Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях

Виконав: студент 4 курсу
Групи 5ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Швець В. В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Чехместрук Р. Ю.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Дудник О. В.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. О. Н. Романюк
(ініціали та прізвище)
«09» червня 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Швецю Василю Васильовичу

1. Тема роботи – «Розробка програмного комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях».

Керівник роботи: Чехместрук Роман Юрійович, к.т.н., доцент, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: середовище розробки – Visual studio, мова програмування – Python.

4. Зміст текстової частини: вступ; аналіз стану технологій; порівняльний аналіз аналогів; аналіз методів розв'язання задачі; постановка задачі; розробка алгоритму дії програмного застосунку; розробка модуля аналітики статистики найпопулярніших позицій; розробка модуля для виводу графіку активності; тестування програмного застосунку; висновки; список використанних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів, використані технології; алгоритм створення діаграми популярності позицій, тестування,

висновки, апробація результатів роботи і публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Чехместрук Р. Ю., к.т.н., доцент кафедри ПЗ	24.03.2025	01.06.2025

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз підходів до інтелектуального моніторингу клієнтського потоку в кав'ярнях	25.03.2025 – 30.03.2025	вик
2	Розробка моделі та алгоритмів програмного застосунку	30.03.2025 – 19.04.2025	вик
3	Розробка програмних модулів системи інтелектуального моніторингу клієнтського потоку в кав'ярнях	19.04.2025 – 10.05.2025	вик
4	Розробка програмного застосунку	10.05.2025 – 24.05.2025	вик
5	Тестування програмного застосунку	24.05.2025 – 27.05.2025	вик
6	Оформлення матеріалів до захисту БКР	27.05.2025 – 30.05.2025	вик

Студент _____ Швець В. В.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Чехместрук Р.Ю.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Швець В. В. Розробка програмного комплексу для інтелектуального моніторингу клієнтського потоку в кав’ярнях: бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025.

На укр. мові. Бібліогр. : 25 назв ; рис. : 22 ; табл. 2.

У бакалаврській кваліфікаційні роботі розроблено методи та програмні засоби комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях. Проведено аналіз сучасних підходів до інтелектуального моніторингу клієнтського потоку. Обгрунтовано необхідність розробки методів виконання інтелектуального моніторингу клієнтського потоку з метою підвищення точності вимірювання отриманих даних.

Розроблено методи вимірювання інформації з клієнтського потоку для визначення найпопулярніших пропозицій, отримання графіку активності клієнтів та засоби, що дозволяють отримувати необхідну для вимірювань інформацію з клієнтського потоку.

На основі проведених у бакалаврській кваліфікаційні роботті досліджень, розроблено алгоритми і програмні засоби для інтелектуального моніторингу клієнтського потоку.

Проведене тестування підтвердило достовірність теоретичних досліджень методів інтелектуального моніторингу клієнтського потоку та засобів їх реалізації. Розроблені засоби вимірювання отриманої інформації з клієнтського потоку можна використати в кав'ярнях для подальшого розвитку бізнесу.

ABSTRACT

UDC 004.42

Shvets V.V. Development of a software complex for intelligent monitoring of customer flow in coffee shops: bachelor's qualification work in specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2025.

In Ukrainian. Bibliography: 25 titles; Figures: 22; Table 2.

In the bachelor's thesis, methods and software tools for intelligent monitoring of customer flow in coffee shops were developed. An analysis of modern approaches to intelligent monitoring of customer flow is carried out. The necessity of developing methods for performing intelligent monitoring of the customer flow in order to increase the accuracy of measuring the data obtained is substantiated.

Methods for measuring information from the customer flow are developed to determine the most popular offers, obtain a graph of customer activity, and tools that allow obtaining the necessary information from the customer flow.

Based on the research conducted by the bachelor’s thesis, algorithms and software tools for intelligent monitoring of customer flow were developed.

The testing confirmed the reliability of theoretical studies of methods of intelligent monitoring of customer flow and means of their implementation. The developed tools for measuring the information obtained from the customer flow can be used in coffee shops for further business development.

ЗМІСТ

ВСТУП

Обгрунтування вибору теми дослідження

Сучасний етап розвитку інформаційних технологій вимагає постійного вдосконалення підходів до збору, аналізу та обробки даних, зокрема в сфері обслуговування. Однією з актуальних задач є створення систем, здатних точно та надійно відображати інформацію про потік клієнтів у закладах громадського харчування, зокрема в кав’ярнях [1].

Традиційні методи моніторингу клієнтської активності зазвичай базуються на візуальних підходах — встановлення камери, підрахунок людей за допомогою сенсорів або спостереження персоналу. Проте такі методи часто є обмеженими за точністю, потребують складного технічного обладнання і не завжди забезпечують повну відповідність реальній кількості клієнтів. У цьому контексті, використання даних про замовлення, створені клієнтами, відкриває нові можливості для об'єктивного та автоматизованого обліку відвідувачів без необхідності використання дорогих апаратних засобів.

Інформація про замовлення є точним і достовірним джерелом даних, що відображає не лише факт візиту клієнта, але й дозволяє аналізувати його поведінку, уподобання, частоту відвідувань та інші корисні показники. Такий підхід значно знижує навантаження на персонал, автоматизує процес збору статистики та відкриває нові горизонти для глибокого аналітичного аналізу діяльності закладу.

Висока точність отриманих аналітичних даних має критичне значення для ефективного управління бізнесом. Завдяки об'єктивному розумінню клієнтського потоку можливо своєчасно приймати управлінські рішення, прогнозувати попит на послуги, планувати графіки персоналу, розробляти акційні пропозиції або оновлення меню. Також, це забезпечує конкурентну перевагу та підвищує рівень обслуговування.

На сьогоднішній день більшість аналогічних рішень, які пропонують аналітику клієнтів для закладів харчування, або є вузькоспеціалізованими, або вимагають підключення до сторонніх сервісів та мають обмежену функціональність. Часто такі

системи є платними, з високою вартістю обслуговування, а також не дозволяють налаштувати аналіз саме під потреби конкретного закладу. Тому розробка власного програмного забезпечення для моніторингу клієнтського потоку на основі даних про замовлення є актуальним завданням, що дозволить створити ефективний, гнучкий та економічно вигідний інструмент для бізнесу, тому розробка зумовлена зростаючою потребою в точному, швидкому збіру інформації про клієнтський потік. Традиційні методи вимірювання є трудомісткими, залежать від людського фактора і не завжди забезпечують високу точність.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є підвищення точності інтелектуального моніторингу клієнтського потоку у кав'ярнях за рахунок використання даних про замовлення, які створюються клієнтами під час відвідування закладу. Такий підхід дозволяє автоматизувати збір статистичної інформації без застосування камер або сенсорів, забезпечуючи високу точність та достовірність обліку. Проєкт передбачає розробку низки програмних модулів, які будуть об'єднані в єдину інформаційну систему. Серед основних модулів – модуль обробки даних про замовлення, модуль аналітики клієнтського потоку, модуль побудови графіків, а також модуль графічного інтерфейсу для взаємодії з користувачем.

Відповідно до поставленої мети в бакалаврській кваліфікаційні роботі потрібно вирішити такі **задачі**:

- аналіз предметної галузі;
- розробити алгоритм збору інформації про замовлення клієнтів, що дозволить автоматично фіксувати факти відвідування без потреби у візуальному спостереженні;
- створенити модуль обробки замовлень з можливістю аналізу кількості клієнтів за певні часові інтервали;
- реалізувати алгоритм оцінки популярності послуг на основі частоти замовлень конкретних позицій з меню;

- створити модуль графічного інтерфейсу користувача, орієнтованого на працівників адміністрації, з інтуїтивно зрозумілим дизайном;

- тестування розроблених програмних продуктів;

Об'єкт дослідження – процес вимірювання інтелектуального моніторингу клієнтського потоку у кав'ярнях на основі даних про замовлення.

Предмет дослідження – методи та програмні засоби розробки застосунку для аналізу інформації про клієнтів, які дозволяють автоматизувати облік і покращити управління процесами обслуговування в кав’ярні.

Методи дослідження. У процесі дослідження застосовувалися: теорія інтелектуального моніторингу; теорія алгоритмів для обробки та аналізу даних із клієнтського потоку.

Наукова новизна отриманих результатів:

1. Подальшого розвитку отримав метод побудови системи для керування інтелектуальним потоком клієнтів, у якому, на відміну від існуючих використано технологію JSON, що дозволило спростити розробку та підвищити швидкість відтворення діаграмм на 0.765 мілісекунд.

2. Подальшого розвитку отримав метод визначення статистики відвідування кав’ярні клієнтським потоком для керування бізнес процесами кав’ярень у якому, на відміну від існуючих визначаються вибірка позицій у меню клієнтами.

Практична цінність отриманих результатів полягає в тому, що на основі теоретичних досліджень розроблено алгоритми та програми для інтелектуального моніторингу клієнтського потоку у кав’ярнях.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення БКР доповідались та обговорювались на Міжнародній науково-практичній конференції «Research in Science, Technology and Economics» у 2025 році.

Публікації. Основні результати досліджень опубліковано у матеріалах конференції «Research in Science, Technology and Economics» [2].

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто 4 розділи та було використано 25 літературних джерел.

1 АНАЛІЗ ПІДХОДІВ ДО ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ КЛІЄНТСЬКОГО ПОТОКУ У КАВ'ЯРНЯХ

1.1 Аналіз стану технологій інтелектуального моніторингу клієнтського потоку

Сучасні інформаційні технології відіграють важливу роль у сфері обслуговування, зокрема в закладах громадського харчування, таких як кав'ярні. Однією з актуальних тенденцій є впровадження систем моніторингу клієнтського потоку, які дозволяють ефективно керувати ресурсами, планувати завантаженість персоналу та підвищувати рівень обслуговування [3].

Одним із ключових аспектів розвитку цих технологій є зростання швидкості та потужності обчислювальних систем. Сучасні комп'ютери та обчислювальні методи дозволяють швидко та ефективно обробляти великі обсяги даних, що є необхідним для створення якісного та сучасного програмного забезпечення [4].

Завдяки технологіям комп'ютерного зору, сенсорним пристроям, Wi-Fi трекінгу, а також спеціалізованим програмним рішенням, адміністратори закладів можуть отримувати детальну інформацію про поведінку клієнтів: скільки людей відвідали заклад, коли спостерігається найбільший наплив, які послуги найбільш популярні, тощо [5].

Прогрес не стоїть на місці і постійно з'являються нові, більш зручні та швидкі методи моніторингу, більшість програмного забезпечення, встановленного в закладах занадто застаріле, і потребує оновлення. Окремою перевагою розробляємого застосунку буде використовування аналізу популярності послуг через дані замовлень.

Отже, аналіз стану технологій моніторингу клієнтського потоку в закладах приготування кавових напоїв підтверджує актуальність переходу до більш точних і ефективних методів збору та обробки інформації про поведінку відвідувачів. Розробка програмних засобів для автоматизованого відстеження клієнтської активності та аналізу популярності послуг стає необхідною умовою для оптимізації управлінських рішень у сфері обслуговування. Використання напрацювань та ресурсів провідних компаній, що спеціалізуються на розробці інформаційних систем, дозволить вирішувати складні завдання та створювати інноваційні рішення, які

покращують зручність використання та надають закладу конкурентні переваги. Зважаючи на постійний розвиток цифрових технологій у цій галузі, важливо стежити за новітніми досягненнями та тенденціями, щоб забезпечити релевантність і ефективність розроблених програмних продуктів.

1.2 Аналіз методів розв’язання задачі

Розробка програмних засобів для інтелектуального моніторингу клієнтського потоку вимагає комплексного підходу до вирішення завдань, пов’язаних зі збором, обробкою та візуалізацією даних про поведінку клієнтів у закладах обслуговування. Існують різні методи реалізації таких систем, кожен з яких має свої переваги та недоліки. У цьому розділі розглянуто найбільш ефективні підходи до створення програмних модулів збору та аналізу клієнтських даних.

Одним із найпоширеніших методів реалізації систем моніторингу клієнтів є використання відеоспостереження з вбудованими алгоритмами комп’ютерного зору. За допомогою камер та датчиків руху система може фіксувати переміщення клієнтів і здійснювати базовий підрахунок відвідувачів. Проте такий підхід має певні недоліки: по-перше, він вимагає значних апаратних ресурсів і складної технічної інфраструктури; по-друге, точність виявлення іноді залишається низькою, особливо в умовах змінного освітлення чи великого скупчення людей; по-третє, цей спосіб не дозволяє визначити, що саме замовили клієнти і це робить неможливим усі підрахунки для статистики замовлень.

Отже, такий метод не завжди є ефективним у реальних умовах роботи кав’ярні.

Інший підхід базується на застосуванні технологій штучного інтелекту, зокрема нейронних мереж, для аналізу поведінки клієнтів. Така система може визначати не лише кількість відвідувачів, а й фіксувати час їх перебування, частоту візитів, а також оцінювати популярність певних зон обслуговування. Одним із прикладів такого підходу є використання моделей на кшталт YOLO або MediaPipe, які забезпечують високу точність виявлення об'єктів у кадрі в реальному часі. Але попри ефективність, ці алгоритми потребують попереднього навчання на великому обсязі даних, а також налаштування під специфіку конкретного закладу.

Найефективнішим підходом є використання методів збору інформації про замовлення. Цей метод дозволяє точно визначити і отримати інформацію про популярність певних послуг та кількістю часу проведенного у закладі клієнтом, до того ж цей спосіб дозволяє записати клієнта у базу даних кав'ярні і в залежності від частоти візитів система сама буде підбирати замовлення для клієнту. Такий підхід гарантує високу точність та стандартизацію результатів і не потребує зайвих витрат для підрахунку та нагрузок для системи.

Розглянувши переваги та недоліки кожного методу, було вирішено використовувати метод збору інформації про замовлення клієнта з закладу. Оскільки даний метод не потребує великої трати ресурсів і має найширший функціонал з усіх запропонованих, тому саме він є найретельнішим для реалізації у програмному застосунку.

1.3 Порівняльний аналіз аналогів

Використання технологій для моніторингу клієнтських потоків набуває все більшого поширення в умовах цифрової трансформації сучасного бізнесу. Такі системи активно впроваджуються в різноманітні галузі, зокрема у сферу торгівлі, громадського харчування, банківські сервіси та сервісні підприємства. Моніторинг клієнтів дозволяє не лише підвищити якість обслуговування, а й оптимізувати бізнес-процеси, скоротити витрати, а також приймати обґрунтовані управлінські рішення на основі зібраних статистичних даних.

Розробка програмних засобів для ефективного моніторингу потоку клієнтів із застосуванням RFID-технологій є сучасним та перспективним напрямом. Використання радіочастотної ідентифікації дає змогу отримувати точні дані в режимі реального часу, автоматизувати процеси і значно знизити вплив людського фактора. Такі системи підвищують точність аналізу поведінки клієнтів, дозволяють ефективно регулювати навантаження на персонал і ресурси закладу.

Серед найбільш відомих та ефективних програмних рішень у сфері моніторингу клієнтського потоку можна виділити такі продукти:

- RetailNext;

- Poster POS;
- POS Sector;
- Jsolutions;
- PayKit.

Одним із прикладів таких програмних засобів є RetailNext, інтерфейс якого зображено на рисунку 1.1. Це сучасна платформа, розроблена компанією RetailNext, яка дозволяє не лише здійснювати моніторинг потоку клієнтів, а й інтегрувати фінансові операції, керування складом, створення звітів та візуалізацію аналітичних даних. Система також має можливості адаптації під конкретні потреби бізнесу та використовується в багатьох міжнародних торгових мережах [6].

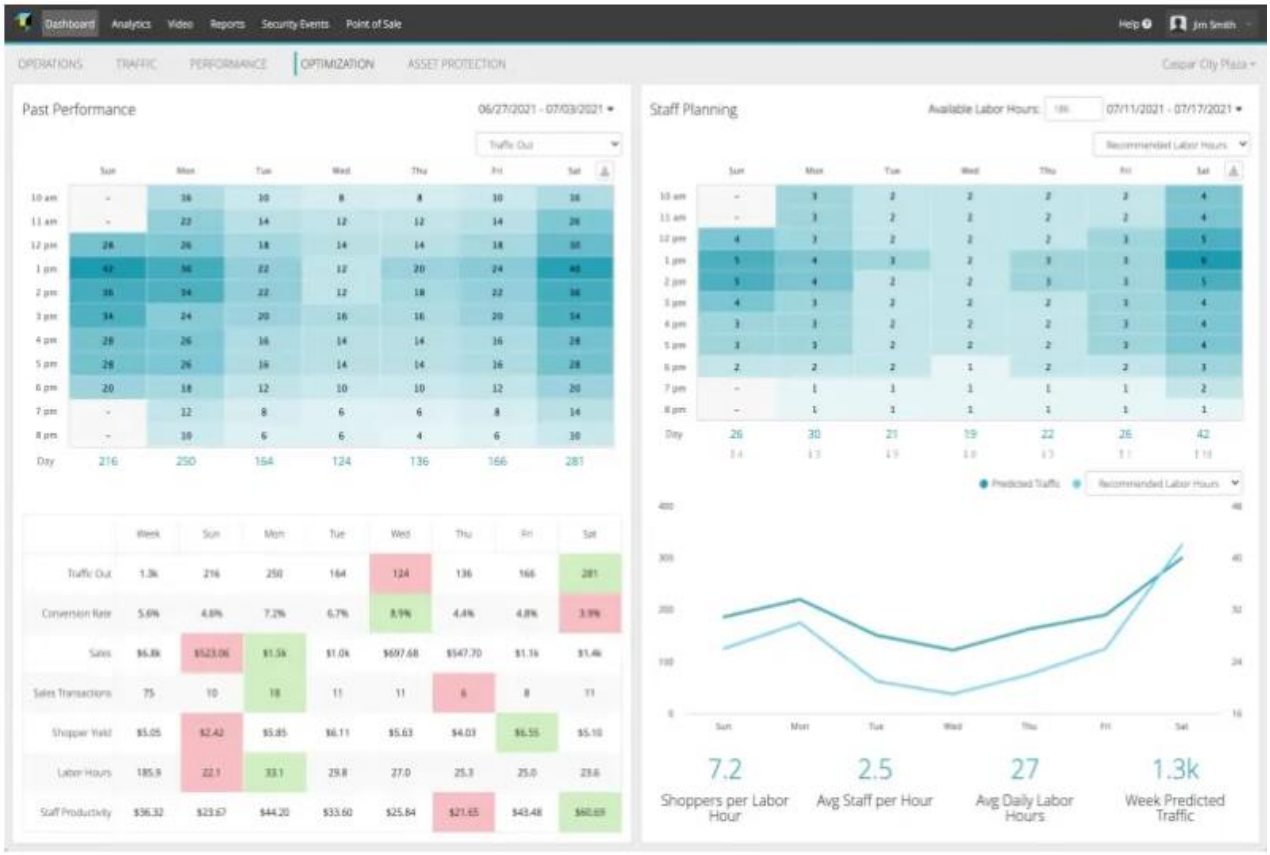


Рисунок 1.1 – Приклад роботи застосунку RetailNext

Іншим прикладом ефективного програмного забезпечення для моніторингу потоку клієнтів є застосунок Poster POS, інтерфейс якого зображено на рисунку 1.2. Цей застосунок виступає як потужний інструмент, що дозволяє не лише

відслідковувати кількість клієнтів у реальному часі, а й здійснювати детальний аналіз їхньої поведінки та вподобань. Poster POS забезпечує користувачам можливість створення гнучких графіків та діаграм, що дозволяють візуалізувати популярність певних послуг або товарів у залежності від різних демографічних характеристик клієнтів, таких як вік, стать, час відвідування та середній чек.

