Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота
за освітнім ступенем “бакалавр”
на тему:

АВТОМАТИЗОВАНА СИСТЕМА ПОДАЧІ ТА ОБРОБКИ СТУДЕНТСЬКИХ ЗВЕРНЕНЬ У ЗВО

Виконав:
здобувач IV курсу
групи КН-41
спеціальності 122 «Комп’ютерні науки»
Микола Миколайович Бруяка

Науковий керівник:
кандидат технічних наук,
доцент Степанія Михайлівна Бабич

Рівне – 2026

ЗМІСТ

СПИСОК ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

ЗВО – заклад вищої освіти.

СУБД – система управління базами даних.

API (Application Programming Interface) – інтерфейс прикладного програмування.

UX/UI (User Experience / User Interface) – користувацький досвід та користувацький інтерфейс.

JSON (JavaScript Object Notation) – текстовий формат обміну даними.

ВСТУП

Актуальність роботи. У закладах вищої освіти значну роль відіграє організація інформаційної взаємодії між студентами та адміністрацією. Процес подачі та обробки студентських звернень охоплює широкий спектр питань, пов'язаних з навчальною, організаційною, соціальною та адміністративною діяльністю. Поточні способи обробки звернень, які базуються на паперових документах є не ефективними, оскільки призводять до затримок у прийнятті рішень, підвищують ризик втрати інформації і знижують рівень контролю заявок.

Одним із шляхів вирішення проблем є впровадження автоматизованих інформаційних систем, що забезпечують централізоване зберігання даних, прозорість процесів та швидку обробку звернень. Використання цих рішень дає мережі, та значно спрощує взаємодію між адміністрацією та студентами.

Автоматизовані системи подачі та обробки звернень базуються на використанні баз даних, які забезпечують структуроване зберігання інформації, а також систем, що реалізують зручну та інтуїтивно зрозумілу взаємодію з користувачем. Такі системи дозволяють не лише обробляти поточні звернення, але й накопичувати дані, необхідні для аналізу та оптимізації управлінських процесів у закладах вищої освіти [20]. Розробка автоматизованої системи подачі та обробки студентських звернень є актуальним завданням.

Метою роботи є розробка та програмна реалізація автоматизованої системи подачі та обробки студентських звернень у закладі вищої освіти, а також тестування ефективності його функціоналу.

Для досягнення поставленої мети в роботі були сформульовані та вирішені такі **завдання**:

- аналіз існуючих рішень електронної взаємодії та обробки звернень у закладах вищої освіти;

- визначити функціональні та нефункціональні вимоги до автоматизованої системи;
- розробити структуру бази даних для зберігання інформації про користувачів та звернення;
- створити вебінтерфейс для подачі та обробки студентських звернень з урахуванням ролей користувачів;
- реалізувати основні функції системи засобами сучасних вебтехнологій;
- перевірити працездатність розробленої системи на тестових даних.

Об’єкт дослідження – процес організації комунікації та управління інформацією між студентами та адміністрацією закладу вищої освіти.

Предмет дослідження – програмні засоби та технології розробки автоматизованих систем подачі й обробки студентських звернень у ЗВО.

Методи дослідження:

1. аналіз наукових і навчальних джерел з інформаційних систем та вебтехнологій;
2. структурно-функціональний та об’єктно-орієнтований аналіз при проєктуванні системи;
3. моделювання структури бази даних та логіки роботи вебзастосунку;
4. програмна реалізація та тестування розробленого вебінтерфейсу.

Практичне значення дослідження. Розроблена автоматизована система може бути використана в закладах вищої освіти для оптимізації процесу подачі та обробки студентських звернень, підвищення прозорості управлінських рішень та зменшення часу реагування на звернення студентів.

Запропонований підхід до проєктування вебінтерфейсу та структури бази даних може бути застосований під час створення інших інформаційних систем, орієнтованих на автоматизацію внутрішніх процесів у закладах освіти та організаціях різного профілю.

Апробація і впровадження результатів дослідження. Основні теоретичні та практичні результати дослідження доповідалися та обговорювалися на звітній науковій конференції викладачів, співробітників і здобувачів вищої освіти Рівненського державного гуманітарного університету за 2025 рік (м. Рівне, 14 травня 2026 року).

Структура роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел. Загальний обсяг роботи становить 44 сторінки. Список використаних джерел включає 20 найменувань.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Особливості організації студентських звернень у закладах вищої освіти

Під час функціонування закладів вищої освіти виникають питання щодо необхідності взаємообміну даними між студентами та підрозділами адміністрація ЗВО. Звернення є невід'ємною і важливою частиною цього процесу, вони охоплюють такі частини навчального процесу як: академічні питання, документальні запити, соціально-фінансові питання, навчальні плани, соціальне забезпечення і т.д. [16].

Майже завжди подача та обробка звернень відбувається, або у паперовому форматі, або неформально, за допомогою прямого зв'язку з викладачем, наприклад: в телефонному режимі, за допомогою електронної пошти або месенджерів. Ці формати є недосконалими та мають низку проблем: ризик втрати даних, відсутність спільного місця зберігання інформації, не можна швидко та ефективно проаналізувати статистику звернень.

Також варто зазначити, що без чіткого алгоритму оброблень звернень, ускладняється зв'язок між здобувачами вищої освіти та виконавцями звернень, що тягне за собою зниження рівня задоволеності та ефективності керування освітнім процесом. Тому задачі впровадження автоматизованих систем для подачі заявок у закладах вищої освіти, які дадуть можливість ефективно контролювати процес роботи з студентськими зверненнями є доцільними.

