МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

**КОРОЛЕНКО СЕРГІЙ ОЛЕКСІЙОВИЧ**

Допускається до захисту:
В.о. завідувача кафедри інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«____»_____ 20__ р.

# СИСТЕМА ЕЛЕКТРОННОЇ ЧЕРГИ ГРОМАДСЬКОГО ЗАКЛАДУ

Спеціальність 122 Комп'ютерні науки

**Кваліфікаційна (бакалаврська) робота**

Керівник:
Веселовська Н.Р. , професор кафедри
інформаційних технологій,
д.т.н., професор
_____

Оцінка: _____/_____/_____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____

Вінниця - 2025

## АНОТАЦІЯ

Короленко Сергій Олексійович «Система електронної черги громадського закладу»

Дипломна бакалаврська робота за спеціальністю 122 - «Комп'ютерні науки» - Донецький національний університет імені Василя Стуса, Факультет інформаційних і прикладних технологій, Вінниця, 2025 рік. У кваліфікаційній (бакалаврській) роботі досліджено особливості побудови та функціонування систем електронної черги, проаналізовано сучасні підходи до автоматизації процесів обслуговування відвідувачів у громадських закладах. Для реалізації використано фреймворк Flask, мову програмування Python, систему управління базами даних MySQL, а також технології HTML, CSS та JavaScript.

Ключові слова: електронна черга, веб-додаток, Flask, Python, MySQL, автоматизація обслуговування.

53 с. 17 рис., 2 табл., 44 джерела.

## ABSTRACT

Korolenko Serhii Oleksiiovych " Queue System of a Public Institution"

Bachelor's thesis in specialty 122 - "Computer Science" - Vasyl' Stus Donetsk National University, Faculty of Information and Applied Technologies, Vinnytsia, 2025. The qualification (bachelor's) thesis investigates the features of the construction and functioning of electronic queue systems, analyzes modern approaches to automating the processes of serving visitors in public institutions. The Flask framework, Python programming language, MySQL database management system, and HTML, CSS, and JavaScript technologies were used for implementation.

Keywords: electronic queue, web application, Flask, Python, MySQL, service automation.

# ЗМІСТ

# ВСТУП

**Актуальність теми:**

У сучасних умовах цифровізації суспільства організації все частіше стикаються з необхідністю оптимізації процесів взаємодії з клієнтами. Одним із таких процесів є організація черг. До традиційних форм черг відносяться: паперові списки, ручна реєстрація, усна фіксація, які на сьогодні втрачають актуальність через низьку ефективність, велику кількість конфліктних ситуацій, відсутність обліку та аналітики [5; 11; 34]. З цієї причини все більше установ - як державних, так і приватних - впроваджують електронні системи черг [1; 7; 36]. Однак багато таких рішень є надмірно складними, дорогими або не мають функціональності, необхідної для зручного користування як клієнтами, так і персоналом [3; 8; 20]. Тому питання удосконалення електронних черг з урахуванням гнучкості, зручності, інтеграції з базами даних і можливості віддаленого доступу - є вкрай актуальним. У межах цієї роботи реалізовано комп'ютеризовану систему електронної черги з використанням Python-фреймворку Flask та СУБД MySQL [22; 23; 35], орієнтовану на практичне впровадження в умовах державних сервісів або медичних установ.

**Мета дослідження:**

Метою цієї кваліфікаційної роботи є розробка та впровадження удосконаленої комп'ютеризованої системи електронної черги [4], яка враховує недоліки існуючих реалізацій та реалізує ключові функції: реєстрацію, вхід, виклик клієнта, підтвердження по електронній пошті, адміністрування та автоматичне оновлення даних.

**Завдання дослідження:**

1. Провести теоретичний аналіз предметної області електронних черг, визначити їх типи, недоліки та сучасні тенденції.

2. Ознайомитися з існуючими програмними аналогами, виділити їх переваги та обмеження.
3. Сформулювати функціональні та нефункціональні вимоги до системи електронної черги.
4. Розробити структуру бази даних для зберігання інформації про користувачів та адміністраторів.
5. Реалізувати веб-застосунок за архітектурою MVC з використанням Flask та MySQL.
6. Додати модулі реєстрації з підтвердженням email, вхід адміністраторів, виклик та завершення обслуговування.
7. Провести тестування реалізованої системи, перевірити її працездатність у типових сценаріях.

**Об’єкт і предмет дослідження:**

Об’єкт дослідження - процес організації та управління електронною чергою в сервісних структурах.

Предмет дослідження - комп’ютеризована система електронної черги, реалізована з використанням Python (Flask) та MySQL.

**Теоретична та практична реалізація роботи:**

Теоретична реалізація полягає в узагальненні підходів до побудови сучасних систем керування чергами, критичному аналізі методів автоматизації сервісів та обґрунтуванні архітектурних рішень.

Практична реалізація проявляється у розробці готового програмного продукту, який може бути використаний в установах, що мають потребу в ефективній організації черговості обслуговування. Зокрема, систему можна впровадити в медичних центрах, центрах надання адміністративних послуг, університетах, сервісних службах та інших установах з великим потоком відвідувачів.

**Апробація отриманих результатів:**

Основні положення, висновки та результати кваліфікаційної (бакалаврської) роботи на тему «Удосконалення системи електронної черги громадського закладу» були апробовані шляхом підготовки тез доповіді та участі в науково-практичній конференції.

**Структура роботи:**

Робота складається зі вступу, трьох основних розділів, висновків, списку використаних джерел і додатків.

1. У першому розділі розглянуто теоретичні основи предметної області, поняття електронної черги, проблеми традиційних рішень, а також проведено огляд аналогів.
2. Другий розділ присвячено проєктуванню системи, опису її структури, вибору технологій та створенню бази даних.
3. У третьому розділі описано реалізацію веб-додатку, функціональні модулі, а також проведено тестування системи.

# РОЗДІЛ 1.
# Теоретичні основи та аналіз предметної області

## 1.1 Поняття та класифікація електронних черг

Електронна черга - це автоматизована інформаційна система [1; 4; 34], призначена для організації, керування та оптимізації процесу обслуговування клієнтів у різних сферах діяльності. Її основною метою є підвищення ефективності обслуговування, зменшення часу очікування та виключення фізичного скупчення людей у черзі.

У загальному вигляді система електронної черги складається з таких основних компонентів:

- термінал реєстрації або попереднього запису;
- система диспетчеризації (програмне забезпечення);
- інформаційне табло або екран;
- автоматизовані робочі місця операторів;
- мобільні/веб-додатки (опціонально).

Класифікація електронних черг може здійснюватися за кількома критеріями :
За характером організації обслуговування:

- Поетапна черга - клієнт проходить кілька етапів обслуговування (наприклад, реєстрація → консультація → оплата);
- Одномоментна черга - клієнт отримує послугу в одному пункті, без переходу;
- Гібридна - поєднання декількох типів обслуговування.

За способом керування чергою:

- Пріоритетна черга - клієнтам призначаються різні рівні пріоритету (наприклад, пенсіонери, інваліди);
- Звичайна (FIFO) - обслуговування відбувається в порядку надходження.

За місцем обслуговування:

- Локальні системи - встановлюються в одному об’єкті;

- Мережеві системи - функціонують у кількох філіях та мають централізоване управління.

Таким чином, електронна черга є гнучкою інструментальною системою, здатною адаптуватися до різних сценаріїв обслуговування [6; 7].

### 1.2 Проблеми традиційних систем черг та мотивація автоматизації

Традиційні системи черг, що базуються на фізичному очікуванні без формалізованого обліку, широко використовувалися в державних установах, лікарнях, банках, поштових відділеннях тощо. Проте, з розвитком цифрових технологій, такі підходи дедалі більше втрачають актуальність через низку об'єктивних недоліків, що знижують ефективність роботи організацій [5; 11; 38] та викликають незадоволення у клієнтів.

До основних проблем традиційних черг відносяться:

1. Відсутність централізованого контролю за послідовністю обслуговування. У паперових списках або усному обговоренні легко виникають суперечки, порушення черговості, конфлікти між відвідувачами, що веде до втрати довіри до установи.
2. Фізичне скупчення людей в одному місці. Особливо критично це проявляється в періоди епідеміологічних загроз (наприклад, COVID-19), коли скупчення клієнтів у холах та коридорах є фактором ризику для здоров'я [39].
3. Суб'єктивність і вплив людського чинника. Обслуговування «поза чергою», на основі особистих знайомств або суб'єктивної оцінки працівника, є поширеною проблемою, яка нівелює прозорість процесу.
4. Неможливість оперативного збору статистичних даних. У традиційних чергах відсутній механізм фіксації часу обслуговування, кількості клієнтів, середнього часу очікування, що унеможливлює аналітичний контроль навантаження та подальшу оптимізацію.

5. Відсутність попереднього запису або дистанційного доступу. Клієнт змушений особисто бути присутнім для реєстрації в черзі, що спричиняє додаткові витрати часу і ресурсів.
6. Низький рівень зручності для осіб з інвалідністю. Відсутність цифрової інфраструктури не дозволяє таким особам отримати послуги в доступному форматі без необхідності фізичної присутності.

Усе це формує стійкий попит на модернізацію [2; 12; 16] систем керування чергою через впровадження автоматизованих електронних рішень. Основними перевагами автоматизації є:

1. Цифровий контроль за черговістю. Програмна система забезпечує точний облік клієнтів [1; 20], унеможливлюючи порушення порядку.
2. Можливість дистанційної реєстрації. Через веб-інтерфейс або мобільний застосунок [13; 17] клієнт може зарезервувати час або місце в черзі, що особливо зручно для осіб із зайнятим графіком.
3. Оперативна аналітика. Система автоматично формує статистику щодо завантаженості, часу обслуговування, ефективності працівників тощо - ці дані можуть бути використані керівництвом для прийняття управлінських рішень.
4. Оптимізація ресурсів. Зменшення кількості чергових адміністраторів, усунення паперового документообігу, зниження навантаження на персонал.
5. Інтеграція з іншими модулями. Наприклад, з електронною поштою (для підтвердження запису), з табло (для виводу поточних чергових номерів), із базою даних користувачів.
6. Покращення клієнтського досвіду. Завдяки зручному інтерфейсу, прозорості системи, персональним сповіщенням (через e-mail або SMS), клієнти отримують більш комфортний сервіс.

Таким чином, автоматизація процесу керування чергою дозволяє не лише зменшити навантаження на персонал та уникнути черг фізичного типу, а й забезпечити якісно новий рівень обслуговування. Такий підхід має критичне

значення для організацій, які прагнуть підвищити ефективність роботи, мінімізувати конфлікти та підвищити задоволеність клієнтів. Ці переваги є основною мотивацією впровадження електронних черг у державних та комерційних структурах, а також служать підґрунтям для розробки удосконаленої системи, описаної в цій роботі.

### 1.3 Огляд існуючих підходів та програмних рішень

На ринку програмного забезпечення існує ряд готових рішень для керування електронними чергами, як українських, так і міжнародних відповідно до таблиці 1.1.

Таблиця 1.1 - Порівняльна характеристика сучасних систем Е-черги

| Система | Тип реалізації | Сфера застосування | Особливості |
|---|---|---|---|
| Qwaiting | SaaS | Банки, лікарні | Хмарна архітектура, аналітика |
| QLess | Мобільна/веб-платформа | Рітейл, держпослуги | Інтеграція з CRM |
| Qmatic | Стаціонарна + мобільна | Великі установи | Глибока кастомізація |
| Moviik | Хмарна | Університети, міграційні служби | Простота впровадження |
| QueueBee | Локальна + онлайн | Поліклініки, сервісні центри | Гнучка система пріоритетів |
| Дія/ЦНАП | Державна | Адміністративні послуги | Інтеграція з держсистемами |

Переваги таких рішень [28; 30]:

- масштабованість,
- багатоплатформність,

- аналітика в реальному часі.

Недоліки:

- закритий вихідний код,
- складність адаптації під специфічні задачі,
- залежність від сервера або хмари,
- потреба у ліцензії.

Таким чином, актуальним є створення кастомізованої, локальної системи, що враховує конкретні потреби установи, з можливістю подальшого масштабування.

### 1.4 Постановка задачі та вибір методів дослідження

У рамках даної кваліфікаційної роботи ставиться мета вдосконалення комп'ютеризованої системи електронної черги, яка б дозволяла забезпечити ефективний, зручний та автоматизований процес керування черговістю обслуговування користувачів у різних організаціях. Серед основних аспектів модернізації передбачається:

- Підвищення гнучкості конфігурування черги та можливість роботи з декількома адміністраторами одночасно;
- Інтуїтивно зрозумілий інтерфейс для клієнтів та адміністраторів, який сприятиме мінімізації часу навчання користувачів;
- Інформаційна інтеграція з базами даних, системами виводу інформації на табло та з іншими зовнішніми сервісами через API;
- Безпечна реєстрація користувачів з підтвердженням електронною поштою;
- Автоматизоване оновлення інформації в реальному часі, зокрема списку очікування та номера поточного користувача;
- Обмеження доступу адміністратора до дій інших адміністраторів, що забезпечує паралельну роботу без конфліктів.

