Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)
(рівень вищої освіти)

на тему: "**Розроблення вебзастосунку для організації святкових фуршетів засобами Python**"

Виконав: студент IV курсу, групи **КН–41**
спеціальності 122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Грицьків Ю.В.
(прізвище та ініціали)

Керівник **Думанський О.І.**
(прізвище та ініціали)

Рецензент **Карашецький Б.П.**
(прізвище та ініціали)

Львів – 2025 р.

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 – "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____ Борецька І.Б.
"10" червня 2025 року

З А В Д А Н Н Я
на дипломну роботу студенту

Грицьківу Юрію Васильовичу
(прізвище, ім'я, по батькові)

1.Тема роботи "**Розроблення вебзастосунку для організації святкових фуршетів засобами Python**"
Керівник роботи *Думанський Остап Іванович, канд. фіз.–матем. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджений наказом вищого навчального закладу від "15" листопада 2024 року № С-882
2.Термін подання студентом (роботи) 10.06.2025 р.
3.Вихідні дані до роботи (проєкту). *Аналіз шляхів вирішення поставлених задач, організаційна структура застосунку , огляд алгоритмів та програмних засобів для їх реалізації.*
4.Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ.
Розділ 1. Стан проблемної області.
Розділ 2. Інформаційне та математичне забезпеченн.
Розділ 3. Програмне та технічне забезпечення
Висновки. Список використаних літературних джерел.
5. Перелік графічного матеріалу: *слайди для доповіді.*
6. Дата видачі завдання 18 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Огляд літературних та інших джерел згідно досліджуваної теми.	18.11.2024 р. – 20.12.2024 р.	*Виконано*
2.	Аналіз досліджуваної теми та вибір відповідних варіантів її розроблення	10.01.2025 р. – 25.01.2025 р.	*Виконано*
3.	Постановка задачі та її формалізація	01.02.2025 р.	*Виконано*
4.	Вибір та обгрунтування методів і засобів проведення дослідження	10.02.2025 р. – 20.02.2025 р.	*Виконано*
5.	Розроблення концептуальної схеми БД та структури таблиць	05.03.2025 р. – 15.03.2025 р.	*Виконано*
6.	Програмна реалізація завдання	20.03.2025 р. – 05.04.2025 р.	*Виконано*
7.	Тестування програмного продукту та отриманих результатів	10.04.2024 р. – 20.04.2024 р.	*Виконано*
8.	Розроблення пояснювальної записки дипломної роботи	01.05.2025р. – 20.05.2025 р.	*Виконано*
9.	Корегування пояснювальної записки згідно вимог, розроблення презентації	02.06.2025 р. – 08.06.2025 р.	*Виконано*

Студент _____ **Грицьків Ю.В.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Думанський О.І.**
(підпис) (прізвище та ініціали)

Анотація

Результатом моєї дипломної роботи є інформаційна система для організації святкового столу в ресторані. Для цього використана методика прийняття рішень згідно теорії графів, зокрема дерева прийняття рішень. Структурно система надає послуги працівникам ресторану. Система дозволяє замовляти страву, організовувати святкування, відправляти електронні листи організатору, скасовувати замовлення. Програма створена засобами мови програмування python з використанням бібліотеки tkinter для побудови інтерфейсу користувача та опрацювання його дій а також для додання авторизації та цільових можливостей інтеграції в робочий процес. Тестування цієї програми показало її цілісність та структурованість, вона має властивості, що були поставлені в технічному завданні при проектуванні.

Ключові слова: прибуткова організація, ресторан, дерева рішень, установи і організації, UI, Python, Re, Tkinter ,ООП, SMTP .

Annotation

The result of my diploma thesis is an information system for organizing a festive table in a restaurant. The system is based on a decision-making methodology according to graph theory, particularly decision trees. Structurally, the system provides services to restaurant staff. It allows for ordering dishes, organizing celebrations, sending emails to the event organizer, and canceling orders.
The program was developed using the Python programming language with the Tkinter library to build the user interface and handle user interactions, as well as to add authorization and targeted integration capabilities into the workflow. Testing of this program demonstrated its integrity and structured design, meeting the requirements specified in the technical documentation during the design phase.

Keywords: profitable organization, restaurant, decision trees, institutions and organizations, UI, Python, Re, Tkinter, OOP, SMTP.

ТЕХНІЧНЕ ЗАВДАННЯ

Було розроблено та програмно реалізувати на основі комп'ютерних технологій завдання стосовно розроблення інформаційної системи аналізу функціонування ресторану, тобто необхідно розробити систему, яка буде надавати послуги працівникам ресторану. Працівники за допомогою графічного інтерфейсу вводять всю необхідну інформацію в програму своєї установи і мають можливість редагувати або видаляти вже введену раніше інформацію.

У розробленій інформаційній системі вводиться наступна інформація:

- ✓ про страву;
- ✓ про вільні місця;
- ✓ про час замовлення
- ✓ про особу замовника.

Система повинна давати можливість відстежувати всі замовлення в ресторані.

ПЕРЕЛІК СКОРОЧЕНЬ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

PIL – Python Imaging Library

Tk – Tkinter

Re – RegEx - Regular Expression

SMTP - Simple Mail Transfer Protocol (SMTP, Простий Протоко Пересилання Пошти) — це комунікаційний протокол для пересилання електронної пошти.

ООП - Об’єктно-орієнтоване програмування (object-oriented

programming) – це метод програмування, побудований на представленні програми у виді сукупності взаємодіючих об’єктів, кожен з яких є екземпляром певного класу, а клас є членом певної ієрархії успадкування.

UI - Дизайн інтерфейсу користувача

ЗМІСТ

ВСТУП

Ресторанний бізнес є одним із найприбутковіших у світі. Формування високоефективного національного ресторанного господарства відіграє важливу роль у цій індустрії за обсягом матеріальних, фінансових ресурсів, забезпеченістю трудовим потенціалом і загальним обсягом доходів у бізнесі. Утворення стабільної клієнтської бази, пошук та розроблення нових шляхів розвитку організації ресторанного бізнесу, постійне оновлення концепції закладів з урахуванням динамічного ринку послуг в умовах політичної й економічної нестабільності та недостатньої кількості інвестиційних коштів є перспективними напрямами для розвитку економіки країни загалом. Лише після такої 6 деталізації доцільно пропонувати і розвивати конкретний напрям ведення ресторанного господарства з відповідним спектром послуг.

Актуальність теми. Можна зазначити, що розвиток ресторанної справи в Україні знаходиться на стадії зародження. Але ця галузь є перспективною, тому що вона сприяє підвищенню іміджу власника бізнесу та є привабливою (прибутковість закладів 15–20%) для інвестування на тривалий термін часу. Необхідно констатувати, що в національній ресторанній індустрії з'являються нові напрями діяльності: високе елітне обслуговування, тематичні ресторани, фаст-фуди, спортивні бари, обслуговування за типом «кейтеринг» тощо. Виходячи із цього, слід зазначити, що досвідчені ресторатори та ресторатори-початківці повинні ретельно досліджувати ринок ресторанного бізнесу, використовуючи основи ресторанного маркетингу, беручи до уваги уподобання і бажання споживачів, зважаючи на певні типи ресторанних закладів.

Об'єкт дослідження *– галузь електронної комерції, принципи побудови та функціонування додатків для організації свят.*

Предмет дослідження – системи впорядкування святкового столу.

Метою дипломної роботи є розробка системи святкових столів, яка дозволить підвищити ефективність ресторанів в цілому та облегшити

опрацювання. Система повинна забезпечувати максимально зручний інтерфейс, мінімальну ціну імплементації, та високу швидкодію.

Для реалізації мети необхідне вирішення таких задач:

- Розроблення інтерфейсу,
- Розроблення системи відправки повідомлень,
- Розроблення засобів опису структури підприємства.

Завдання – для досягнення поставленої мети визначено декілька завдань:

- Огляд існуючого програмного забезпечення в сфері генерації коду.
- Аналіз чинників, що впливають на ефективність програмного забезпечення.
- Формулювання алгоритму роботи програмного рішення.
- Реалізація програмного рішення.
- Апробація роботи пз на прикладних завданнях.

Практична значимість – відображається в ширшому охопленні етапів проектування та стилізації веб-інтерфейсів; підвищення доступності для користувачів незнайомих з веб-технологіями.

Розділ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Загальна інформація

Слід зазначити досить велику кількість розробок в даному руслі, попри те що вони так чи інакше орієнтовані або чисто для користувачів без досвіду, або на полегшення життя існуючих ресторанів.

З розвитком технологій, отримання базових програмних продуктів стає дедалі простішим для широкого кола користувачів, адже що простіше стає задовольняти потреби користувачів, що розробляти сервіси, які самостійно справляються з такими не хитрими завданнями.