Крім того, застосунок підтримує функціонал створення звітів за будь-який період, що значно полегшує управління бізнесом. Це програмне забезпечення активно використовується в закладах громадського харчування, таких як кафе, ресторани, кав'ярні та фаст-фуди, де особливо важливо контролювати навантаження на персонал, швидко обробляти замовлення та аналізувати поведінку відвідувачів для подальшої оптимізації роботи закладу. Завдяки своїй інтуїтивно зрозумілій панелі керування Poster POS дозволяє власникам бізнесу оперативно приймати рішення, що базуються на актуальних і точних даних [7].

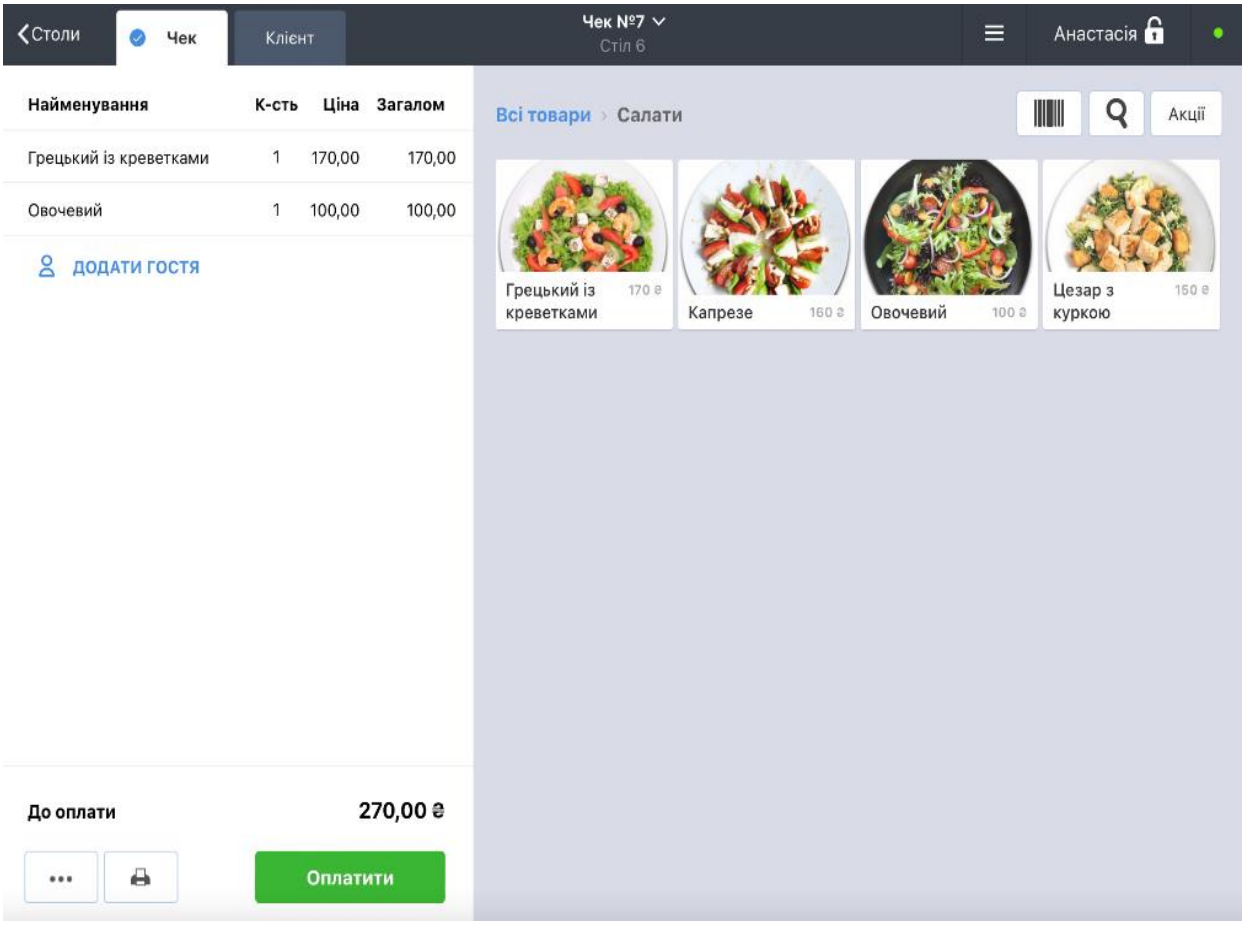


Рисунок 1.2 – Приклад роботи застосунку Poster POS

POS Sector є ще одним ефективним програмним продуктом, який був спеціально розроблений для автоматизації процесів обслуговування у сфері роздрібної торгівлі та закладів громадського харчування. Його інтерфейс можна переглянути на рисунку 1.3. Цей застосунок підтримує широкий спектр функцій, необхідних для щоденного управління діяльністю ресторанів, кафе, барів та магазинів. Система забезпечує просте і зрозуміле керування замовленнями, ведення обліку продажів, а також контроль складських залишків.

POS Sector має гнучке налаштування, що дозволяє адаптувати його як до потреб стандартного ресторану з офіціантами, столами та поділом замовлень за залами, так і до умов роботи фаст-фудів або кафе, де замовлення здійснюються безпосередньо біля каси. Крім того, програмне забезпечення дозволяє формувати докладні звіти про фінансову діяльність, аналізувати популярність страв, ефективність роботи персоналу та виявляти пікові години завантаженості.

Цей функціонал робить POS Sector зручним і вигідним рішенням для власників бізнесу, які прагнуть автоматизувати робочі процеси, зменшити людський фактор у прийнятті рішень і підвищити загальну продуктивність роботи закладу. Завдяки підтримці сучасних пристроїв, система також дозволяє інтеграцію з планшетами та мобільними терміналами, що забезпечує ще більшу мобільність у роботі персоналу [8].

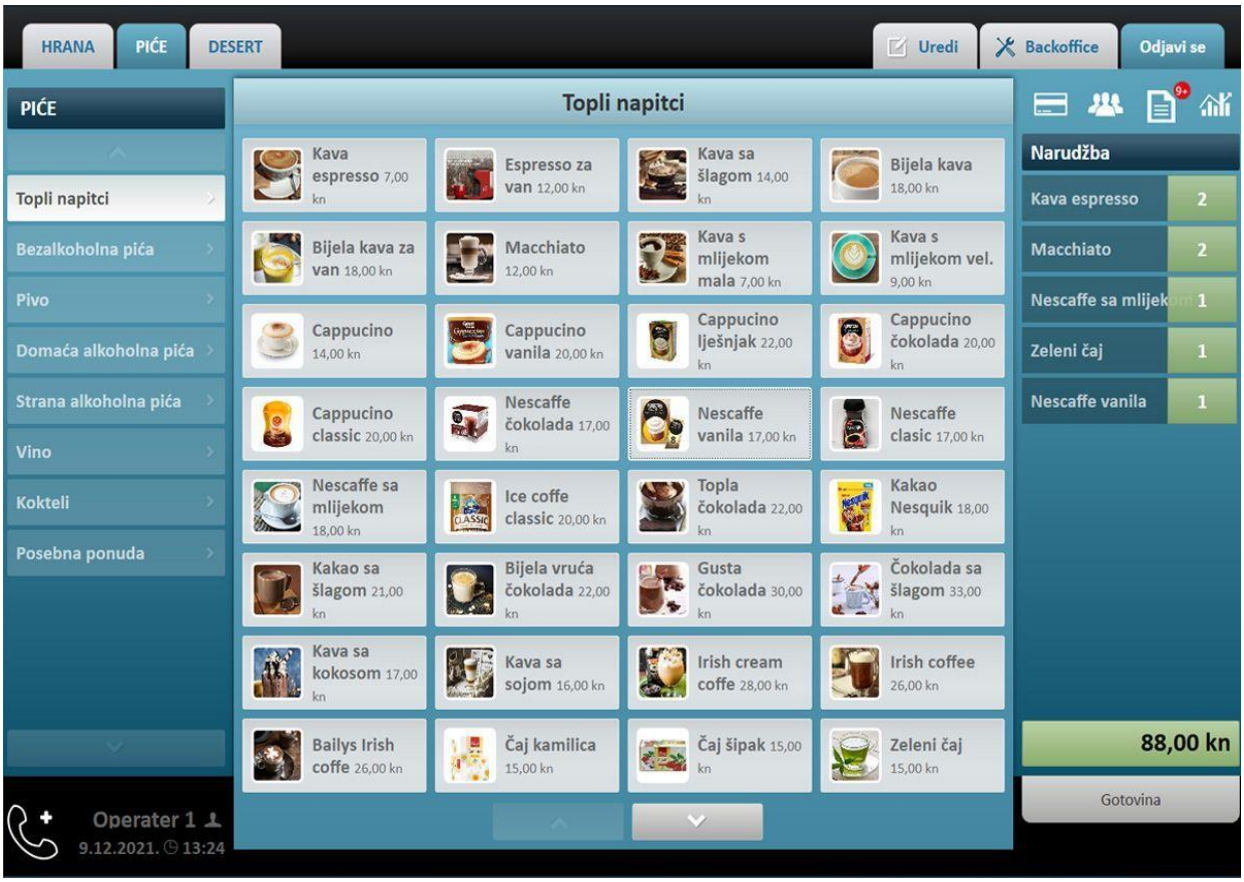


Рисунок 1.3 – Приклад роботи застосунку POS sector

Jsolutions, зображений на рисунку 1.4, є сучасним програмним засобом, що пропонує хмарну інфраструктуру для автоматизації управлінських та облікових процесів у підприємствах різного масштабу. Ця система орієнтована на забезпечення централізованого контролю за діяльністю закладу, дозволяючи керівникам оперативно приймати рішення, ґрунтуючись на актуальних даних, що зберігаються у хмарному середовищі. Однією з ключових переваг Jsolutions є її здатність до інтеграції з різноманітним торговим обладнанням — від касових апаратів до терміналів і сканерів [9].

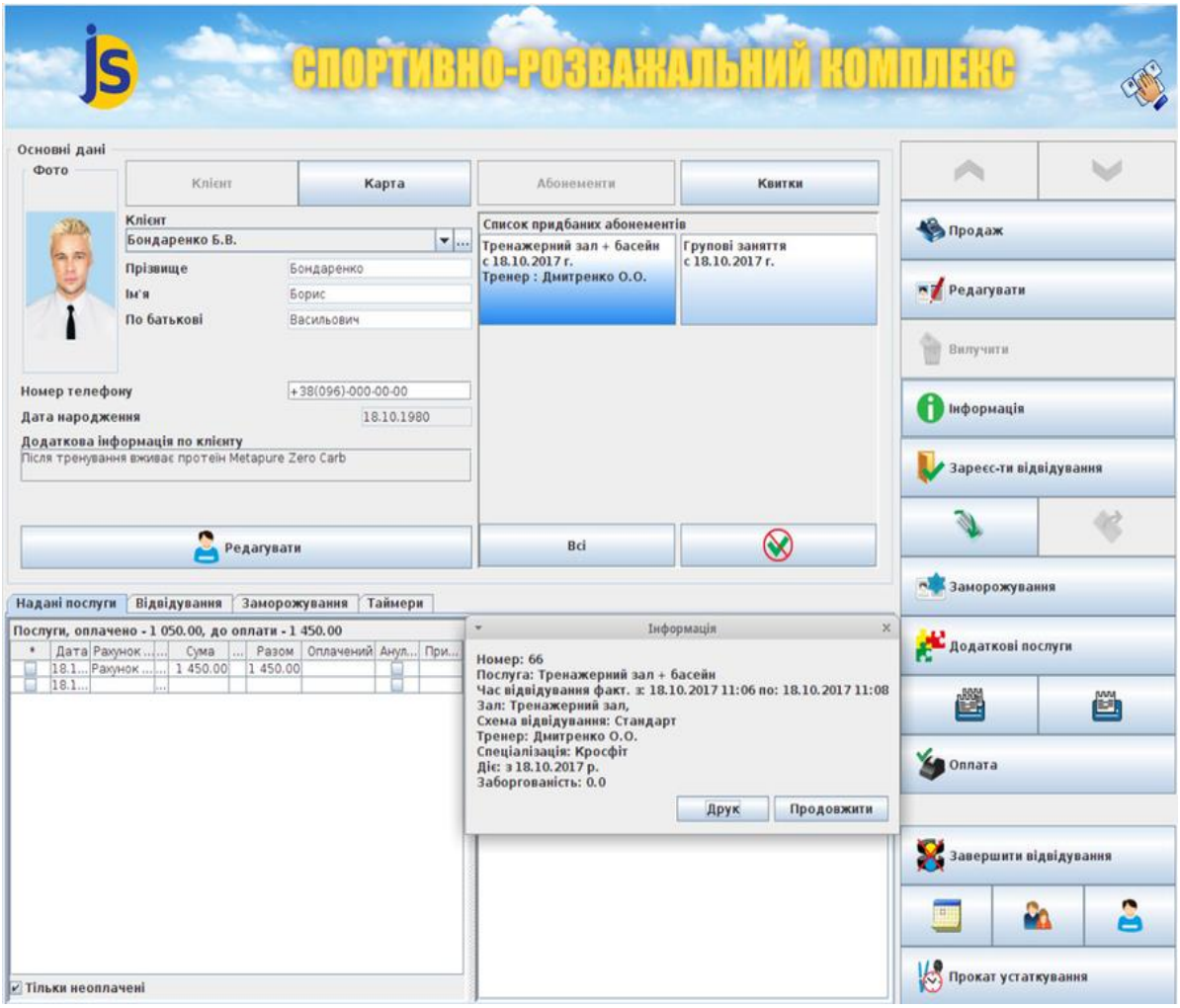


Рисунок 1.4 – Приклад роботи застосунку Jsolutions

PayKit є POS-додатком для каси, товарного та складського обліку, також у додаток була інтегрована система фіскального реєстратору, у наявності функціоналу

присутній електронний документообіг та еквайрінг для безготівкових платежів інтерфейс застосунку показаний на рисунку 1.5 [10].

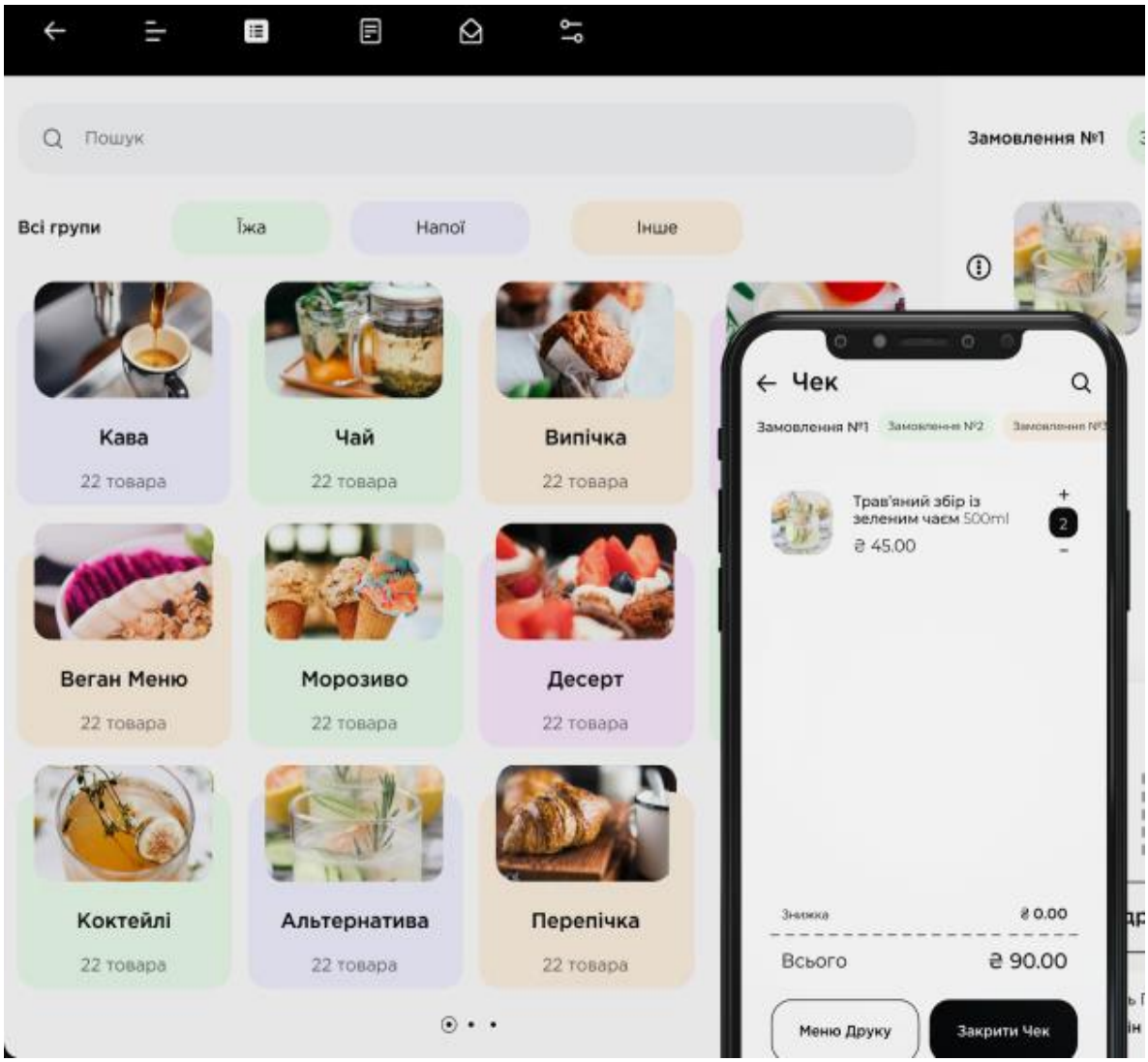


Рисунок 1.5 – Приклад роботи застосунку PayKit

Розробка програмних засобів моніторингу клієнтського потоку із використанням графічного модуля потребує досвіду у створенні програмного забезпечення, аналітики даних та проєктування інтерфейсу користувача. Це включає реалізацію внутрішньої системи, яка здатна збирати інформацію про кількість відвідувачів, аналізувати динаміку клієнтського потоку та визначати найпопулярніші послуги за певний період. Компонент користувацького інтерфейсу передбачає створення інтуїтивно зрозумілих панелей управління та візуалізацій, що забезпечать

зручну взаємодію адміністратора із розробленими модулями й аналітичними алгоритмами.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	RetailNext	Poster POS	POS Sector	JSolutions	PayKit	Власна розробка
Точність вимірювань	-	+	-	-	+	+
Аналіз відвідуваності клієнтів	+	+	+	+	+	+
Автоматизація вимірювання	-	-	+	+	+	+
Аналіз популярності замовлень	+	-	-	-	-	+
Швидкість та зручність вимірювання	-	+	-	+	+	+
Загальна оцінка	40%	60%	40%	60%	80%	100%

Відповідно до таблиці порівняльних характеристик розробка власних програмних засобів для моніторингу потоку клієнтів з аналізом популярності замовлень є доцільною. Отриманий продукт зможе покрити недоліки існуючих рішень та забезпечити розширені можливості.

Отже, розробка програмних засобів для моніторингу клієнтського потоку в закладах приготування кавових напоїв відкриває широкі можливості для вдосконалення управлінських процесів, покращення сервісу та оптимізації роботи

персоналу. Завдяки точному збору та аналізу даних про поведінку клієнтів, адміністрація може приймати більш обґрунтовані рішення щодо організації обслуговування, планування робочого часу та популяризації послуг, що сприяє підвищенню загальної ефективності роботи закладу.