Основні задачі дослідження:

1. Провести аналіз недоліків існуючих реалізацій електронних черг у державному та комерційному секторі.
2. Побудувати функціональну модель удосконаленої системи з урахуванням практичних потреб.
3. Визначити архітектуру та набір технологій, придатних для реалізації функціоналу в умовах сучасних вимог до надійності та масштабованості.
4. Розробити прототип системи, здійснити його тестування та оцінити ефективність у порівнянні з аналогами.

Методологія дослідження:

Прототипування - метод швидкої розробки інтерфейсів, який дозволив швидко створити перші версії сторінок користувача та адміністратора. Завдяки цьому можна було оперативно внести зміни до логіки системи без затрат на повну реалізацію. Прототипування було реалізоване за допомогою HTML/CSS (зі шрифтами Google Fonts, зокрема Poppins), а також шаблонізатора Jinja2, вбудованого у Flask.

Інженерія програмного забезпечення - методичний підхід до розробки, що передбачає структуроване проектування системи. Включає фази вимог, проєктування, реалізації, тестування та обслуговування. В основі лежать принципи чіткого поділу модулів, повторного використання коду та забезпечення надійності.

Аналіз аналогів - включав критичний огляд популярних рішень (Qwaiting, Qmatic, Дія, ЦНАП, МВС) щодо функціональності, гнучкості, інтеграційних можливостей. Це дозволило виділити ключові компоненти, які потребують поліпшення: швидкість оновлення інформації, можливість багатокористувацької адміністрації, зворотний зв’язок з користувачем тощо.

RUP (Rational Unified Process) - застосовується як ітеративна модель розробки ПЗ, яка передбачає циклічне вдосконалення рішення:

- Ініціація - аналіз предметної області, формулювання вимог [3; 4; 8].
- Проєктування - розробка структури БД, логіки взаємодії.

- Реалізація - кодування системи, тестування, інтеграція модулів.
- Впровадження - розгортання прототипу, перевірка на практиці.

Технології та засоби реалізації [22; 23; 25; 26; 27; 35]:

Flask - легкий веб-фреймворк на Python, обраний через гнучкість, модульність і підтримку RESTful API. Забезпечує обробку запитів, маршрутизацію, інтеграцію з шаблонами.

MySQL - реляційна СУБД, що застосовується для збереження даних користувачів, адміністративної інформації та стану черги. Має підтримку зовнішніх ключів, що дозволяє реалізувати зв'язки між таблицями users та admins.

SMTP + smtplib - використовується для реалізації механізму надсилання електронних листів підтвердження реєстрації користувача.

Jinja2 - шаблонізатор, який дозволяє формувати HTML-сторінки з динамічними даними (список черги, викликаний клієнт тощо).

JavaScript + AJAX - використовується на стороні клієнта для періодичного оновлення інтерфейсу без перезавантаження [25; 26; 31] сторінки (наприклад, Now Serving).

Werkzeug - бібліотека, що використовується для хешування паролів адміністраторів з метою безпечного зберігання в базі даних.

Усі ці методи та технології було обрано з огляду на їх відкритість, простоту розгортання та адаптацію до потреб проєкту. У підсумку це дозволяє розробити ефективну, безпечну та зручну систему електронної черги, що здатна до масштабування й подальшого вдосконалення.

### 1.5 Висновки до розділу 1

У першому розділі було здійснено комплексний аналіз теоретичних основ і предметної області, що стосується електронних черг як складових інформаційно-довідкових систем. Надано чітке визначення поняття «електронна черга», охарактеризовано її основні типи - поетапні, одномоментні, пріоритетні,

що дозволило закласти фундамент для систематизації функціональних можливостей подібних рішень.

Особливу увагу було приділено аналізу проблем традиційних паперових та ручних систем керування чергою. Серед них виокремлено: повільність обробки, незручність для клієнтів, відсутність гнучкості та високий рівень людського чинника. Це стало обґрунтуванням доцільності переходу на автоматизовані програмні рішення.

Огляд вітчизняних та зарубіжних реалізацій [1; 19; 36; 42; 43] (Qwaiting, QLess, Qmatic, Moviik, QueueBee, «Дія», ЦНАП м. Києва, електронна черга МВС України) дав змогу визначити поточний стан розвитку електронних черг, виявити їхні сильні сторони (інтеграція з електронною поштою та SMS, гнучкий запис, навантаження на адміністраторів, зворотний зв'язок) та слабкі місця (обмежена підтримка кількох адмінів, слабкий UI/UX, відсутність персоналізації та обмеження інтеграцій).

На основі проведеного аналізу було чітко сформульовано задачі даного дослідження, які охоплюють модернізацію логіки черги, розширення адміністративних прав без конфліктів, реалізацію підтвердження реєстрації через електронну пошту, а також забезпечення зручного та надійного інтерфейсу. Визначено застосовну методологію дослідження - поєднання інженерії програмного забезпечення, прототипування, аналізу аналогів та ітеративної моделі розробки RUP.

Отже, зміст проведеної роботи закладає надійну теоретичну та методологічну основу для подальшого проектування, реалізації й тестування вдосконаленої комп'ютеризованої системи електронної черги, що буде детально розглянуто в наступних розділах бакалаврської роботи.

# РОЗДІЛ 2.
# Розробка методики для «Удосконалення комп'ютеризованої системи електронної черги»

## 2.1. Формулювання функціональних і нефункціональних вимог

Етап формулювання вимог є критично важливим у процесі проєктування будь-якої інформаційної системи, оскільки він визначає, що саме повинна виконувати система та в яких умовах. У випадку комп'ютеризованої системи електронної черги, яка повинна забезпечити ефективне управління чергою в реальному часі, вимоги поділяються на функціональні та нефункціональні [3; 5].

Функціональні вимоги:

Цей клас вимог описує основну поведінку системи, яку очікує користувач або адміністратор.

1. Реєстрація користувача:
   - Система повинна надавати веб-інтерфейс, де користувач може ввести свої дані (ім'я, прізвище, електронну пошту).
   - Після заповнення форми реєстрації, система автоматично генерує унікальний номер черги [26] у форматі "AXX", де XX - порядковий номер.
   - Система повинна зберігати дані користувача у базі даних та надіслати лист підтвердження на вказану email-адресу, який містить унікальний токен для активації.
2. Підтвердження через email:
   - Користувач зобов'язаний підтвердити свою електронну адресу для того, щоб з'явитися у загальній черзі.
   - Після переходу за посиланням із листа, система має змінити статус «is_confirmed» [22; 23] на «True».
3. Вхід до адміністративної панелі:

- Адміністратор повинен мати можливість увійти в систему за допомогою логіну та пароля.
- Паролі зберігаються у вигляді хешів, що генеруються бібліотекою Werkzeug.security [32].
- Успішна аутентифікація відкриває доступ до панелі керування чергою.

4. Виклик користувача:
   - Адміністратор може обрати користувача зі списку "очікуючих" і здійснити його виклик.
   - При цьому статус користувача змінюється на «called», а в полі «called_by_admin_id» зберігається ID поточного адміністратора.
   - Інші адміністратори не бачать викликаного користувача у своєму списку.
5. Завершення обслуговування:
   - Після надання послуги, адміністратор натискає кнопку Фініш, і статус користувача змінюється на «finished».
   - Система автоматично дозволяє виклик наступного клієнта в черзі.
6. Автоматичне оновлення інформації:
   - Реалізовано API-інтерфейси для:
     - /api/now_serving - повертає поточний номер у черзі.
     - /api/queue_list - повертає список усіх очікуючих клієнтів.
     - /api/admin_queue - для завантаження актуального стану черги в адмін-панелі.

Нефункціональні вимоги:

Цей тип вимог визначає якість системи - наскільки ефективно, безпечно, зручно вона працює.

1. Безпека:
   - Всі паролі зберігаються у хешованому вигляді (SHA256 + Salt) для запобігання витоку даних.

- Доступ до адміністративної панелі обмежено лише аутентифікованими користувачами.
- Підтвердження пошти захищене токенами, які мають термін дії.

2. Масштабованість:
   - Система підтримує додавання кількох адміністраторів.
   - База даних розрахована на розширення структури таблиць (наприклад, логування дій або обробку черг у декількох відділеннях).
3. Доступність:
   - Клієнтський інтерфейс розроблений на базі HTML/CSS з адаптацією під мобільні пристрої.
   - Використання шрифту Poppins забезпечує сучасний і легкий для сприйняття інтерфейс.
4. Надійність:
   - Система не допускає одночасного виклику одного користувача декількома адміністраторами.
   - Записи у базу даних проходять перевірку на дублікат email або черговий номер.
5. Інтеграція:
   - Для відправки підтверджень використовується SMTP-сервер (наприклад, Gmail або локальний Postfix) [24; 40].
   - Можлива подальша інтеграція з табло або системами оповіщення (через API).
6. Юзабіліті (Usability):
   - Мінімальна кількість кроків для дій користувача [30] (реєстрація → підтвердження → виклик).
   - Усі повідомлення формуються зрозумілою мовою, передбачена валідація форм.

Таким чином, формалізація вимог надає чітке бачення архітектури системи, дозволяє зменшити ризики неправильного трактування задач та полегшує перевірку результатів при реалізації й тестуванні функціоналу.

### 2.2. Проєктування логічної структури системи

Ефективна реалізація системи електронної черги потребує попереднього проєктування її логічної структури. Це дає змогу чітко зрозуміти, як дані будуть передаватися між користувачами, адміністраторами, системними процесами та базою даних. На цьому етапі було розроблено дві основні моделі: діаграму потоків даних (DFD) та ER-діаграму (схему сутність-зв'язок) [10; 14; 42].

Модель потоків даних (Data Flow Diagram):

DFD-діаграма дозволяє описати інформаційні потоки в системі та основні процеси [2; 29], які ці потоки генерують або споживають . Вона допомагає візуалізувати функціональні компоненти, підсистеми та взаємодії між ними.

У розробленій системі електронної черги виділено такі основні учасники процесів:

- Користувач - особа, яка реєструється в системі для отримання чергового талона та подальшого виклику.
- Адміністратор - оператор або співробітник установи, який керує чергою: викликає, обслуговує та завершує клієнтів.
- Система черги - сукупність серверних процесів, які обробляють запити, зберігають дані, надсилають листи, оновлюють статуси.
- База даних - сховище для інформації про користувачів, адміністраторів, чергові номери та стани [15].

Ключові процеси:

- Реєстрація користувача.
- Підтвердження реєстрації через email.
- Вхід адміністратора.
- Виклик користувача.

- Завершення обслуговування.
- Оновлення табло черги.

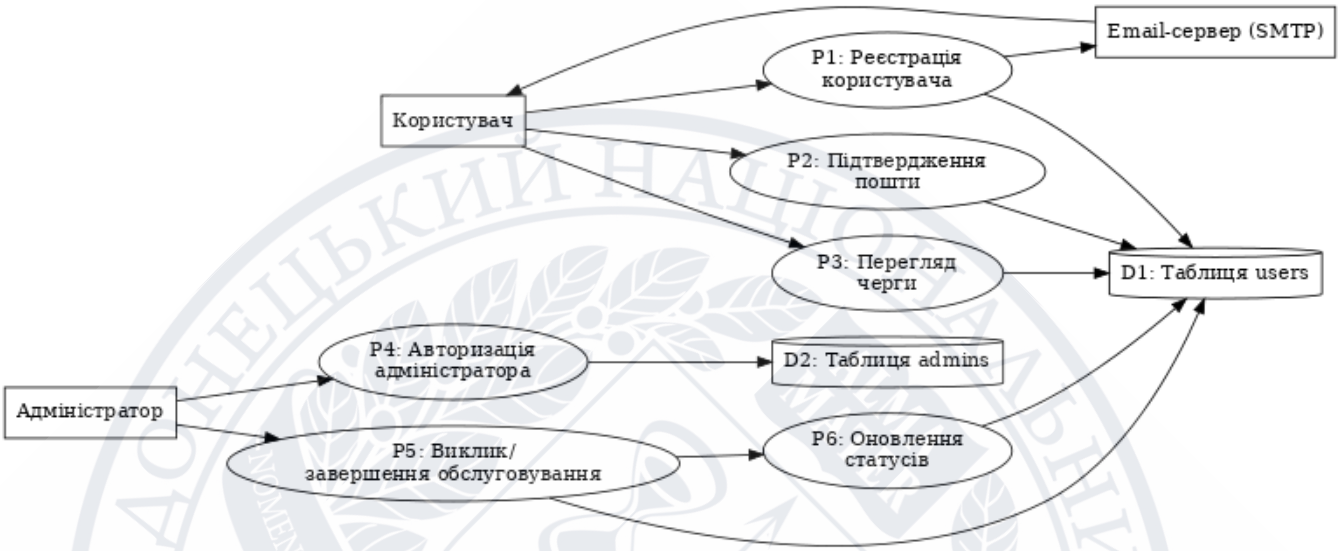



Рисунок 2.1 - DFD-модель системи електронної черги

DFD також включає взаємодію з зовнішніми сервісами - SMTP-сервером для надсилання підтвердження, а також HTML/CSS та JavaScript-фронтендом [13; 25; 26] для відображення актуального стану черги.

Entity-Relationship - модель бази даних:

Для забезпечення коректної обробки даних у системі було спроєктовано реляційну структуру бази даних. Основою є сутності «users» та «admins», між якими встановлено логічний зв’язок через зовнішній ключ [9; 23].