1.2 Необхідність автоматизації процесу подачі та обробки звернень

Автоматизація – це один з головних напрямків розвитку процесів інформаційних систем **[**10**]**. Ці рішення дозволяють спростити та оптимізувати робочі процеси, розвантажити персонал та підвищити оперативність прийняття рішень.

Автоматизована система подачі та обробки студентських звернень повинна надавати:

1. зручний механізм створення та подачі звернень;
2. збереження та захист всієї інформації у структурованому вигляді;
3. безперервний моніторинг статусу звернень на кожному етапі їх обробки;
4. можливість коментування та зворотного зв’язку,розмежування доступу відповідно до ролей користувачів;
5. оперативно змінювати стан виконання запиту та зворотного зв’язку з обох сторін.

Використання систем як основного засобу дозволяє використати інтеграцію та кросплатформеність, без необхідності використання додаткового програмного забезпечення зі сторони користувача. Виходячи з цього. Автоматизація буде сприяти зросту якості та ефективності взаємодій у ЗВО.

1.3 Аналіз існуючих інформаційних систем та вебрішень

Наразі заклади вищої освіти використовую різні програмні рішення, які частково або повністю реалізують функціонал взаємодії. Їх можна розділити на три основні групи:

- *великі корпоративні системи документообігу («Вчасно»)*. Основним недоліком таких систем є те, що вони створені для державних установ і т.д. Для студента, якому потрібно швидко замовити довідку інтерфейс

таких систем може бути складним. Також більшість таких систем є комерційними. Крім того ці системи потребують значне фінансування;

- *навчальні платформи(Moodle)*. Використання навчальних платформ для подачі заяв – це погане рішення. Moodle створений сам під навчальні задачі, у ньому відсутні потрібні інструменти для коректної роботи системи подачі звернень.

Аналіз показує, що оптимальним рішенням є розробка власної системи, яка буде включати в себе необхідних функціонал та простоту у використані, відсутність візуального «сміття» та адаптацію до різних умов.

1.4 Основні вимоги до автоматизованої системи студентських звернень

Базуючись на аналізі існуючих рішень та предметної області можна сформулювати основні вимоги [13]. Їх можна розділити на 3 групи: функціональні, технічні та безпекові. Так як система вміщає в себе 2 різні групи користувачів, то і вимоги до їх функціоналу будуть відрізнятися.

1. *Функціональні вимоги*.

1.1. Для студентів:

- Проста аутентифікація. Здобувач вищої освіти має швидко та інтуїтивно зрозуміло заходити на проєкт.
- Подача звернень. Можливість вибору шаблону та підрозділу для звернень. Змога самому писати текст та добавляти вкладені файли.
- Відстеження статусів. Користувач повинен бути проінформованим про хід роботи над його заявкою як на сайті, так і за допомогою додаткових джерел (Телеграм бот).
- Зворотній зв'язок. Можливість бачити і відповідати на коментарі виконавця.

1.2. Для адміністраторів:

- Єдиний реєстр. Всі звернення повинні бути в одній таблиці з можливістю їх сортування.
- Керування статусами. Адміністратор має змогу змінювати статус заявки в один клік для кожного процесу обробки.
- Логування. Важливо, щоб система записувала, хто обробив заявку та який коментар залишив, щоб звести до мінімуму проблеми в майбутньому.
- База користувачів. Доступ до даних всіх зареєстрованих користувачів.

2. *Технічні вимоги*

- Швидкодія. Проєкт не повинен навантажувати сервера. Робота на базі PHP + SQL надає швидке завантаження сторінок.
- Автономність. Можливість розгортання на локальному сервері (XAMPP) [18]. Без використання складного налаштування.

3. *Вимоги безпеки*

- Цілісність. При видалені користувача, також автоматично видаляються його заявки з бази динних (каскадне видалення).
- Розмежування прав. Користувачі з нижчими правами не мають ніякими способами отримати доступ до адмінпанелі та заявок інших студентів. Перевірка відбувається на рівні сервера за допомогою сесій.
- Захист від SQL-ін’єкцій. Все, що взаємодіє з базою даних, повинно використовувати підготовлені запити, для захисту від взлому через поля введення.

1.5 Вибір мови програмування та технології серверної частини

Вибір мови програмування є одним з найважливіших рішень, тому що це визначить продуктивність, швидкість та безпеку. У межах даної роботи вибір стояв з 3 найпопулярніших технологій веброзробки: PHP, Python (Django) та JavaScript (Node.js).

Аналіз:

1. Python (Django/Flask). Має початковий високий рівень безпеки. Порівняно з іншими мовами програмування має складнішу конфігурацію середовища.
2. JavaScript (Node.js). Надає високу швидкість обробки великої кількості одночасних з'єднань. Але для системи, де більшу роль відіграють операції запису в базу даних, переваги Node.js не є вагомими, а складність підтримки є вищою.
3. PHP. Ця мова була обрана для розробки. Вона має широкий інструментарій для захисту паролів, фільтрації даних та підготовленими запитами MySQL [2], [5]. Ця мова програмування була створена спеціально для кращої інтеграції з HTTP-протоколами [1].

1.6 Аналіз та вибір системи управління базами даних

Другим важливим питанням став вибір системи управління базами даних. Основний вибір був між реляційними і нереляційними системами.

Взявши до уваги, що система матиме чітку структуру даних (студент – факультет – заявка – адміністратор – чат), було вирішено, що використання NoSQL рішень призведе до дублювання даних.

З реляційних СУБД було вибрано MySQL завдяки тому, що вона зазвичай іде у зв'язці з PHP, що забезпечує стабільну роботу, і показує вищу швидкість вибірки даних [6].