Будь-яке універсальне програмне рішення завжди програє додаткам, створеним під конкретний бізнес і його специфічні бізнес-процеси. Всі існуючі програми занадто універсальні. Або вони вимагають значно доопрацювання, або

доведеться підлаштовувати бізнес під готову програмну систему. Однак мета - створити ефективний бізнес формат, який можна буде масштабувати на федеральний і навіть транснаціональний рівень. Компанія може це зробити тільки з власною системою, яка враховуватиме всю специфіку і деталі бізнесу.
Вигода в довгостроковій перспективі

Розробка власної інформаційної системи є дорогим рішенням у середньостроковій перспективі. Однак у довгостроковій перспективі з урахуванням масштабування бізнесу розробка свого ПЗ є більш вигідним проектом, так як всі авторські права на систему належать компанії і не доведеться виробляти ліцензійні відрахування після кожного запуску нового роздрібного об'єкта.

Власна інформаційна система буде збільшувати ефективність роздрібних операцій. У майбутньому наші франчайзі отримуватимуть не тільки торгову марку, меню і стандарти, а й інформаційну систему, створену під унікальні бізнес-процеси. Єдина інформаційна система дозволить також контролювати роботу франчайзі, зокрема - якість проведених операцій, що дозволить підтримувати єдині стандарти по всій мережі

В останні роки розвиваються комп'ютерні системи, що здатні забезпечити ефективне керування різними видами підприємств. Причому попит продовжує рости саме на інтегровані системи керування. Автоматизація окремих функцій (бухгалтерський облік, збут готової продукції) вважається вже пройденим етапом для багатьох підприємств. Сьогоднішній стан ринку комп'ютерних систем обумовлено, у першу чергу, історичним розвитком комп'ютерних систем і приходом західних розроблювачів і партнерів на ринок.

Більшість вітчизняних комп'ютерних систем з'явилося на рубежі 90-х років. У силу об'єктивних причин ринкової економіки, першими змогли виділити необхідні фінансові засоби підприємства торгівлі і сфери послуг. Таким чином, практично всі системи почали розвиватися як облікові бухгалтерські системи. Багато з них продовжують залишатися чисто обліковими, дозволяючи автоматизувати одну або кілька функцій підприємства, але не даючи цілісної картини для управління.

Основою багатогранної діяльності ресторанів є виробництво, яке є сукупністю процесів перетворення сировини на готову продукцію. Управління виробничим процесом здійснюється шляхом оперативного планування завдань для кожного цеху і відділення з урахуванням комплексної реалізації продукції та оптимального використання потужності виробництва. Основою оперативного планування є виробнича програма, яка визначає асортимент й обсяг виготовленої продукції за робочу зміну. Необхідність оперативного планування зумовлена особливостями виробничої діяльності ресторанів: широким асортиментом реалізованої продукції, частою зміною асортименту страв, сировини, коливанням попиту тощо. Наявність виробничої програми (плану-меню) дає змогу менеджерам ресторанного сервісу урізноманітнити харчування протягом окремих днів тижня, співвідносити випуск страв із графіком їх почасової реалізації, своєчасно здійснювати закупівлю сировини в необхідному обсязі й асортименті, планувати чисельний і кваліфікаційний склад працівників виробництва та сфери обслуговування, дбати про підвищення технічного рівня підприємств.

1.2 Огляд існуючих рішень

Інформаційна система Poster POS.

Одна з компаній займається розробкою власної інформаційної системи Poster POS. Інформаційна система Poster POS представляє з себе додаток. Сама система та база даних знаходяться на віддаленому сервері. Для роботи в системі необхідно постійне підключення комп'ютера або мобільного пристрою до Інтернету. Poster POS призначена для управління операціями такими як замовленнями організаціями столу чи роботу з клієнтами. Між Poster POS і спеціалізованими програмами буде налагоджено обмін даними. У Poster POS здійснюватиметься прийом і управління замовленнями, товарний і складський облік, управління запасами, управління персоналом, клієнтська база. З інформаційною системою буде інтегрований додатки для клієнтів. Інформаційна система є одним з головних конкурентних переваг нашої бізнес-концепції. Вона замислювалася спочатку як ядро нашого бізнесу.

Інформація, створена користувачами, може бути використана для поліпшення системи, проте користувачі мають можливість відмовитися від такого використання.

Користувачі мають можливість переглядати історію всіх своїх замовлень, а також можуть звернутися з проханням про видалення своїх даних.

Переваги: швидкість та зручність обслуговування, підвищення безпеки, автоматизація обліку та аналітики, гнучкість та масштабованість

Недоліки: залежність від інтернету та електропостачання, висока вартість впровадження, необхідність навчання персоналу, ризики кібербезпеки

Віртуальна АТС (ВАТС)

Це програма, що об'єднує всі номери компанії в єдину мережу, зберігає всю історію й записи розмов із клієнтами. Крім того, вона автоматизує роботу й підвищує ефективність співробітників і якість обслуговування клієнтів.

До ВАТС **Binotel** можна підключити ваші поточні стаціонарні та мобільні номери, а також нові телефонні номери Binotel та інших операторів

Binotel має велику бібліотеку шаблонів, які користувач може використовувати для початку роботи над своїм рестораном. Кожен шаблон можна змінювати та налаштовувати під конкретні потреби.

Редактор дозволяє додавати, видаляти та редагувати елементи додатку за допомогою перетягування та опускання. Користувачі можуть додавати тексти, зображення, відео, галереї, форми зворотного зв'язку та багато іншого.

Binotel надає безкоштовне хостингове обслуговування для всіх своїх користувачів, а також забезпечує безпеку веб-сайтів, включаючи SSL-шифрування для захисту особистих даних відвідувачів.

Переваги: Миттєве сповіщення про дзвінок клієнта у спливаючому вікні.Уся історія дзвінків і записи розмов із клієнтами в CRM.Набір номера одним кліком із картки клієнта**.**

Недоліки: Проблемність інтеграції функціоналу для програміста. Неможливість розгорнути на альтернативному хостинг сервісі.

1.3. Дерево цілей розроблюваної системи:

Розділ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Python

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Розроблена на початку 1990-х років Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування , зокрема: об'єктно орієнтована, процедурна, аспектно - орієнтована та функціональна.

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке називається IDLE і яке написане мовою Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

2.2 *PIL*

Python Imaging Library — open-source бібліотека мови Python , призначена для роботи з растровою графікою.

Можливості бібліотеки:

- підтримка бінарних, напівтонових, індексованих, повнокольорових і CMYK зображень;
- підтримка форматів BMP, EPS, GIF, JPEG, PDF, PNG, PNM, TIFF і деяких інших у режимі читання та запису;
- підтримка форматів (ICO, MPEG, PCX, PSD, WMF та інших) тільки для читання;
- перетворення зображень з одного формату у інший;
- редагування зображень (використання різноматніх фільтрів, масштабування, малювання, матричні операції і т.п.);
- використання бібліотеки з Tkinter та PyQt.

Потребує наявності бібліотек zlib (для PNG), libjpeg, freetype2 (для OpenType/TrueType).

2.3 Tkinter

Tkinter — багато платформна графічна бібліотека інтерфейсів на основі засобів Tk (широко розповсюджена у світі GNU/Linux та інших UNIX подібних систем, портована в тому числі і на Microsoft Windows, Apple Mac OS), поширювана з відкритими вихідними текстами, написана Стіном Лумхольтом (Steen Lumholt) і Гвідо ван Россумом. Входить у стандартну бібліотеку Python.

Бібліотека Tkinter не реалізує власний інтерфейс до бібліотеки Tk, а забезпечує конвертування звернень Python в звернення Tcl — мови, яка тісно інтегрована з Tk. Таким чином Tkinter є обгорткою для Tcl/Tk.

2.4 Regular Expression

Regular Expression — це послідовність символів, які визначають шаблон для пошуку відповідностей.

Цей модуль забезпечує регулярні операції з узгодження виразів, подібні до ті, що знайдені в Перлі.

Як шаблони, так і рядки, які потрібно шукати, можуть бути рядками Unicode (str) а також 8-бітні струни (bytes). Однак рядки Unicode та 8-бітні рядки не можна змішувати: тобто ви не можете співставити рядок Unicode з шаблоном байтів або навпаки; аналогічно, коли просять про заміну, заміну рядок повинен бути одного типу, як і шаблон, і рядок пошуку.

Регулярні вирази використовують символ зворотної косої риски ('\') вказати спеціальні форми або дозволити використовувати спеціальні символи без посилань їх особливе значення. Це узгоджується із використанням Python символ з тією ж метою в рядкових літералах; наприклад, відповідати буквальна зворотна коса риса, можливо, доведеться написати '\\\\' як візерунок рядок, тому що регулярний вираз повинен бути \\, і кожен зворотний нахил повинен бути виражений як \\ всередині звичайного рядка Python буквальний. Також зауважте, що будь-які недійсні послідовності втечі в Python використання зворотної косої риски в рядкових літералах тепер генерує a SyntaxWarning і в майбутньому це стане SyntaxError. Така поведінка відбудеться, навіть якщо це дійсна послідовність втечі для регулярного виразу.