1.4 Постановка задачі дослідження

Проаналізувавши переваги та недоліки існуючих систем моніторингу клієнтського потоку, було визначено основні завдання, необхідні для успішної розробки програмного забезпечення, яке не базується на візуальному підрахунку клієнтів, а використовує дані про замовлення, що створюються клієнтами. Такий підхід дозволяє забезпечити більш точний та об’єктивний аналіз активності відвідувачів у кав'ярні:

- розробити алгоритм збору інформації про замовлення клієнтів, що дозволить автоматично фіксувати факти відвідування без потреби у візуальному спостереженні;
- створити модуль обробки замовлень з можливістю аналізу кількості клієнтів за певні часові інтервали;
- реалізувати алгоритм оцінки популярності послуг на основі частоти замовлень конкретних позицій з меню;
- створити модуль графічного інтерфейсу користувача, орієнтованого на працівників адміністрації, з інтуїтивно зрозумілим дизайном;
- провести тестування програмного забезпечення згідно поставлених задач.

1.5 Висновки

У першому розділі було розглянуто стан програмних засобів для моніторингу потоку клієнтів. Були розглянуті програмні засоби такі як RetailNext, Poster POS, POS Sector, Jsolutions, Paykit.

Проаналізувавши переваги та недоліки існуючих програмних засобів, було сформульовано завдання для подальшої розробки власного програмного забезпечення моніторигу потоку клієнтів. Ці завдання включають розробку алгоритмів аналізу популярних послуг через дані замовлень, розробку графічного

інтерфейсу. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки власних програмних засобів.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1 Обгрунтування вибору мови програмування

Java — це високорівнева, об’єктно-орієнтована мова програмування, яка була створена компанією Sun Microsystems у 1995 році (згодом придбаною Oracle). Основною ідеєю при розробці мови було досягнення кросплатформеності, тобто можливості запуску програм на будь-якій операційній системі без необхідності перекомпіляції. Цього вдалося досягти завдяки віртуальній машині Java (JVM), яка дозволяє виконувати зкомпільований байт-код незалежно від апаратної платформи чи ОС.

Після компіляції вихідний код Java перетворюється на байт-код, який не є машинним кодом конкретного процесора. Замість цього він виконується JVM, яка інтерпретує або компілює байт-код у машинний код під час виконання. Такий підхід забезпечує високу безпеку, ефективну роботу з пам’яттю та стабільність виконання програм. Java займає важливе місце в екосистемі програмного забезпечення. Її активно використовують для розробки корпоративних систем, мобільного програмного забезпечення (особливо для Android), ігрових застосунків та вбудованих рішень.

Серед найвідоміших фреймворків для Java-розробки — Spring, Jakarta EE (колишня Java EE), а також Play Framework. Вони забезпечують розробників широким набором інструментів та бібліотек для створення масштабованих, надійних та ефективних рішень. Попри численні переваги, мова іноді піддається критиці за надмірну багатослівність синтаксису та повільний запуск програм, що пов’язано з додатковим рівнем абстракції — JVM. Проте завдяки постійним оновленням і оптимізаціям, Java залишається актуальною і конкурентоспроможною.

На сьогоднішній день Java зберігає статус однієї з найпопулярніших мов програмування у світі. Особливо вона затребувана в корпоративному секторі, де важливі стабільність, масштабованість і довготривала підтримка рішень Java зберігає статус однієї з найпопулярніших мов програмування у світі.

Архітектура Java наведена на рисунку 2.1.

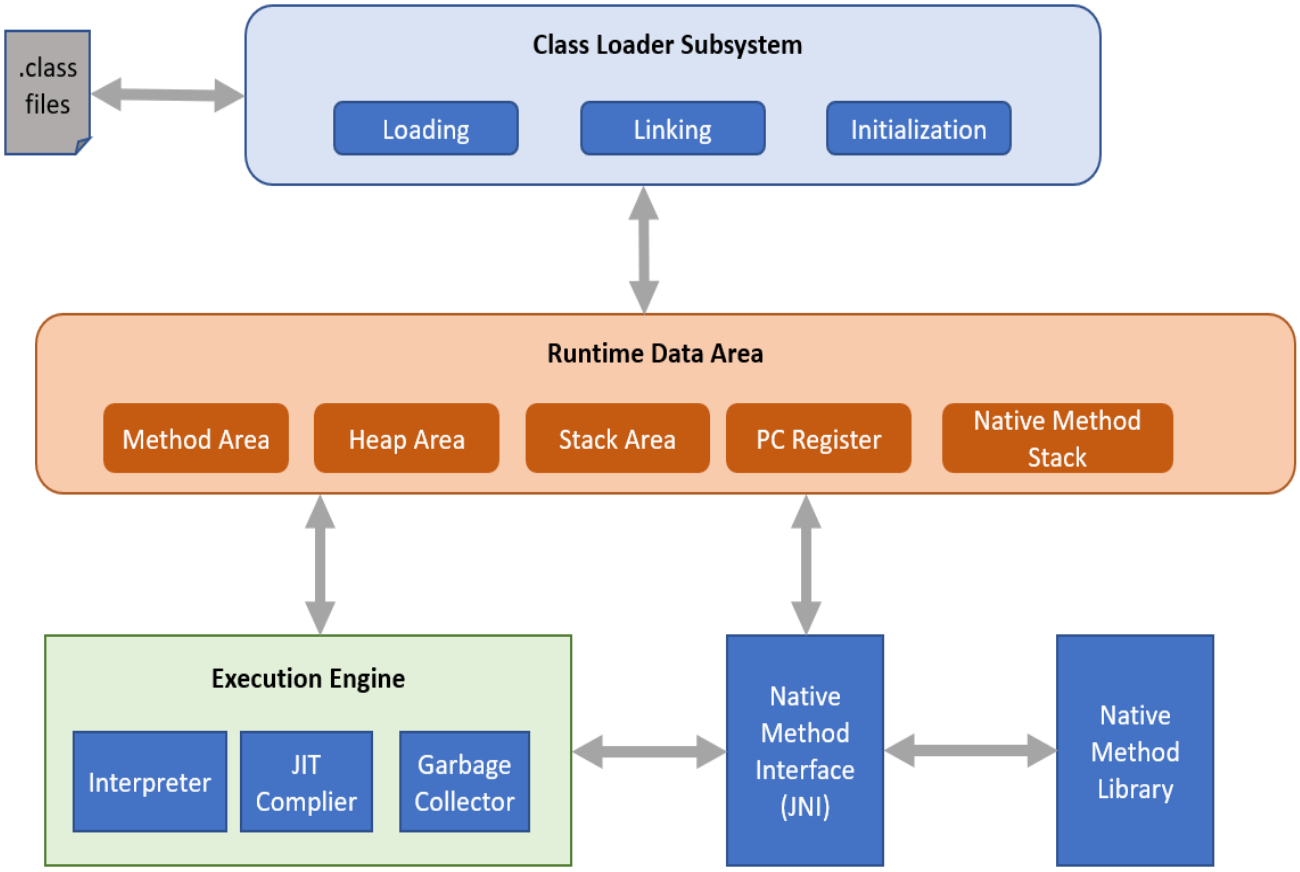


Рисунок 2.1 – Архітектура Java

C++ — це високорівнева, компільована мова програмування з підтримкою об’єктно-орієнтованої парадигми, створена Б’ярне Страуструпом у 1980-х роках як розширення мови C. Вона поєднує ефективність низькорівневого програмування з можливостями високорівневої абстракції, що робить її надзвичайно потужним інструментом для розробки продуктивного програмного забезпечення. Однією з основних особливостей C++ є підтримка множинних парадигм програмування: процедурного, об’єктно-орієнтованого, функціонального, а з виходом стандарту C++11 і пізніших — навіть узагальненого програмування (generic programming). Це дозволяє програмістам обирати найбільш зручний стиль написання коду в залежності від завдання. C++ відомий своєю високою продуктивністю, точним контролем над ресурсами (особливо пам’яттю), а також можливістю прямої роботи з апаратурою.

Завдяки цьому вона широко використовується в системному програмуванні, розробці операційних систем, ігрових рушіїв, вбудованих систем, баз даних, фінансових інструментів та іншого високопродуктивного програмного забезпечення. Водночас, C++ часто вважається складнішою у вивченні, особливо для новачків, через більш складний синтаксис, необхідність управління пам’яттю вручну та високу ймовірність помилок, таких як витоки пам’яті або помилки доступу. Незважаючи на появу нових мов програмування, C++ залишається актуальним та незамінним у багатьох критичних сферах, де важливі надійність, швидкість та ефективність. Завдяки постійному розвитку (стандарти C++11, C++14, C++17, C++20 і новіші), ця мова продовжує відповідати вимогам сучасного програмування.

Архітектура C++ наведена на рисунку 2.2.

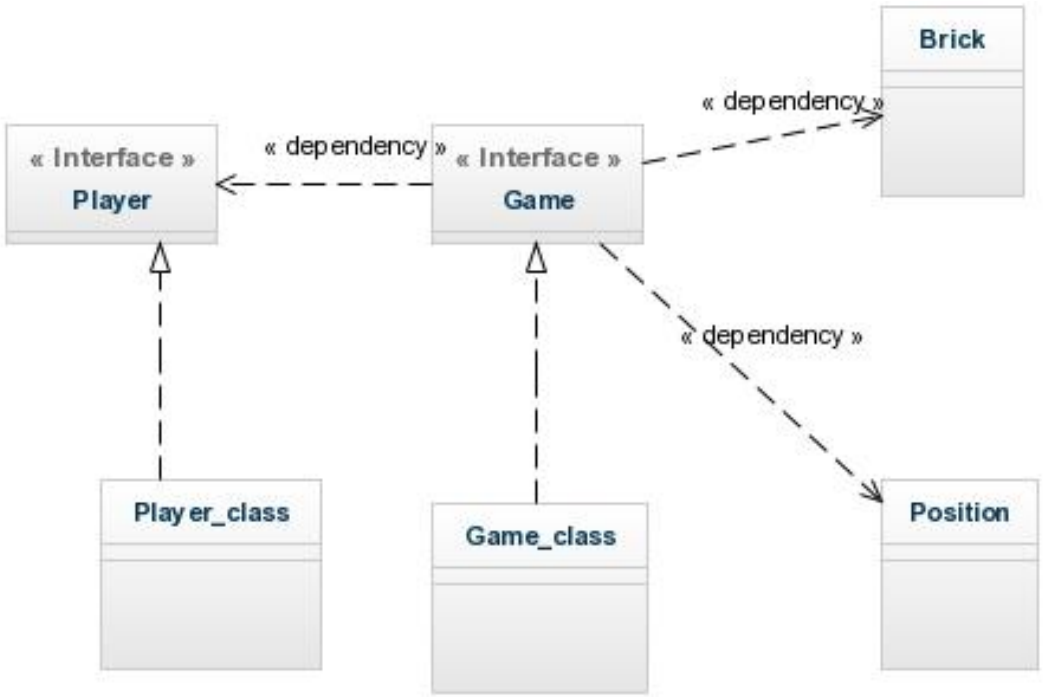


Рисунок 2.2 – Архітектура C++

Python — це високорівнева, інтерпретована мова програмування з підтримкою об’єктно-орієнтованої парадигми, яку створив Гвідо ван Россум у 1991 році. Головною перевагою мови є її лаконічний, інтуїтивно зрозумілий синтаксис, що

робить Python ідеальним вибором для початківців і водночас ефективним інструментом для досвідчених розробників. Python належить до мультипарадигмальних мов, тобто підтримує процедурний, функціональний та об’єктно-орієнтований стиль програмування. Вона забезпечує динамічну типізацію, автоматичне управління пам’яттю та містить велику кількість вбудованих модулів, які дозволяють розв’язувати широкий спектр задач без необхідності використання сторонніх бібліотек.

Ця мова широко використовується в різних сферах: від веб-розробки і аналізу даних до машинного навчання, штучного інтелекту, наукових досліджень, автоматизації процесів, а також у сфері освіти.

Попри всі переваги, Python не завжди є найкращим вибором для задач, які потребують високої продуктивності чи роботи з ресурсомісткими обчисленнями — у таких випадках перевагу часто надають C++, Java або Rust.

Втім, Python продовжує стрімко зростати в популярності у всьому світі. Завдяки універсальності, активній спільноті, постійному оновленню та підтримці з боку великих компаній, він залишається однією з найбільш затребуваних мов програмування як у сфері бізнесу, так і в академічному середовищі та стартапах.

Архітектура Python наведена на рисунку 2.3.

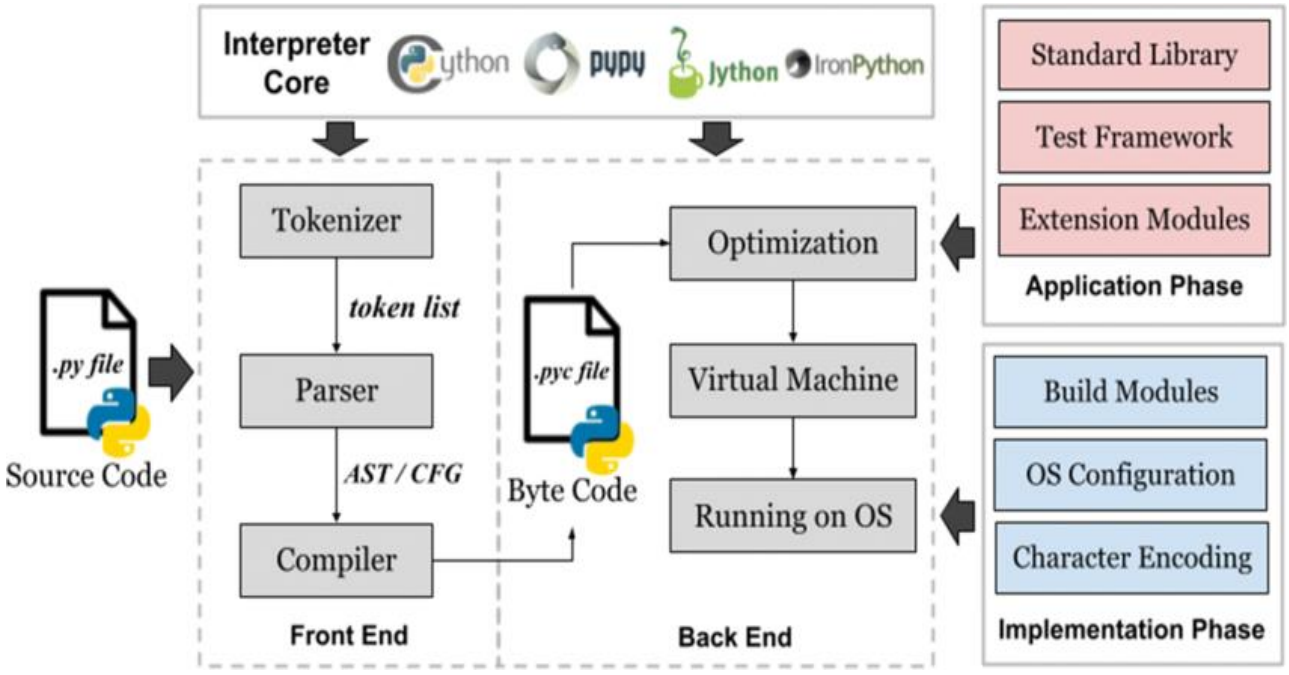


Рисунок 2.3 – Архітектура Python

Для реалізації програмного продукту для моніторингу потоку клієнтів у закладах приготування кавових напоїв буде використана проста і популярна мова програмування Python, а також будуть використані бібліотеки Tkinter, Matplotlib та Pandas. Також у створенні застосунку буде використовуватись текстовий формат зберігання даних JSON.

Python – це інтерпретована, високорівнева мова програмування з простим синтаксисом, що використовується для розробки застосунків для тривимірного моделювання, штучного інтелекту та інших задач [11].

Бібліотека Tkinter — це стандартний інтерфейс для побудови графічного інтерфейсу користувача (GUI) в мові програмування Python. Вона є обгорткою над бібліотекою Tk, яка спочатку була розроблена для мови Tcl, але з часом набула широкої популярності й у Python. Бібліотека використовується для створення десктопних додатків з графічним інтерфейсом. Це зручно, коли потрібно створити інтерфейс для взаємодії з користувачем [12].

Бібліотека Matplotlib — це потужний інструмент для побудови графіків та візуалізації даних у Python. Вона є однією з найпопулярніших бібліотек для створення 2D-графіки і часто використовується в наукових дослідженнях, аналізі даних, машинному навчанні, фінансовій аналітиці тощо. Її найпопулярніший підмодуль — pyplot, який дозволяє створювати графіки в стилі MATLAB. Matplotlib дозволяє візуалізувати будь-які числові або категоріальні дані, що допомагає краще зрозуміти структуру та закономірності в них [13].

Бібліотека Pandas — це одна з найпопулярніших і найпотужніших бібліотек у мові програмування Python, призначена для ефективної обробки, аналізу та підготовки структурованих даних. Вона забезпечує гнучкий інструментарій для роботи з табличними структурами, такими як DataFrame та Series, що дозволяє зручно зчитувати, змінювати, групувати, фільтрувати й агрегувати великі обсяги інформації. Завдяки цим можливостям Pandas є надзвичайно корисною під час створення звітів, очищення даних, попередньої обробки для машинного навчання та побудови складних аналітичних моделей.

Бібліотека широко застосовується в різних галузях — від наукових досліджень і академічної аналітики до фінансового моделювання, бізнес-аналітики, обліку продажів і прогнозування. Вона підтримує імпорт та експорт даних із файлів CSV, Excel, JSON, баз даних SQL та інших джерел. Окрім цього, Pandas інтегрується з іншими популярними бібліотеками Python, такими як NumPy, Matplotlib, Scikit-learn та TensorFlow, що робить її центральною складовою у багатьох проєктах, пов'язаних із аналізом даних або автоматизацією бізнес-процесів. У межах мого проєкту ця бібліотека є ключовим інструментом для роботи зі статистикою замовлень, побудови звітів та обробки клієнтських даних [14].

Формат JSON (JavaScript Object Notation) — це легкий текстовий формат, що використовується для зберігання та обміну структурованими даними між різними програмами, сервісами або пристроями. Він став надзвичайно популярним у сфері розробки програмного забезпечення завдяки своїй простоті, універсальності та сумісності з багатьма мовами програмування, зокрема з Python. Основною перевагою JSON є його здатність представляти складні структури даних у вигляді простого тексту, який легко читається не лише машинами, але й людьми. Він базується на парі ключ–значення та дозволяє зберігати дані у вигляді об'єктів, списків, рядків, чисел та логічних значень [15].

У Python формат JSON особливо корисний для зберігання конфігурацій, експорту даних, передачі інформації між клієнтськими й серверними частинами застосунку, а також для взаємодії з API. За допомогою вбудованого модуля json розробники можуть легко серіалізувати (перетворювати в JSON) та десеріалізувати (повертати у вигляд Python-об'єктів) різноманітні структури даних, такі як словники або списки. Це спрощує обмін даними в межах систем або між окремими сервісами. JSON є незамінним у сучасній розробці, особливо у програмуванні, де постійно виникає потреба у швидкому та зручному форматі передачі даних. Формат був розроблений Дугласом Крокфордом і з того часу став одним із стандартів у галузі програмної інженерії [16].

Розробка програмних модулів системи моніторингу клієнтського потоку в закладах приготування кавових напоїв передбачає кілька етапів. По-перше, потрібно

створити модуль, який буде відповідати за зручну та інтуїтивно зрозумілу взаємодію із системою. Цей модуль буде відповідати за взаємодію користувача із програмним продуктом та навігацією із функціоналом застосунку. Далі буде розроблений модуль для збирання та зберігання інформації про замовлення клієнтів, інформація буде зберігатись у файлі формата JSON, і з нього ж вона буде братись для подальшого аналізу.