РОЗДІЛ 2
ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТА СТРУКТУРИ БАЗИ ДАНИХ

2.1 Загальна характеристика автоматизованої системи

Розроблена автоматизована система подачі та обробки студентських звернень концептуально побудована як Service Desk. Це вебзастосунок з централізованою базою даних, який працює за принципом єдиного вікна. Основа система це розмежування функціоналу для різних ролей.

Загальна логіка:

- Студенти. Мають доступ лише до особистого кабінету, де можуть бачити виключно свої заявки або подати нові. При створенні нових запитів здобувач вищої освіти може вибрати готовий шаблон, додати власний текстовий опис і додати файл для підтвердження проблеми. Головний плюс – можливість бачити статус своїх заявок в реальному часі.
- Виконавці (Адміністрація). Отримують доступ до робочої панелі лише свого відділу. Можуть бачити та виконувати заявки які направлені на їх підрозділ. Щодо роботи з заявками, виконавець може: прийняти їх у роботу, повернути для уточнення, виконати або завершити, також він має доступ до чату зі студентом, який подав заявку для уточнення даних.
- СуперАдмін. Має інструменти для створення нових користувачів (тільки суперадмін може створювати нових виконавців, у той час як студенти можуть самі зареєструватися) та редагування бази студентів.

Для спрощення користування було закладено декілька функціональних особливостей.

- Тікети. Кожна заявка – це окремий чат. Виконавець і студент можуть у будь-який момент зв’язатися для уточнення або змін у середині конкретної заявки.
- Автоматичний розподіл навантаження. Система автоматично вирішує кому з виконавців надати нову заявку. Рішення системи базується на завантаженості працівників окремого підрозділу. Тобто за працівником який має найменше активних заявок, закріпиться нова.
- Сповіщення через додаткові джерела. До системи підключений власний Телеграм бот, який дає змогу студентами отримувати сповіщення про зміну статусу їх заявок в будь-який момент, не заходячи постійно на сайт. Користувач може прив’язати, за допомогою номеру телефона, свій акаунт до бота.

У цілому, система оптимізувала бюрократію і перетворила її в простий сервіс, зрозумілий кожному

2.2 Архітектура системи

Архітектура системи побудована на основі «Cient-Server Architecture» [3]. Систему оформлено як кросплатформний вебзастосунок, розмежування на базу даних, клієнтську частину та серверну (рис. 2.1).

Серверна частина (Backend).

Основною мовою програмування виступає PHP. Вибір впав на неї із-за гнучкості та швидкості розгортання.

Архітектурно бекенд поділений на логічні модулі.

- Модуль аутентифікації. Проводить перевірку даних користувачів. У цьому моделі також добавлений алгоритм хешування паролів, для захисту інформації. Розмежування доступів виконується через механізм серверних сесій PHP ($_SESSION), що блокує несанкціоновані доступи до закритих сесій.

- Модуль обробки запитів. Створює нову заявки, обробляє їх, змінює статуси та веде історію в журнал логування. Зв'язок з базою даних проходить через розширення MYSQL.
- API-модуль. Створено окремий скрипт, який є шлюзом між базою даних та серверами Telegram.

Клієнтська частина (Frontend).

Клієнтська частина створюється на сервері, і подається користувачі, як HTML розмітка. Для зовнішнього вигляду використовується Bootstrap 5, що дало змого використовувати як комп’ютери, так і смартфони для роботи з вебінтерфейсом.