Рішення полягає у використанні позначення сирого рядка Python для регулярного вираження візерунки; зворотні косої риси не обробляються жодним особливим чином у буквальному буквалі з префіксом 'r'. Так r"\n" є двозначною струною, що містить '\' і 'n', поки "\n" є односимвольним рядком, що містить новий рядок. Зазвичай шаблони будуть виражені в коді Python, використовуючи цей сирий позначення рядків.

Важливо зазначити, що більшість регулярних виразних операцій доступні як функції та методи на рівні модуля складені регулярні вирази. Функції - ярлики які не вимагають, щоб ви спочатку склали об'єкт regex, але пропустіть його параметри тонкої настройки.

2.5 Smtplib

Simple Mail Transfer Protocol (SMTP, Простий Протокол Пересилання Пошти) — це комунікаційний протокол для пересилання електронної пошти.

Як Інтернет-стандарт, SMTP вперше визначений у 1982 в RFC 821 та оновлений у 2008 в RFC 5321 до ESMTP (*Extended SMTP*, укр. *Розширений SMTP*), який і використовується в даний час.[*коли?*] Поштові сервери та інші агенти передачі повідомлень використовують SMTP для надсилання та отримання поштових повідомлень.

Пропрієтарні (власницькі) системи, такі як Microsoft Exchange Server і HCL Domino, а також системи вебпошти, такі як Outlook.com, Gmail та Yahoo! Mail можуть використовувати нестандартні протоколи всередині системи, але всі вони використовують SMTP під час надсилання або отримання електронної пошти за межами власної системи. SMTP-сервери зазвичай використовують TCP та порт 25, який призначений IANA для SMTP. SMTP з'єднання з SSL шифруванням використовують порт 587 або 465.

У той час, як поштові сервери та інші агенти передачі повідомлень використовують SMTP для надсилання та отримання поштових повідомлень, працюючі на користувацькому рівні клієнтські поштові програми зазвичай використовують SMTP лише для надсилання повідомлень на поштовий сервер для ретрансляції, використовуючи, за замовчуванням порт 587 або 465 на поштовому сервері, відповідно до RFC 8314. Для отримання повідомлень клієнтські програми використовують протоколи POP або IMAP, які є стандартом, але пропрієтарні сервери також часто реалізують власні протоколи, наприклад Exchange ActiveSync.

Екземпляр SMTP_SSL поводиться точно так само, як екземпляри SMTP. SMTP_SSL слід використовувати для ситуацій, коли SSL потрібен від початку з’єднання, а використання starttls() не підходить. Якщо *host* не вказано, використовується локальний хост. Якщо *port* дорівнює нулю, використовується стандартний порт SMTP через SSL (465). Необов’язкові аргументи *local_hostname*, *timeout* і *source_address* мають те саме

значення, що й у класі SMTP. *context*, також необов’язковий, може містити SSLContext і дозволяє налаштовувати різні аспекти безпечного з’єднання. Будь ласка, прочитайте Міркування безпеки, щоб дізнатися про найкращі практики.

2.6 CustomTkinter

CustomTkinter - це настільна бібліотека Python на базі Tkinter, яка забезпечує сучасний вигляд та повністю настроюванні віджети. За допомогою CustomTkinter ми можем послідовний вигляд на всіх платформах ПК (Windows, macOS, Linux).

CustomTkinetr був розроблений TomSchimanSky (кудо йому для розвитку цього).

Розділ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Реляційна алгебра

Реляційна модель даних (РМД) - логічна модель даних, прикладна теорія побудови баз даних, яка є додатком до завдань обробки даних таких розділів математики як теорії множин і логіка першого порядку. На реляційної моделі даних будуються реляційні бази даних.

Реляційна модель даних включає такі компоненти:

- Структурний аспект (складова) - дані в базі даних є набором відносин.
- Аспект (складова) цілісності - відносини (таблиці) відповідають певним умовам цілісності. РМД підтримує декларативні обмеження цілісності рівня домену (типу даних), рівня відносини і рівня бази даних.
- Аспект (складова) цілісності - відносини (таблиці) відповідають

певним умовам цілісності. РМД підтримує декларативні обмеження цілісності рівня домену (типу даних), рівня відносини і рівня бази даних.

- Аспект (складова) обробки (маніпулювання) - РМД підтримує оператори маніпулювання відносинами (реляційна алгебра, реляційне числення). Крім того, до складу реляційної моделі даних включають теорію нормалізації.

Термін "реляційний" означає, що теорія заснована на математичному понятті ставлення (relation). Як неформального синоніма терміну "відношення" часто зустрічається слово таблиця. Необхідно пам'ятати, що "таблиця" є поняття нестроге і неформальне і часто означає не "ставлення" як абстрактне поняття, а візуальне уявлення відносини на папері або екрані.

Некоректне і нестроге використання терміну "таблиця" замість терміна "ставлення" нерідко призводить до нерозуміння. Найбільш часта помилка полягає в міркуваннях про те, що РМД має справу з "плоскими", або

"двовимірними" таблицями, тоді як такими можуть бути тільки візуальні представлення таблиць. Відносини ж є абстракціями, і не можуть бути ні "плоскими", ні "неплоским".

Для кращого розуміння РМД слід відзначити три важливі обставини:

- модель є логічною, тобто відносини є логічними (абстрактними), а не фізичними (збереженими) структурами;
- для реляційних баз даних вірний інформаційний принцип : все інформаційне наповнення бази даних представлено одним і тільки одним способом, а саме - явним завданням значень атрибутів у кортежі відносин; зокрема, немає ніяких покажчиків (адрес), що зв'язують одне значення з іншим;
- наявність реляційної алгебри дозволяє реалізувати декларативне програмування і декларативне опис обмежень цілісності, на додаток до навігаційного (процедурним) програмування і процедурної перевірці умов.

3.2. Операції реляційної алгебри

Реляційна алгебра - замкнута система операцій над відносинами в реляційної моделі даних, або Реляці йна алгебра — відгалуження логіки першого порядку, множина відношень замкнених операторами. Оператори застосовуються до відношень, в результаті застосування отримується нове відношення.

В математиці, алгебра відношень є алгебраїчною структурою щодо математичної логіки та теорії множин.

РМД стала першою працездатною моделлю даних, оскільки мала ефективний інструментарій - операції реляційної алгебри. Основною одиницею обробки є відношення, а не його кортежі.

Реляційна алгебра включає дві групи операцій.

1. Традиційні операції над множинами (модифіковані з урахуванням того, що їх операндами є відношення) - об'єднання, перетин, різниця (віднімання), декартовий твір і розподіл.

2. Спеціальні реляційні операції - вибірка, проекція, з'єднання.

Розділ 4. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Код програми

При запуску в додаток генерує UI оформлення для вікна замовлення місць.

```
labels = {

        "Ширина столу (см):": r"^\d+$",

        "Довжина столу (см):": r"^\d+$",

        "Кількість місць:": r"^\d+$",

        "Час початку (ГГ:ХХ):": r"^([01]?\d|2[0-3]):[0-5]\d$",

        "Час завершення (ГГ:ХХ):": r"^([01]?\d|2[0-3]):[0-5]\d$",

        "Дата бронювання (ДД.ММ.РРРР):": r"^\d{2}\.\d{2}\.\d{4}$"

    }

    for label_text, regex in labels.items():

        frame = ctk.CTkFrame(self.root, height=25,fg_color="#ffd3ac")

        frame.pack(pady=5, fill="x", padx=10)

        ctk.CTkLabel(frame,
text=label_text,text_color='#664b37',fg_color="#ffd3ac").pack(side="left", padx=5)

        entry = ctk.CTkEntry(frame,
height=20,text_color='#664b37',fg_color="#ffd3ac")

        #entry.configure(ttk.highlightthickness==2, ttk.highlightcolor=="red",
ttk.bg=="lightblue"    )

        entry.pack(side="right", expand=True, fill="x")

                if previous_data and label_text in previous_data:

            entry.insert(0, previous_data[label_text])

        self.entries[label_text] = (entry, regex)
```

```
self.forme_var = tk.StringVar(value="Квадратний")

forme_frame =ctk.CTkFrame(self.root)

forme_frame.pack(pady=5, fill="x", padx=10)

ctk.CTkLabel(forme_frame, text="Форма
столу:",fg_color="#ffd3ac",text_color='#664b37').pack(side="left", padx=5)

forems = ["Квадратний", "Овальний", "Прямокутний"]

for forem_text in forems:

    ctk.CTkRadioButton(forme_frame, text=forem_text, variable=self.forme_var,
value=forem_text,fg_color="#ffd3ac",text_color='#664b37').pack(side="left")

ctk.CTkButton(self.root, text="Зарезервувати",
command=self.reserve_table,fg_color="#ffd3ac",text_color='#664b37').pack(pady=2
0)
```