Після виявлення та визначення замовлень клієнтів, буде розроблений модуль для будування списку замовлень. За допомогою цього модуля користувач може налаштовувати меню кав'ярні, додавати нові пропозиції, редагувати їх ціну, вся інформація з меню буде зберігатись у вищеописаному форматі JSON [17].

Наступним буде створеним модуль для визначення популярності позицій. Суть цього модулю буде в тому, що адміністрації кав'ярні буде відомо, яка з позицій у кав'ярні має популярність серед клієнтів і на основі отриманих даних можна створювати позиції, які будуть уявляти з себе комбінації двох найпотребніших позицій.

Ще один із необхідних модулів це модуль активності замовлень за годинами, цей модуль може використовуватись в цілях заманювання клієнтів, наприклад якщо у вечірній час активність клієнтів знижується, то завдяки цьому модулю це буде відомо адміністрації кав'ярні і вона може запропонувати вечірні знижки на тістечка.

Розробка цих програмних модулів потребує розуміння, графічного програмування та програмування на Python та роботи з її бібліотеками. Буде проведено тестування програмних модулів для забезпечення точності та коректної роботи програмного застосунку.

Отже, розробка програмних модулів системи моніторингу клієнтського потоку в закладах приготування кавових напоїв на мові програмування Python, з використанням бібліотек Tkinter, Matplotlib, Pandas а також текстового формату для зберігання даних JSON це комплексне завдання, яке складається з багатьох етапів.

Модулі повинні бути ретельно спроектовані, реалізовані та протестовані, щоб забезпечити користувачу зручну, ефективну, та правильну роботу з програмним забезпеченням для моніторингу клієнтського потоку. Маючи необхідні інструменти

та досвід, можна створити програмне забезпечення, яке задовольнить потреби користувача а також значно спростить збирання інформації про замовлення клієнтів.

2.2 Розробка алгоритму дії програмного застосунку

При розробці програмного забезпечення для моніторингу потоку клієнтів кав'ярні необхідно обрати найбільш оптимальні технології для розробки кожного модуля програмних засобів. Мова програмування Python та її графічні бібліотеки tkinter є потужним інструментом для розробки графічного інтерфейсу користувача. Використання tkinter забезпечить можливість створення зручного інтерфейсу для взаємодії з користувачем та відображення інформації яку він отримує під час роботи у застосунку.

Бібліотека Matplotlib — це потужний інструмент для побудови графіків та візуалізації даних у Python. Вона є однією з найпопулярніших бібліотек для створення 2D-графіки і часто використовується в наукових дослідженнях, аналізі даних, машинному навчанні, фінансовій аналітиці тощо. Її найпопулярніший підмодуль — pyplot, який дозволяє створювати графіки в стилі MATLAB. Matplotlib дозволяє візуалізувати будь-які числові або категоріальні дані, що допомагає краще зрозуміти структуру та закономірності в них. Ця бібліотека зазвичай використовується у побудові лінійних графіків, гістограм кругових діаграм. Також вона здатна візуалізувати тимчасові ряди, створювати теплові карти, графіки функцій та субграфіки, здатна представляти графічні звіти у бізнес-аналітиці.

Аналогом Matplotlib може виступати Plotly ця бібліотека здатна складати графіки для інтерактивної візуалізації даних, отримані графіки користувач може масштабувати і переглядати на різних пристроях. Вона підтримує інтеграцію з Jupyter Notebook, Dash. Вона підтримує побудову 3D-графіків і діаграм, однак у зв'язку з складністю реалізації і відсутності необхідності у використанні 3D графіки, було рішено відмовитись від Plotly у користь Matplotlib, оскільки вона набагато легша у реалізації і має здатна до потужної оптимізації.

Бібліотека Pandas — це одна з найпопулярніших бібліотек у Python для обробки, аналізу та підготовки даних. Вона широко використовується у наукових

дослідженнях, машинному навчанні, фінансовому аналізі, бізнес-аналітиці та будь-якій сфері, де потрібно працювати з табличними або часовими даними. Pandas дозволяє працювати з табличними структурами даних типу DataFrame – двовимірна таблиця аналог Excel-таблиць, та Series – одномірний набір даних. Бібліотека Pandas здатна до очищення і фільтрування даних, обробляє пропущені значення і здатна обчисляти статистики, також вона побудована поверх NumPy і дозволяє ефективно працювати з великими обсягами даних.

Формат JSON (JavaScript Object Notation) — це зручний та легкий у використанні текстовий формат, який широко застосовується для зберігання та обміну даними між різними системами та додатками. Його основною перевагою є простота структури, яка робить цей формат придатним як для машинного оброблення, так і для читання людьми. Саме завдяки цій простоті JSON став надзвичайно популярним у галузі програмування, особливо в розробці вебдодатків, мобільного програмного забезпечення, API-сервісів та інтеграційних рішень.

У мові програмування Python формат JSON використовується дуже активно. Зокрема, він служить для збереження інформації у вигляді файлів, передачі даних між клієнтом і сервером, а також для обробки структурованої інформації. До типових прикладів таких структур можна віднести словники, списки, вкладені об'єкти, які легко перетворюються у формат JSON і назад. Python має вбудований модуль json, який дозволяє серіалізувати (перетворювати об'єкти у формат JSON) і десеріалізувати (повертати у вихідну форму) ці об'єкти.

Варто зазначити, що формат JSON є дуже схожим на об'єктну нотацію мови JavaScript, з якої він і походить, але завдяки універсальності його підтримують практично всі сучасні мови програмування. Також одним із важливих плюсів є те, що цей формат є незалежним від платформи, що дозволяє використовувати його у кросплатформенних рішеннях.

Формат JSON був створений програмістом Дугласом Крокфордом, який зробив його публічним стандартом і сприяв широкому впровадженню цього формату в IT-індустрії. Завдяки його зусиллям JSON сьогодні став одним із найпоширеніших способів представлення даних у цифровому світі.

Поєднання цих технологій розробки дозволяє створити програмні модулі для системи моніторингу клієнтського потоку в закладах приготуввання кавових напоїв. Python та бібліотеки Pandas і Matplotlib забезпечують надійну та ефективну кодову базу для взаємодії та маніпуляцій з статистикою замовлень клієнтів, в той час як tkinter надає можливості та інструменти для створення зручного та приємного інтерфейсу користувача. Використовування JSON забезпечує швидкий та точний аналіз статистики замовлень, та надійно зберігає дані на пристрої, що є необхідним для успішної роботи застосунку.

Отже, вибір технологій для розробки програмних модулів системи моніторингу клієнтського потоку в закладах приготування кавових напоїв в Python, його бібліотеках Pandas, Matplotlib та tkinter, а також текстовому формату для зберігання та обміну даними JSON, є найкращим вибором для створення ефективного і точного програмного модулю для моніторину потоку клієнтів.

Поєднання вищеперелічених інструментів надає комплексний і якісний підхід до розробки, що дозволяє створювати продукт з високим рівнем функціональності, надійності та зручного користування.

Для розробки програмного застосунку, для інтелектуального моніторингу клієнтського потоку у кав'ярнях, необхідно розробити загальний алгоритм, згідно якого буде працювати програмний модуль (рис 2.4).

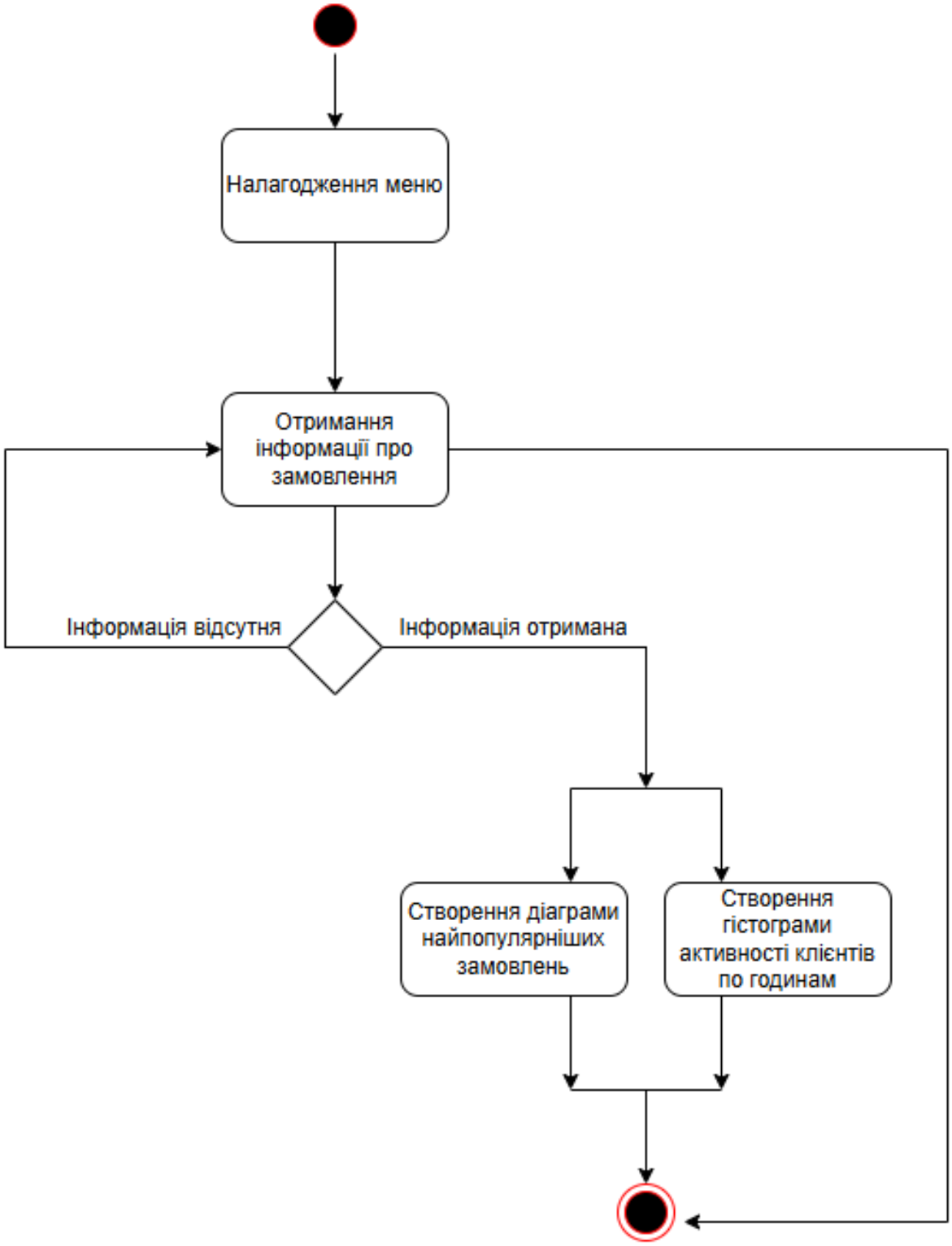


Рисунок 2.4 – Діаграма діяльності програмних засобів

Розробка діаграми діяльності була виконана у програмному забезпеченні draw.io [18].

2.3 Розробка графічного інтерфейсу програмного застосунку

Під час створення інтерфейсу програмного забезпечення значна увага приділялася його зрозумілості, інтуїтивності та зручності взаємодії для кінцевого користувача. Метою було досягти такого вигляду інтерфейсу, який би не викликав труднощів при навігації та забезпечував приємний візуальний досвід під час користування програмою. Усі елементи були розміщені з урахуванням логіки взаємодії користувача, а компонування екранів розроблялося відповідно до принципів ергономіки інтерфейсів. Враховано потреби різних категорій користувачів, у тому числі тих, хто не має великого досвіду у роботі з програмним забезпеченням [19].

Основним кольором інтерфейсу було обрано білий, який асоціюється з чистотою, простотою та відкритістю. Білий фон забезпечує високу контрастність і дозволяє зосередити увагу користувача на основному контенті. Для текстових елементів застосовано чорний колір, що забезпечує найкращу читабельність на світлому фоні. Така кольорова палітра надає програмі лаконічного, стриманого вигляду, підкреслює професійність і не створює візуального перевантаження, що позитивно впливає на комфорт тривалого використання [20].

Після запуску застосунку користувача зустрічає вікно завантаження, яке дає зрозуміти, що програма успішно запускається. Після короткого завантаження з’являється головне вікно з базовою інформацією про призначення застосунку, його можливості та перелік доступних дій. Інтерфейс чітко структурований: у верхній частині розміщене меню з основними функціями, а в центральній — основний блок для взаємодії з даними та візуалізацією. Оскільки програмні засоби передбачають взаємодію із великим обсягом текстової інформації, було вирішено створити віртуальне робоче середовище із чітким розмежуванням за допомогою відповідних вкладок меню, яке зображено на рисунку 2.5.

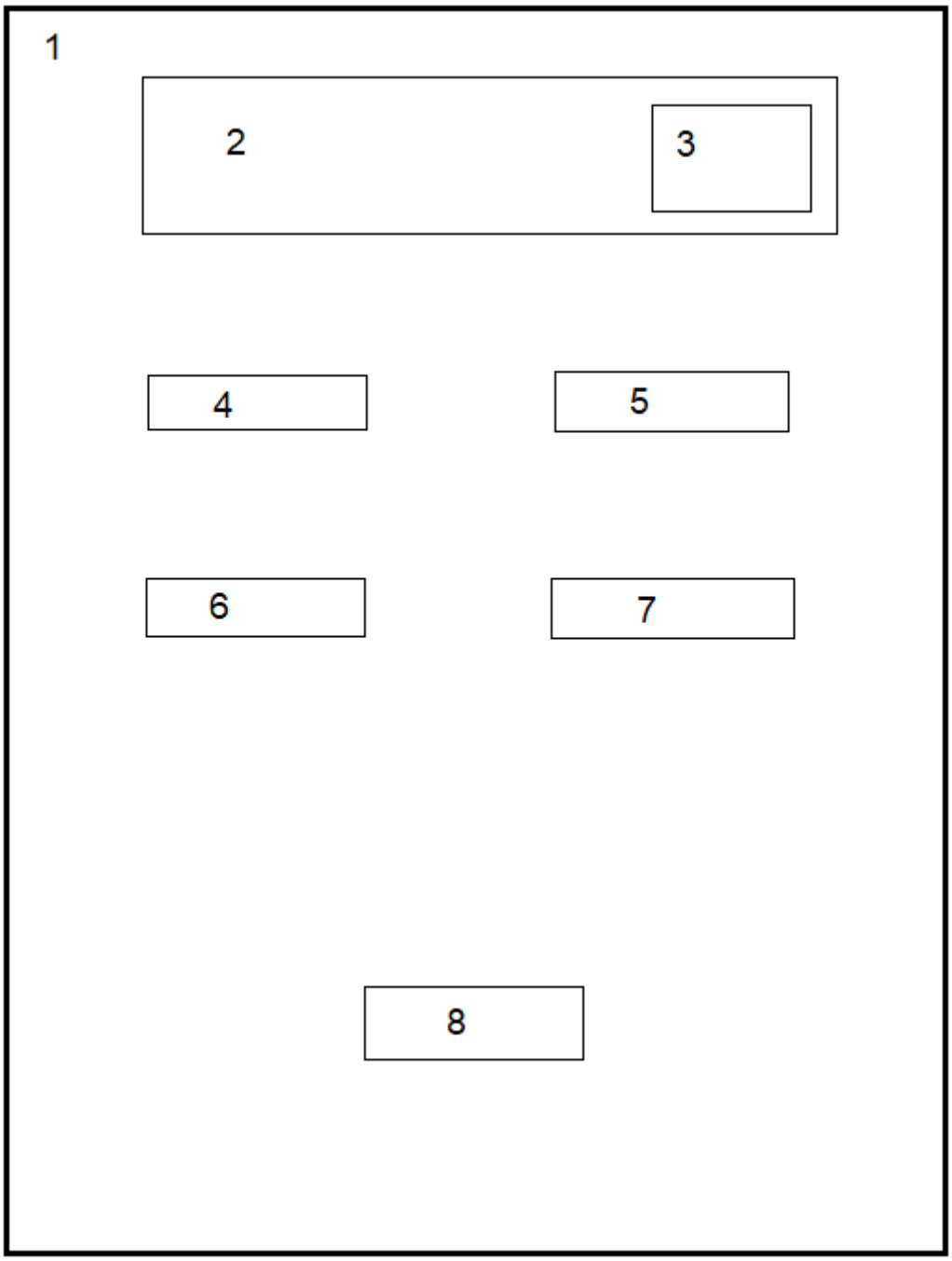


Рисунок 2.5 – Структурна схема головного вікна застосунку

На продемонстрованому вище рисунку представлено структурну схему головного застосунку яка графічно зазначає основні фунції які були розроблені у застосунку:

- під цифрою «1» зображено головне вікно у якому розташовані функціональні кнопки;

- під цифрою «2» зображено блок з назвою застосунку;

- під цифрою «3» зображено місце для логотипу застосунку;

- під цифрою «4» зображено місце для функціональної кнопки «Про програму» яка виводить на екран інформацію про застосунок

- під цифрою «5» зображено місце для функціональної кнопки «Редагування меню кав’ярні» при натисканні користувача переносить на екран, у якому він може редагувати меню кав’ярні, додавати або видаляти конкретні позиції, встановлювати їм ціну;

- під цифрою «6» зображено місце для функціональної кнопки «Вивід інформації про активність клієнтів» при натисканні користувача переносить на екран, у якому він може отримати інформацію про те, у який час найактивніше клієнти відвідують кав’ярні і оформлюють замовлення;

- під цифрою «7» зображено місце для функціональної кнопки «Вивід інформації про найпопулярніші позиції» при натисканні користувача переносить на екран, у якому він може отримати інформацію про те, які саме позиції у кав'ярні мають більше усього замовлень;

- під цифрою «8» зображен місце для функціональної кнопки «Закрити застосунок» при натисканні на неї користувач завершує роботу і закриває застосунок.

Окрім створення графічного інтерфейсу застосунку було прийнято рішення для створення простого логотипу.

Логотип є одним із ключових елементів візуальної ідентичності будь-якого програмного продукту. Він виконує функцію впізнаваності — саме завдяки логотипу користувачі можуть швидко асоціювати програму з її призначенням, функціоналом або компанією-розробником. Логотип допомагає створити позитивне перше враження, формує довіру до продукту та підкреслює його унікальність серед конкурентів. У разі розробки комерційного застосунку логотип також відіграє важливу роль у просуванні бренду на ринку [21].

Існує кілька основних типів логотипів. Один із найпоширеніших – текстовий логотип (логотип-шрифт)**,** що базується на унікальному написанні назви продукту або компанії. Такий логотип легко читається, добре масштабується і запам’ятовується. Інший тип – символічний або графічний логотип, який складається з абстрактного або конкретного зображення без тексту. Цей тип часто використовується у мобільних

застосунках або на піктограмах. Також можливий комбінований логотип, який поєднує текстову частину з графічним елементом – це найуніверсальніший варіант, який дозволяє використовувати частини логотипу окремо в залежності від контексту.

При створенні логотипу для програмного забезпечення важливо враховувати його тематику, цільову аудиторію, стиль інтерфейсу та загальну візуальну концепцію. Логотип повинен бути простим, адаптивним, легко впізнаваним і візуально гармонійним. У випадку програмного продукту для моніторингу потоку клієнтів, логотип може містити елементи, пов’язані з аналітикою, людьми, графіками або інтерфейсними іконками — щоб підкреслити функціонал програми вже з першого погляду [22].