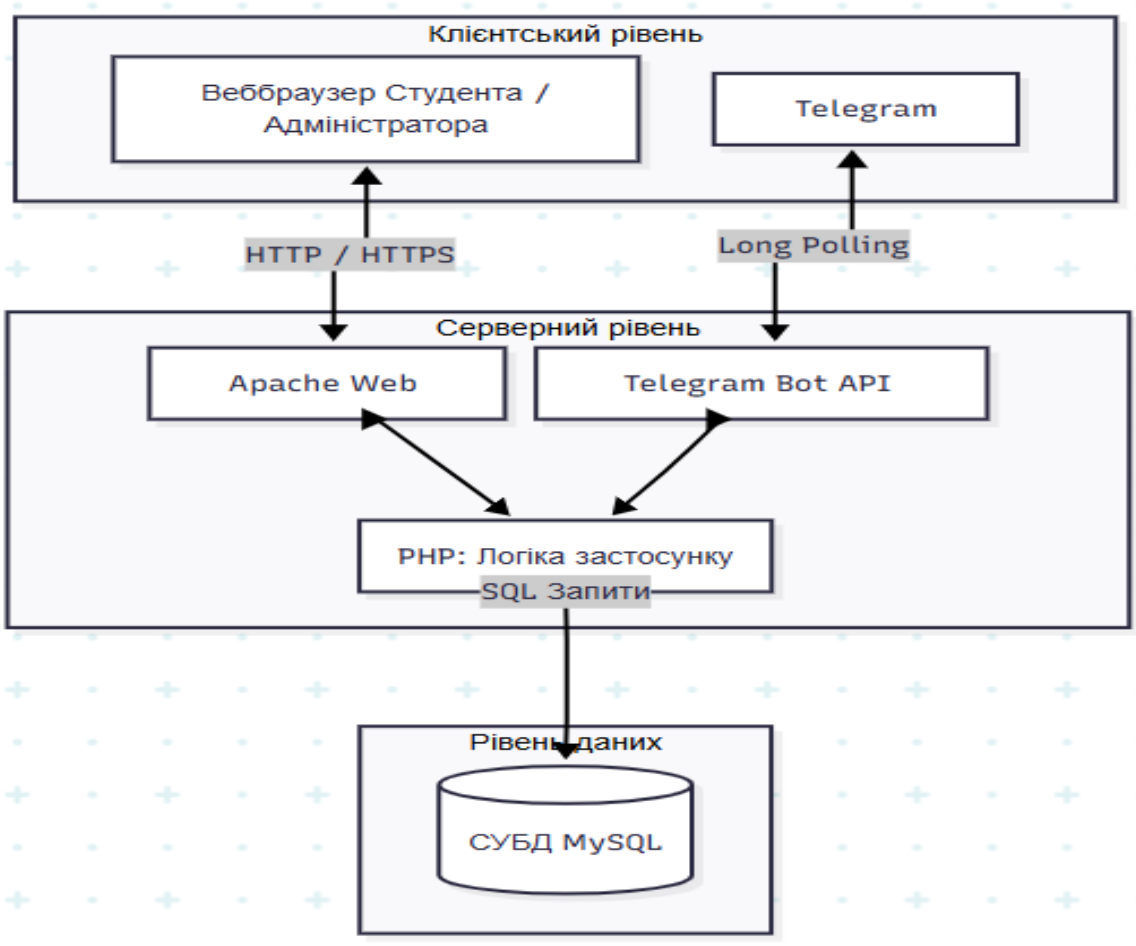


Рисунок 2.1. Архітектура системи

2.3 Проєктування структури бази даних

База даних є основою системи, тому було вибрано реляційну модель СУБД, вона дає змогу встановити зв’язки, при цьому не дублювати дані [4], [12].

У базі даних «student_requests_db» кожен окремий елемент зберігається в одному місці, а доступ з інших таблиць ми маємо через посилання ідентифікаторів.

Усі таблиці можна поділити на 3 групи.

1. Довідники.

- «faculties» та «departments». Включають у себе назви факультетів та відділів.
- «request_tempalates». Збергіає готові шаблони. З боку студента – не потрібно вигадувати запит з 0, з боку адміністрації – отримання стандартних запитів, які легше обробити.

2. Користувачі та профілі.

- «USERS». Основна таблиця. Зберігає лише логіни, паролі та ролі.
- «STUDENTS». Таблиця для студентів. Зберігає ім’я, номер телефону та телеграм айді, що є основою для контакту акаунту на стай з телеграм ботом.
- «STAFF». Таблиця для виконавців. Обов’язкова прив’язка до конкретного департаменту (department_id).

