунок В.4 – Слайд № 4

Діаграма варіантів використання

5

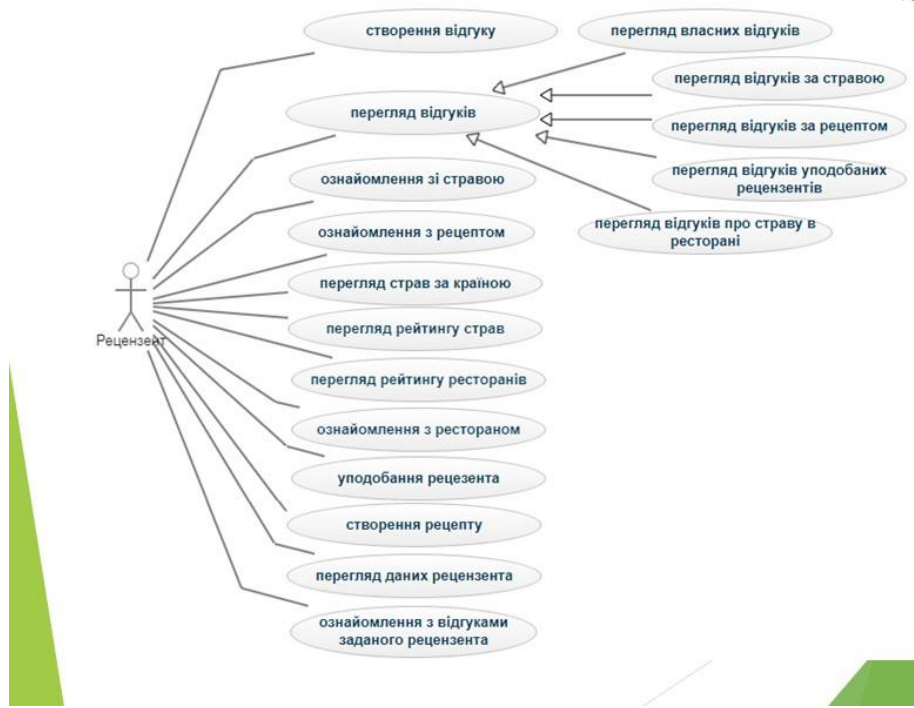


Рисунок В.5 – Слайд № 5

Вибір вебфреймворку розробки мовою Python

6

Показник	Django	Flask
Створення структури бази даних на основі коду	+	-
Швидкий вибір даних з бази даних	+	-
Автоматичне створення засобів адміністрування	+	-
Формування динамічних вебсторінок	+	-
Використовувана мова програмування	Python	Python