Ця частина коду відповідає застворення UI дизайну вікна замовлення страв. Для редагування попереднього вікна є кнопка **Назад**(***Рисунок 4.1***) яка запускає функцію(def go_back) що поветрає вікно замовлення Місць.

```
class MenuApp:

def __init__(self, root, previous_data, orders, contact_data=None):

    self.previous_data = previous_data

    self.orders = orders

    self.contact_data = contact_data if contact_data else {}

    self.root = root

    self.root.title("Меню")

    self.root.geometry("900x700")

    self.root.configure(bg="#D2B48C")
```

ctk.CTkButton(self.root,text="Назад",command=self.go_back).pack(pady=5, anchor="w"

```
self.main_frame = tk.Frame(root, bg="#D2B48C")
self.main_frame.pack(fill="both", expand=True)
self.menu_frame = tk.Frame(self.main_frame, bg="#D2B48C")
self.menu_frame.pack(side="left", fill="both", expand=True)
self.order_frame = tk.Frame(self.main_frame, bg="#8B4513", width=300)
self.order_frame.pack(side="right", fill="y", padx=5, pady=5)
self.notebook = ttk.Notebook(self.menu_frame)
self.notebook.pack(fill="both", expand=True)
```

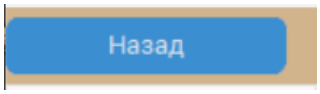

Рисунок 4.1 – Кнопка повернення столу

```
def go_back(self):
    self.root.destroy()
    root = tk.Tk()
    TableReservationApp(root, self.previous_data, self.orders, self.contact_data)
    root.mainloop()

def go_to_cart(self):
    self.root.destroy()
    root = tk.Tk()
```

```
        CartApp(root, self.previous_data, self.orders, self.contact_data)
        root.mainloop()
    def create_tab(self, section_name, menu_items):
        tab = tk.Frame(self.notebook, bg="#D2B48C")
        self.notebook.add(tab, text=section_name)
        canvas = tk.Canvas(tab, bg="#D2B48C")
        scrollbar = tk.Scrollbar(tab, orient="vertical", command=canvas.yview)
        scrollable_frame = tk.Frame(canvas, bg="#D2B48C")
        scrollable_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))
        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)
        canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")
        for name, price, weight, img_path in menu_items:
            frame = tk.Frame(scrollable_frame, bg="#D2B48C")
            frame.pack(fill="x", padx=5, pady=2)
            img = self.load_image(img_path, size=(50, 50))
            img_label = tk.Label(frame, image=img, bg="#D2B48C")
            img_label.image = img
            img_label.pack(side="left", padx=5)
            text_label = tk.Label(frame, text=f"{name} ({weight} г) - {price}", font=("Arial", 12), bg="#D2B48C")
```

```
        text_label.pack(side="left", fill="x", expand=True)
        entry = tk.Entry(frame, width=5)
        quantity = self.orders.get(name, (0, img_path))[0]
        entry.insert(0, str(quantity))
        entry.pack(side="right", padx=5)
        entry.bind("<KeyRelease>", lambda event, dish=name, ent=entry, img=img_path: self.update_order(dish, ent, img))
        self.entries[name] = entry
```

Після заповнення необхідних даних відкривається вікно Сторінка результату замовлення(***Рисунок 4.5***) В ній демонструється обрана страва. Під списком є однорядкові текстові поля в яких необхідно заповнити ім’я, телефон електронну адресу користувача. Після заповнення дані будуть надіслано на електрону пошту.

```
class CartApp:
    def __init__(self, root, previous_data, orders, contact_data=None):
        self.previous_data = previous_data
        self.orders = orders
        self.contact_data = contact_data if contact_data else {}
        self.root = root
        self.root.title("Кошик")
        self.root.geometry("500x600")
        self.root.configure(background="#a4c639")
        ctk.CTkButton(self.root, text="Назад", command=self.go_back).pack(pady=5, anchor="w")
```

```
        ctk.CTkLabel(self.root, text="Ваше замовлення", font=("Arial", 12, "bold")).pack(pady=10)

        # Створюємо фрейм з canvas для скролу
        container = ctk.CTkFrame(self.root, height=300)
        container.pack(pady=5, fill="x")
        container.pack_propagate(False)
        canvas = tk.Canvas(container, height=300, background="#a4c639")
        scrollbar = ttk.Scrollbar(container, orient="vertical", command=canvas.yview)
        scrollable_frame = ttk.Frame(canvas)
        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
        )
        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=scrollbar.set)
        canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")
        for dish, (quantity, img_path) in self.orders.items():
            if quantity > 0:
                frame = ttk.Frame(scrollable_frame)
                frame.pack(pady=5, fill="x")
```

```
                try:
                    img = Image.open(img_path)
                    img = img.resize((50, 50), Image.LANCZOS)
                    photo = ImageTk.PhotoImage(img)
                    label_img = tk.Label(frame, image=photo)
                    label_img.image = photo
                    label_img.pack(side="left", padx=5)
                except:
                    pass
                ctk.CTkLabel(frame, text=f"{dish}: {quantity} шт.",bg_color="#a4c639").pack(side="left", padx=10)
            ctk.CTkLabel(self.root, text="Контактні дані", font=("Arial", 12, "bold")).pack(pady=10)
            self.contact_entries = {}
            contact_labels = {
                "Ім'я:": r"^.+$",
                "Телефон:": r"^\+?\d{10,15}$",
                "Email:": r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
            }
            for label, regex in contact_labels.items():
                frame = ctk.CTkFrame(self.root)
                frame.pack(pady=5, fill="x", padx=10)
                ctk.CTkLabel(frame, text=label).pack(side="left", padx=5)
```

```
            entry = ctk.CTkEntry(frame)
            entry.pack(side="right", expand=True, fill="x")
            if label in self.contact_data:
                entry.insert(0, self.contact_data[label])
            self.contact_entries[label] = (entry, regex)
        ctk.CTkButton(self.root, text="Підтвердити замовлення", command=self.confirm_order).pack(pady=10)
    def go_back(self):
        contact_data = {label: entry.get().strip() for label, (entry, regex) in self.contact_entries.items()}
        self.root.destroy()
        root = tk.Tk()
        MenuApp(root, self.previous_data, self.orders, contact_data)
        root.mainloop()
    def confirm_order(self):
        try:
            contact_info = {label: entry.get().strip() for label, (entry, regex) in self.contact_entries.items()}
            for label, regex in self.contact_entries.items():
                if not re.match(regex[1], contact_info[label]):
                    raise ValueError(f"Неправильний формат: {label}")
            order_details = {
                "Дата": selected_table.day if selected_table else "Невідомо",
```

```
                "Час": f"{selected_table.start} - {selected_table.end}" if selected_table
else "Невідомо",
                "Форма столу": selected_table.forme if selected_table else
"Невідомо",
                "Місця": selected_table.seats if selected_table else "Невідомо",
                "Ім'я": contact_info["Ім'я:"],
                "Телефон": contact_info["Телефон:"],
                "Email": contact_info["Email:"],
                "Страви": "\n".join([f"{dish}: {qty} шт." for dish, (qty, _) in
self.orders.items() if qty > 0])
            }
            send_email(order_details)
            messagebox.showinfo("Успіх", "Ваше замовлення прийнято та
відправлено на пошту!")
            self.root.destroy()
        except ValueError as e:
            messagebox.showerror("Помилка!", str(e))
```

```
354
355 def send_email(order_details):
356         sender_email = "uriyghritskiv1@gmail.com"
357         sender_password = "oird ptih hndq zeym"
358         receiver_email = "uriyghritskiv1@gmail.com"
359         subject = "Нове замовлення з програми бронювання столу"
360
361         body = f"""
362         Ви отримали нове замовлення:
363
364         Дата бронювання: {order_details['Дата']}
365         Час: {order_details['Час']}
366         Форма столу: {order_details['Форма столу']}
367         Місця: {order_details['Місця']}
368
369         --- Контактні дані ---
370         Ім'я: {order_details["Ім'я"]}
371         Телефон: {order_details['Телефон']}
372         Email: {order_details['Email']}
373
374         --- Страви ---
375         {order_details['Страви']}
376         """
377
378         msg = EmailMessage()
379         msg.set_content(body)
380         msg["Subject"] = subject
381         msg["From"] = sender_email
382         msg["To"] = receiver_email
383
```

Рисунок 4.2 – Оформлення

4.2 Тестування контрольного прикладу

На сторінці додатку містяться поле для вводу, кнопка організації столу, його збереження і заповнення (рис. 4.3). При помилці можна завжди повернутися до попередніх налаштувань.