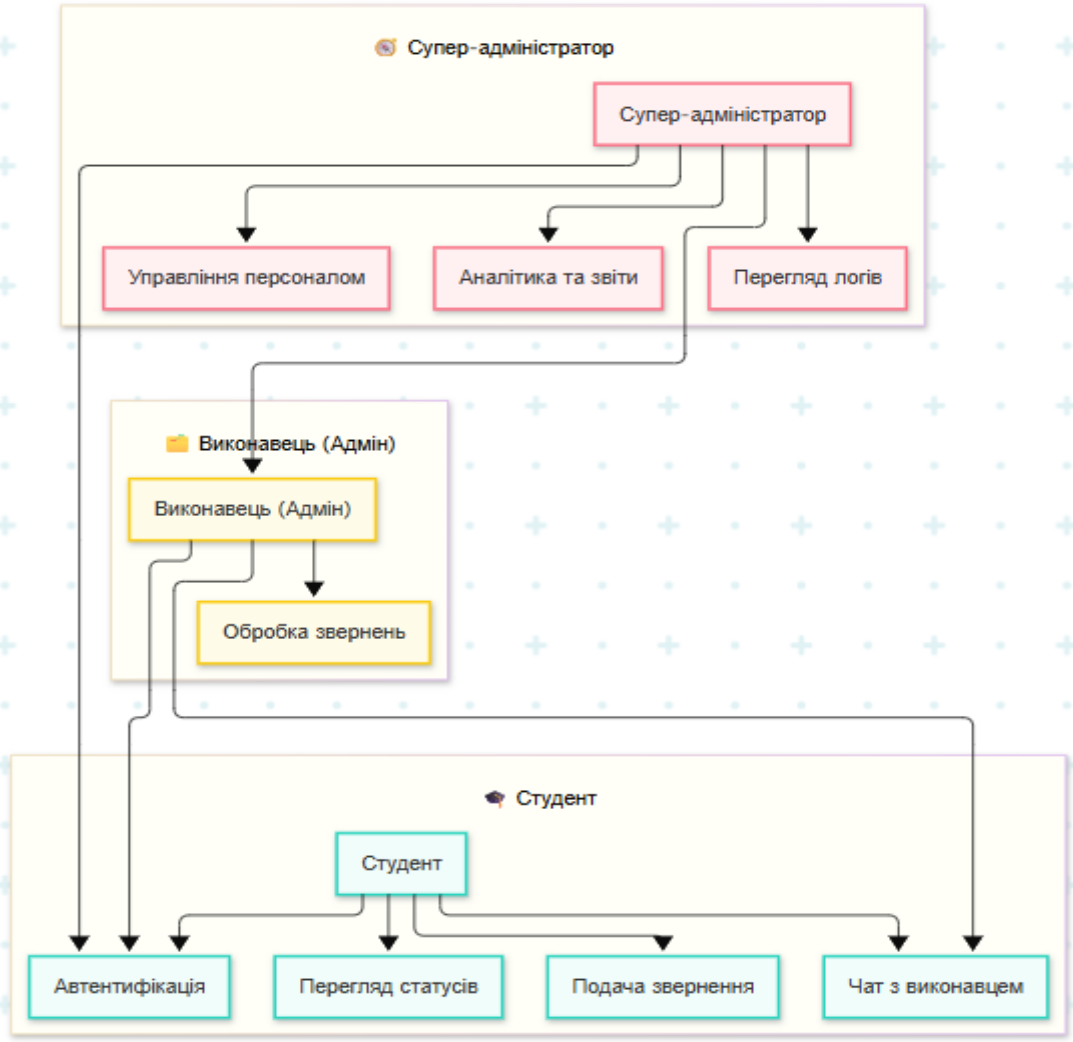


Рисунок 2.2. Діаграма прецедентів системи

3. Ядро системи

- «requests». Містить зміст звернення, та метадані: пріоритет, шлях до файлу, ID виконавця, до якого прив’яже система.
- «ticket_messages». Забезпечує роботу чату. Кожне повідомлення кріпиться до (request_id), даючи змогу мати історію діалогу для кожної окремої проблеми.
- «request_processing». «Чорна скринька», яка зберігає всі зміни статусу та коментарі виконавців.

Щоб у системі не було непотрібних даних, у базі даних налаштовані зовнішні ключі. Для прикладу, коли видаляється акаунт користувача з

таблиці «users» система автоматично видалить його з «students» та всі повідомлення. Нижче представлена ER-діаграма (рис. 2.3.)

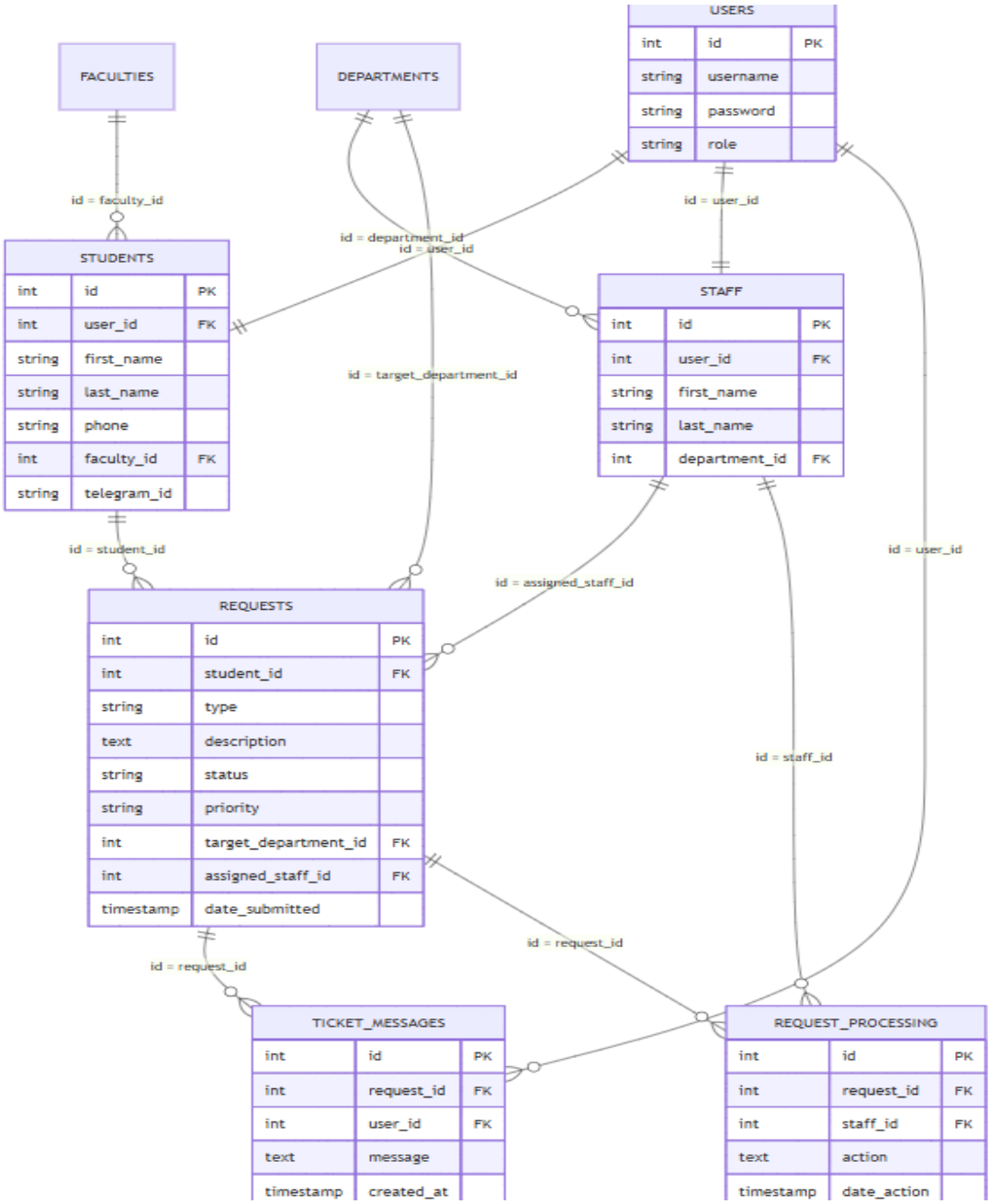


Рисунок 2.3. Узагальнена архітектура системи

2.4 Автоматизація та її функції

Система не лише являє собою сховище для заявок, але й бере небезпосередню роль у керуванні процесами, без прямого втручання зі сторони адміністрації.