Рисунок В.6 – Слайд № 6

Схема структури бази даних

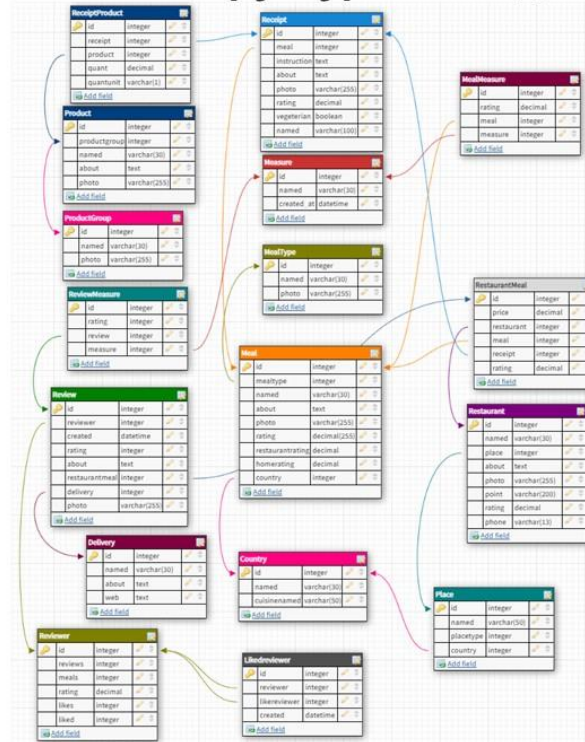


Рисунок В.7 – Слайд № 7

Схема функціонування програми

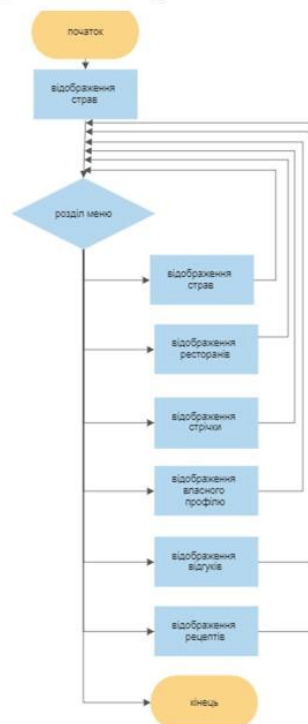


Рисунок В.8 – Слайд № 8

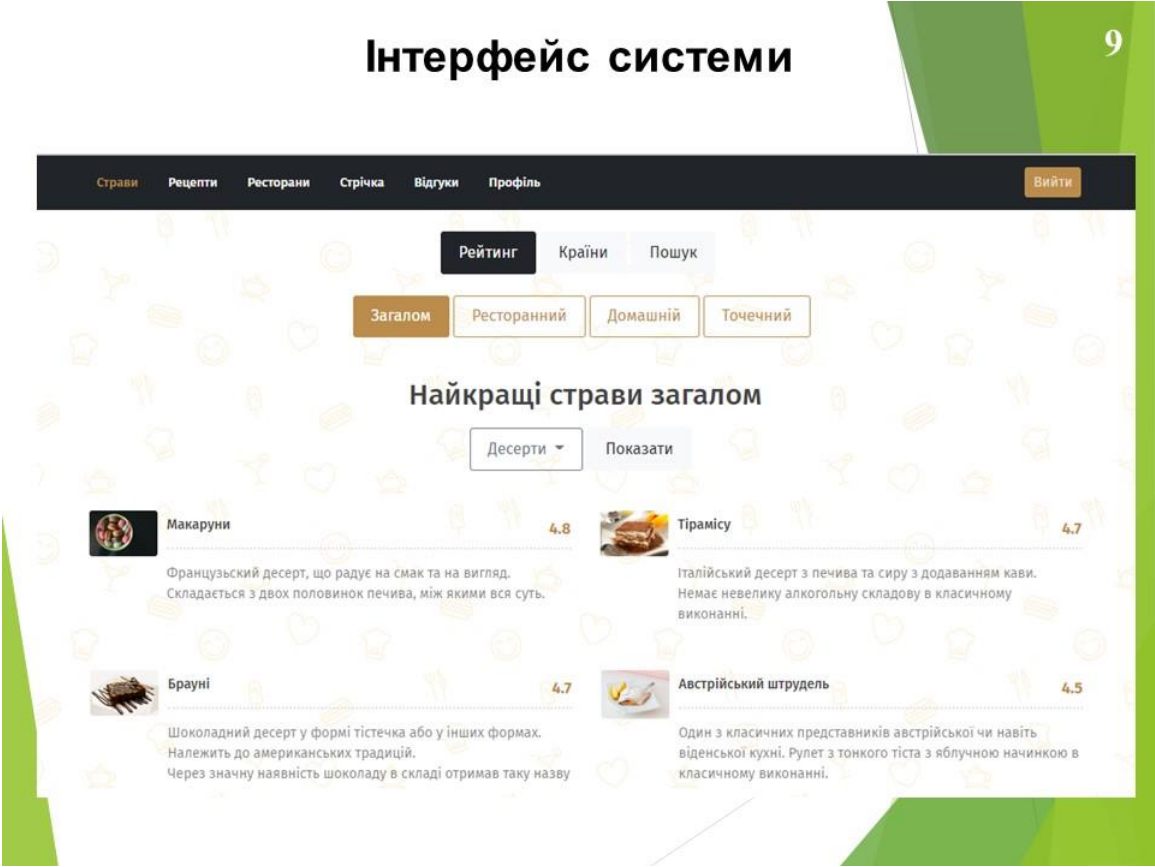


Рисунок В.9 – Слайд № 9

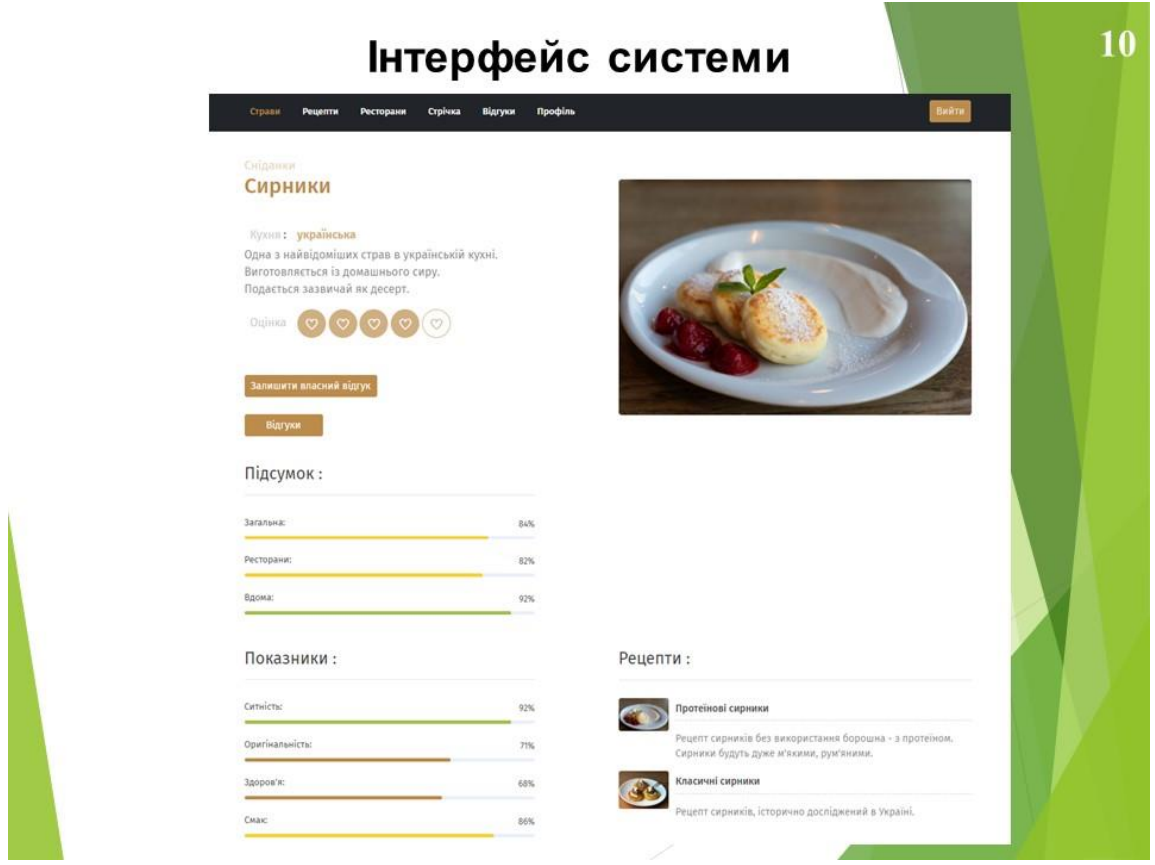


Рисунок В.10 – Слайд № 10

Інтерфейс системи

11

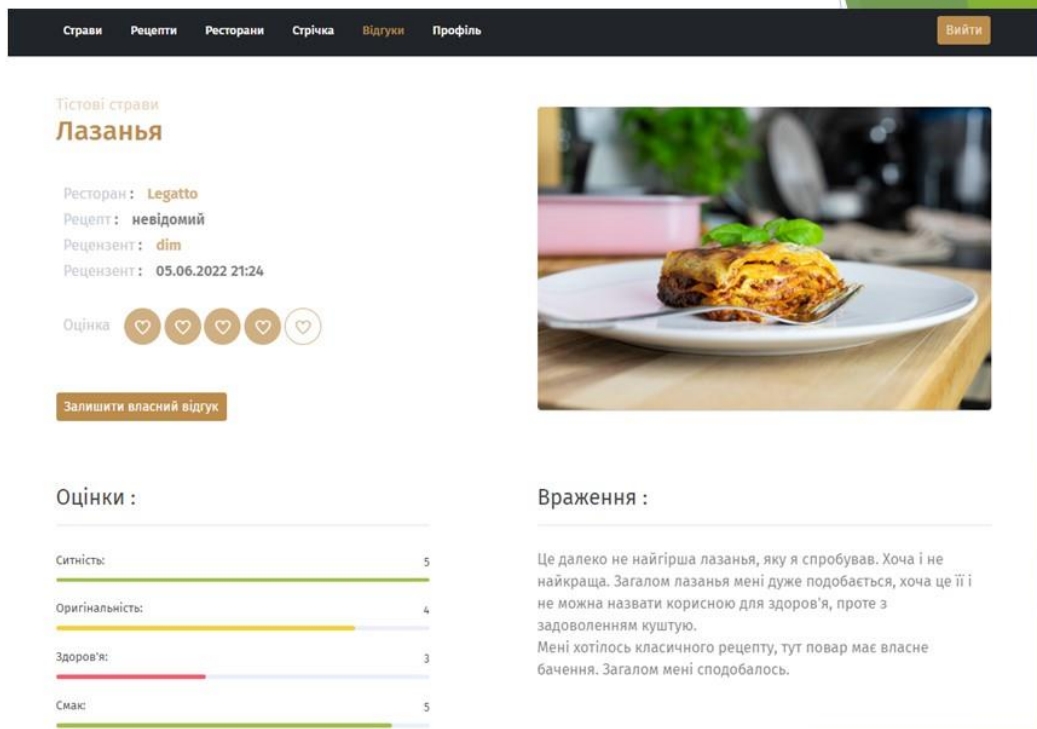


Рисунок В.11 – Слайд № 11

Висновки

12

За результатами аналізу проблеми виконано аналіз аналогів і їх порівняння з розробкою, визначено її особливості, що стосуються загалом ідеї концентрації саме на стравах.

За результатами проєктування представлено результати порівняння фреймворків для розробки вебзастосунків мовою Python, обрано фреймворк Django у цій якості, що визначило структуру програми на основі використання шаблону проєктування MVT, представлено функціональні вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв'язки між ними.

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми, включаючи класи моделей і уявлень, та загальну логіку роботи користувача, який виступає фактично рецензентом, з програмою. Під час виконання розробки та за результатами її виконання реалізовано тестування програми, що дозволило отримати результат, який відповідає запланованим для реалізації у технічному завданні вимогам.

Рисунок В.12 – Слайд № 12