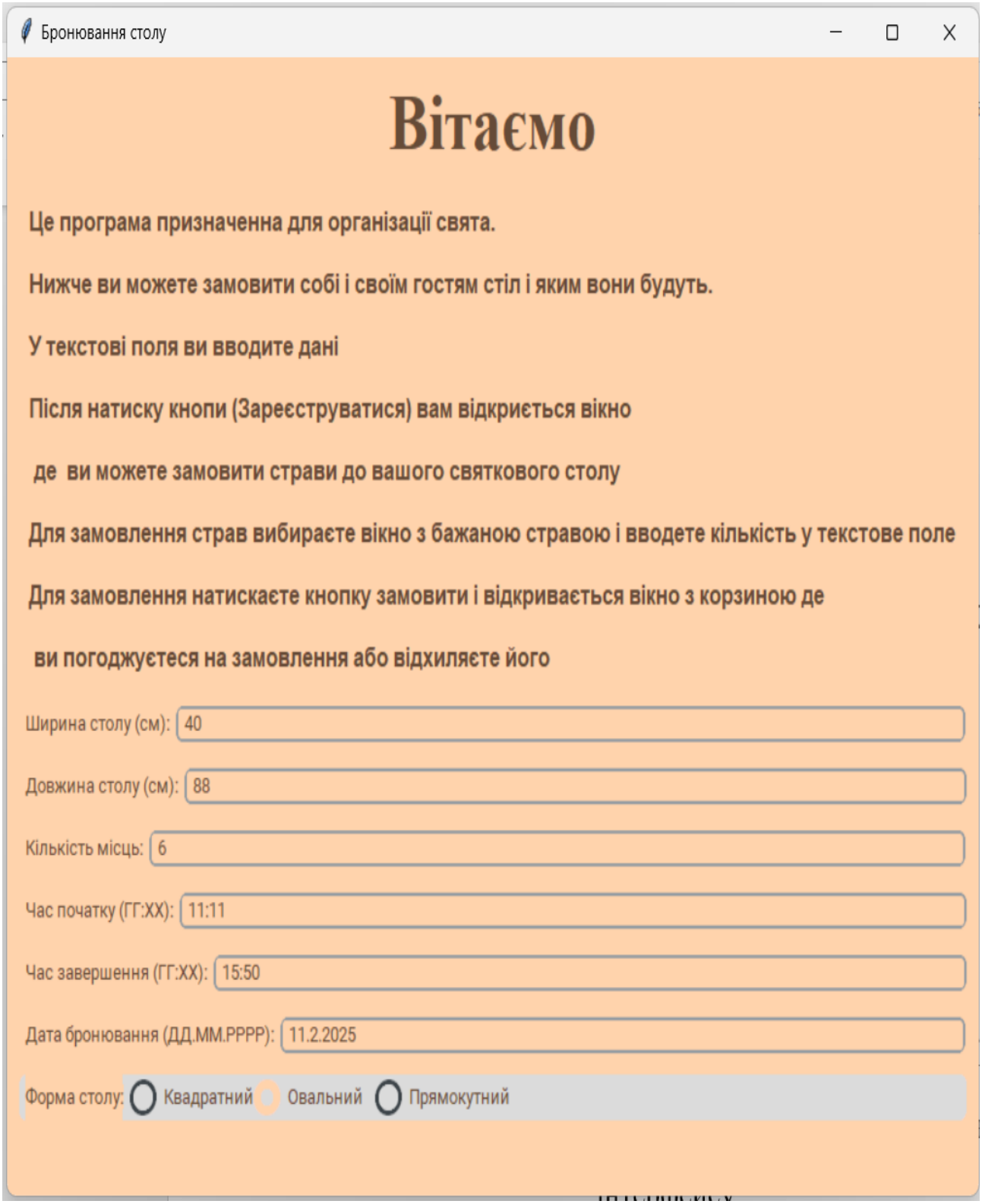

Рисунок 4.3 – Сторінка додатку

Після надсилання запиту відкривається нове вікно де можна замовляти різноманітні страви (рис. 4.4).При виборі страви з правого краю вікна можна побачити замовлені страви. Зліва зверху є кнопка для повернення до попереднього вікна

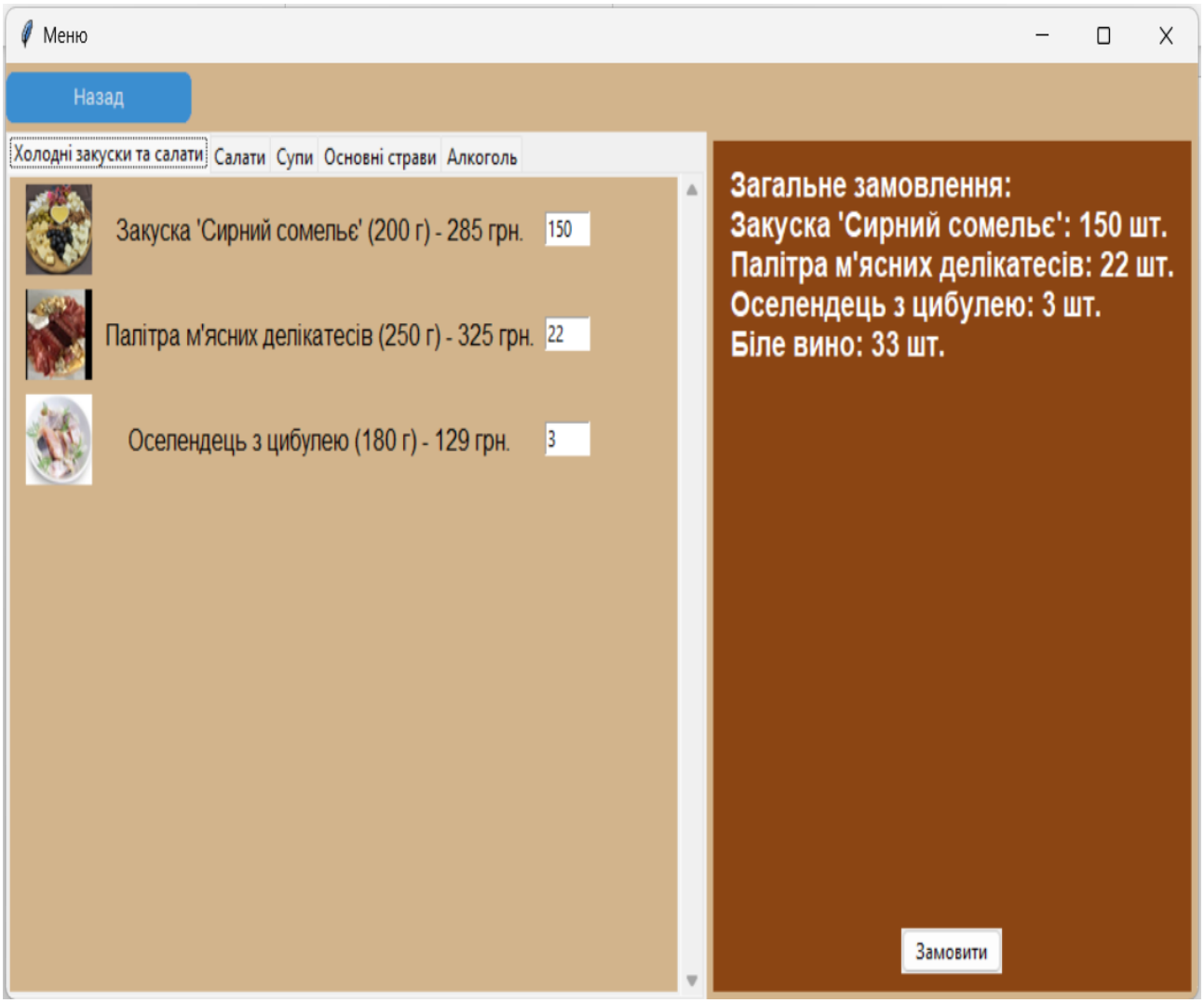

Рисунок 4.4– Сторінка замовлення страв

Після вибору кнопки “Замовити” в профілі користувача створюється репозиторій із файлом розмітки, щойно згенерованим (рис. 4.5).