Основні сутності:

1. users - таблиця, що зберігає інформацію про зареєстрованих користувачів:
    - «id» - унікальний ідентифікатор.
    - «first_name», «last_name», «email» - персональні дані.
    - «queue_number» - номер у черзі.
    - «status» - поточний статус клієнта («waiting», «called», «finished»).
    - «is_confirmed» - логічне поле, що позначає підтвердження поштою.

- «called_by_admin_id» - зовнішній ключ, який вказує, який адміністратор викликав користувача.

2. admins - таблиця з обліковими даними адміністраторів:
   - «id» - ідентифікатор адміністратора.
   - «username» - логін.
   - «password_hash» - хеш пароля.

Взаємозв'язки:

- Один адміністратор може викликати багато користувачів (зв'язок один до багатьох).
- Кожен користувач асоціюється з одним адміністратором, який його викликав.

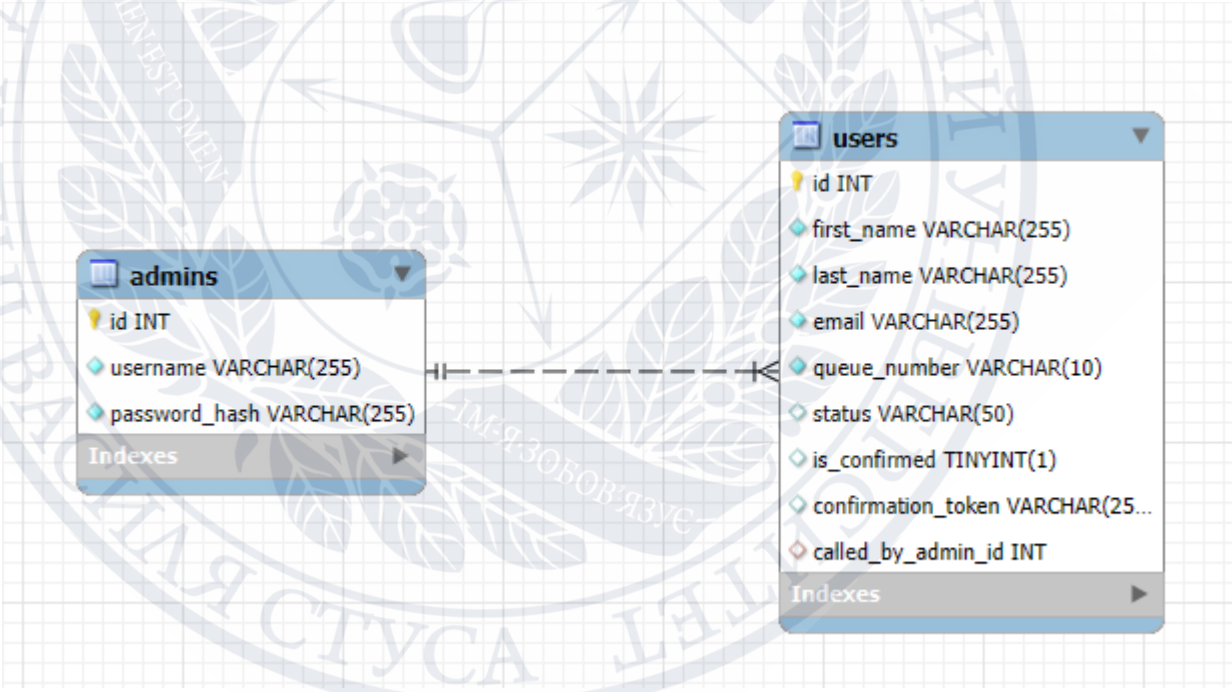



Рисунок 2.2 - ER модель бази даних

Переваги логічного проєктування:

Використання логічного проєктування на етапі розробки інформаційної системи є ключовим чинником, що забезпечує її структурованість, гнучкість та надійність. Системне моделювання взаємозв'язків між компонентами дозволяє

не лише краще зрозуміти вимоги до функціонування, а й заздалегідь виявити потенційні ризики.

Основні переваги логічного проєктування полягають у наступному:

1. Зниження ризику помилок на етапі реалізації.
   - Завдяки чіткому опису потоків даних і структур даних ще до початку програмування зменшується ймовірність виникнення логічних суперечностей та дублювання функціональності.
2. Покращення модифікованості та розширюваності системи.
   - Структуроване логічне ядро дозволяє безболісно вносити зміни у функціонал або масштабувати систему без кардинального переписування коду.
3. Уніфікація архітектурного підходу до побудови коду.
   - Завдяки попередньому моделюванню формується основа майбутньої архітектури: визначаються модулі, їхні вхідні/вихідні параметри, відповідальність кожного компонента.
4. Оптимізація командної роботи над проєктом.
   - Наявність діаграм DFD, ER або UML полегшує розподіл завдань між учасниками команди: бекенд-розробники, фронтенд-фахівці та фахівці з баз даних можуть паралельно реалізовувати відповідні частини проєкту.
5. Підвищення ефективності документування та підтримки.
   - Логічна структура слугує також основою для написання технічної документації, що значно полегшує подальше супроводження системи або її інтеграцію з іншими рішеннями.

Таким чином, логічне проєктування не є лише теоретичним етапом, а важливою практичною складовою, яка суттєво впливає на якість розробленого програмного забезпечення, його відповідність вимогам замовника та перспективу розвитку в майбутньому.

### 2.3 Обґрунтування вибору технологій

У ході розробки удосконаленої системи електронної черги було обґрунтовано використання певного технологічного стеку на основі вимог до продуктивності, безпеки, гнучкості та простоти впровадження. Нижче представлено детальний розбір вибраних інструментів та порівняння з альтернативами. Обраний технологічний стек було сформовано на основі порівняльного аналізу альтернативних рішень з урахуванням гнучкості, простоти інтеграції та відповідності функціональним вимогам системи (див. табл. 2.1).

Таблиця 2.1 - Порівняння вибраних технологій з альтернативами

| Компонент | Обрана технологія | Альтернатива | Переваги | Причини відмови від альтернативи |
|---|---|---|---|---|
| Фреймворк | Flask | Django | Простота, гнучкість, мінімалізм | Зайва складність, надлишкова функціональність |
| СУБД | MySQL | PostgreSQL | Простота налаштування, продуктивність | Вища складність, більші ресурси |
| Інтерфейс | HTML/CSS + JS + AJAX | React.js | Легкість, швидкість реалізації | Складне середовище, окремий API |
| Email-верифікація | SMTP + smtplib | SendGrid / Mailgun | Просте впровадження | Потреба в акаунтах |

Flask було обрано як основний фреймворк для створення веб-додатку через його легкість [21; 22; 28], гнучкість та швидку інтеграцію з іншими бібліотеками Python. Flask є мікрофреймворком, який забезпечує лише базову

функціональність, надаючи розробникам свободу вибору необхідних компонентів. Переваги Flask полягають у його зрозумілості, великій кількості навчальних ресурсів, сумісності з Jinja2 (сучасним шаблонізатором), а також у легкій структурі коду, яка є зручною для невеликих і середніх проєктів.

Для порівняння, Django - це більш потужний фреймворк, який включає більшість компонентів «з коробки», зокрема ORM, панель адміністратора, вбудовану автентифікацію та безпекові засоби [41]. Однак, через його складність і надлишковість для порівняно простого застосунку, такого як система електронної черги, його використання є недоцільним. Flask краще підходить, коли необхідно мати повний контроль над структурою застосунку.

Система управління базами даних MySQL була обрана завдяки своїй стабільності, простоті в розгортанні та активній спільноті [23; 38]. Вона чудово підходить для реалізації транзакційних операцій, а також має широке розповсюдження серед хостинг-провайдерів. Альтернативою могла б бути PostgreSQL, яка має більш розвинені функції, особливо щодо роботи з JSON, CTE-запитами та транзакціями. Проте складніша конфігурація та більш високі вимоги до ресурсів роблять її менш зручною для проєктів середнього масштабу.

Інтерфейс користувача реалізовано з використанням HTML5/CSS3 [24; 25; 26] з підключенням шрифту Poppins. Цей вибір забезпечив сучасний та естетично привабливий вигляд системи. Використання JavaScript дозволило реалізувати динамічну поведінку веб-інтерфейсу [18], а саме: оновлення елементів черги в реальному часі без перезавантаження сторінки за допомогою технологій AJAX. Альтернативою міг бути фреймворк React, проте він потребує значно більшої підготовки, налаштування середовища та створення окремого API.

Для підтвердження електронної пошти після реєстрації був застосований SMTP-протокол через бібліотеку smtplib [40]. Це дозволяє автоматично надсилати листи користувачам для верифікації, запобігаючи фіктивній реєстрації. Альтернативи у вигляді сторонніх сервісів, як-от SendGrid або Mailgun, надають додаткову аналітику, але вимагають окремої авторизації та прив'язки до акаунтів, що ускладнює розгортання у локальному середовищі.

Таким чином, обраний стек технологій - Flask, MySQL, HTML/CSS, JavaScript, SMTP - було обрано з огляду на баланс між простотою реалізації, надійністю та можливістю масштабування. Усі компоненти забезпечують необхідну функціональність, не перевантажуючи систему зайвою складністю.

### 2.4. Розробка алгоритмів і логіки оновлення черги

У цьому підрозділі розглянемо основні алгоритмічні процеси, що забезпечують функціонування системи електронної черги. Зокрема, буде описано логіку виклику користувача, механізм блокування доступу до вже викликаних записів [12] іншими адміністраторами, а також процес підтвердження електронної пошти.

Алгоритм відображення кнопки завершення виклику:

Цей алгоритм регулює доступ до кнопки «Завершити», яка з’являється лише в тому випадку, якщо саме поточний адміністратор викликав відповідного користувача. Такий підхід забезпечує цілісність даних та унеможливлює втручання інших адміністраторів.

Основні етапи:

- Отримати ідентифікатор поточного адміністратора.
- Отримати дані про всіх користувачів зі статусом called.
- Перевірити, чи викликав користувача поточний адміністратор.
- Якщо так - показати кнопку «Завершити», інакше - приховати її.


Рисунок 2.3 - Схема блокування користувачів для інших адміністраторів

Алгоритм виклику користувача адміністратором:

Цей алгоритм відповідає за оновлення статусу користувача в базі даних з "waiting" на "called", після чого користувач стає видимим у табло виклику.

Основні кроки:

- Перевірити, чи в сесії присутній авторизований адміністратор.
- Перевірити, чи немає вже викликаного користувача цим адміністратором.
- Змінити статус користувача на "called".
- Записати called_by_admin_id відповідно до ID адміністратора.
- Відобразити змінену чергу лише для відповідального адміністратора.

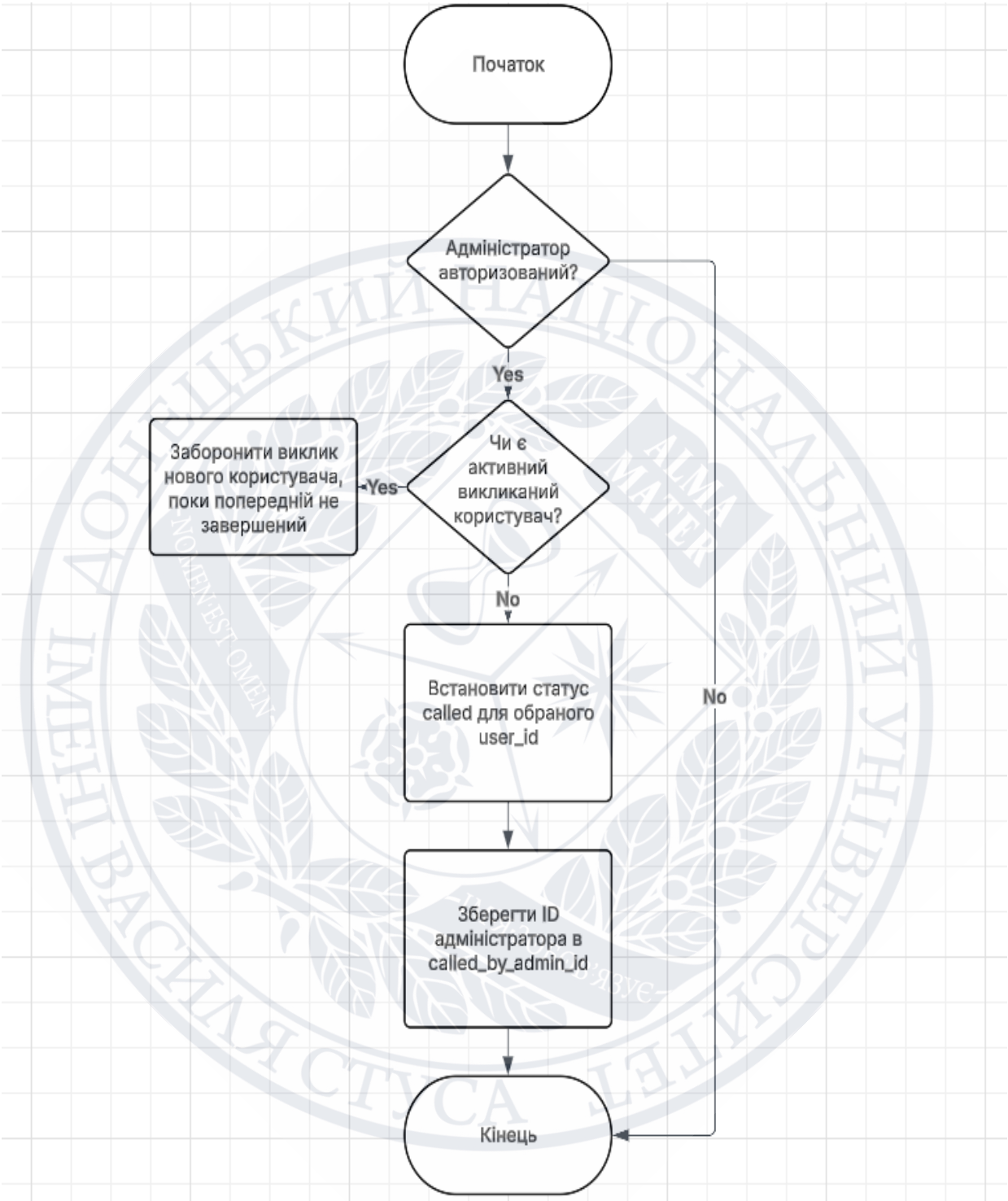



Рисунок 2.4 - Схема алгоритму виклику користувача

Алгоритм підтвердження електронної пошти:

Це критично важливий крок для запобігання фейковим реєстраціям. Після створення користувача формується лист із посиланням на підтвердження.