З ключових функцій автоматизації можна виділити наступні:

- Авторозподіл заявок. У системі створено алгоритм автоматичного вибору виконавця. У момент подачі студентом заявки серверний скрипт виконує запит до таблиць і знаходить працівника з найменшою кількістю активних задач.
- Синхронізація з Telegram. Студенту потрібно лише поділитися контактом, і його автоматично синхронізує з сайтом і здобувач вищої освіти постійно отримуватиме статус своїх заявок, без потреби самому постійно моніторити сайт.
- Шаблони. Шаблони дають змогу пришвидшити процес заповнення форм завдяки механізму автоматичного заповнення. Використовуючи JavaScript-обробник на стороні клієнти можна підтягнути потрібний текст і одночасно заповнити поле опису.
- Архівування. У системі продумано скрипт, у заявках, які мають статус «вирішено» або «відмінено» понад 5 днів статус змінюється на «архівовано».

Систему можна назвати автоматизованою, тому що вона поєднує в собі роботу людей з автоматичним обробленням програмних алгоритмів, які пришвидшують і спрощують подачу та обробку заяв у закладах вищої освіти.

2.5 Інтеграція з месенджером Telegram та архітектура бота

Щоб спростити студентам отримання інформації, була додана мультиплатформенність. Вона введена в систему через інтеграцію з

месенджером Telegram, що дає змогу користувачам отримувати інформацію про статуси заявок, без постійної авторизації на сайті.

Для зв’язку використовується Telegram Bot API [8]. Взаємодія між виконується через скрипт «bot_sync.php», який працює за принципом вхідних запитів. Використання методу «getUpdates» дозволяє серверу з певною періодичністю запитувати сервера Телеграм про нові повідомлення від користувачів. Для оброблених повідомлень система зберігає їх у файлі «bot_offset.txt», що не дає обробити запит студента двічі, або пропустити.

Бот не є повною заміною сайту, він використовується як додатковий інформативний інструмент.

Основні функції бота:

- Перегляд активних заявок. Бот витягує останні 5 звернень, та показує їх поточний статус.
- Підтримка. Бот дає невелику інструкції, як використовувати систему.
- Сповіщення. Бот надсилає повідомлення, коли виконавці змінюють статус заявки.

Весь обмін даними відбувається через протокол HTTPS із використанням JSON-структур [19]. Для безпеки токен бота зберігається у прихованій змінній на сервері. Реалізація бота у вигляді окремого скрипта «bot_sync.php» дозволяє тримати його у фоновому режимі, що забезпечує високу швидкість реакції на запити користувачів.

2.6 Проєктування графічного інтерфейсу та UX-логіки системи

Для реалізації візуальної сторони вебінтерфейсу було використано фреймворк Bootstrap 5. Базою для всіх сторінок служить 12-колонкова сітка Bootstrap, що дає змогу динамічно змінювати розташування елементів. Увесь вміст системи загорнутий у «.container» .Цей підхід надає автоматичні відступи та центрування зон. Сторінки кобінатів діляться на логічні блоки з використанням класів .row та .col.. Такі налаштування реалізують Z-патерн,

що є найкращим для зчитування людським оком. Починаючи перегляд з верхнього лівого кута, де знаходиться роль, користувач рухається направо до кнопки виходу, після чого спускається вниз до таблиці звернень.

Базуючись на законі Фіттса, були побудовані ключові принципи розташування елементів керування [14]:

1. Так як кнопки підтвердження є логічним завершенням циклу, було вирішено розташувати їх праворуч або по центру внизу форми.
2. Була використана базова психологія кольорів:
   - btn-primary (синій) – основні дії (створення, перегляд);
   - btn-success (зелений) – позитивні фінали (виконання заявки, реєстрація);
   - btn-warning (жовтий) – дії, що потребують уваги (статус «потрібно уточнення»);
   - btn-danger (червоний) – критичні дії або відмова.
3. Елементи керування чатом на сторінці тікетів згруповані та відокремлені, що запобігає плутанню ознайомлення з документом і безпосередній діалог.
4. Окремо увагу було приділено інтерактивності. Для кнопок та полів вводу використано CSS-переходи, тобто користувач має візуальний відклик на дії, що дає відчуття «живого» застосунку.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ПРОГРАМНА ЛОГІКА СИСТЕМИ

3.1 Загальні принципи реалізації системи

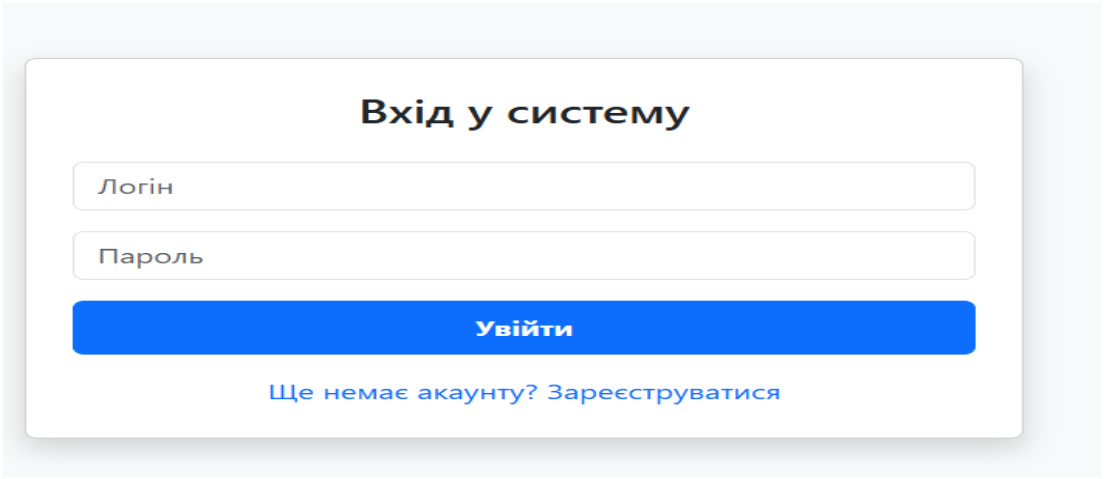
Основна ідея програмної реалізації полягає в тому, щоб створити не тільки сайт з формами, а інтерактивне середовище для пришвидшення обробок звернень. Логіка побудована, щоб зменшити ручну роботу.