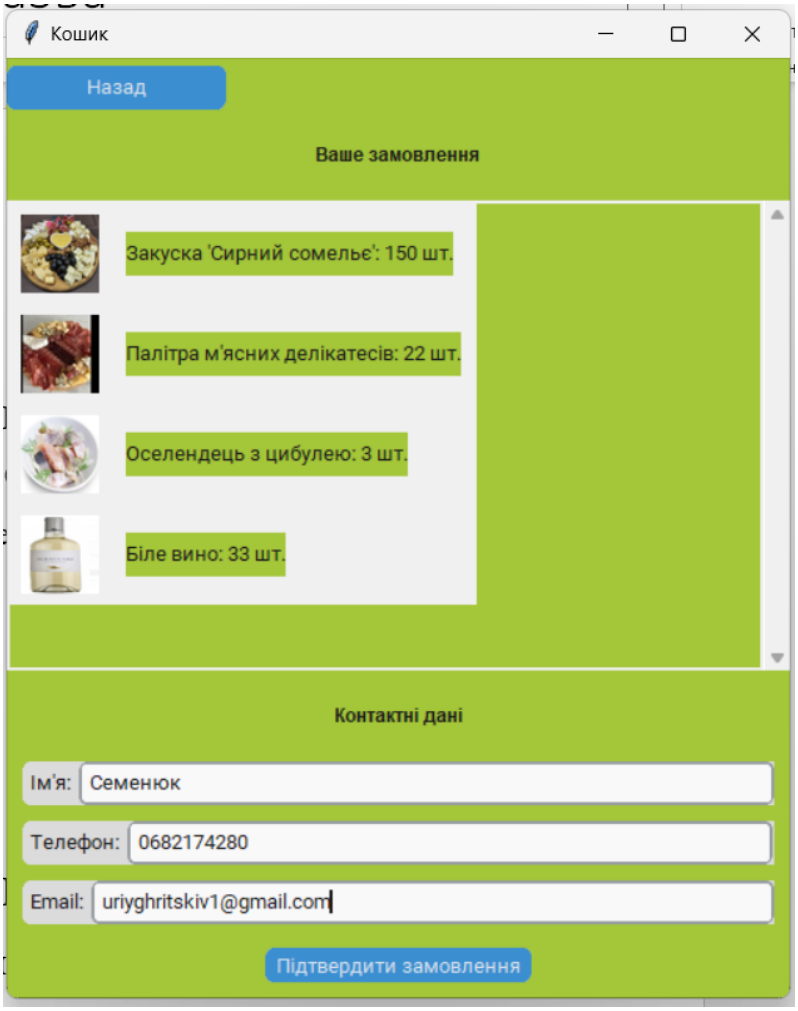

Рисунок 4.5 – Сторінка результату замовлення

Назва його містить часовий штамп, коли його було створено.

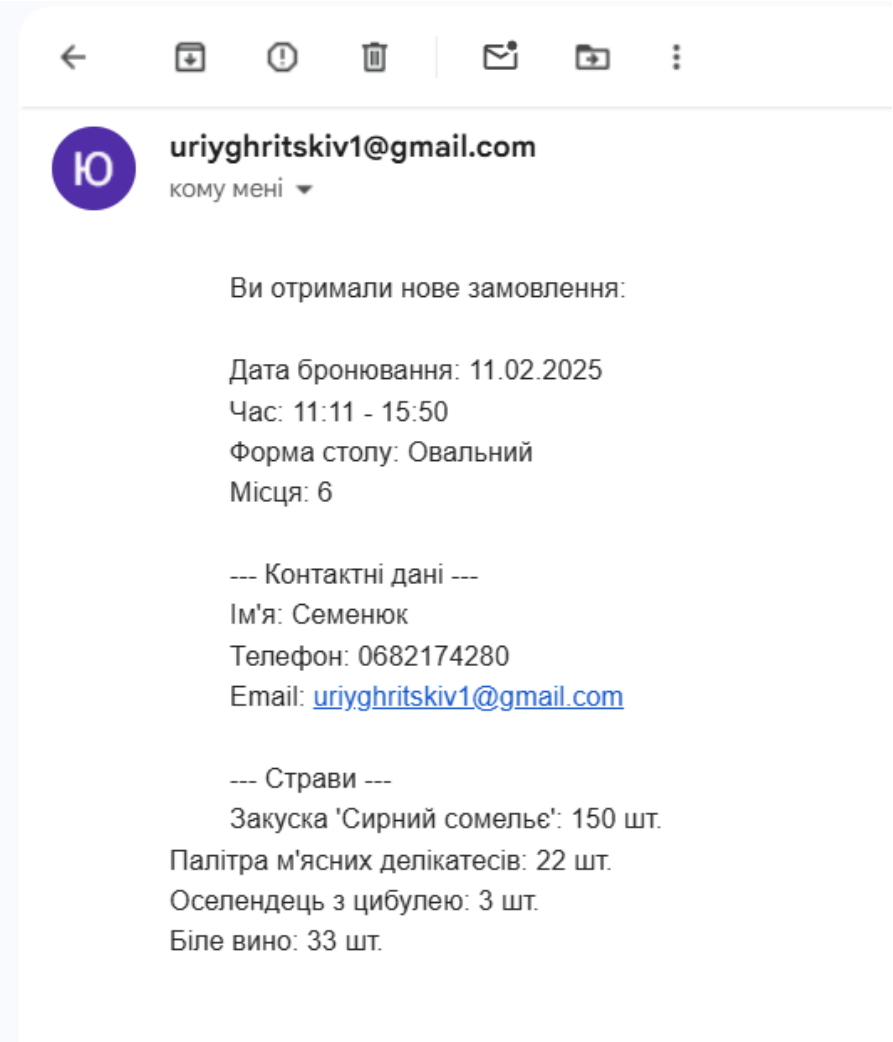

Рисунок 4.6 – Екземпляр відправленого повідомлення

ВИСНОВКИ

Результатом роботи над проектом став додаток, що пропонує роботу над замовленнями в ресторані, що значно полегшує процес обслуговування.

Розроблений програмний продукт був ретельно протестований на всіх етапах розробки та перевірений на відповідність вимогам технічного завдання. У результаті можна стверджувати, що застосунок повністю відповідає поставленим завданням і забезпечує високу швидкість та якість обслуговування.

Практичне значення результатів полягає в наступному:

- Проведено дослідження предметної області – ресторанної справи;
- Отримано можливість швидкої адаптації та експорту коду: програмний продукт дозволяє зручно експортувати згенерований код для подальшої роботи, що підвищує гнучкість та ефективність процесу розробки.
- Підвищується якість та точність стилізації: додаток забезпечує високу якість та узгодженість стилізації вікон, що сприяє підвищенню загальної якості продукту.
- Зроблено огляд найбільш відомих існуючих інформаційних систему у ресторанній справі;
- Сформульовано мету розробки та описано призначення інформаційної системи, побудовано
- Проведено аналіз технологій та засобів розробки інформаційної системи та вибрані засоби розв’язання поставленої задачі
- Розроблено інформаційну систему для ресторану;
- Продемонстровано роботу побудованої системи на тестовому прикладі та детально описано його виконання

Таким чином, впровадження цього додатку значно підвищує ефективність та продуктивність роботи ресторанів та забезпечує високу якість результату.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Сахно Є. Ю. Менеджмент сервісу. Теорія та практика : навчальний посібник / Ю. Сахно, М. С. Дорош. – К. : Центр учбової літератури, 2010. – 328 с.
2. Сало Я. М. Організація обслуговування населення на підприємствах ресторанного сервісу. Ресторанна справа : довідник офіціанта / Я. М. Сало. – Львів : Афіша 2007. – 301 с.
3. П’ятницька Г. Т. Розвиток ресторанного господарства в Україні: структурні трансформації, фінансова стійкість підприємств, прогнози / Г. Т. П’ятницька, О. М. Григоренко, В. С. Найдюк // Економіст. – 2013. – № 11. – С. 37–45.
4. П’ятницька Г. Т. Інноваційні ресторанні технології: основи теорії : навчальний посібник для вищих навчальних закладів / Г. Т. П’ятницька, Н. О. П’ятницька. – К. : Кондор-Видавництво, 2013. – 250 с.
5. Організація обслуговування у закладах ресторанного господарства : підручник : [для вищ. навч. закл.] / за ред. Н. О. Пятницької. – 2-ге вид., переробл. та доповн. – К. : Центр учбової літератури, 2011 – 584 с.
6. Нечаюк Л. І. Готельно-ресторанний бізнес: менеджмент : навч. посіб. для студ. вищих навч. закладів / Л. І. Нечаюк, Н. О. Нечаюк // Київський національний ун-т культури і мистецтв. – К. : Центр навч. літератури, – 2009. – 344 с
7. Мережа роздрібної торгівлі та ресторанного господарства підприємств на 1 січня 2013 року : статистичний бюлетень. – К. : Державна служба статистики України, 2013. – 123 с.
8. Мостова Л. М. Організація обслуговування на підприємствах ресторанного господарства : навчальний посібник / Л. М. Мостова, О. В. Новікова. – К. : Ліра-К, 2010. – 308 с.
9. Мазаракі А. А. Проектування закладів ресторанного господарства : підручник для ВУЗів / за ред.проф. А. А. Мазаракі. – 2-ге видання, доповнене і виправлене. – К. : Київ. нац. торг.-екон. ун-т, 2011. – 339 с