Для створення логотипу було принято рішення використовувати саме графічний тип логотипу, легкий для створення і простий логотип легко запам'ятовується клієнтами і викликає асоціації з комфортом і чимось знайомим. На рисунку 2.6 показано ескіз логотипу для програмного застосунку у вигляді горнятка кави.

Рисунок 2.6 – Ескіз логотипу програмного застосунку

Після створення ескізу, було обрано кольори для заповнення ескізу, а саме сірий, коричневий та світло-коричневий, поєднання цих кольорів асоціюється з теплою атмосферою кав'ярні. Готовий логотип зображено на рисунку 2.7.

Рисунок 2.7 – Готовий логотип програмного застосунку

2.4 Висновки

У другому розділі було обрано мову програмування та додаткові засоби для створення програмного застосунку. Було розроблено основний алгоритм роботи програмного застосунку. Розглянуто процес створення графічного інтерфейсу програмного застосунку, та було створено логотип до нього.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ КЛІЄНТСЬКОГО ПОТОКУ В КАВ’ЯРНЯХ

3.1 Розробка алгоритмів і програм інтелектуального моніторингу

Згідно розробленої блок-схеми, робота програмного застосунку починається із запуску головного вікна, яке представлено класом MainWindow. Це головний елемент графічного інтерфейсу користувача, який ініціалізує всі вкладки застосунку та забезпечує навігацію між ними. Після запуску інтерфейсу, користувач має змогу перейти до вкладки меню кав’ярні, реалізованої у класі MenuTab. Тут користувач переглядає доступні позиції з меню, а також може додавати нові страви або напої. Для управління даними меню застосовується допоміжний клас MenuManager, який зчитує та зберігає інформацію про меню у відповідному форматі [23].

Наступним важливим кроком є можливість генерації замовлень, що реалізовано через вкладку GenerateClientsTab. При її активації викликається клас OrderGenerator, який автоматично створює 200 випадкових замовлень, кожне з яких прив'язане до умовного клієнта (наприклад, «Клієнт №1»). Ці замовлення містять від однієї до трьох позицій з меню та зберігають інформацію про дату й час створення у форматі останнього тижня. Усі ці замовлення представлені об'єктами класу Order, які у свою чергу складаються з набору об’єктів класу Item, що містять назву та вартість конкретної позиції з меню.

Далі користувач може перейти на вкладку OrdersTab, де відображаються всі нещодавно згенеровані замовлення у зручному вигляді. Для аналітичної обробки зібраної інформації використовується вкладка PopularityTab, що дозволяє виявити найпопулярніші позиції з меню на основі кількості замовлень. Інформація виводиться у вигляді діаграми, що допомагає візуально оцінити переваги клієнтів. Окрім цього, користувач може скористатися вкладкою HourlyActivityTab, яка формує гістограму замовлень по годинах протягом останнього тижня. Для кращого сприйняття тут не враховуються хвилини — лише години, що дає змогу аналізувати пікові періоди активності клієнтів.

Таким чином, блок-схема відображає чітку структурну організацію застосунку, де кожна вкладка має свою відповідальність, а логіка обробки даних чітко відокремлена від представлення інтерфейсу. Це дозволяє забезпечити гнучкість і масштабованість системи в разі подальшого її розвитку.

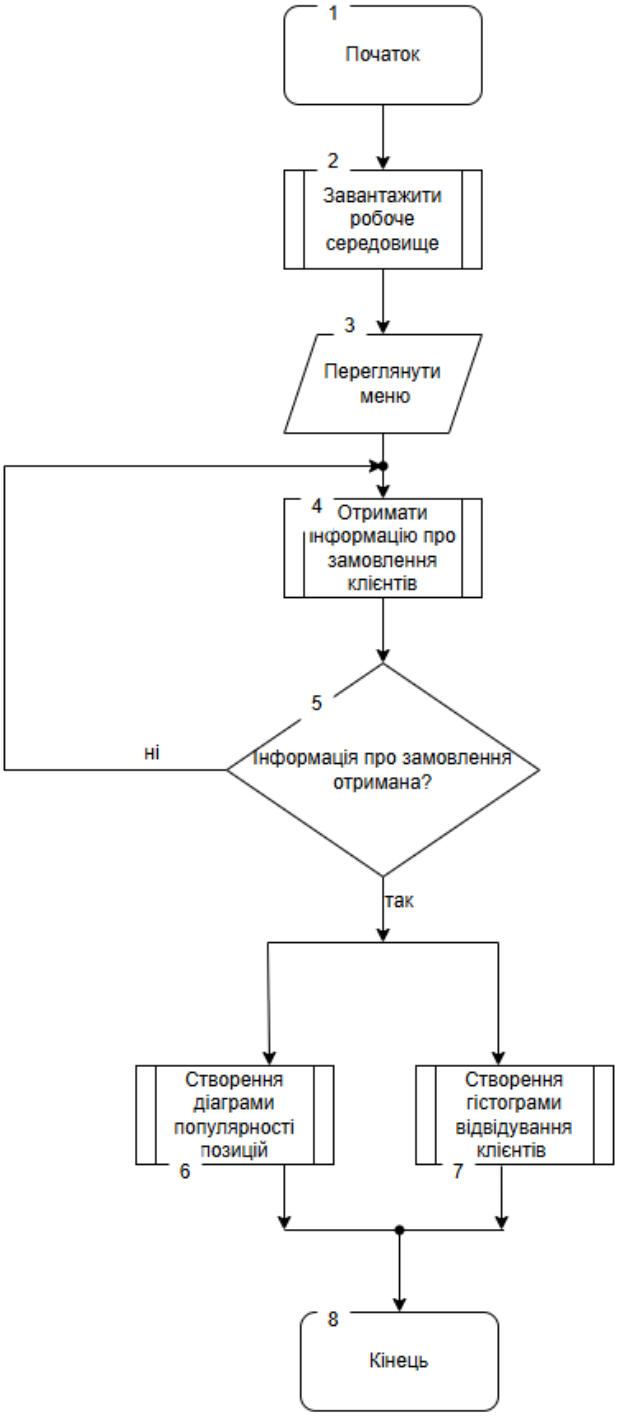


Рисунок 3.1 – Алгоритм роботи програмного застосунку

3.2 Розробка модуля редагування та перегляду меню кав'ярні

Під час розробки модуля, призначеного для редагування та перегляду меню кав’ярні, було прийнято рішення використовувати текстовий формат JSON (JavaScript Object Notation) для зберігання та обміну інформацією про позиції меню. Цей формат обрано через його зручність, просту структуру, легку читаність та підтримку більшістю мов програмування. JSON дозволяє ефективно зберігати ключову інформацію про кожну позицію меню, зокрема її назву, ціну у гривнях, категорію, доступність та інші важливі параметри.

Модуль реалізований таким чином, щоб забезпечити максимально зручну взаємодію користувача з інтерфейсом. У готовій до використання версії модуля реалізовано можливість додавання нових пунктів до меню через спеціальну форму. Користувач може ввести назву напою чи страви, вказати її вартість у гривнях та за потреби додати додаткову інформацію — наприклад, короткий опис чи позначити статус наявності. Після введення нової інформації вона автоматично додається у структуру JSON, яка зберігається у відповідному файлі або базі даних, що забезпечує швидкий доступ до змін у майбутньому.

Крім функції додавання, модуль також підтримує перегляд існуючих позицій меню у зручному табличному вигляді або списку. Це дозволяє адміністраторам швидко орієнтуватися у всьому асортименті продукції та оперативно вносити зміни в разі потреби. Графічний інтерфейс реалізовано з урахуванням принципів зручності та інтуїтивності, щоб навіть нові користувачі змогли легко розібратися з його функціоналом. Алгоритм роботи меню детально продемонстровано на рисунку 3.2

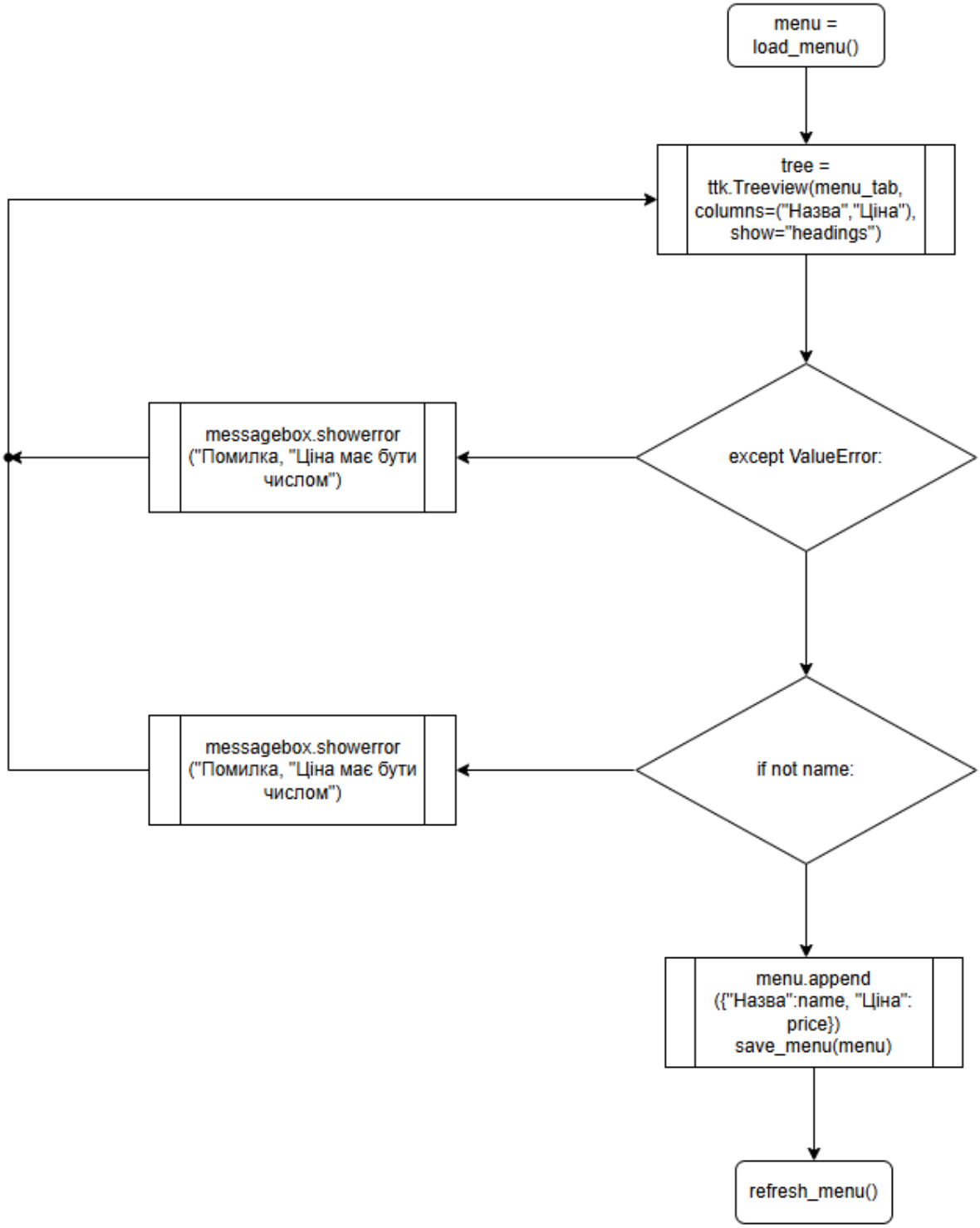


Рисунок 3.2 – Блок-схема алгоритму додавання позиції у меню

3.3 Розробка модуля для аналітики статистики найпопулярніших позицій

Наступний алгоритм покроково та детально ілюструє процес створення діаграми, яка відображає найпопулярніші послуги серед клієнтів. Алгоритм реалізується у вигляді окремого програмного модуля, що взаємодіє з базою даних замовлень користувачів. На першому етапі програма здійснює зчитування інформації про всі зроблені замовлення клієнтів, які були зафіксовані у системі протягом певного періоду — наприклад, тижня або місяця. Ці дані включають назви послуг, їхню кількість, час замовлення та інші мета-дані, які можуть бути використані для аналітики.

Після отримання інформації дані передаються до внутрішнього механізму аналізу, який проводить сортування, агрегацію та підготовку до візуалізації. На цьому етапі застосовується бібліотека matplotlib, яка дозволяє побудувати наочну діаграму у вигляді гістограми або кругової діаграми — залежно від реалізації. Візуалізація показує, які послуги користуються найбільшим попитом серед клієнтів, що дозволяє отримати цінну аналітику для подальших рішень у сфері маркетингу або оптимізації асортименту.

У випадку, якщо на цьому етапі програма не може знайти жодного замовлення або база даних не містить необхідної інформації, вона ініціює повідомлення про помилку. Це повідомлення виводиться користувачеві у графічному інтерфейсі із зазначенням причини (наприклад, "Немає замовлень за обраний період"). Після цього алгоритм повертається до початкової точки, дозволяючи користувачеві спробувати знову або перевірити введені параметри.

У фінальній частині алгоритму, після успішного оброблення даних, програма будує графік, який виводиться безпосередньо у графічному інтерфейсі користувача. Діаграма відображає найпопулярніші позиції, що дозволяє візуально оцінити уподобання клієнтів у зручному форматі. Робота цього алгоритму наочно представлена на рисунку 3.3, де зображено всі етапи обробки та відображення статистики.

Головною задачею модуля для аналітики статистики найпопулярніших позицій є точне вимірювання основних показників при аналізі замовлень клієнтів кав’ярні.

Для реалізації цієї задачі було створено функцію show_popularity, що приймає в себе в якості параметра замовлення клієнтів, програмний код функції зображений на рисунку 3.6. Крім того, було розроблено функцію для візуального виводу діаграми за допомогою стандартної бібліотеки мови програмування Python – matplotlib.

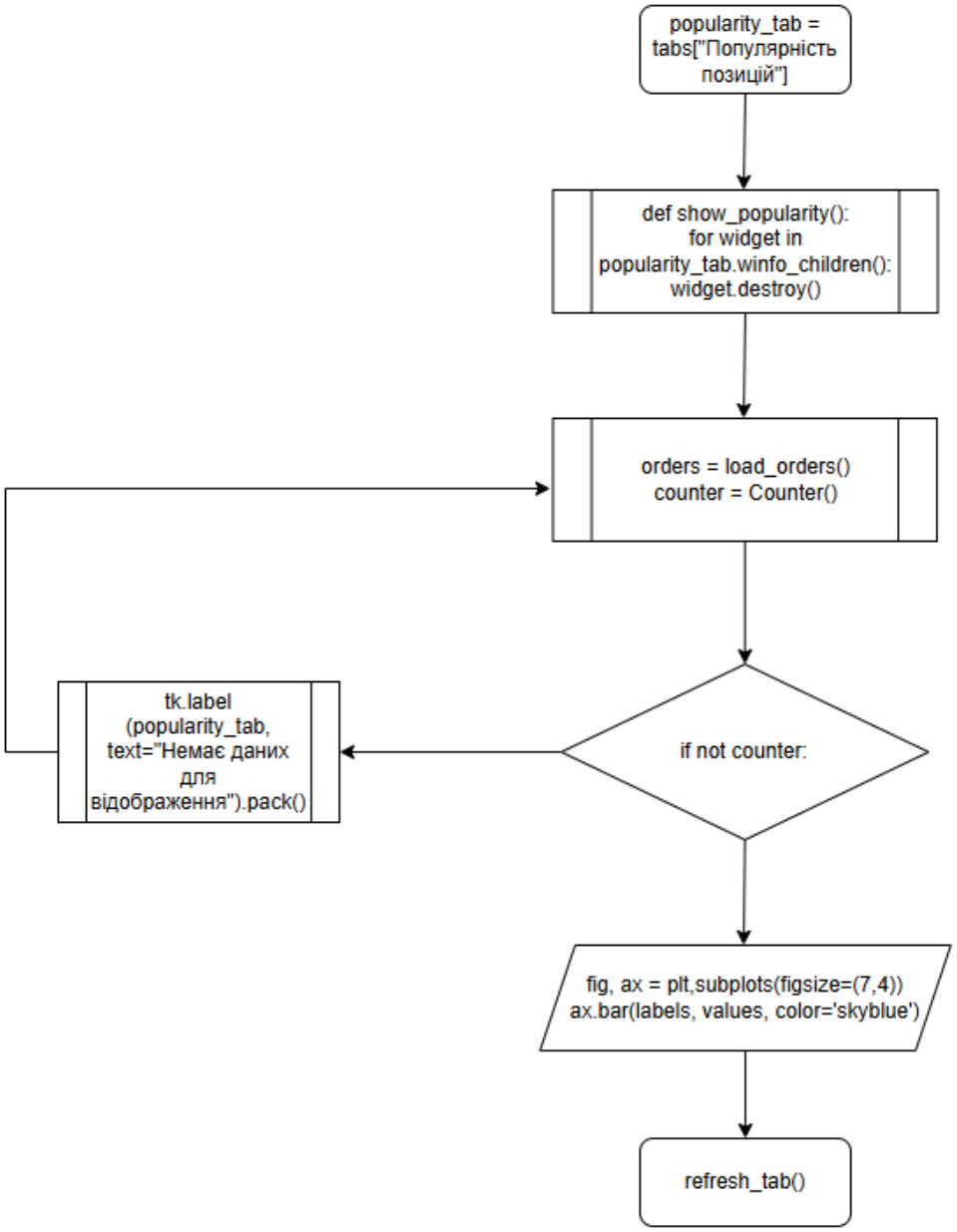


Рисунок 3.3 – Блок-схема алгоритму показу найпопулярніших позицій у кав’ярні

3.4 Розробка модуля для виводу графіку активності клієнтів кав’ярні за годинами

Наступний модуль програмного забезпечення відповідає за виконання операцій сортування інформації про замовлення та візити клієнтів до кав’ярні згідно з часом — зокрема, по годинам протягом дня. Цей функціонал є надзвичайно корисним для адміністрації кав’ярні, оскільки дозволяє отримати детальне уявлення про пікові години активності клієнтів. Аналіз таких даних допомагає не лише зрозуміти поведінку споживачів, а й сформувати більш ефективну стратегію управління закладом.

Завдяки цьому модулю керівництво кав’ярні має змогу ідентифікувати години найвищого та найнижчого навантаження, що у свою чергу дає змогу приймати зважені рішення щодо маркетингових кампаній. Наприклад, у періоди найменшої відвідуваності можна застосовувати знижки, спеціальні пропозиції чи акції з метою залучення більшої кількості клієнтів. Це сприяє підвищенню загального прибутку закладу за рахунок рівномірного розподілу потоку відвідувачів упродовж дня.

Операція сортування реалізована за допомогою відповідних алгоритмів обробки даних, що дозволяє швидко та ефективно згрупувати замовлення відповідно до часу їх створення. Інтерфейс модуля розроблений таким чином, щоб користувач міг легко переглянути кількість замовлень у кожну годину, проаналізувати активність і зробити відповідні висновки для прийняття рішень.

Інформація про активності за годинами візуально зображається завдяки виводу гістограми, яку можливо побудувати використовуючи бібліотеку matplotlib.

На рисунку 3.4 зображено функцію, яка відповідає за роботу модуля.