Послідовність:

- При реєстрації створюється запис у базі зі статусом unconfirmed.
- Користувач отримує лист із унікальним посиланням (ідентифікатор + токен).
- Після переходу за посиланням статус змінюється на waiting.
- Користувач з’являється в черзі.

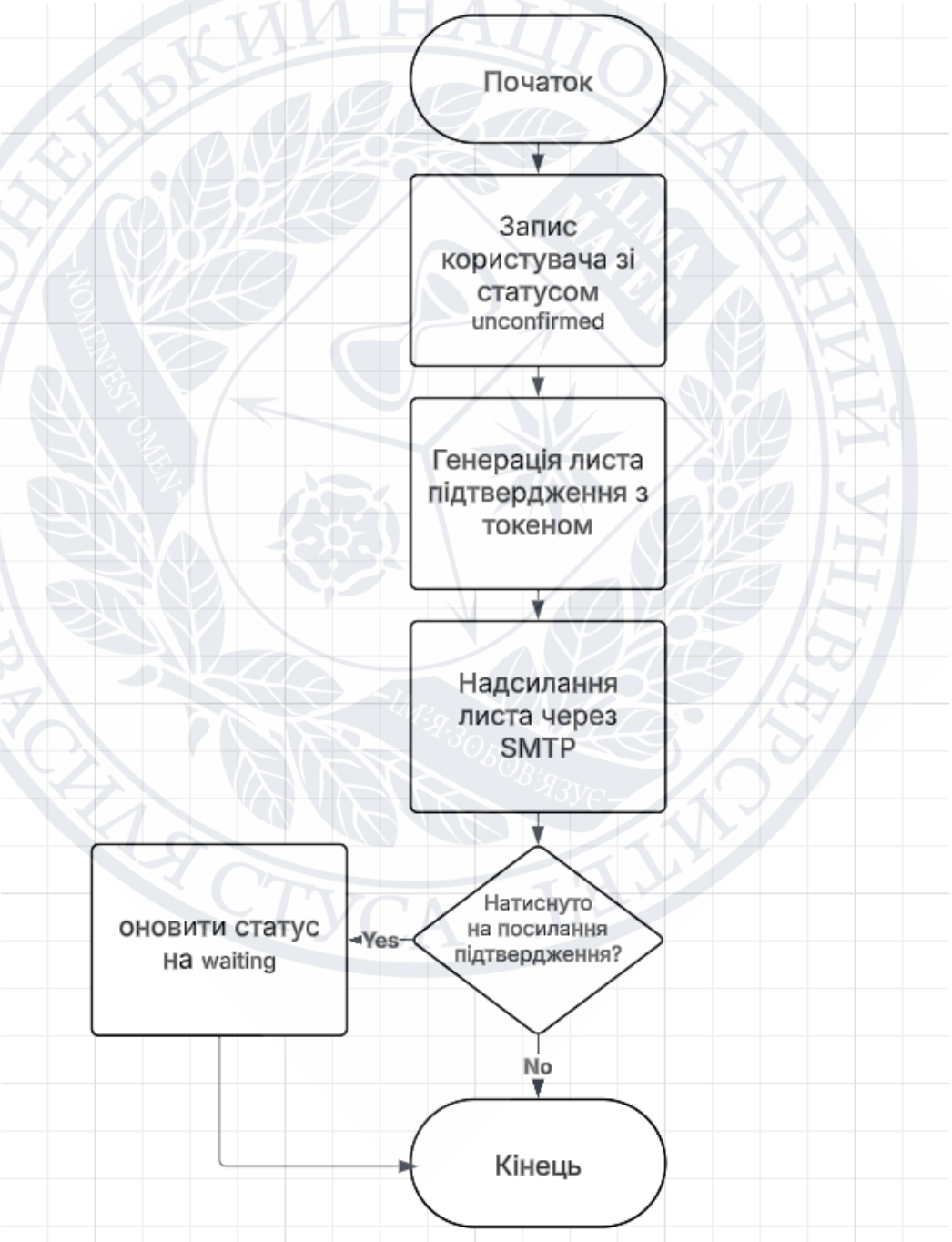



Рисунок 2.5 - Схема підтвердження електронної пошти

Реалізація оновлення у реальному часі [25; 27]:

Черга оновлюється без перезавантаження сторінки завдяки JavaScript (AJAX) та API /api/queue_list. Аналогічно, на табло відображається останній викликаний користувач через /api/now_serving.

### 2.5 Висновки до розділу 2

У другому розділі було проведено систематичний аналіз шляхів реалізації та вдосконалення системи електронної черги, виходячи з конкретних функціональних і нефункціональних вимог. На основі поставлених цілей було сформульовано вимоги до майбутньої системи, які охоплюють критично важливі аспекти, як забезпечення безпеки, масштабованості, доступності та інтеграції. У ході логічного проектування були побудовані DFD-діаграма, яка відображає основні потоки даних та функціональні компоненти, і ER-модель бази даних, що демонструє взаємозв'язки між основними сутностями - користувачами та адміністраторами. Ці моделі дали змогу чітко структурувати архітектуру системи, що, у свою чергу, дозволило зменшити ризики виникнення логічних помилок, оптимізувати комфортну командну роботу адміністраторів та забезпечити можливість масштабування.

Під час вибору технологій було проведено порівняльний аналіз альтернатив. Як результат, було обґрунтовано доцільність використання стеку Flask + MySQL + HTML/CSS/JS + SMTP [22; 23; 26; 31], який забезпечує баланс між легкістю реалізації та надійністю. Зокрема, Flask було обрано через його гнучкість, MySQL - через простоту в налаштуванні, JavaScript + AJAX - для реалізації динамічного інтерфейсу, SMTP - для інтеграції підтвердження електронної пошти.

Таким чином, у межах розділу 2 було розроблено методику та закладено повноцінний фундамент для реалізації ефективної, гнучкої та безпечної системи електронної черги. Ці результати є необхідною передумовою для переходу до етапу практичної реалізації, тестування та оцінювання працездатності розробленого програмного продукту у наступному розділі.

# РОЗДІЛ 3.
# Розробка та реалізація комп'ютеризованої системи електронної черги

## 3.1 Архітектура системи електронної черги

Розроблена система електронної черги базується на архітектурному підході MVC (Model-View-Controller) [3; 4; 41], який дозволяє ефективно розділяти логіку обробки даних, бізнес-процеси та користувацький інтерфейс. Такий підхід є стандартним у сучасному веб-програмуванні, оскільки забезпечує гнучкість при оновленні, масштабуванні, супроводі та тестуванні системи.

Загальна структура

Уся система складається з трьох головних рівнів:

- Model (Модель) - відповідає за взаємодію з базою даних MySQL, включає операції створення, вибірки, оновлення та видалення записів, а також бізнес-логіку, пов'язану зі статусами черги.
- View (Подання) - представлення інформації у вигляді веб-сторінок, що розроблені з використанням HTML5, CSS3 та JavaScript.
- Controller (Контролер) - основна логіка, що координує запити користувача, взаємодіє з моделлю та повертає відповідь у вигляді подання. Цей компонент реалізовано за допомогою фреймворку Flask.

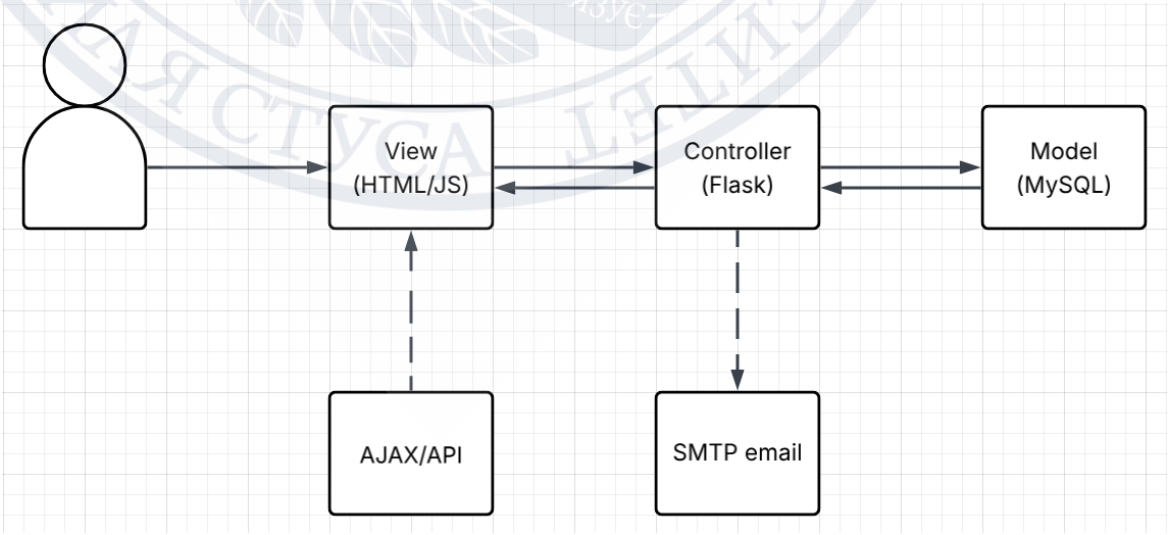



Рисунок 3.1 - Загальна архітектура системи в контексті MVC

Компоненти системи:

1. Фронтенд:

Інтерфейс системи поділяється на кілька ключових екранів, кожен з яких виконує окрему функцію:

- Головна сторінка (index.html) - показує поточний стан черги, а також користувача, якого зараз обслуговують.
- Форма реєстрації (register.html) - дає змогу користувачу ввести свої дані для отримання електронного квитка у черзі. Після реєстрації користувач отримує email з підтвердженням.

Сторінка підтвердження email - користувач переходить за посиланням із листа, що активує його участь у черзі.

Панель адміністратора (admin.html) - адміністратор може викликати наступного клієнта, завершити обслуговування та переглядати список очікуючих.

2. Бекенд (Flask)

Серверна частина реалізована у вигляді окремого Python-додатку на Flask [22; 21; 35]. Основні функції:

- Обробка POST/GET-запитів на реєстрацію, логін, виклик клієнтів.
- Авторизація через сесію.
- Надсилання email з токеном підтвердження через бібліотеку smtplib [24; 40].
- Обробка REST-запитів для оновлення інтерфейсу в реальному часі (AJAX).

3. База даних (MySQL)

База даних включає дві основні таблиці:

- users - зберігає дані користувачів (ПІБ, email, статус, ID адміністратора, що викликав, токен підтвердження, позначка верифікації).

- admins - облікові записи адміністраторів, що мають доступ до панелі керування.

API-інтерфейси:

Для реалізації динамічного оновлення черги використано кілька REST-ендпоінтів:

- /api/queue_list - повертає список очікуючих користувачів.
- /api/now_serving - віддає номер поточного викликаного клієнта.
- /api/admin_queue - для адмін-панелі з фільтрацією за викликаними ID.

Ці запити обробляються на фронтенді за допомогою JavaScript (функції fetch/AJAX) [26; 30; 33], що дозволяє безперервно оновлювати дані без перезавантаження сторінки.

Система верифікації користувачів:

Для забезпечення достовірності запису користувачів використовується підтвердження електронної пошти. Після подання форми система генерує унікальний токен і надсилає лист за вказаною адресою. Після переходу за посиланням запис активується, і користувач з'являється в загальній черзі.

Архітектура системи реалізована таким чином, щоб забезпечити розширюваність, модульність та зручність в обслуговуванні. Поділ на логічні компоненти дозволяє проводити окреме тестування частин системи, впроваджувати нові функції без порушення загальної роботи та адаптувати систему до змін вимог користувачів або інфраструктури.

### 3.2 Реалізація модулів системи:

Розроблена система електронної черги реалізована у вигляді модульної веб-системи, побудованої на архітектурі MVC

(Model-View-Controller) з використанням фреймворку Flask. Завдяки модульному підходу до реалізації кожна ключова функція системи була

ізольована у вигляді окремого логічного блоку, що забезпечує гнучкість, масштабованість і легкість супроводу.

У цьому підрозділі послідовно розглянуто реалізовані функції з ілюстрацією відповідних інтерфейсів, змін у базі даних і механізмів взаємодії між компонентами системи.