10. Кривошей В. В. Трудовий капітал ресторанного господарства: теорія та методологія управління : монографія / В. В. Кривошей. – Х. : ФОРТ, 2011. – 255 с.
11. Зубар Н. М. Логістика у ресторанному господарстві : навчальний посібник / Н. М. Зубар, М. Ю. Григорак. – К. : Центр навчальної літератури, 2010. – 312 с. 4
12. Долина Н. Як три деньки не поборщуєш, на серці замоторошнить: українська національна кухня добре збалансована за кількістю жирів, вуглеводів, білків / Н. Долина // Урядовий кур'єр. – 2012. – № 36, 24 лютого. – С. 8.
13. Агафонова Л. Г. Туризм, готельний та ресторанний бізнес: ціноутворення, конкуренція, державне регулювання : навч посіб. для студ. вищ. навч. закл. / Київський ун-т туризму, економіки і права / Л. Г. Агафонова, О. С. Агафонова. − К. : Знання України, 2002. – 352 с.
14. Азарян О. М. Сегментація підприємств ресторанного бізнесу в Україні / О. М. Азарян, О. В. Сушко // Схід. – 2004. – Вип. 5 (63). – С. 48.
15. Архіпов В. В. Ресторанна справа: асортимент, технології управління якістю в сучасному ресторані / В. В. Архіпов, Т. В. Іваннікова, А. В. Архіпова та ін. – К. : ІНОКС, 2007. – 382 с.
16. Давидова О. Ю. Інформаційно-комп'ютерні інновації в ресторанному бізнесі / О. Ю. Давидова, Н. В. Полстяна // Комунальне господарство міст. – 2012. – № 106. – С. 403–408.– (Серія «Економічні науки»).
17. Долгопол, О. О. Формування організаторських умінь майбутніх молодших спеціалістів сфери ресторанного обслуговування у процесі професійної підготовки у коледжі : автореф. дис. ... канд. пед. наук : спец. 13.00.04 «Теорія та методика професійної освіти» / О. О. Долгопол ; МОН України, Українська інженерно-педагогічна академія. – Х., 2014. – 20 с.
18. https://customtkinter.tomschimansky.com/documentation/
19. https://sites.google.com/comp-sc.if.ua/python-easy/tkinter
20. https://pythonprogramming.altervista.org/create-more-windows-with-tkinter/
21. https://knowledge-base.joinposter.com/en/