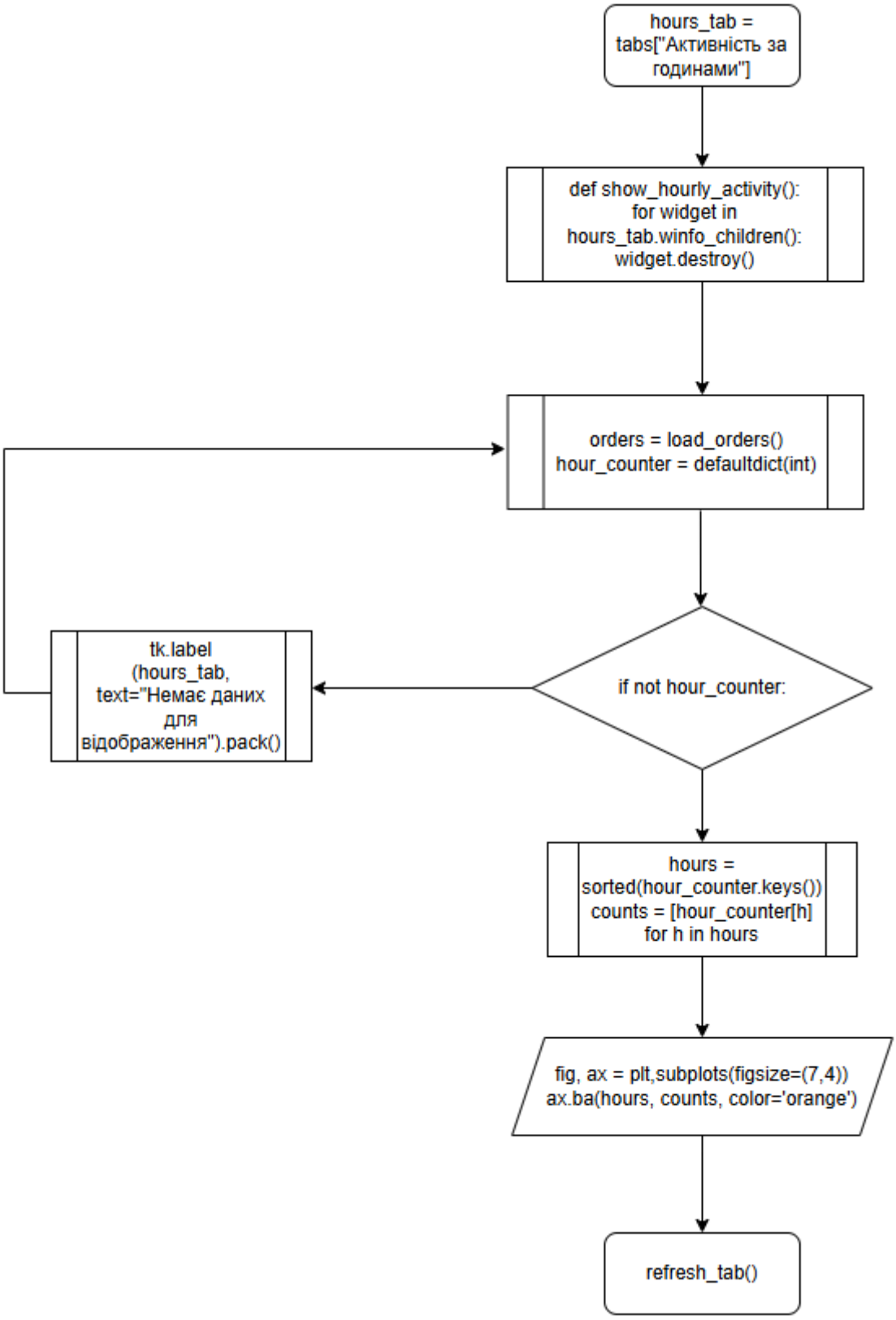


Рисунок 3.4 – Блок-схема алгоритму модуля відображення активності клієнтів за годинами

3.5 Розробка модуля для створення графічного інтерфейсу програмного застосунку

Основним і найважливішим модулем, завдяки якому забезпечується візуальне сприйняття програмного застосунку для моніторингу потоку клієнтів у кав’ярні, є графічний модуль. Цей модуль реалізовано з використанням потужної бібліотеки tkinter, що є стандартним інструментом для створення графічних інтерфейсів користувача в Python. Завдяки tkinter застосунок набуває інтуїтивно зрозумілого та привабливого вигляду, що забезпечує комфортну взаємодію з користувачем.

Графічний інтерфейс цього модуля дозволяє адміністраторам кав’ярні в реальному часі переглядати потік клієнтів, відстежувати різні параметри відвідувань, такі як час перебування, кількість замовлень і навіть час пік активності, що дозволяє приймати зважені рішення щодо організації роботи закладу. Крім того, цей модуль дозволяє відображати статистику, графіки та діаграми, що дає можливість користувачам аналізувати візуальну інформацію про відвідувачів у зручному і доступному форматі.

Програмний код цього модуля наведено на рисунку 3.5, де зображено структуру та компоненти, що забезпечують функціонування графічного інтерфейсу та взаємодію з іншими частинами системи.

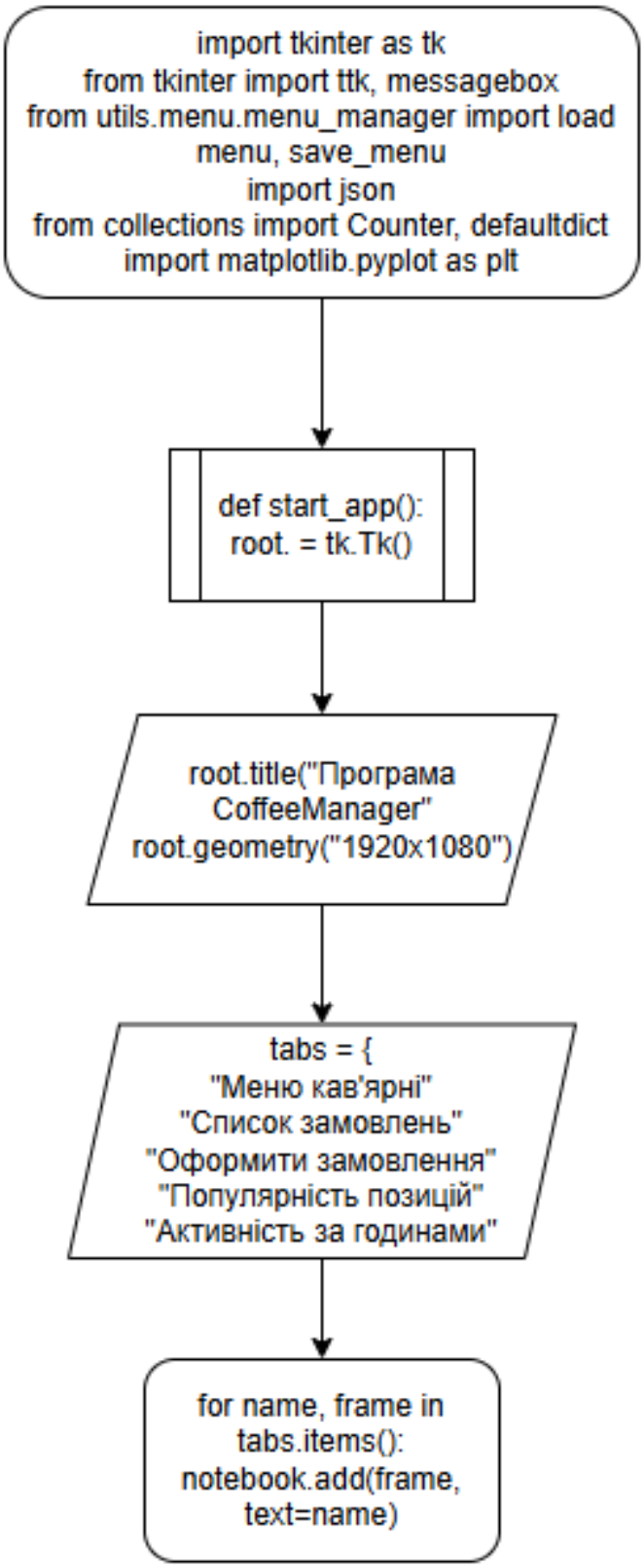


Рисунок 3.10 – Блок-схема алгоритму модулю для створення графічного інтерфейсу програмного застосунку

3.6 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувались для розробки програмних модулів. Провевши аналіз програмного продукту та його задач було обрано мову програмування Python. Для розробки графічного інтерфейсу користувача була обрана стандартна бібліотека Python – Tkinter. Для роботи із інформацією про замовлення клієнтів було обрано бібліотеки Matplotlib та Pandas. У розділі також було описано розробку основних програмних модулів застосунку.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Тестування програмного забезпечення

Тестування програмного забезпечення являє собою процес перевірки роботи програмного продукту з метою виявлення помилок, збоїв або недоліків, а також підтвердження його відповідності визначеним вимогам користувача чи замовника. Головною метою цього процесу є забезпечення високої якості та стабільної роботи програмного забезпечення до моменту його впровадження або реального використання [24].

Перш ніж розпочати тестування, необхідно інсталювати тестову версію застосунку на комп’ютер з операційною системою Windows або MacOS, яка буде використовуватись у процесі перевірки. Також слід підготувати всю супровідну документацію, серед якої мають бути специфікації вимог, а також заздалегідь складений план тестування. Процес перевірки складається з декількох етапів і типів тестування. Першим з них є функціональне тестування, що передбачає перевірку коректного виконання всіх функцій програми. До цього входить тестування введення і виведення інформації, перевірка реакції системи на некоректні або помилкові дані, а також перевірка відповідності реалізованої логіки очікуваним бізнес-процесам.

Наступним етапом є тестування на відмовостійкість, мета якого — виявлення помилок та збоїв у роботі програмного забезпечення. Це тестування передбачає оцінку здатності системи відновлювати свою функціональність після виникнення критичних ситуацій або несправностей. У рамках цього етапу використовуються різні сценарії, які можуть передбачати навмисне введення некоректних даних, використання неочікуваних параметрів або моделювання взаємодії з програмою в умовах, що імітують надзвичайні обставини, наприклад, втрату зв’язку з іншими компонентами чи порушення роботи системи.

Після завершення функціонального та відмовостійкого тестування виконується ще один важливий етап — перевірка продуктивності програмного забезпечення. Це тестування орієнтоване на аналіз швидкодії, під час якого оцінюється, наскільки

ефективно виконується обробка операцій і завдань. Основна увага приділяється вимірюванню часу реакції програми на різні дії користувача або зовнішні запити.

Окремим етапом процесу тестування виступає перевірка сумісності, яка має на меті оцінити коректність роботи програмного забезпечення на різних операційних системах, версіях програмного забезпечення та типах апаратного забезпечення. У ході такого тестування з'ясовується, чи здатна програма функціонувати в різних середовищах та конфігураціях, а також чи не виникають при цьому збої або конфлікти під час взаємодії з іншими застосунками.

Під час першого етапу тестування програмного забезпечення перевіряється реакція програми на відсутність інформації про замовлення. Якщо користувач спробує переглянути діаграми без жодної інформації, програма має виявити цю помилку та повідомити користувача про відсутність інформації, результат тестування показаний на рисунку 4.1.

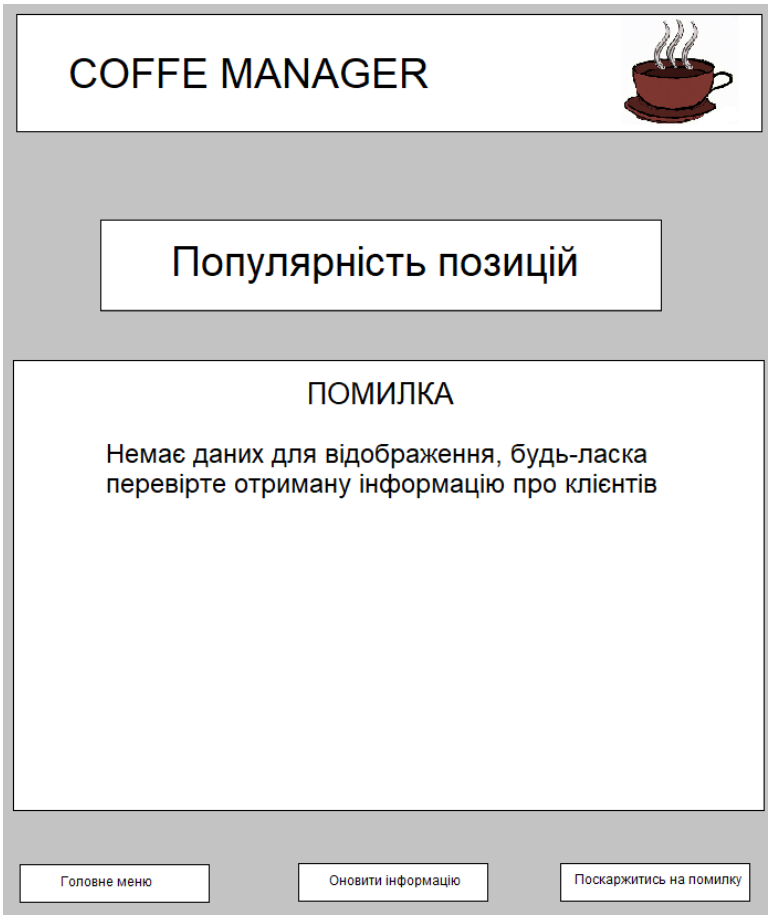


Рисунок 4.1 – Спроба переглянути діаграму популярності позицій без жодної інформації

Після проведення тестування програмного забезпечення на спробу переглянути діаграму популярності позицій без жодної інформації, було проведено додаткове тестування програмного застосунку, спрямоване на спробу додати нову пропозицію у меню без вказання її ціни. У результаті цього тестування, яке зображено на рисунку 4.2, було виявлено, що програмне забезпечення повідомляє користувача про помилку.

Рисунок 4.2 – Спроба додати нову пропозицію без вказання її ціни

Після успішної перевірки, наступною спробою було додавання пропозиції без назви на рисунку 4.3 продемонстрована реакція системи під час додавання пропозиції без назви. Результати тестування свідчать про те, що програмне забезпечення успішно обробляє такі випадки та повідомляє про це користувача програмного застосунку.

Рисунок 4.3 – Результат спроби додати послугу у меню без назви

У разі, якщо у користувача виникатимуть труднощі під час експлуатації програми або він потребуватиме додаткових роз'яснень щодо функціональності

застосунку, він може скористатися вбудованою системою довідки. Довідка доступна через спеціальну кнопку, розташовану в лівому верхньому кутку інтерфейсу програми. Користувач має змогу швидко отримати коротку та доступну інформацію, яка містить основні відомості про програмний продукт, його можливості та інструкції щодо ефективного використання. Кнопка для відкриття довідки, яку можна натискати для отримання інформації, зображена на рисунку 4.4.

Рисунок 4.4 – Головне меню програмного застосунку

Отже, в результаті проведення тестування було визначено, що програмний застосунок реагує на нестандартні ситуації, такого формату як введення некоректних даних або їх відсутність, при виникнені такої ситуації програма інформує про це користувача повідомленням.

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо рекомендованої та мінімальної конфігурації розміщено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовані	Мінімальні
1	2	3
Процесор	Intel Core i7	Intel Core i3
Оперативна пам'ять	8 ГБ RAM	2 ГБ RAM
Вільний простір на диску	6 ГБ	1 ГБ
Відеокарта	NVIDIA GeForce GTX 800 або аналог	NVIDIA GeForce GTX 650 або аналог
Монітор	Роздільназдатність 1920x1080 пікселів	Роздільна здатність 1280x720 пікселів
Операційна система	Windows 10 / macOS Big Sur	Windows7/ macOS Mojave

Після завершення процесу інсталяції та першого запуску програмного забезпечення, користувач автоматично переходить на інформаційний екран програмного застосунку, який виконує роль початкового вікна. На цьому етапі відображається базова інформація про програму, а також починається процес завантаження віртуального робочого середовища. Користувач очікує на завершення ініціалізації системи, після чого стає доступним увесь функціонал програмного забезпечення. Подальша взаємодія з додатком відбувається через спеціально реалізоване інтерактивне меню, яке містить усі основні дії, що дозволяють повноцінно користуватись програмою та переходити між модулями для виконання потрібних операцій.

Першим етапом взаємодії із програмним застосунком передбачається отримання інформації про замовлення клієнтів. Після чого користувач за наявністю отриманої інформації може будувати діаграми популярності позицій та відображати відвідуваність клієнтів у годинах за допомогою гістограми, меню кав’ярні показано на рисунку 4.5.

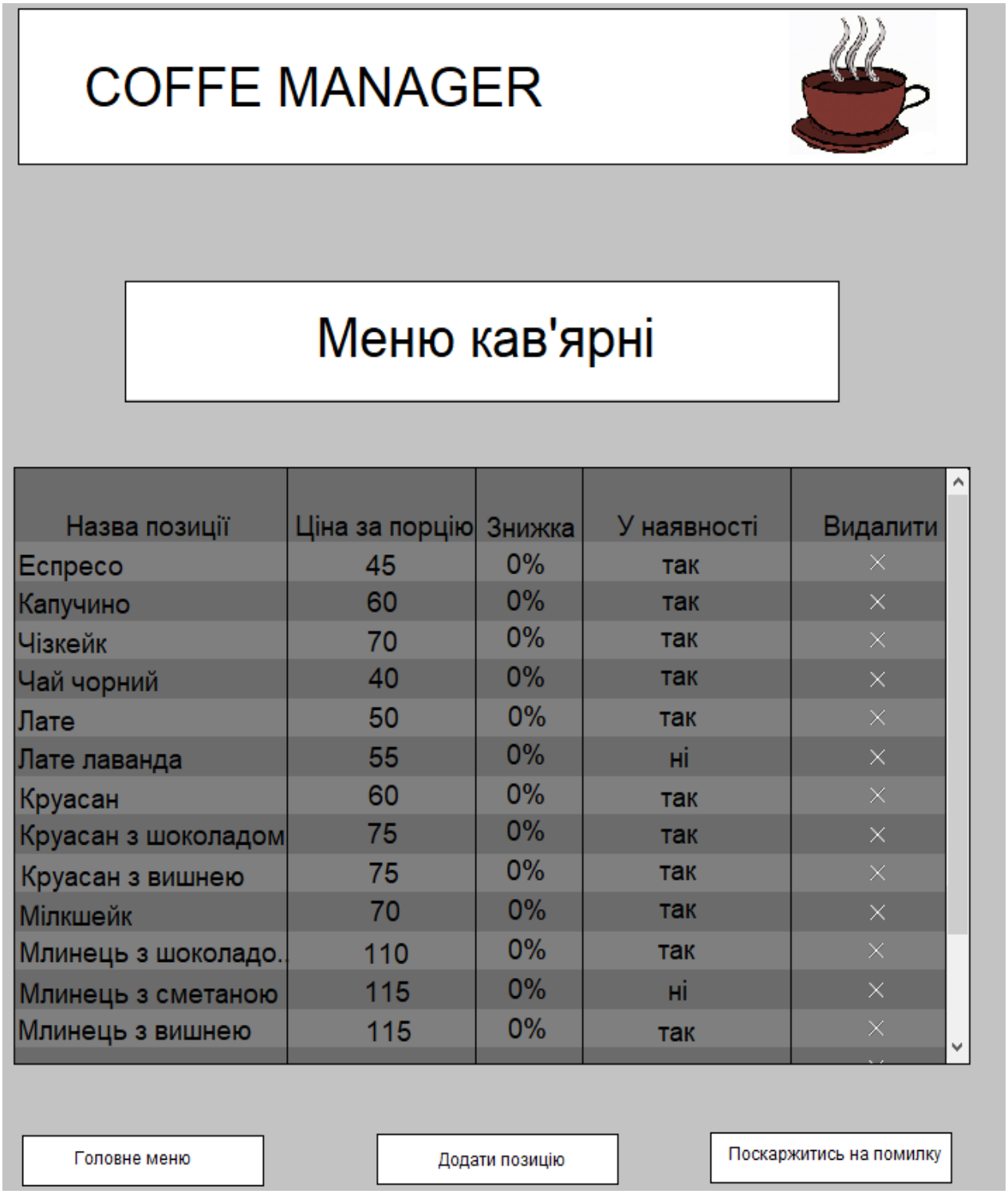


Рисунок 4.5 – Екран «Меню кав'ярні»

4.3 Висновки

У четвертому розділі було проведено тестування програмних модулів програмних застосунку моніторингу клієнтського потоку в закладах приготування кавових напоїв. Була перевірена реакція програмних засобів на нестандартні та помилкові ситуації. Також було розроблено інструкцію користувача по початковій експлуатації програмного продукту.