Модуль реєстрації користувача:

Процес починається з переходу користувача на сторінку реєстрації, де вводяться ім’я, прізвище та електронна пошта.

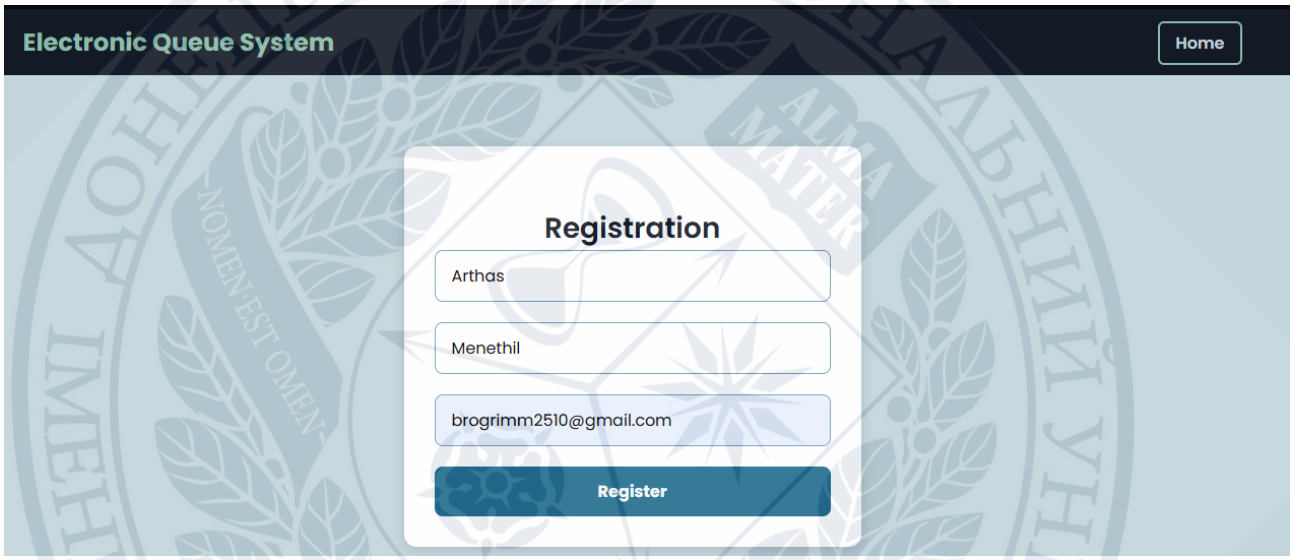



Рисунок 3.2 - Сторінка реєстрації нового користувача

Після натискання кнопки «Підтвердити», дані передаються серверу, де створюється новий запис у таблиці `users` з генерованим унікальним токеном підтвердження. Початково статус користувача встановлюється як `waiting`, а поле `is_confirmed` — як `0`.

| id | first_name | last_name | email | queue_number | status | is_confirmed | confirmation_token | called_by_admin_id |
|---|---|---|---|---|---|---|---|---|
| 1 | Serhii | Korolenko | brogrimm2510@gmail.com | A41 | finished | 1 | NULL | 1 |
| 2 | Serhii | Korolenko | brogrimm2510@gmail.com | A42 | finished | 1 | NULL | 1 |
| 3 | Alex | Korolenko | brogrimm2510@gmail.com | A43 | finished | 1 | NULL | 1 |
| 4 | Olexandr | Korolenko | brogrimmdota2@gmail.com | A44 | finished | 1 | NULL | 1 |
| 5 | Diana | Shyshkova | brogrimm2510@gmail.com | A45 | finished | 1 | NULL | 1 |
| 6 | Anna | Meleniya | brogrimmsystem@gmail.com | A46 | waiting | 1 | NULL | NULL |
| 7 | Oleg | Voroshilov | brogrimm2510@gmail.com | A47 | finished | 1 | NULL | 1 |
| 8 | Vlad | Gaevski | brogrimmdota2@gmail.com | A48 | waiting | 1 | NULL | NULL |
| 9 | Arthas | Menethil | brogrimm2510@gmail.com | A49 | waiting | 0 | 48d39ede-9b13-4f80-abe3-b563e6a40a79 | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Рисунок 3.3 - Поточний вигляд нового запису в базу даних

Модуль підтвердження електронної пошти:

Після успішного створення облікового запису система надсилає лист на вказану пошту. У листі міститься унікальне посилання для підтвердження, яке включає токен. Перехід за ним призводить до оновлення `is_confirmed` у базі на `1`, що активує можливість потрапити до черги.

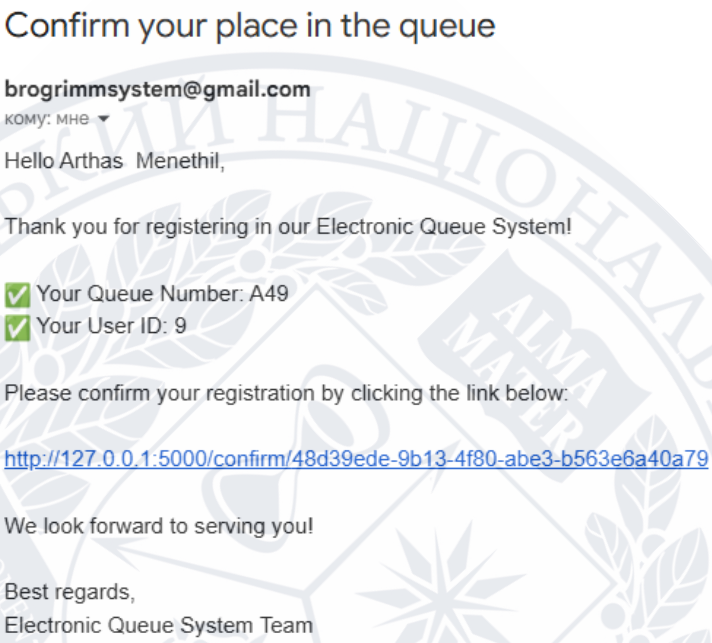



Рисунок 3.4 - Приклад листа підтвердження реєстрації

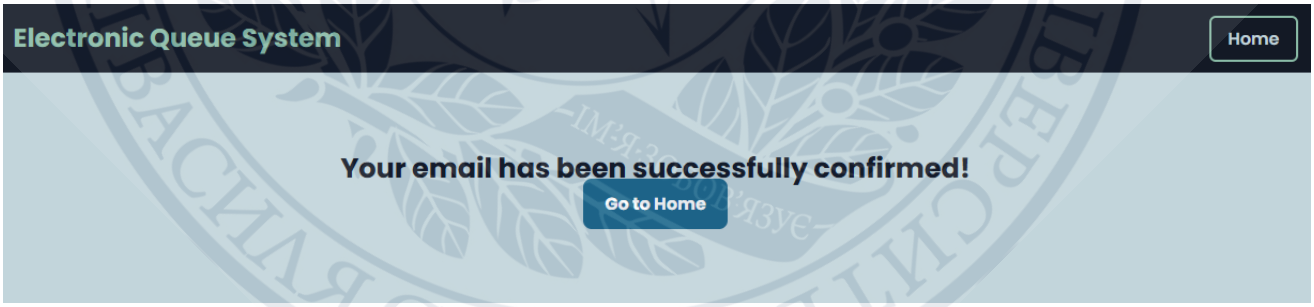
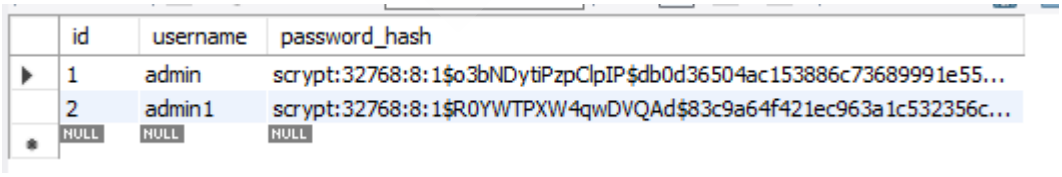



Рисунок 3.5 - Сторінка підтвердження реєстрації

Модуль авторизації адміністратора:

Адміністратор проходить автентифікацію через форму входу. Паролі зберігаються у хешованому вигляді (алгоритм scrypt).



| id | username | password_hash |
|---|---|---|
| 1 | admin | scrypt:32768:8:1$o3bNDytiPzpClpIP$db0d36504ac153886c73689991e55... |
| 2 | admin1 | scrypt:32768:8:1$R0YWTPXW4qwDVQAd$83c9a64f421ec963a1c532356c... |
| NULL | NULL | NULL |

Рисунок 3.6 - Поточний вигляд таблиці `admins`

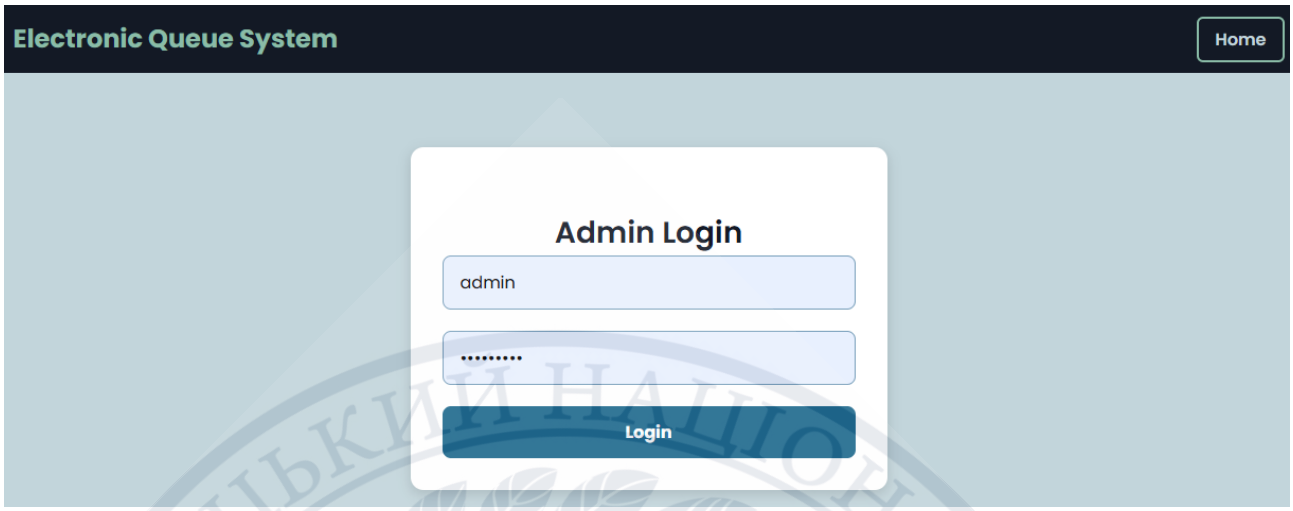



Рисунок 3.7 - Форма входу до адміністративної панелі

Після входу йому відкривається інтерфейс панелі управління, де він бачить чергу з клієнтами зі статусами `waiting` або `called`.

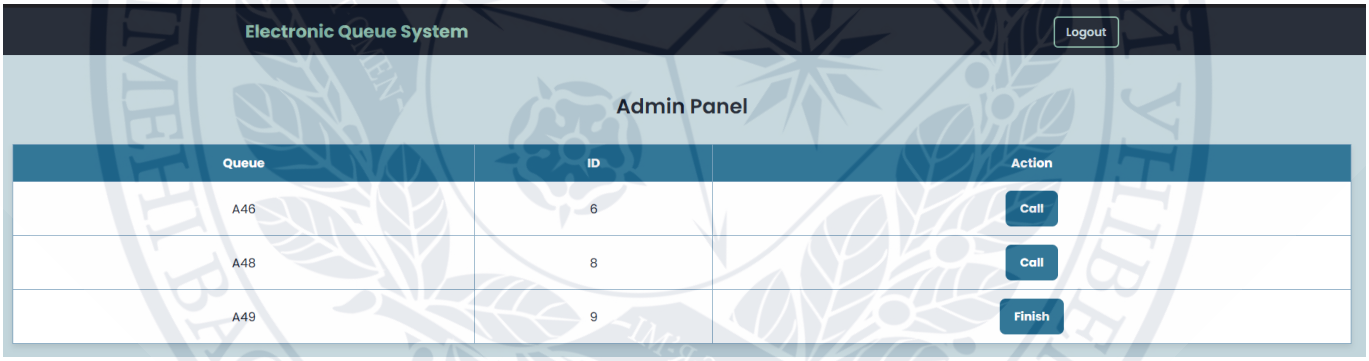
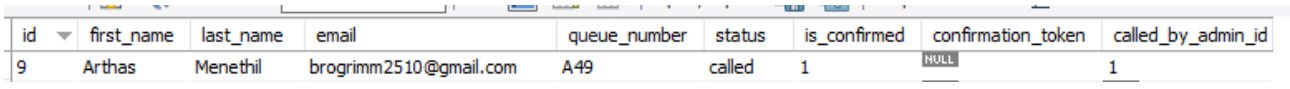



Рисунок 3.8 - Таблиця черги у панелі адміністратора

Модуль виклику користувача:

Адміністратор може викликати наступного користувача, натиснувши кнопку «Викликати». Це змінює статус користувача на `called`, а в полі `called_by_admin_id` зберігається ID адміністратора, що викликав. Це дозволяє запобігти конфліктному обслуговуванню.