1. Поділ на рівні доступу. Роль користувача перевіряється в кожному скрипті. Студенту доступні лише свої заявки, виконавцю – заявки відділу, до якого він відноситься, адміністратору – всі заявки.
2. Фіксування. Будь-яка дія одночасно записується в базі даних і міняє вигляд інтерфейсу для другої сторони.
3. Автоматизація маршрутів. Система сама вирішує, кому передати заяву, синхронізувати Телеграм з сайтом і архівувати старі заявки.

3.2 Реалізація функціоналу автентифікації та авторизації

Захист у системі виконується на двох рівнях:

1. Вхід (login.php / register.php): Паролі не зберігаються звичайним текстом. Під час реєстрування пароль обробляється функцією «password_hash()» з алгоритмом bcrypt.
2. Контроль сесій. Після входу зі сторони сервера створюється сесія «$_SESSION». Для захисту адмін-панелі від входу за прямим посиланням, на кожній сторінці стоїть перевірка: якщо роль користувача відрізняється від потрібної, системи посилає його на сторінку входу.



3.

Рисунок 3.1. Сторінка входу

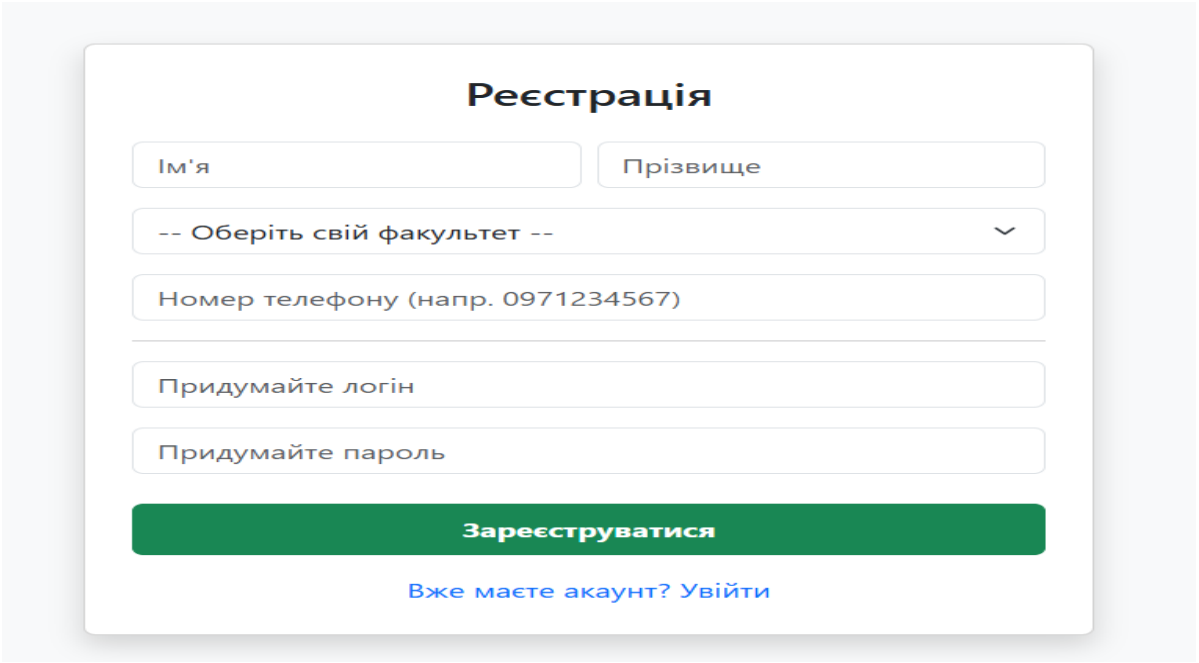


Рисунок 3.2. Сторінка реєстрації

3.3 Реалізація функціоналу подачі та перегляду звернень

Кабінет студент базується у файлі «dashboard.php». Під час запуску виконується запит з об’єднанням таблиць «requests» та «departments», який отримує звернення користувача і створює динамічну таблицю.