ВИСНОВКИ

У бакалаврській кваліфікаційні роботі було розроблено методи та програмні засоби комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях.

Проведено аналіз сучасних підходів до інтелектуального моніторингу клієнтського потоку. Обгрунтовано необхідність розробки методів виконання інтелектуального моніторингу клієнтського потоку з метою підвищення точності вимірювання отриманих даних. Бакалаврська кваліфікаційна робота оформлена з урахуванням рекомендацій [25].

Під час аналізу засобів розробки було обгрунтовано вибір мови програмування Python, бібліотек Tkinter та Matplotlib, та відповідних технологій для реалізації системи для інтелектуального моніторингу клієнтського потоку в кав’ярнях.

У першому розділі роботи проведено аналіз предметної області, розглянуто існуючі програмні рішення з інтелектуальним моніторингом клієнтського потоку, окреслено їхні переваги та недоліки. Особливу увагу приділено критеріям ефективності аналізу інформації отриманої з замовлень клієнтів.

У другому розділі представлено розгорнутий опис графічної моделі програмного застосунку, був розроблений логотип для програмного застосунку, який застосовано у розробці.

У третьому розділі описано архітектуру застосунку, викладено інформацію щодо процесу розробки програмних модулів, необхідних для повноцінного функціонування програмного застосунку. Наведено огляд сучасних технологій та інструментальних засобів, застосованих для розробки, з обґрунтуванням їх вибору. Описано внутрішню структуру та принципи взаємодії розроблених модулів для моніторингу клієнтського потоку.

У четвертому розділі проведено тестування розробленого проєкту. Продемонстровано основні екрани застосунку, описано сценарії використання. Тестування довело працездатність та коректність роботи програмного застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Coffee Shop Inventory Management: Tips, tools and techniques [Електронний ресурс] – Режим доступу до ресурсу: https://supy.io/blog/coffee-shop-inventory-management-tools-and-techniques/

2. Швець В. В. Чехместрук Р. Ю. Development of a software package for intelligent monitoring of customer flow in coffee shops // Матеріали конференції «Research in Science, Technology and Economics», Люксембург, 2025. [Електронний ресурс] – Режим доступу до ресурсу: https://isu-conference.com/wp-content/uploads/2025/05/Luxembourg_Luxembourg_28.05.25.pdf

3. Розвиток комп'ютерних технологій [Електронний ресурс] – Режим доступу до ресурсу: https://drukarnia.com.ua/articles/istoriya-rozvitku-komp-yuternoyi-tekhniki-V4GaH

4. Computer Vision Algorithms and Applications / Richard Szeliski - Springer London, 2010. 812p.

5. Plunkett's Wireless, Wi-Fi, Rfid & Cellular Industry Almanac: Wireless, Wi-Fi, Rfid & Cellular Industry Market Research, Statistics, Trends & Leading / Jack W. Plunkett - Plunkett Research, 2007. 460p.

6. Застосунок RetailNext [Електронний ресурс] – Режим доступу до ресурсу: https://retailnext.net/

7. Застосунок Poster POS [Електронний ресурс] – Режим доступу до ресурсу: https://joinposter.com/ua

8. Застосунок POS Sector [Електронний ресурс] – Режим доступу до ресурсу: https://pos-systems.com.ua/ua/

9. Застосунок Jsolutions [Електронний ресурс] – Режим доступу до ресурсу: https://jsolutions.ua/ua/sistema-upravleniya-skladom

10. Застосунок PayKit [Електронний ресурс] – Режим доступу до ресурсу: https://www.facebook.com/paykitpos/

11. Хошаба О. М. Методичні вказівки для виконання лабораторних робіт з дисципліни «Програмне забезпечення високопродуктивних обчислень» для студентів

спеціальності 121 «Інженерія програмного забезпечення» [Електронний ресурс] – Режим доступу до ресурсу: https://iq.vntu.edu.ua/repository/card.php?id=8220

12. The Golden Book of Python 2024 Edition / Diego Rodrigues - Diego Rodrigues, 2024. 226p.

13. The Quick Python Book / Naomi Ceder – Magging, 2021. 472p.

14. JavaScript JSON CookBook / Ray Rischpater - Packt Publishing, 2015. 192p.

15. JSON Quick Syntax Reference / Wallace Jackson - Apress, 2016. 142p.

16. JSON Data Basics / Frank Wellington - Publifye AS, 2025. 249p.

17. Python GUI Programming with Tkinter / Alan D. Moore - Packt Publishing, 2018. 452p.

18. Конструктор UML-діаграм [Електронний ресурс] – Режим доступу до ресурсу: https://draw.io

19. Різниця між 2D і 3D графікою [Електронний ресурс] – Режим доступу до ресурсу: https://cgischool.ua/2d-i-3d-grafika-riznytsya-ta-yaku-specialnist-obraty/

20. Романюк О. Н., Катєльніков Д. І. Веб-дизайн і комп'ютерна графіка [Електронний ресурс] – Режим доступу до ресурсу: https://iq.vntu.edu.ua/method/getfile.php?fname=59360.pdf&x=1&card_id=2964&id=59360

21. Що таке логотип, і для чого він потрібен? [Електронний ресурс] – Режим доступу до ресурсу: https://evopack.com.ua/logotyp-firmovyj-znak-emblema-v-chomu-vidminnosti/

22. Як створити логотип [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/blog/how-to-create-a-logo

23. Що таке алгоритми в програмуванні? [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/shho-take-algorytmy-kroky-pryklady-konstrukcziyi-myslennya/

24. Тестування програмного забезпечення: типи, види та застосування [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/

25. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

59

Додаток А – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
"25" березня 2025 р.

Технічне завдання

На бакалаврську кваліфікаційну роботу «Розробка програмного комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доцент. Р.Ю. Чехместрук
"24" березня 2025 р.
Виконав:
студент гр.5ПІ-21б В.В. Швець
"24" березня 2025 р.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмного комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях».

Галузь застосування – інформаційні технології

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 року ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки

Метою роботи є підвищення точності інтелектуального моніторингу клієнтського потоку у кав’ярнях за рахунок використання даних про замовлення, які створюються клієнтами під час відвідування закладу.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Coffee Shop Inventory Management: Tips, tools and techniques [Електронний ресурс] – Режим доступу до ресурсу: https://supy.io/blog/coffee-shop-inventory-management-tools-and-techniques/

2. Python GUI Programming with Tkinter / Alan D. Moore - Packt Publishing, 2018. 452p.

3. JSON Data Basics / Frank Wellington - Publifye AS, 2025. 249p.

4. The Golden Book of Python 2024 Edition / Diego Rodrigues - Diego Rodrigues, 2024. 226p.

5. Технічні вимоги

Середовище розробки: Visual Studio, мова програмування: Python, вхідні дані: необроблена інформація про клієнтський потік, вихідні дані: створені діаграми популярності пропозицій, відвідування клієнтів.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської кваліфікаційної роботи;
– технічне завдання;
– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз підходів до інтелектуального моніторингу клієнтського потоку в кав'ярнях	25.03.2025 – 30.03.2025
2	Розробка моделі та алгоритмів програмного застосунку	30.03.2025 – 19.04.2025
3	Розробка програмних модулів системи інтелектуального моніторингу клієнтського потоку в кав'ярнях	19.04.2025 – 10.05.2025
4	Розробка програмного застосунку	10.05.2025 – 24.05.2025
5	Тестування програмного застосунку	24.05.2025 – 27.05.2025
6	Оформлення матеріалів до захисту БКР	27.05.2025 – 30.05.2025

10 Порядок контролю та прийняття

Усі етапи бакалаврської кваліфікаційної роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

62

Додаток Б – Протокол перевірки на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення факультет інформаційних технологій та комп'ютерної інженерії, 5ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 7,3 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Чехместрук Р. Ю., к.т.н., доцент
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Швець В. В.
(підпис) (прізвище, ініціали)