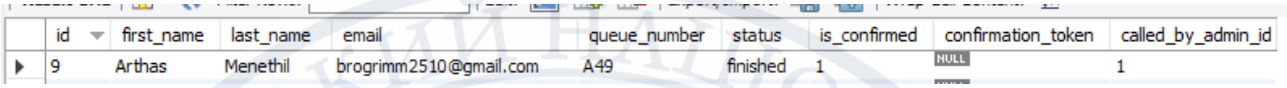


| id | first_name | last_name | email | queue_number | status | is_confirmed | confirmation_token | called_by_admin_id |
|---|---|---|---|---|---|---|---|---|
| 9 | Arthas | Menethil | brogrimm2510@gmail.com | A49 | called | 1 | NULL | 1 |

Рисунок 3.9 - Зміна статуса нового запису на `called`

Модуль завершення обслуговування:

Після обслуговування клієнта адміністратор натискає кнопку «Завершити», що змінює статус на `finished` та очищує поле `called_by_admin_id`. Таким чином система переходить у стан готовності до обробки наступного користувача.



| id | first_name | last_name | email | queue_number | status | is_confirmed | confirmation_token | called_by_admin_id |
|---|---|---|---|---|---|---|---|---|
| 9 | Arthas | Menethil | brogrimm2510@gmail.com | A49 | finished | 1 | NULL | 1 |

Рисунок 3.10 - Зміна статуса нового запису на `finished`

Модуль автооновлення (API та AJAX)

Для забезпечення актуальності даних в інтерфейсі використовується набір REST API-ендпоінтів:

- `/api/now_serving` – повертає поточного клієнта зі статусом `called`.
- `/api/queue_list` – повертає список користувачів у черзі зі статусом `waiting`.
- `/api/admin_queue` – використовується у панелі адміністратора для оновлення таблиці в режимі реального часу.

Ці запити ініціюються клієнтом асинхронно за допомогою JavaScript та AJAX, без повного перезавантаження сторінки.

### 3.3 Тестування реалізованої системи

Тестування є ключовим етапом життєвого циклу розробки програмного забезпечення, метою якого є забезпечення відповідності програмного продукту заявленим вимогам, виявлення помилок і перевірка стабільності роботи системи у різних сценаріях використання. У рамках реалізації комп'ютеризованої системи електронної черги було проведено низку видів тестування, серед яких функціональне, модульне, інтеграційне, тестування некоректних сценаріїв, а також тестування адміністративного функціоналу.

Функціональне тестування:

Метою функціонального тестування було перевірити відповідність реалізованої системи заданим функціональним вимогам. Перевірялися ключові процеси, зокрема:

- реєстрація користувача та генерування номеру черги;
- відправка листа → підтвердження на email з токеном [25; 35];
- перехід за посиланням підтвердження, з подальшою активацією облікового запису;
- відображення підтверджених користувачів у черзі;
- виклик користувача адміністратором та завершення обслуговування.
- обробку статусів waiting, called, finished;

Усі перевірені дії працювали стабільно, без збоїв та помилок. Результати тестування свідчать про відповідність реалізації вимогам.

Модульне тестування:

Цей тип тестування охоплював окремі компоненти системи — функції та класи, які реалізовують конкретну логіку. Модульно перевірялись:

- методи взаємодії з базою даних (insert, update, select);
- функції API (/api/now_serving, /api/queue_list, /api/admin_queue);
- відправка поштових повідомлень через SMTP (send_email()).

Модулі демонстрували правильну логіку роботи, обробку винятків та коректне оновлення інформації в БД.

Інтеграційне тестування:

Цей рівень тестування дозволяє оцінити, як різні частини системи працюють разом. Були протестовані такі зв’язки:

- реєстраційна форма → перевірка → БД → відправка листа → підтвердження токену;
- таблиця черги → оновлення статусу → API → відображення на клієнтській частині;
- виклик клієнта адміністратором → зміна черги → головне табло.

Усі інтеграційні потоки показали очікувану поведінку без логічних конфліктів.

Тестування на некоректні дії:

Окремо перевірялась поведінка системи у випадках помилок користувача або повторних дій:

- спроба повторного підтвердження уже активованого профілю;
- введення некоректного email;
- реєстрація одного email декілька разів;

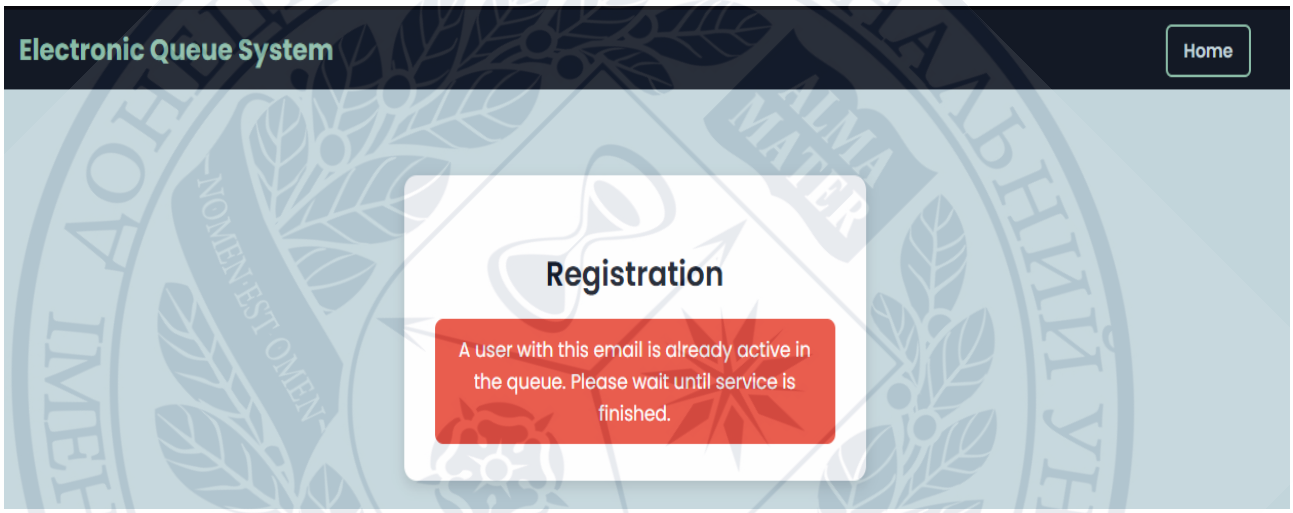



Рисунок 3.11 - Повідомлення про помилку при повторній реєстрації

Система у всіх випадках показала правильну реакцію — або повертаючи повідомлення про помилку, або ігноруючи повторну дію.

Тестування адміністративного функціоналу:

Було перевірено:

- авторизацію адміністратора;
- можливість виклику клієнта та завершення обслуговування;
- унікальність виклику (тільки один адміністратор може викликати конкретного клієнта);
- правильне оновлення таблиці черги в режимі реального часу.

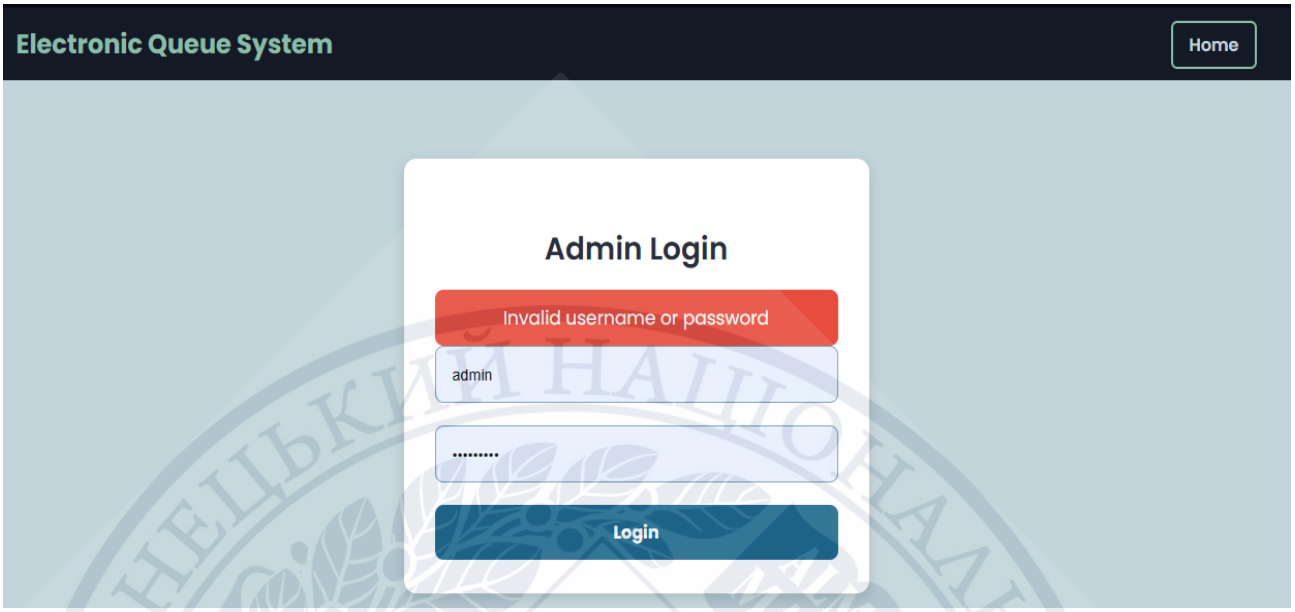



Рисунок 3.12 - Повідомлення про помилку під час хибної авторизації адміністратора

Усі функції адміністративної частини працювали стабільно, а сесії зберігались коректно.

На основі проведеного тестування можна зробити висновок, що система електронної черги відповідає поставленим функціональним вимогам, забезпечує коректну логіку роботи, має стійкість до помилок та гарантує комфортну взаємодію для кінцевих користувачів і адміністраторів.

### 3.4. Аналіз потенційної інтеграції та перспектив розвитку

Система електронної черги, розроблена в межах даної роботи, демонструє високу функціональну завершеність у своєму базовому вигляді. Вона здатна ефективно обробляти типові сценарії взаємодії користувача з чергою, включаючи реєстрацію, підтвердження участі, виклик і завершення обслуговування. Водночас, як і будь-яка сучасна інформаційна система, вона має потенціал до подальшого розширення, удосконалення функціоналу та масштабування як за технічними характеристиками, так і за сферою застосування.

У цьому підрозділі висвітлюються можливі напрями розвитку системи, додаткові функціональні можливості, які можуть бути реалізовані у майбутніх ітераціях, а також аналізуються поточні обмеження й недоліки, які було виявлено в ході тестування та практичного використання.

Можливі напрями масштабування системи

1. Інтеграція з зовнішніми інформаційними системами:

Одним з основних трендів у сфері цифровізації є створення екосистем, де окремі сервіси взаємодіють між собою на базі відкритих API або за допомогою централізованих державних платформ. У цьому контексті розглядається доцільність інтеграції розробленої системи із зовнішніми інформаційними ресурсами, такими як:

- Державні реєстри або системи ідентифікації [19; 43] (наприклад, "Дія" ), що дозволяє автоматично підставляти дані користувача при реєстрації, уникнути помилок при введенні, підвищити довіру та безпеку;
- Системи документообігу, які можуть слугувати як архіви для обробки звернень, формування звітності або юридичного підтвердження візитів.
- Такі інтеграції значно підвищать ефективність роботи персоналу та зменшать кількість рутинних операцій.

2. Інтеграція з чат-ботами (наприклад, Viber або WhatsApp [36])

У сучасних умовах користувачі очікують можливості взаємодії з сервісами через зручні, звичні канали комунікації. Чат-боти на базі Viber/WhatsApp можуть стати додатковим інтерфейсом системи черги. Така інтеграція передбачає:

- Попередній запис без входу на сайт;
- Отримання повідомлень про зміну статусу;
- Автоматичні нагадування про наближення черги;
- Відповіді на часті запитання та технічну підтримку в текстовому режимі.

Реалізація такої взаємодії розширює охоплення аудиторії, забезпечуючи більш інклюзивний доступ до сервісу, включаючи маломобільних користувачів, осіб похилого віку або тих, хто надає перевагу смартфонам.

3. Розгортання інфраструктури офлайн-доступу: табло та термінали

Невід'ємною частиною інтегрованої електронної черги є фізичні компоненти, які забезпечують інформування та автономну взаємодію з системою у режимі реального часу. До них належать:

- Інформаційне табло: пристрої в залі очікування, що відображають поточний статус черги - наприклад, номер, що обслуговується, і наступного клієнта. Це знижує навантаження на персонал та покращує прозорість обслуговування.
- Електронні термінали реєстрації: автоматизовані пристрої, розміщені безпосередньо у приміщеннях організації (ЦНАП, поліклініки, центри обслуговування), які дозволяють користувачу самостійно пройти процедуру реєстрації без допомоги адміністратора. Зокрема, це може бути реалізовано через сенсорні панелі або планшети, що працюють у спеціальному захищеному режимі.
- Інтеграція подібних компонентів дає змогу реалізувати гібридну модель черги - як онлайн, так і офлайн - з єдиною базою даних, що дозволяє обслуговувати більший потік клієнтів і уникати дублювання.