22.https://coderslegacy.com/python/python-gui/

23.https://www.w3schools.com/python/default.asp

\

ДОДАТКИ

```
import tkinter as tk

from tkinter import ttk, messagebox

from PIL import Image, ImageTk

import re

import smtplib

from email.message import EmailMessage

import customtkinter as ctk

class Table:

    def __init__(self, width, length, forme, seats, price, start, end, day):

        self.width = width

        self.length = length

        self.forme = forme

        self.seats = seats

        self.price = price

        self.start = start

        self.end = end

        self.day = day

selected_table = None

cart_items = []

class TableReservationApp:

    def __init__(self, root, previous_data=None, previous_orders=None, contact_data=None):

        self.root = root

        self.root.title("Бронювання столу")

        self.root.geometry("800x850")

              self.root.configure(background="#ffd3ac")

              ttk.Label(self.root, text="Вітаємо",background="#ffd3ac",foreground='#664b37', font=("Times New Roman", 36, "bold")).pack(pady=10)

             ttk.Label(self.root,

                text="Це програма призначенна для організації свята.\r\nНижче ви можете замовити собі і своїм гостям стіл і яким вони будуть.\r\nУ текстові поля ви вводите дані\r\nПісля натиску кнопки (Зареєструватися) вам відкриється вікно \r\n де  ви можете замовити страви до вашого святкового столу\r\nДля замовлення страв вибираєте вікно з бажаною стравою і вводете кількість у текстове поле\r\nДля замовлення натискаєте кнопку замовити і відкривається вікно з корзиною де \r\n ви погоджуєтеся на замовлення або відхиляєте його",
```

```
            background="#ffd3ac",foreground='#664b37',
            font=("Arial", 12, "bold" )).pack(pady=10)
        self.previous_orders = previous_orders if previous_orders else {}
        self.contact_data = contact_data if contact_data else {}

        self.entries = {}
        labels = {
            "Ширина столу (см):": r"^\d+$",
            "Довжина столу (см):": r"^\d+$",
            "Кількість місць:": r"^\d+$",
            "Час початку (ГГ:ХХ):": r"^([01]?\d|2[0-3]):[0-5]\d$",
            "Час завершення (ГГ:ХХ):": r"^([01]?\d|2[0-3]):[0-5]\d$",
            "Дата бронювання (ДД.ММ.РРРР):": r"^\d{2}\.\d{2}\.\d{4}$"
        }
        for label_text, regex in labels.items():
            frame = ctk.CTkFrame(self.root, height=25,fg_color="#ffd3ac")
            frame.pack(pady=5, fill="x", padx=10)
            ctk.CTkLabel(frame, text=label_text,text_color='#664b37',fg_color="#ffd3ac").pack(side="left", padx=5)
            entry = ctk.CTkEntry(frame, height=20,text_color='#664b37',fg_color="#ffd3ac")
            #entry.configure(ttk.highlightthickness==2, ttk.highlightcolor=="red", ttk.bg=="lightblue"    )
            entry.pack(side="right", expand=True, fill="x")
                    if previous_data and label_text in previous_data:
                entry.insert(0, previous_data[label_text])
            self.entries[label_text] = (entry, regex)
        self.forme_var = tk.StringVar(value="Квадратний")
        forme_frame =ctk.CTkFrame(self.root)
        forme_frame.pack(pady=5, fill="x", padx=10)
        ctk.CTkLabel(forme_frame, text="Форма столу:",fg_color="#ffd3ac",text_color='#664b37').pack(side="left",
padx=5)
        forems = ["Квадратний", "Овальний", "Прямокутний"]
        for forem_text in forems:
            ctk.CTkRadioButton(forme_frame, text=forem_text, variable=self.forme_var,
value=forem_text,fg_color="#ffd3ac",text_color='#664b37').pack(side="left")
```

```
        ctk.CTkButton(self.root, text="Зарезервувати",
command=self.reserve_table,fg_color="#ffd3ac",text_color='#664b37').pack(pady=20)

    def reserve_table(self):
        global selected_table
        try:
            data = {}
            for label, (entry, regex) in self.entries.items():
                value = entry.get().strip()
                if not re.match(regex, value):
                    raise ValueError(f"Неправильний формат: {label}")
                data[label] = value
            data["Форма столу:"] = self.forme_var.get()
            selected_table = Table(
                width=int(data["Ширина столу (см):"]),
                length=int(data["Довжина столу (см):"]),
                forme=data["Форма столу:"],
                seats=int(data["Кількість місць:"]),
                price=0,
                start=data["Час початку (ГГ:ХХ):"],
                end=data["Час завершення (ГГ:ХХ):"],
                day=data["Дата бронювання (ДД.ММ.РРРР):"]
            )
            messagebox.showinfo("Успіх!", "Стіл успішно заброньовано!")
            self.root.destroy()
            root = tk.Tk()
            MenuApp(root, data, self.previous_orders, self.contact_data)  # Передаємо `previous_orders`
            root.mainloop()
        except ValueError as e:
            messagebox.showerror("Помилка!", str(e))
class MenuApp:
    def __init__(self, root, previous_data, orders, contact_data=None):
        self.previous_data = previous_data
        self.orders = orders
```

```
self.contact_data = contact_data if contact_data else {}
self.root = root
self.root.title("Меню")
self.root.geometry("900x700")
self.root.configure(bg="#D2B48C")
ctk.CTkButton(self.root, text="Назад", command=self.go_back).pack(pady=5, anchor="w")
self.main_frame = tk.Frame(root, bg="#D2B48C")
self.main_frame.pack(fill="both", expand=True)

self.menu_frame = tk.Frame(self.main_frame, bg="#D2B48C")
self.menu_frame.pack(side="left", fill="both", expand=True)
self.order_frame = tk.Frame(self.main_frame, bg="#8B4513", width=300)
self.order_frame.pack(side="right", fill="y", padx=5, pady=5)
self.notebook = ttk.Notebook(self.menu_frame)
self.notebook.pack(fill="both", expand=True)
self.sections = {
   "Холодні закуски та салати": [
      ("Закуска 'Сирний сомельє'", "285 грн.", "200", "chees.png"),
      ("Палітра м'ясних делікатесів", "325 грн.", "250", "meat.png"),
      ("Оселендець з цибулею", "129 грн.", "180", "fish_oni.png"),
            ],
   "Салати": [
      ("Цезар з куркою", "195 грн.", "230", "cesar.png"),
      ("Грецький салат", "160 грн.", "200", "greeks.png"),
      ("Олів'є", "175 грн.", "250", "olive.png"),
   ],
   "Супи": [
      ("Борщ український", "190 грн.", "350", "borch.png"),
      ("Крем-суп з грибами", "220 грн.", "300", "mushroom.png"),
   ],
   "Основні страви": [
      ("Куряче філе під сиром", "250 грн.", "400", "checkenches.png"),
      ("Стейк з телятини", "450 грн.", "500", "steik.png"),
```

```
        ],
        "Алкоголь": [
            ("Біле вино", "250 грн.", "400", "wwine.png"),
            ("Червоне вино", "450 грн.", "500", "rwine.png"),
        ],
    }

    self.images = {}
    self.entries= {}
    for section in self.sections:
        self.create_tab(section, self.sections[section])
    self.create_order_summary()
    self.update_total_order()
def go_back(self):
    self.root.destroy()
    root = tk.Tk()
    TableReservationApp(root, self.previous_data, self.orders, self.contact_data)
    root.mainloop()
def go_to_cart(self):
    self.root.destroy()
    root = tk.Tk()
    CartApp(root, self.previous_data, self.orders, self.contact_data)
    root.mainloop()
def create_tab(self, section_name, menu_items):
    tab = tk.Frame(self.notebook, bg="#D2B48C")
    self.notebook.add(tab, text=section_name)
    canvas = tk.Canvas(tab, bg="#D2B48C")
    scrollbar = tk.Scrollbar(tab, orient="vertical", command=canvas.yview)
    scrollable_frame = tk.Frame(canvas, bg="#D2B48C")
    scrollable_frame.bind("<Configure>", lambda e: canvas.configure(scrollregion=canvas.bbox("all")))
    canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
    canvas.configure(yscrollcommand=scrollbar.set)
    canvas.pack(side="left", fill="both", expand=True)
```

```
        scrollbar.pack(side="right", fill="y")
        for name, price, weight, img_path in menu_items:
            frame = tk.Frame(scrollable_frame, bg="#D2B48C")
            frame.pack(fill="x", padx=5, pady=2)
            img = self.load_image(img_path, size=(50, 50))
            img_label = tk.Label(frame, image=img, bg="#D2B48C")
            img_label.image = img
            img_label.pack(side="left", padx=5)
            text_label = tk.Label(frame, text=f"{name} ({weight} г) - {price}", font=("Arial", 12), bg="#D2B48C")
            text_label.pack(side="left", fill="x", expand=True)
            entry = tk.Entry(frame, width=5)
            quantity = self.orders.get(name, (0, img_path))[0]
            entry.insert(0, str(quantity))
            entry.pack(side="right", padx=5)
            entry.bind("<KeyRelease>", lambda event, dish=name, ent=entry, img=img_path: self.update_order(dish, ent,
img))
            self.entries[name] = entry
    def update_order(self, dish, entry, img_path):
        entry_value = entry.get()
        if not entry_value.isdigit():
            entry.delete(0, tk.END)
            entry.insert(0, "0")
            return
        count = min(max(int(entry_value), 0), 150)
        entry.delete(0, tk.END)
        entry.insert(0, str(count))
        self.orders[dish] = (count, img_path)
        self.update_total_order()

    def update_total_order(self):
        order_summary = "Загальне замовлення:\n"
        for dish, (quantity, _) in self.orders.items():
            if quantity > 0:
```

```
            order_summary += f"{dish}: {quantity} шт.\n"
        self.total_label.config(text=order_summary)
    def create_order_summary(self):
        self.total_label = tk.Label(self.order_frame, text="Загальне замовлення:\n", font=("Arial", 14, "bold"), bg="#8B4513", fg="white", justify="left")
        self.total_label.pack(padx=10, pady=10, anchor="nw")
        self.order_button = ttk.Button(self.order_frame, text="Замовити", command=self.go_to_cart)
        self.order_button.pack(side="bottom", pady=10)
    def load_image(self, path, size=(50, 50)):
        try:
            image = Image.open(path)
            image = image.resize(size, Image.LANCZOS)
            return ImageTk.PhotoImage(image)
        except:
            return None
class CartApp:
    def __init__(self, root, previous_data, orders, contact_data=None):
        self.previous_data = previous_data
        self.orders = orders
        self.contact_data = contact_data if contact_data else {}
        self.root = root
        self.root.title("Кошик")
        self.root.geometry("500x600")
        self.root.configure(background="#a4c639")
        ctk.CTkButton(self.root, text="Назад", command=self.go_back).pack(pady=5, anchor="w")
        ctk.CTkLabel(self.root, text="Ваше замовлення", font=("Arial", 12, "bold")).pack(pady=10)
        # Створюємо фрейм з canvas для скролу
        container = ctk.CTkFrame(self.root, height=300)
        container.pack(pady=5, fill="x")
        container.pack_propagate(False)
        canvas = tk.Canvas(container, height=300, background="#a4c639")
        scrollbar = ttk.Scrollbar(container, orient="vertical", command=canvas.yview)
        scrollable_frame = ttk.Frame(canvas)
```

```
scrollable_frame.bind(
    "<Configure>",
    lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
)
canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
canvas.configure(yscrollcommand=scrollbar.set)
canvas.pack(side="left", fill="both", expand=True)
scrollbar.pack(side="right", fill="y")
for dish, (quantity, img_path) in self.orders.items():
    if quantity > 0:
        frame = ttk.Frame(scrollable_frame)
        frame.pack(pady=5, fill="x")
        try:
            img = Image.open(img_path)
            img = img.resize((50, 50), Image.LANCZOS)
            photo = ImageTk.PhotoImage(img)
            label_img = tk.Label(frame, image=photo)
            label_img.image = photo
            label_img.pack(side="left", padx=5)
        except:
            pass
        ctk.CTkLabel(frame, text=f"{dish}: {quantity} шт.",bg_color="#a4c639").pack(side="left", padx=10)
ctk.CTkLabel(self.root, text="Контактні дані", font=("Arial", 12, "bold")).pack(pady=10)

self.contact_entries = {}
contact_labels = {
    "Ім'я:": r"^.+$",
    "Телефон:": r"^\+?\d{10,15}$",
    "Email:": r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
}
for label, regex in contact_labels.items():
    frame = ctk.CTkFrame(self.root)
    frame.pack(pady=5, fill="x", padx=10)
```

```
            ctk.CTkLabel(frame, text=label).pack(side="left", padx=5)
            entry = ctk.CTkEntry(frame)
            entry.pack(side="right", expand=True, fill="x")
            if label in self.contact_data:
                entry.insert(0, self.contact_data[label])
            self.contact_entries[label] = (entry, regex)
        ctk.CTkButton(self.root, text="Підтвердити замовлення", command=self.confirm_order).pack(pady=10)
    def go_back(self):
        contact_data = {label: entry.get().strip() for label, (entry, regex) in self.contact_entries.items()}
        self.root.destroy()
        root = tk.Tk()
        MenuApp(root, self.previous_data, self.orders, contact_data)
        root.mainloop()
    def confirm_order(self):
        try:
            contact_info = {label: entry.get().strip() for label, (entry, regex) in self.contact_entries.items()}
            for label, regex in self.contact_entries.items():
                if not re.match(regex[1], contact_info[label]):
                    raise ValueError(f"Неправильний формат: {label}")
            order_details = {
                "Дата": selected_table.day if selected_table else "Невідомо",
                "Час": f"{selected_table.start} - {selected_table.end}" if selected_table else "Невідомо",
                "Форма столу": selected_table.forme if selected_table else "Невідомо",
                "Місця": selected_table.seats if selected_table else "Невідомо",
                "Ім'я": contact_info["Ім'я:"],
                "Телефон": contact_info["Телефон:"],
                "Email": contact_info["Email:"],
                "Страви": "\n".join([f"{dish}: {qty} шт." for dish, (qty, _) in self.orders.items() if qty > 0])
            }
            send_email(order_details)
            messagebox.showinfo("Успіх", "Ваше замовлення прийнято та відправлено на пошту!")
            self.root.destroy()
        except ValueError as e:
```

```
            messagebox.showerror("Помилка!", str(e))
    def send_email(order_details):
        sender_email = "uriyghritskiv1@gmail.com"
        sender_password = "oird ptih hndq zeym"
        receiver_email = "uriyghritskiv1@gmail.com"
        subject = "Нове замовлення з програми бронювання столу"
            body = f"""
        Ви отримали нове замовлення:
        Дата бронювання: {order_details['Дата']}
        Час: {order_details['Час']}
        Форма столу: {order_details['Форма столу']}
        Місця: {order_details['Місця']}
            --- Контактні дані ---
        Ім'я: {order_details["Ім'я"]}
        Телефон: {order_details['Телефон']}
        Email: {order_details['Email']}
        --- Страви ---
        {order_details['Страви']}
        """
        msg = EmailMessage()
        msg.set_content(body)
        msg["Subject"] = subject
        msg["From"] = sender_email
        msg["To"] = receiver_email

        try:
            with smtplib.SMTP_SSL("smtp.gmail.com", 465) as server:
                server.login(sender_email, sender_password)
                server.send_message(msg)
            print("Лист успішно надіслано!")
        except Exception as e:
            print(f"Помилка під час надсилання листа: {e}")
    if __name__ == "__main__":
```

```
root = tk.Tk()

app = TableReservationApp(root)

root.mainloop()
```