Додаток В – Лістинг програми

```
ORDERS_FILE = "data/orders.json"

def show_about_window():
    about_window = Toplevel()
    about_window.title("Про програму")
    about_window.geometry("400x300")

    about_text = """
    CoffeManager

    CoffeManager - програмний застосунок для автоматичного моніторингу і аналізу клієнтського потоку у кав'ярнях.

    Розробник: Швець Василь Васильович
    Група 5ПІ-21б

    Програмний застосунок є абсолютно безкоштовним.

    Якщо у вас виникли питання щодо використання програмного застосунку, або ви зіткнулися з будь-якими проблемами - звертайтесь до службової адреси технічної підтримки:
    vasylshvets5pi@gmail.com
    """

    label = tk.Label(about_window, text=about_text, justify=tk.LEFT, padx=10, pady=10)
    label.pack(fill="both", expand=True)

def start_app():
    root = tk.Tk()
    root.title("Система моніторингу замовлень у кав’ярні")
    root.geometry("900x650")

    # Створення меню
    menu_bar = tk.Menu(root)
```

```
    # Створення меню "Про програму"

    about_menu = tk.Menu(menu_bar, tearoff=0)

    about_menu.add_command(label="Про програму", command=show_about_window)

    menu_bar.add_cascade(label="Довідка", menu=about_menu)

    # Підключаємо меню до головного вікна
    root.config(menu=menu_bar)

    notebook = ttk.Notebook(root)
    notebook.pack(expand=True, fill="both")

    tabs = {
        "Меню кав’ярні": tk.Frame(notebook),

        "Генерація клієнтів": tk.Frame(notebook),

        "Список замовлень": tk.Frame(notebook),

        "Популярність позицій": tk.Frame(notebook),

        "Активність за годинами": tk.Frame(notebook)
    }

    for name, frame in tabs.items():
        notebook.add(frame, text=name)

    menu_tab = tabs["Меню кав’ярні"]
    menu = load_menu()

    tree = ttk.Treeview(menu_tab, columns=("Назва", "Ціна"), show="headings")
    tree.heading("Назва", text="Назва")

    tree.heading("Ціна", text="Ціна, грн")
    tree.pack(pady=10, fill="both", expand=True)

def refresh_menu():
```

```
    for row in tree.get_children():
        tree.delete(row)
    for item in menu:
        tree.insert("", "end", values=(item["назва"], item["ціна"]))


def add_item():
    name = entry_name.get().strip()
    try:
        price = float(entry_price.get())

    except ValueError:
        messagebox.showerror("Помилка", "Ціна має бути числом")
        return

    if not name:
        messagebox.showerror("Помилка", "Назва не може бути порожньою")
        return

    menu.append({"назва": name, "ціна": price})
    save_menu(menu)
    refresh_menu()

    entry_name.delete(0, tk.END)

    entry_price.delete(0, tk.END)

form_frame = tk.Frame(menu_tab)
form_frame.pack(pady=10)


tk.Label(form_frame, text="Назва:").grid(row=0, column=0, padx=5)
entry_name = tk.Entry(form_frame)
entry_name.grid(row=0, column=1, padx=5)


tk.Label(form_frame, text="Ціна:").grid(row=0, column=2, padx=5)
entry_price = tk.Entry(form_frame)
entry_price.grid(row=0, column=3, padx=5)
```

```
    btn_add = tk.Button(form_frame, text="Додати позицію", command=add_item)
    btn_add.grid(row=0, column=4, padx=10)

    refresh_menu()

generate_tab = tabs["Генерація клієнтів"]


    listbox = tk.Listbox(generate_tab, width=100)
    listbox.pack(pady=10, fill="both", expand=True)



    def generate_and_show():
        listbox.delete(0, tk.END)
        orders = generate_orders()
        for order in orders:
            items_str = ", ".join([item["назва"] for item in order["позиції"]])
            listbox.insert(tk.END, f"{order['клієнт']} ({order['час']}): {items_str}")


    btn_gen = tk.Button(generate_tab, text="Згенерувати клієнтів",
command=generate_and_show)
    btn_gen.pack(pady=10)

    orders_tab = tabs["Список замовлень"]



    orders_tree = ttk.Treeview(orders_tab, columns=("Клієнт", "Час", "Позиції"),
show="headings")
    orders_tree.heading("Клієнт", text="Клієнт")

    orders_tree.heading("Час", text="Час")

    orders_tree.heading("Позиції", text="Позиції")

    orders_tree.pack(fill="both", expand=True, pady=10)
```

```
    def load_orders():
        try:
            with open(ORDERS_FILE, "r", encoding="utf-8") as f:
                return json.load(f)
        except FileNotFoundError:
            return []

    def refresh_orders():
        orders_tree.delete(*orders_tree.get_children())
        orders = load_orders()
        for order in orders:
            items_str = ", ".join([item["назва"] for item in order["позиції"]])
            orders_tree.insert("", "end", values=(order["клієнт"], order["час"], items_str))

    btn_refresh = tk.Button(orders_tab, text="Оновити список замовлень",
command=refresh_orders)
    btn_refresh.pack(pady=5)

    refresh_orders()

    popularity_tab = tabs["Популярність позицій"]

    def show_popularity():
        for widget in popularity_tab.winfo_children():
            widget.destroy()

        orders = load_orders()
        counter = Counter()
        for order in orders:
            for item in order["позиції"]:
                counter[item["назва"]] += 1

        if not counter:
            tk.Label(popularity_tab, text="Немає даних для відображення").pack()
            return

        labels, values = zip(*counter.most_common())

        fig, ax = plt.subplots(figsize=(7, 4))
        ax.bar(labels, values, color='skyblue')
```

```
        ax.set_title("Популярність позицій")
        ax.set_ylabel("Кількість замовлень")
        ax.set_xticklabels(labels, rotation=45, ha="right")

        canvas = FigureCanvasTkAgg(fig, master=popularity_tab)
        canvas.draw()
        canvas.get_tk_widget().pack(fill="both", expand=True)

    btn_chart = tk.Button(popularity_tab, text="Показати діаграму", command=show_popularity)
    btn_chart.pack(pady=10)

    hours_tab = tabs["Активність за годинами"]


def show_hourly_activity(tab):
        for widget in tab.winfo_children():
            widget.destroy()

        total_clients = 200
        hours_distribution = {



        hours = sorted(hours_distribution.keys())
        counts = [hours_distribution[hour] for hour in hours]

        fig, ax = plt.subplots(figsize=(7, 4))
        ax.bar(hours, counts, color='orange')
        ax.set_title("Активність за годинами (штучні дані)")
        ax.set_xlabel("Година")
        ax.set_ylabel("Кількість замовлень")
        ax.set_xticks(hours)
        ax.set_xticklabels([f"{h:02d}:00" for h in hours], rotation=45, ha="right")

        canvas = FigureCanvasTkAgg(fig, master=tab)
        canvas.draw()
        canvas.get_tk_widget().pack(fill="both", expand=True)

    btn_chart2 = tk.Button(hours_tab, text="Показати графік", command=lambda: show_hourly_activity(hours_tab))
```

```
    btn_chart2.pack(pady=10)

root.mainloop()

@startuml

title UML-діаграма застосунку моніторингу замовлень у кав’ярні

package "Інтерфейс користувача (GUI)" {
    class MainWindow {
        +start_app()
    }


    class MenuTab {
        +refresh_menu()
        +add_item()
    }


    class GenerateClientsTab {
        +generate_and_show()
    }


    class OrdersTab {
        +refresh_orders()
    }


    class PopularityTab {
        +show_popularity()
    }


    class HourlyActivityTab {
        +show_hourly_activity()
    }
}
```

```
package "Утиліти" {
   class MenuManager {
      +load_menu()
      +save_menu(menu)
   }

   class OrderGenerator {
+generate_orders()
   }
}

package "Дані" {
   class Order {
      -клієнт: str
      -час: str
      -позиції: List[Item]
   }

   class Item {
      -назва: str
      -ціна: float
   }
}

MainWindow --> MenuTab

MainWindow --> GenerateClientsTab

MainWindow --> OrdersTab

MainWindow --> PopularityTab

MainWindow --> HourlyActivityTab


MenuTab --> MenuManager

OrdersTab --> Order
```

```
PopularityTab --> Order

HourlyActivityTab --> Order

GenerateClientsTab --> OrderGenerator
Order --> Item

@startuml
title Діаграма активностей: Побудова діаграми популярності позицій

start

:Натискання на вкладку "Популярність позицій";

:Завантаження списку замовлень (Order list);


if (Список замовлень порожній?) then (так)
   :Показ повідомлення "Немає даних";
   stop
else (ні)
   :Створення словника для підрахунку позицій;
   :Прохід по кожному замовленню;

   :Прохід по кожній позиції у замовленні;
   :Збільшення лічильника для кожної позиції;

   :Сортування позицій за кількістю замовлень;
   :Формування списку назв позицій;

   :Формування списку кількостей;

   :Побудова кругової діаграми (PieChart);
   :Відображення діаграми у вікні;

endif


# --- Оформити замовлення ---
manual_tab = tabs["Оформити замовлення"]
```

```
tk.Label(manual_tab, text="Клієнт:").pack(pady=5)
entry_customer = tk.Entry(manual_tab)
entry_customer.pack(pady=5)

tk.Label(manual_tab, text="Оберіть позиції:").pack(pady=5)

menu_items_var = {}
menu_checkbuttons = []

for item in menu:
    var = tk.BooleanVar()
    chk = tk.Checkbutton(manual_tab, text=f"{item['назва']} ({item['ціна']} грн)",
variable=var)
    chk.pack(anchor='w')
    menu_items_var[item["назва"]] = (var, item)

def submit_order():
    name = entry_customer.get().strip()
    if not name:
        messagebox.showerror("Помилка", "Ім’я клієнта не може бути порожнім")
        return

    selected_items = [item for item, (var, obj) in menu_items_var.items() if var.get()]
    if not selected_items:
        messagebox.showerror("Помилка", "Оберіть хоча б одну позицію")
        return

    order = {
```

```
        "клієнт": name,
        "час": datetime.datetime.now().strftime("%Y-%m-%d %H:%M"),
        "позиції": [menu_items_var[item][1] for item in selected_items]
    }

    orders = load_orders()
    orders.append(order)

    with open(ORDERS_FILE, "w", encoding="utf-8") as f:
        json.dump(orders, f, ensure_ascii=False, indent=2)

    messagebox.showinfo("Успіх", "Замовлення додано!")
    entry_customer.delete(0, tk.END)
    for var, _ in menu_items_var.values():
        var.set(False)

btn_submit_order = tk.Button(manual_tab, text="Оформити замовлення",
command=submit_order)
btn_submit_order.pack(pady=10)

def load_orders():
    try:
        with open(ORDERS_FILE, "r", encoding="utf-8") as f:
            return json.load(f)
    except FileNotFoundError:
        return []

    def refresh_orders():

        orders_tree.delete(*orders_tree.get_children())

        orders = load_orders()
```

```
        for order in orders:

            items_str = ", ".join([item["назва"] for item in order["позиції"]])

            orders_tree.insert("", "end", values=(order["клієнт"], order["час"], items_str))

    btn_refresh = tk.Button(orders_tab, text="Оновити список замовлень", command=refresh_orders)
    btn_refresh.pack(pady=5)

    refresh_orders()

    popularity_tab = tabs["Популярність позицій"]

    def show_popularity():
        for widget in popularity_tab.winfo_children():
            widget.destroy()

        orders = load_orders()
        counter = Counter()

        for order in orders:
            for item in order["позиції"]:
                counter[item["назва"]] += 1

        if not counter:
            tk.Label(popularity_tab, text="Немає даних для відображення").pack()
            return

        labels, values = zip(*counter.most_common())

        fig, ax = plt.subplots(figsize=(7, 4))
        ax.bar(labels, values, color='skyblue')
        ax.set_title("Популярність позицій")

        ax.set_ylabel("Кількість замовлень")
        ax.set_xticklabels(labels, rotation=45, ha="right")

        canvas = FigureCanvasTkAgg(fig, master=popularity_tab)
```

```
        canvas.draw()
        canvas.get_tk_widget().pack(fill="both", expand=True)

    btn_chart = tk.Button(popularity_tab, text="Показати діаграму",
command=show_popularity)
    btn_chart.pack(pady=10)

    hours_tab = tabs["Активність за годинами"]


def show_hourly_activity(tab):
        for widget in tab.winfo_children():
            widget.destroy()

        total_clients = 200
        hours_distribution = {



        hours = sorted(hours_distribution.keys())
        counts = [hours_distribution[hour] for hour in hours]

        fig, ax = plt.subplots(figsize=(7, 4))
        ax.bar(hours, counts, color='orange')
        ax.set_title("Активність за годинами (штучні дані)")

        ax.set_xlabel("Година")
        ax.set_ylabel("Кількість замовлень")

        ax.set_xticks(hours)
        ax.set_xticklabels([f"{h:02d}:00" for h in hours], rotation=45, ha="right")

        canvas = FigureCanvasTkAgg(fig, master=tab)
        canvas.draw()
        canvas.get_tk_widget().pack(fill="both", expand=True)



USERS_FILE = "data/users.json"
```

```
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()



def load_users():
    if not os.path.exists(USERS_FILE):

        return {}
    with open(USERS_FILE, "r", encoding="utf-8") as f:
        return json.load(f)



def save_users(users):
    with open(USERS_FILE, "w", encoding="utf-8") as f:
        json.dump(users, f, ensure_ascii=False, indent=2)



def register_user():
    def register():

        username = entry_username.get().strip()
        password = entry_password.get().strip()

        if not username or not password:
            messagebox.showerror("Помилка", "Логін та пароль не можуть бути
порожніми")
            return

        users = load_users()
```

```
        if username in users:
            messagebox.showerror("Помилка", "Користувач уже існує")
            return

        users[username] = hash_password(password)
        save_users(users)

        messagebox.showinfo("Успіх", "Реєстрація успішна. Тепер увійдіть у систему.")
        reg_win.destroy()

    reg_win = tk.Toplevel()
    reg_win.title("Реєстрація")

    tk.Label(reg_win, text="Логін:").pack(pady=5)
    entry_username = tk.Entry(reg_win)

    entry_username.pack(pady=5)

    tk.Label(reg_win, text="Пароль:").pack(pady=5)

    entry_password = tk.Entry(reg_win, show="*")

    entry_password.pack(pady=5)

    tk.Button(reg_win, text="Зареєструватися", command=register).pack(pady=10)


def login_user():
    username = entry_login.get().strip()
```

```
    password = entry_password.get().strip()

    users = load_users()
    if username in users and users[username] == hash_password(password):
        login_win.destroy()

        start_app()
    else:
        messagebox.showerror("Помилка", "Невірний логін або пароль")



# --- Вікно входу ---
login_win = tk.Tk()
login_win.title("Вхід у систему")
login_win.geometry("300x200")

frame = tk.Frame(login_win)
frame.pack(pady=20)

tk.Label(frame, text="Логін:").grid(row=0, column=0, pady=5)
entry_login = tk.Entry(frame)
entry_login.grid(row=0, column=1, pady=5)

tk.Label(frame, text="Пароль:").grid(row=1, column=0, pady=5)
entry_password = tk.Entry(frame, show="*")
entry_password.grid(row=1, column=1, pady=5)

btn_login = tk.Button(login_win, text="Увійти", width=20, command=login_user)
```

```
btn_login.pack(pady=5)


btn_register = tk.Button(login_win, text="Зареєструватися", width=20,
command=register_user)
btn_register.pack()


login_win.mainloop()
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ КЛІЄНТСЬКОГО ПОТОКУ В КАВ’ЯРНЯХ

Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ КЛІЄНТСЬКОГО ПОТОКУ В КАВ’ЯРНЯХ»

Автор: ст. групи 5ПІ-21б Швець В. В.
Науковий керівник: к.т.н., доц. каф. ПЗ Чехместрук Р. Ю.

Вінниця ВНТУ – 2025

Рисунок Г.1 – Титульний слайд

Актуальність теми

Сучасний етап розвитку інформаційних технологій вимагає постійного вдосконалення підходів до збору, аналізу та обробки даних, зокрема в сфері обслуговування. Однією з актуальних задач є створення систем, здатних точно та надійно відображати інформацію про потік клієнтів у закладах громадського харчування, зокрема в кав’ярнях.

Традиційні методи моніторингу клієнтської активності зазвичай базуються на візуальних підходах — встановлення камери, підрахунок людей за допомогою сенсорів або спостереження персоналу. Проте такі методи часто є обмеженими за точністю, потребують складного технічного обладнання і не завжди забезпечують повну відповідність реальній кількості клієнтів. У цьому контексті, використання даних про замовлення, створені клієнтами, відкриває нові можливості для об'єктивного та автоматизованого обліку відвідувачів без необхідності використання дорогих апаратних засобів.

Рисунок Г.2 – Актуальність теми

Мета, об’єкт та предмет дослідження

Мета та завдання дослідження. Метою дослідження є підвищення точності інтелектуального моніторингу клієнтського потоку у кав’ярнях за рахунок використання даних про замовлення, які створюються клієнтами під час відвідування закладу. Проєкт передбачає розробку низки програмних модулів, які будуть об’єднані в єдину інформаційну систему.

Об'єкт дослідження – процес розробки програмного застосунку для інтелектуального моніторингу клієнтського потоку.

Предмет дослідження – методи та програмні засоби розробки програмного застосунку для інтелектуального моніторингу клієнтського потоку.

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

Завдання дослідження

Основними задачами дослідження є:

- розробити модуль збору інформації про замовлення клієнтів, що дозволить автоматично фіксувати факти відвідування без потреби у візуальному спостереженні;

- створенити модуль обробки замовлень з можливістю аналізу кількості клієнтів за певні часові інтервали;

- реалізувати модуль оцінки популярності послуг на основі частоти замовлень конкретних позицій з меню;

- створити модуль графічного інтерфейсу користувача, орієнтованого на працівників адміністрації, з інтуїтивно зрозумілим дизайном;

- провести тестування розроблених програмних модулів.

Рисунок Г.4 – Завдання дослідження

Наукова новизна отриманих результатів

Подальшого розвитку отримав метод побудови системи для керування інтелектуальним потоком клієнтів, у якому, на відміну від існуючих використано технологію JSON, що дозволило спростити розробку та підвищити швидкість відтворення діаграм на 0.765 мілісекунд.

Подальшого розвитку отримав метод визначення статистики відвідування кав’ярні клієнтським потоком для керування бізнес процесами кав’ярень у якому, на відміну від існуючих визначаються вибірка позицій у меню клієнтами.

Рисунок Г.5 – Наукова новизна

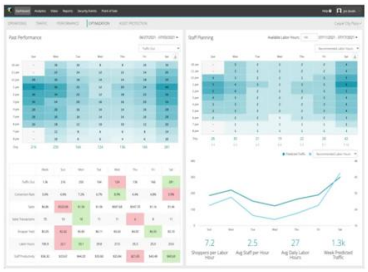
Практична цінність отриманих результатів

Практична цінність отриманих результатів полягає в тому, що на основі теоретичних досліджень розроблено алгоритми та програми для інтелектуального моніторингу клієнтського потоку у кав'ярнях.

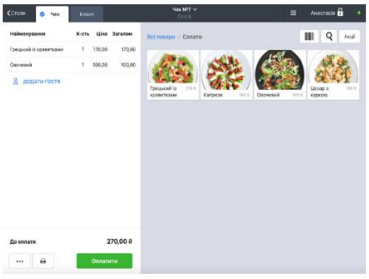
Рисунок Г.6 – Практична цінність

Аналоги

Retailnext

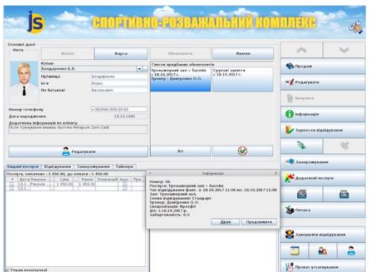


Poster POS



POS Sector

JSolutions



PayKit

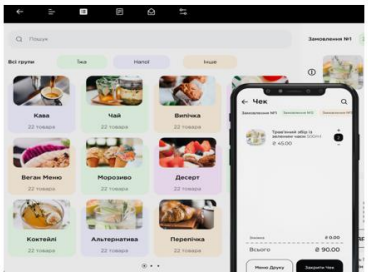


Рисунок Г.7 – Аналоги

Порівняльний аналіз аналогів

Критерії	RetailNext	Poster POS	POS Sector	JSolutions	PayKit	Власна розробка
Точність вимірювань	-	+	-	-	+	+
Аналіз відвідуваності клієнтів	+	+	+	+	+	+
Автоматизація вимірювання	-	-	+	+	+	+
Аналіз популярності замовлень	+	-	-	-	-	+
Швидкість та зручність вимірювання	-	+	-	+	+	+
Сумарний бал	2	3	2	3	4	5

Рисунок Г.8 – Порівняльний аналіз аналогів

Використані технології

Python

JSON

Matplotlib

Tkinter

Рисунок Г.9 – Використані технології

Розробка алгоритму роботи програмного застосунку

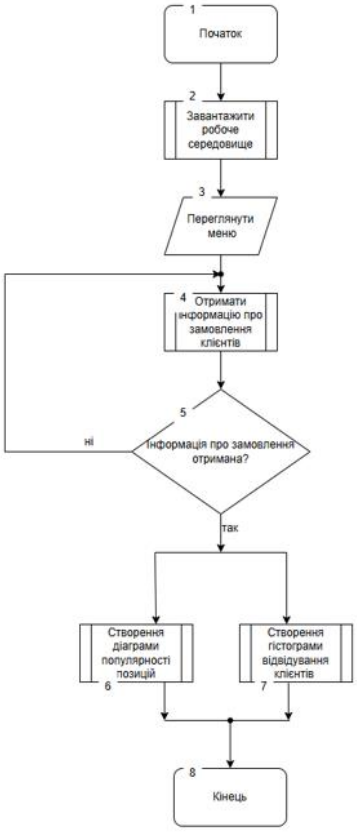


Рисунок Г.10 – Розробка алгоритму роботи програмного застосунку

Розробка алгоритму для відображення активності клієнтів

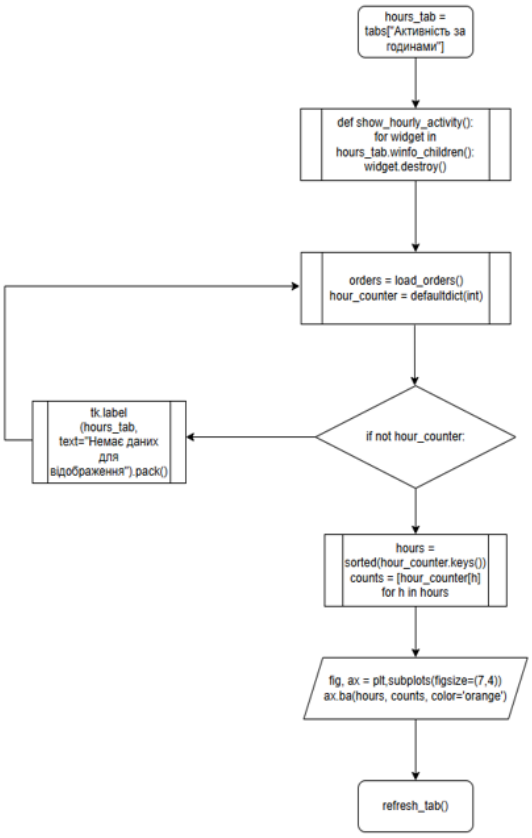


Рисунок Г.11 – Розробка алгоритму для відображення активності клієнтів

Тестування програмного застосунку

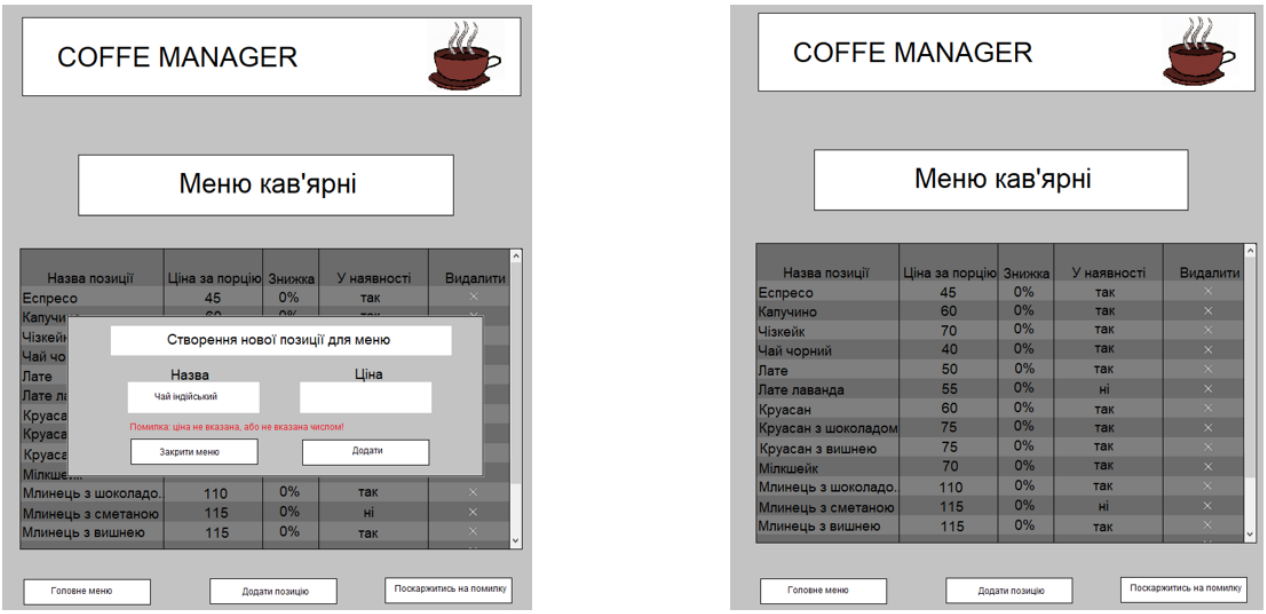


Рисунок Г.12 – Тестування програмного застосунку (частина 1)

Тестування програмного застосунку

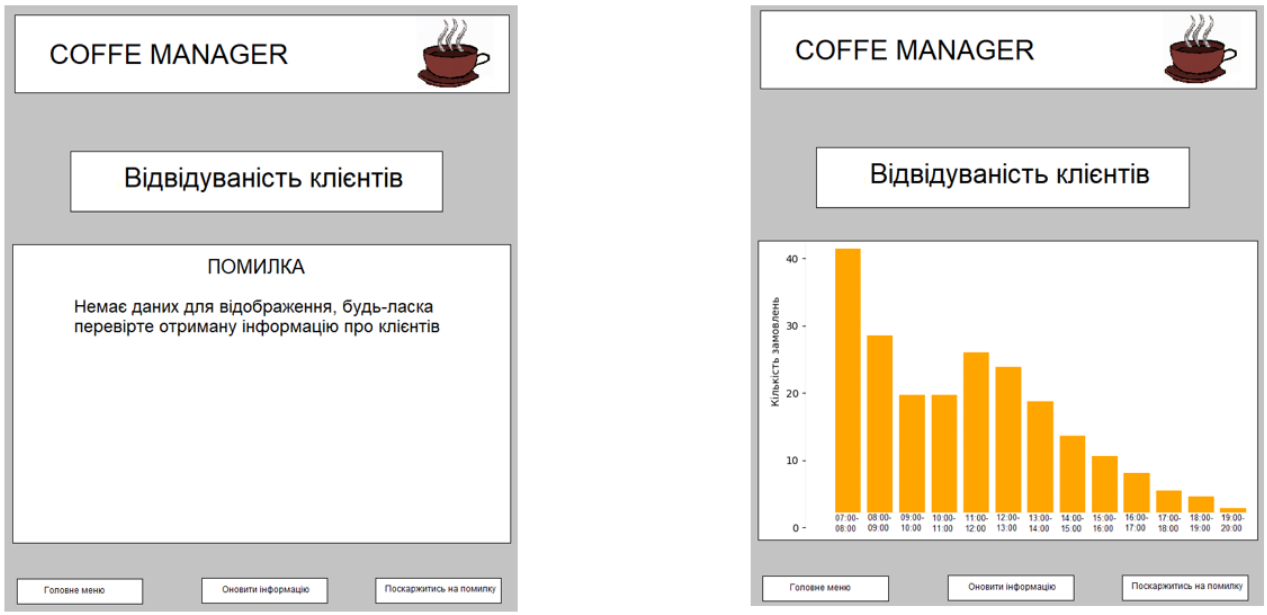


Рисунок Г.13 – Тестування програмного застосунку (частина 2)

Висновки

У бакалаврській кваліфікаційні роботі було розроблено методи та програмні засоби комплексу для інтелектуального моніторингу клієнтського потоку в кав'ярнях.

Проведено аналіз сучасних підходів до інтелектуального моніторингу клієнтського потоку. Обгрунтовано необхідність розробки методів виконання інтелектуального моніторингу клієнтського потоку з метою підвищення точності вимірювання отриманих даних. Бакалаврська кваліфікаційна робота оформлена з урахуванням рекомендацій [25].

Під час аналізу засобів розробки було обгрунтовано вибір мови програмування Python, бібліотек Tkinter та Matplotlib, та відповідних технологій для реалізації системи для інтелектуального моніторингу клієнтського потоку в кав’ярнях.

У першому розділі роботи проведено аналіз предметної області, розглянуто існуючі програмні рішення з інтелектуальним моніторингом клієнтського потоку, окреслено їхні переваги та недоліки. Особливу увагу приділено критеріям ефективності аналізу інформації отриманої з замовлень клієнтів.

У другому розділі представлено розгорнутий опис графічної моделі програмного застосунку, був розроблений логотип для програмного застосунку, який застосовано у розробці.

У третьому розділі описано архітектуру застосунку, викладено інформацію щодо процесу розробки програмних модулів, необхідних для повноцінного функціонування програмного застосунку. Наведено огляд сучасних технологій та інструментальних засобів, застосованих для розробки, з обгрунтуванням їх вибору. Описано внутрішню структуру та принципи взаємодії розроблених модулів для моніторингу клієнтського потоку.

У четвертому розділі проведено тестування розробленого проєкту. Продемонстровано основні екрани застосунку, описано сценарії використання. Тестування довело працездатність та коректність роботи програмного застосунку.

Рисунок Г.14 – Висновки

Апробація результатів роботи і публікації

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення БКР доповідались та обговорювались на Міжнародній науково-практичній конференції «Research in Science, Technology and Economics» у 2025 році.

Публікації. Основні результати досліджень опубліковано в матеріалах конференції: «Research in Science, Technology and Economics»

Рисунок Г.15 – Апробація результатів роботи і публікації