Додатковий функціонал, що може бути реалізований

Хоча розроблена система електронної черги вже успішно виконує свої основні функції, подальший розвиток проєкту передбачає впровадження додаткових можливостей, які здатні суттєво покращити якість взаємодії як для користувачів, так і для адміністраторів. З урахуванням сучасних вимог до гнучкості, адаптивності та аналітичної прозорості інформаційних систем, доцільним є розширення функціоналу в таких напрямках:

1. Інтерактивна аналітична панель адміністратора:

Однією з ключових складових ефективного управління будь-якою цифровою платформою є наявність повноцінного інструментарію для аналітики. Реалізація аналітичної панелі адміністратора дозволить не лише відслідковувати загальні показники, але й отримувати глибше розуміння динаміки роботи системи. Основними можливостями такої панелі можуть бути:

- Генерація звітів у реальному часі щодо поточної та історичної завантаженості черги;
- Аналіз середнього часу очікування, швидкості обслуговування, кількості завершених або скасованих записів;
- Виведення діаграм і графіків на основі зібраних метрик, що дозволяє візуально оцінювати ефективність роботи окремих операторів або змін у завантаженості протягом дня.

Наявність такого модуля дає змогу адміністрації приймати обґрунтовані управлінські рішення, базуючись на реальних цифрах, а також вчасно виявляти проблемні ділянки в обслуговуванні клієнтів.

2. Інтеграція push-сповіщень або SMS-інформування:

У сучасному світі користувачі очікують на швидке та своєчасне інформування. З цією метою доцільним є впровадження механізмів автоматичних сповіщень, що забезпечать додаткову взаємодію із клієнтами системи. Йдеться про:

- Push-сповіщення у вебінтерфейсі або мобільному застосунку, що дозволяють отримувати нагадування, не покидаючи браузера;
- SMS-інтеграцію як резервний або основний канал зв’язку для тих, хто не використовує інтернет або мобільні додатки.

Такі сповіщення можуть містити інформацію про:

- підтвердження реєстрації;
- нагадування про наближення черги;
- зміну статусу обслуговування.

Впровадження цього функціоналу зменшує ймовірність неявки користувачів, сприяє оптимізації черги, а також покращує загальне враження від сервісу завдяки підвищенню його персоналізації.

3. Модуль попереднього запису:

Ще одним важливим кроком до підвищення зручності для кінцевого користувача є реалізація функціоналу попереднього запису на обслуговування. Ця можливість дозволяє:

- заздалегідь обрати дату та час візиту, що найбільше підходить користувачу;
- розподілити навантаження між годинами або днями, оптимізуючи роботу персоналу;
- створити логічну структуру завантаження із прогнозуванням пікових навантажень.

Впровадження системи попереднього запису дозволяє не лише зменшити черги в режимі реального часу, а й створити умови для точного планування ресурсу - як технічного, так і людського.

Усі вищезазначені можливості можна поступово додавати до вже існуючої системи, оскільки її архітектура побудована з урахуванням модульності та масштабованості. Це означає, що нові компоненти можуть бути впроваджені без повного переписування основної логіки або порушення стабільності вже працюючих модулів.

### 3.5 Виявлені недоліки реалізованої системи та оцінка готовності до впровадження

Не зважаючи на реалізацію основних функціональних цілей та позитивні результати тестування, реалізована система електронної черги ще не позбавлена низки технічних та архітектурних обмежень. Їх своєчасне виявлення й усунення є критично важливим для переходу до масштабованої експлуатації, особливо у

випадках використання в державних установах, комерційних структурах або закладах із підвищеними вимогами до безпеки.

1. Відсутність розмежування ролей адміністраторів [37]:

На поточному етапі всі адміністратори мають однаковий рівень доступу до функціоналу системи, що не дозволяє гнучко розподіляти повноваження відповідно до обов'язків. Такий підхід створює ризики, пов'язані з людським фактором, зокрема:

- можливість випадкового або навмисного видалення записів;
- відсутність обмежень на критичні дії, як-от скидання черги або ручне призначення користувачів;

незрозумілий рівень відповідальності серед персоналу.

Доцільним є впровадження ролей доступу - наприклад:

- Супер-адміністратор - має повний контроль над усією системою;
- Оператор - лише викликає клієнтів і переглядає чергу;
- Аналітик - має доступ лише до статистики й звітності.

Це дозволить не лише покращити безпеку системи, але й зробити управління нею більш структурованим і контрольованим.

2. SMTP-підсистема без шифрування (TLS/SSL) [40; 41]:

TLS (Transport Layer Security) - це криптографічний протокол, який забезпечує захищене шифроване з'єднання між клієнтом і сервером у мережі. Він використовується для захисту даних під час передавання, наприклад, у протоколах HTTPS, SMTP тощо.

STARTTLS - це розширення для поштових протоколів (SMTP, IMAP, POP3), яке дозволяє оновити незашифроване з'єднання до захищеного за допомогою TLS. Інакше кажучи, спочатку з'єднання встановлюється як звичайне (plaintext), але за командою клієнта переходить у захищений режим без розриву сесії.

У реалізації механізму надсилання листів використовується стандартний протокол SMTP без активованого шифрування TLS або STARTTLS. Це означає,

що дані (включаючи адреси одержувачів та вміст повідомлень) передаються у відкритому вигляді мережею. Подібна реалізація створює потенційну вразливість до перехоплення трафіку, особливо у випадку публічних або скомпрометованих мереж.

Рекомендованим рішенням є:

- переходити на захищене з'єднання TLS;
- використовувати авторизовані SMTP-сервери з двофакторною перевіркою;
- або інтегрувати сторонні сервіси типу SendGrid або Mailjet, які мають вбудовані безпекові механізми.

3. Відсутність логування дій адміністраторів [6; 16]:

У поточній архітектурі не реалізовано ведення журналу дій (audit log), що є суттєвим обмеженням з точки зору аудиту, відстеження змін та забезпечення безпеки. Це створює такі ризики:

- неможливість встановити, хто викликав певного користувача;
- відсутність історії помилкових дій або системних збоїв;
- ускладнення у процесі технічного супроводу або аналізу інцидентів.

Впровадження системи логування - навіть базового рівня - значно підвищить прозорість роботи адміністраторів, дозволить формувати звіти активності та відновлювати послідовність подій у разі помилок.

4. Потенційна вразливість до CSRF-атак [8; 44]:

CSRF (Cross-Site Request Forgery) - це атака міжсайтової підміни запиту, яка змушує користувача виконати небажану дію на вебсайті, на якому він автентифікований, без його відома (зазвичай через автоматично додану сесійну куки).

На поточний момент для форм, що використовують метод POST, не передбачено захисту у вигляді CSRF-токенів (Cross-Site Request Forgery). Це відкриває можливість виконання підставних запитів від імені авторизованого адміністратора у разі, якщо він перейде за шкідливим посиланням.

Щоб уникнути подібних ризиків, необхідно:

- впровадити CSRF-захист через Flask-WTF або інші розширення;
- використовувати токени, прив'язані до сесії;
- перевіряти Referrer та Origin у запитах.

Оцінка готовності системи до впровадження:

Проведене функціональне тестування та аналіз працездатності розробленої системи дозволяють зробити висновок, що наявна реалізація повністю відповідає основним функціональним вимогам: обробка черги, адміністрування, оновлення у реальному часі, підтвердження через email. Користувацький інтерфейс є інтуїтивно зрозумілим, а всі основні сценарії від реєстрації до завершення обслуговування - успішно реалізовані.

Водночас, наявність описаних недоліків свідчить про те, що система ще не досягла рівня промислової готовності. На поточному етапі її варто позиціонувати як пілотну реалізацію, придатну для впровадження в умовах обмеженої аудиторії - наприклад, у внутрішньому тестуванні установи або як MVP (Minimum Viable Product).

Рекомендовані напрями подальшого розвитку:

- Впровадження розмежування прав доступу для адміністративного персоналу;
- Підключення захищеного SMTP з TLS/SSL;
- Розробка модуля логування дій користувачів і адміністраторів;
- Розширення системи за рахунок зовнішніх API (наприклад, Viber-бот для користувачів);
- Реалізація безпекових протоколів (CSRF, авторизація по токенах, CAPTCHA).

Таким чином, система має всі передумови для подальшого масштабування, але перед її впровадженням у реальне середовище слід провести низку технічних доопрацювань. Це забезпечить не лише функціональну надійність, а й відповідність сучасним стандартам інформаційної безпеки.

**Висновки до розділу 3**

У третьому розділі було безпосередньо реалізовано запропоновані в попередніх розділах методичні підходи до удосконалення комп'ютеризованої системи електронної черги. Проведена робота дозволила не лише впровадити розроблену архітектуру, але й на практиці перевірити її функціональність, ефективність та відповідність сформульованим вимогам.

У процесі реалізації системи було враховано виявлені проблеми існуючих аналогів: обмежена гнучкість, слабка інтеграція з іншими сервісами, низький рівень автоматизації сповіщень користувачів, а також відсутність глибокого аналізу статистики черг. Вирішенням цих проблем стало створення покращеної системи на базі сучасного стеку технологій: Flask (як серверна платформа), MySQL (як СУБД), HTML/CSS/JS (для створення адаптивного клієнтського інтерфейсу), SMTP (для автоматизованих сповіщень).

У межах реалізації системи було впроваджено такі ключові функціональні модулі:

1. Форма реєстрації користувача з підтвердженням на електронну пошту - після реєстрації на вказану адресу надсилається лист із номером у черзі та кнопкою підтвердження. Тільки після підтвердження користувач потрапляє до активної черги.
2. Адміністративна панель з можливістю:
   - перегляду черги в режимі реального часу;
   - виклику користувача (виклик одного клієнта блокує можливість виклику іншими адміністраторами);
   - завершення обслуговування викликаного клієнта.
3. Система авторизації для адміністраторів, яка дозволяє кільком працівникам працювати паралельно, при цьому уникати конфліктів при виклику клієнтів завдяки контролю статусів.

4. Механізм оновлення інформації в реальному часі - черга та поточний виклик оновлюються без потреби ручного перезавантаження сторінки (через AJAX-запити).
5. Інтеграція з SMTP для надсилання листів підтвердження реєстрації, що забезпечує первинну валідацію користувачів та захист від спаму.

Проведене тестування системи підтвердило її працездатність та ефективність в умовах реального використання. Зокрема, система стабільно функціонує при високому навантаженні, коректно обробляє запити користувачів та виконує розсилку сповіщень у визначений час. Було виявлено кілька незначних технічних моментів, які можна оптимізувати у подальшому, наприклад, покращення дизайну для мобільних пристроїв та розширення можливостей статистичного аналізу.

У результаті реалізації системи були сформульовані конкретні пропозиції та рекомендації:

1. Впровадження системи в організації з високим потоком відвідувачів (ЦНАП, медичні установи, МВС).
2. Інтеграція з мобільним додатком для зручності користувачів.
3. Розширення аналітичного модуля для побудови прогнозів завантаженості.
4. Підключення API месенджерів (Viber, WhatsApp) як альтернативного каналу сповіщень.
5. Реалізація багатомовної підтримки системи.
6. Забезпечення зберігання логів та резервного копіювання для підвищення надійності.

Таким чином, третій розділ не лише логічно завершив дослідження, а й довів до практичного втілення поставлену мету - удосконалення комп’ютеризованої системи електронної черги, що має потенціал до масштабування та впровадження в різних сферах обслуговування населення.

# ВИСНОВКИ

У результаті виконання кваліфікаційної (бакалаврської) роботи на тему «Система електронної черги громадського закладу» було реалізовано повний цикл дослідження, проєктування, реалізації та тестування програмного забезпечення, що відповідає сучасним вимогам до ефективного управління потоками користувачів у сервісних установах.

Систематизовано теоретичні основи організації електронних черг, наведено їх класифікацію, визначено проблеми традиційних систем керування чергами та мотиви для автоматизації цих процесів. Визначено типові сценарії застосування електронних черг у публічному та приватному секторах.

Проаналізовано сучасні аналоги (Qwaiting, QLess, QueueBee, ЦНАП м. Києва), виявлено особливості їх реалізації та встановлено недоліки, такі як складність у використанні, відсутність адаптивності, неможливість працювати з кількома адміністраторами, недостатній зворотний зв'язок із користувачем.

Уточнено вимоги до сучасної системи електронної черги. Сформульовано функціональні та нефункціональні вимоги з урахуванням досвіду аналогів і очікувань потенційної аудиторії. До функціональних включено: реєстрацію з підтвердженням пошти, виклик і завершення клієнта, адміністративне керування чергою. До нефункціональних - безпека, масштабованість, доступність, інтегрованість.

Розроблено архітектуру системи за принципами MVC, створено логічну структуру за допомогою DFD- і ER-діаграм, що забезпечують структурованість та простоту в подальшому розширенні системи.

Обґрунтовано вибір технологій: Flask як легкий та гнучкий фреймворк, MySQL як надійна СУБД, SMTP для підтвердження реєстрації, HTML/CSS/JS з AJAX - як інтерфейсна складова. Порівняння з альтернативами (Django, PostgreSQL, React) підтвердило доцільність використаних рішень.

Реалізовано основні модулі системи:

- інтерфейс користувача з реєстрацією;
- підтвердження електронної пошти;
- адміністративна панель з функціями виклику та завершення;
- API-ендпоінти для оновлення черги в реальному часі;
- захист доступу та валідація дій користувачів.

Проведено тестування реалізованої системи: функціональне, модульне, інтеграційне, тестування помилкових сценаріїв. Результати свідчать про відповідність заявленим вимогам і стабільність системи у типових умовах експлуатації.

Виявлено декілька недоліків з яких: відсутність розмежування ролей адміністраторів, слабкий захист SMTP, відсутність CSRF-захисту та логування дій. Запропоновано конкретні шляхи вирішення, серед яких – впровадження TLS, логування, ролей доступу, інтеграції з чат-ботами та аналітичними модулями.

Сформульовано практичні рекомендації щодо впровадження системи:

- для пілотного запуску в держустановах, медичних закладах, університетах;
- поступове додавання нових модулів (аналітика, попередній запис, push-сповіщення);
- інтеграція з зовнішніми платформами (Viber, документообіг, “Дія”).

Таким чином, встановлено, що створена комп’ютеризована система електронної черги є ефективним інструментом, який підвищує якість сервісу, зменшує навантаження на персонал і забезпечує прозорість процесів. Робота має як теоретичну цінність - шляхом узагальнення підходів, так і практичну значущість - як готовий до впровадження програмний продукт.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Асаєнко Ю. С., Неронов С. М., Костікова М. В. Метод створення пристрою обробки різнотипних даних в системах прийняття рішень. Матеріали конференції «Сталий розвиток сучасних інформаційних технологій». Харків, 2025.

2. Бабіч Ю.В. Системи керування чергами в сучасних організаціях // Вісник ХНУРЕ. – 2021. – №4. – С. 12–17.

3. Балабанова І.Т. Інформаційні системи та технології: підручник. – Київ: Центр учбової літератури, 2022. – 378 с.

4. Березовський Ю.А. Сучасні методи автоматизації черг // Інформатика та інформаційні технології. 2020. №3. С. 41–45.

5. Бондаренко Н. Н. Інтеграція Flask з MySQL у веб-додатках // Науковий вісник Дніпровського національного університету. 2014. № 4. С. 20–25.

6. Гавриленко І.М. Моделювання систем обслуговування // Інформ. технол. та комп'ют. інженерія. 2019. №2. С. 78–85.

7. Гриценко Л. Л. Створення адаптивних веб-інтерфейсів з HTML/CSS // Науковий журнал «Сучасні інформаційні технології». 2016. № 2. С. 10–15.

8. Данилюк А. А. Безпека веб-додатків: навч. посіб. Київ: КНУ, 2017. 230 с.

9. Данилюк А. А. Безпека даних у веб-додатках на Flask // Науковий вісник ЧНУ. 2018. № 6. С. 40–45.

10. Іваненко П. П. Використання Flask у розробці веб-додатків // Вісник КНУ імені Тараса Шевченка. 2023. № 5. С. 45–50.

11. Іванченко Т. Т. Системи електронних черг: теорія та практика. Харків: ХНУРЕ, 2019. 198 с.

12. Ковальчук І. І. Веб-програмування: HTML, CSS, JavaScript. Чернівці: ЧНУ, 2022. 280 с.

13. Ковальчук І. І. Розробка інтерфейсів з використанням HTML/CSS // Науковий журнал «Інформаційні технології». 2020. № 4. С. 15–20.

14. Корнага Я., Базака Ю., Мар’єнко Є. Способи оптимізації SQL–запитів для поліпшення роботи з базою даних в високонавантажених системах // Вісник НТУУ «КПІ». 2020. № 37.

15. Корнага Я., Базака Ю., Мар’єнко Є. Способи оптимізації SQL–запитів для поліпшення роботи з базою даних в високонавантажених системах. Вісник НТУУ «КПІ». 2020. № 37. DOI: 10.20535/1560-8956.37.2020.226800.

16. Кравченко С. С. Оптимізація алгоритмів у системах електронних черг // Вісник СумДУ. 2013. № 1. С. 50–55.

17. Крамаренко С. С. Основи баз даних: навч. посіб. Київ: КНЕУ, 2020. 312 с.

18. Литвиненко Т. Т. Впровадження JavaScript у динамічні веб-сторінки // Зб. наук. пр. «Комп’ютерні науки». 2019. № 1. С. 25–30.

19. Литвиненко Т. Т. Основи програмування на Python. Дніпро: ДНУ, 2020. 310 с.

20. Матеріали VIII Всеукраїнської студентської науково-практичної конференції «Вища освіта – студентська наука – сучасне суспільство: напрями розвитку». Київ, 2024.

21. Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених. Одеса, 2025.

22. Матеріали конференції «Сучасні інформаційні технології та системи». Київ, 2024.

23. Мельник В. В. Аналіз продуктивності MySQL у веб-сервісах // Вісник ОНУ. 2017. № 3. С. 55–60.

24. Олійник Ю. Ю. Розробка веб-додатків з використанням Python та Flask // Науковий журнал «Комп’ютерні технології». 2012. № 3. С. 60–65.

25. Петренко О. О. Алгоритми оптимізації в електронних чергах // Наукові записки НУ «Львівська політехніка». 2022. № 3. С. 60–65.

26. Петренко О. О. Розробка веб-додатків з використанням Flask: навч. посіб. Львів: ЛНУ, 2021. 256 с.

27. Сучасні методи штучного інтелекту для навчання програмуванню // Вісник ХНТУ. 2021. № 1. С. 45–52.

28. Сучасні методи штучного інтелекту для навчання програмуванню. Вісник Херсонського національного технічного університету. 2021. № 1. С. 45–52. URL: https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/864.

29. Bonney M. S., de Angelis M., Dal Borgo M. et al. Development of a digital twin operational platform using Python Flask // Data-Centric Engineering. 2022. 3: e1. DOI: 10.1017/dce.2022.1

30. Buitinck L., Louppe G., Blondel M. et al. API design for machine learning software: experiences from the scikit-learn project // arXiv preprint. 2013. arXiv:1309.0238.

31. Creating a Web App From Scratch Using Python Flask and MySQL – TutsPlus. URL: https://code.tutsplus.com/

32. CSS Official Documentation (MDN). URL: https://developer.mozilla.org/en-US/docs/Web/CSS

33. Dash D., Polyzotis N., Ailamaki A. CoPhy: A Scalable, Portable, and Interactive Index Advisor for Large Workloads. arXiv preprint. 2011. arXiv:1104.3214. URL: https://arxiv.org/abs/1104.3214.arXiv

34. Flask Official Documentation. URL: https://flask.palletsprojects.com/en/2.0.x/

35. HTML Living Standard. URL: https://html.spec.whatwg.org/

36. JavaScript Guide (MDN). URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

37. Larson P.-Å., Blanas S., Diaconu C., Freedman C., Patel J. M., Zwilling M. High-Performance Concurrency Control Mechanisms for Main-Memory Databases. arXiv preprint. 2011. arXiv:1201.0228. URL: https://arxiv.org/abs/1201.0228.arXiv

38. MySQL Official Documentation. URL: https://dev.mysql.com/doc/

39. Profile Application Using Python Flask and MySQL – GeeksforGeeks. URL: https://www.geeksforgeeks.org/profile-application-using-python-flask-and-mysql/

40. Python Official Documentation. URL: https://docs.python.org/3/

41. Raschka S., Patterson J., Nolet C. Machine Learning in Python // arXiv preprint. 2020. arXiv:2002.04803.

42. Virtanen P., Gommers R., Oliphant T. E. et al. SciPy 1.0 – Fundamental Algorithms for Scientific Computing in Python // arXiv preprint. 2019. arXiv:1907.10121.

43. Yakutovich A. V., Eimre K., Schütt O. et al. AiiDAlab – an ecosystem for developing, executing, and sharing scientific workflows // arXiv preprint. 2020. arXiv:2010.02731.

44. Zhu Y., Liu J., Guo M., Bao Y., Ma W., Liu Z., Song K., Yang Y. BestConfig: Tapping the Performance Potential of Systems via Automatic Configuration Tuning. arXiv preprint. 2017. arXiv:1710.03439. URL: https://arxiv.org/abs/1710.03439.arXiv

ДОДАТОК А

**Тези доповіді**

**УДК 004.8:613.2**

*Короленко С. О., здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки*
*Веселовська Н. Р., професор кафедри прикладної математики та кібербезпеки*

## РОЗРОБКА СИСТЕМ ЕЛЕКТРОННОЇ ЧЕРГИ ГРОМАДСЬКОГО ЗАКЛАДУ

*Донецький національний університет імені Василя Стуса, м. Вінниця*

У сучасному динамічному світі, що стрімко трансформується під впливом цифрових технологій, зростає потреба у вдосконаленні підходів до організації взаємодії між громадянами та різноманітними сервісними структурами, зокрема державними установами. На тлі зростаючої значущості таких понять як зручність, доступність, ефективність і прозорість обслуговування, все частіше постає необхідність у впровадженні нових рішень, здатних забезпечити якісно новий рівень надання послуг. Одним із напрямів, що активно розвивається в цьому контексті, є автоматизація черг як невід'ємної частини сервісного обслуговування населення.

Традиційні механізми, до яких звикли ще з минулого століття — паперові списки, черги "вживу", суб'єктивні домовленості — дедалі частіше демонструють свою недієвість в умовах зростаючого потоку клієнтів, підвищених вимог до точності та обліку, а також нових викликів, що стосуються безпеки, зокрема в контексті епідеміологічних загроз. Усе це формує соціальний запит на більш гнучкі, технологічні й водночас зручні для користувача рішення, що мають здатність адаптуватися до потреб конкретної установи або навіть ситуації.

У зв'язку з цим зростає актуальність розробки інформаційних систем, які б не лише замінювали паперові черги на електронні таблиці, а й формували принципово нову логіку організації процесу обслуговування. Йдеться про створення систем, які враховують людський фактор, потребу в зворотному зв'язку, необхідність контролю й аналітики, а також готовність до розширення й інтеграції з іншими сервісами. В цьому контексті особливе значення набуває концепція електронної черги — інструменту, який має на меті не просто регулювати потоки відвідувачів, а створювати цілісну систему управління часом, ресурсами й взаємодією.

Розробка подібної системи є складним і багатогранним процесом, що передбачає не лише суто програмну реалізацію, а й врахування психологічних, соціальних та організаційних аспектів. У ході дослідження було поставлено

завдання створити таку систему електронної черги, яка б ураховувала недоліки наявних підходів, водночас залишаючись доступною, простою у використанні та потенційно масштабованою. У фокусі розробки перебувала ідея створення рішення, яке не вимагало б значних технічних знань для впровадження, але могло б забезпечити відчутне полегшення як для клієнтів, так і для співробітників установ.

У процесі реалізації проєкту значна увага приділялася не лише технічним аспектам, а й концептуальному наповненню — структурі взаємодії між користувачем і системою, сценаріям роботи, зручності інтерфейсу. Було здійснено спробу поєднати сучасні технологічні можливості з потребами конкретних установ, таких як центри надання адміністративних послуг, медичні заклади, університети, де потік відвідувачів постійний, а ефективність організації має вирішальне значення.

Також важливо наголосити, що ця система не є виключно технічним проєктом — вона є частиною загального бачення цифрової трансформації сервісного простору. Її впровадження має потенціал впливати не лише на швидкість чи порядок обслуговування, але й на загальне враження громадян від взаємодії з державою чи іншими інституціями. Це враження, сформоване завдяки дрібницям — чи легко записатися, чи зрозуміло відслідковувати свій статус, чи можна уникнути зайвого очікування — і є тим, що формує рівень довіри й задоволеності.

Отже, в межах цієї кваліфікаційної роботи було не лише створено базову програмну реалізацію, а й проведено комплексне осмислення ідеї електронної черги як соціально значущого сервісу. Результатом стало рішення, яке в подальшому може бути адаптоване, розширене, інтегроване з іншими сервісами, зокрема через API або спеціальні модулі. Перспективи його розвитку включають і автоматизовану аналітику, і інтеграцію з мобільними платформами, і навіть елементи штучного інтелекту — однак вже зараз система є повноцінним функціональним продуктом, готовим до пілотного впровадження в умовах реального середовища.

**Список використаних джерел**


1. Бабіч Ю.В. Системи керування чергами в сучасних організаціях // Вісник ХНУРЕ. – 2021. – №4. – С. 12–17.

2. Балабанова І.Т. Інформаційні системи та технології: підручник. – Київ: Центр учбової літератури, 2022. – 378 с.

3. Березовський Ю.А. Сучасні методи автоматизації черг // Інформатика та інформаційні технології. – 2020. – №3. – С. 41–45.

4. Власенко В.О. Технології веб-програмування. – Київ: Вища школа, 2022. – 290 с.

5. Гавриленко І.М. Моделювання систем обслуговування // Інформ. технол. та комп'ют. інженерія. – 2019. – №2. – С. 78–85.

# ДОДАТОК Б

## Декларація щодо унікальності текстів роботи та невикористання матеріалів інших авторів без посилань

**ДЕКЛАРАЦІЯ**
про дотримання академічної доброчесності

Я, _____
_____
*Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)*

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

**ЗОБОВ'ЯЗУЮСЬ:**

дотримуватися принципів та правил академічної доброчесності, щовизначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями,правилами, умовами, визначеними іншими суб'єктами, та не допускати їхпорушення.

**ПІДТВЕРДЖУЮ:**

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

**УСВІДОМЛЮЮ:**

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____
(дата) (підпис)