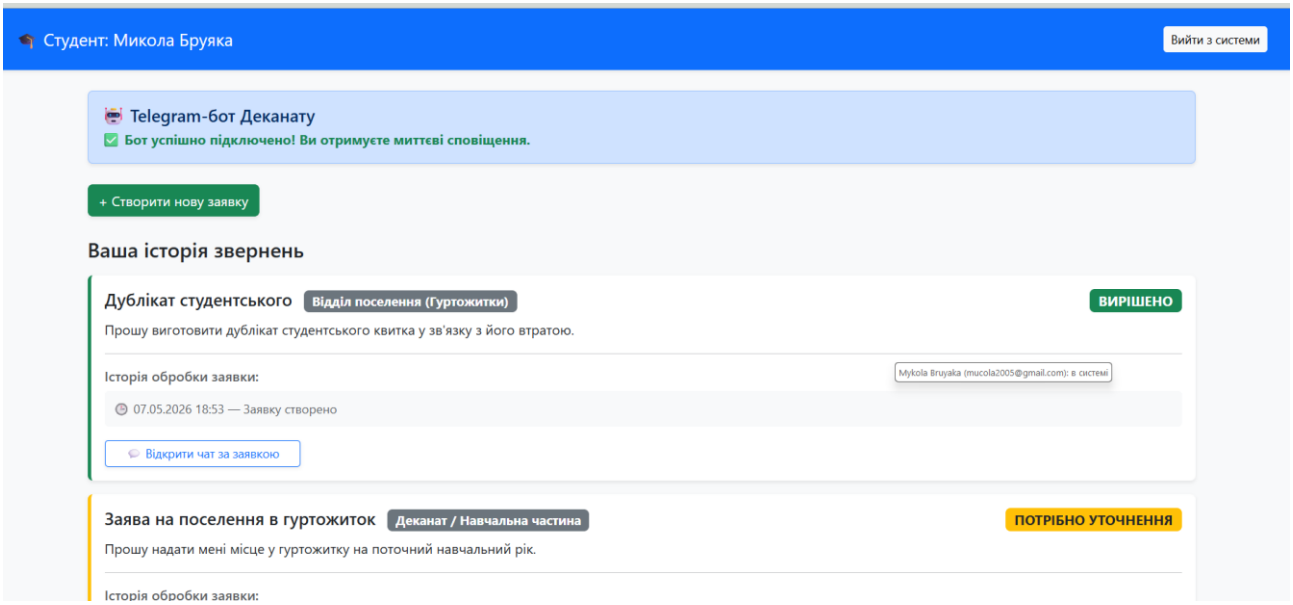


Рисунок 3.3. Особистий кабінет студента

Під час створення нового звернення можливий конфлікт імен на сервері, щоб запобігти цьому додана функція «uniqid()», який генерує унікальний префікс для кожного файлу.

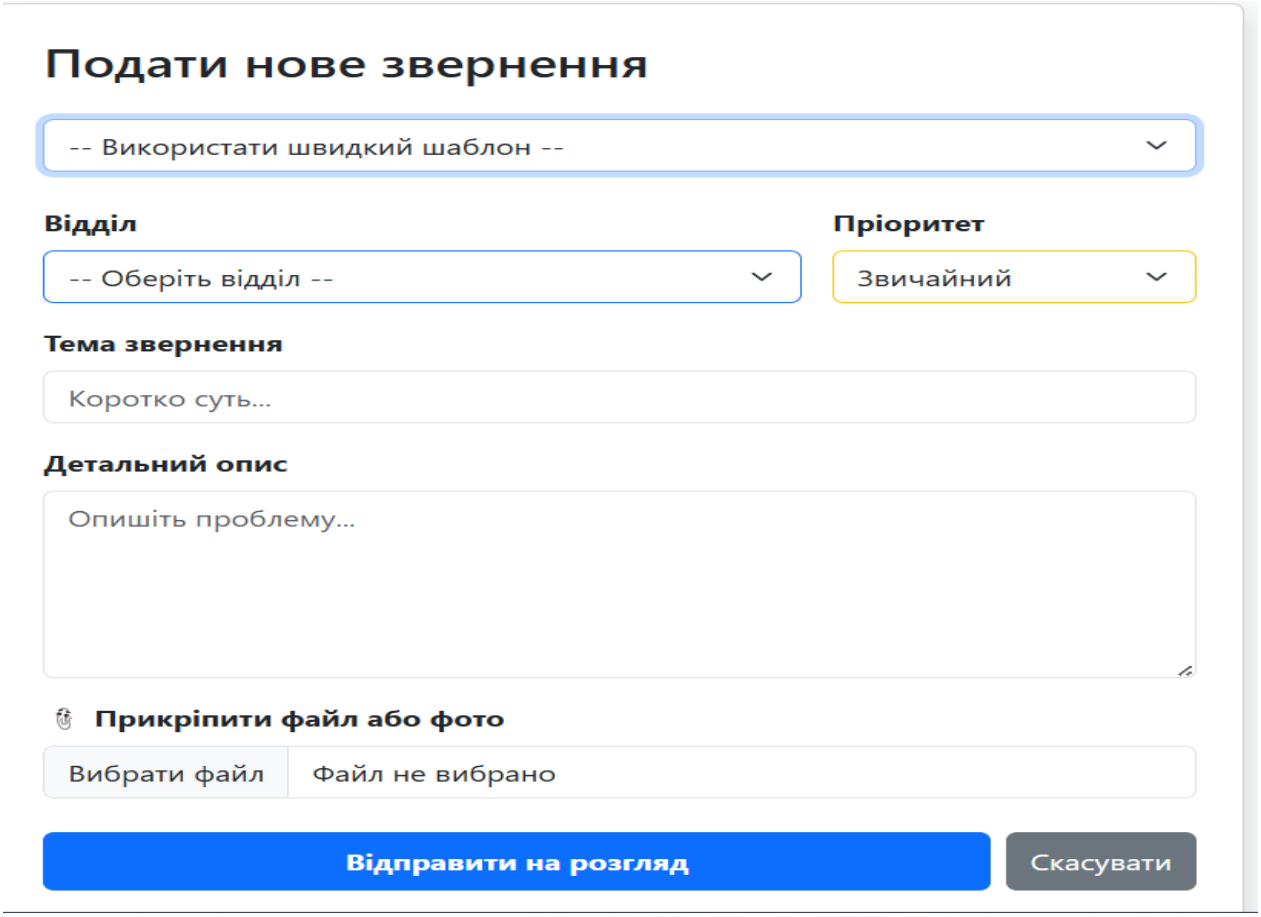


Рисунок 3.4. Створення нового звернення

Для реалізації внутрішнього чату використовуються файл «ticket.php». Логіка будується на записах з таблиці «ticket_messages» і подальшому їх розділенні. Використовуючи класи CSS повідомлення користувача виводяться праворуч, а відповіді ліворуч, що робить інтерфейс системи тікетів подібним до звичних месенджерів.

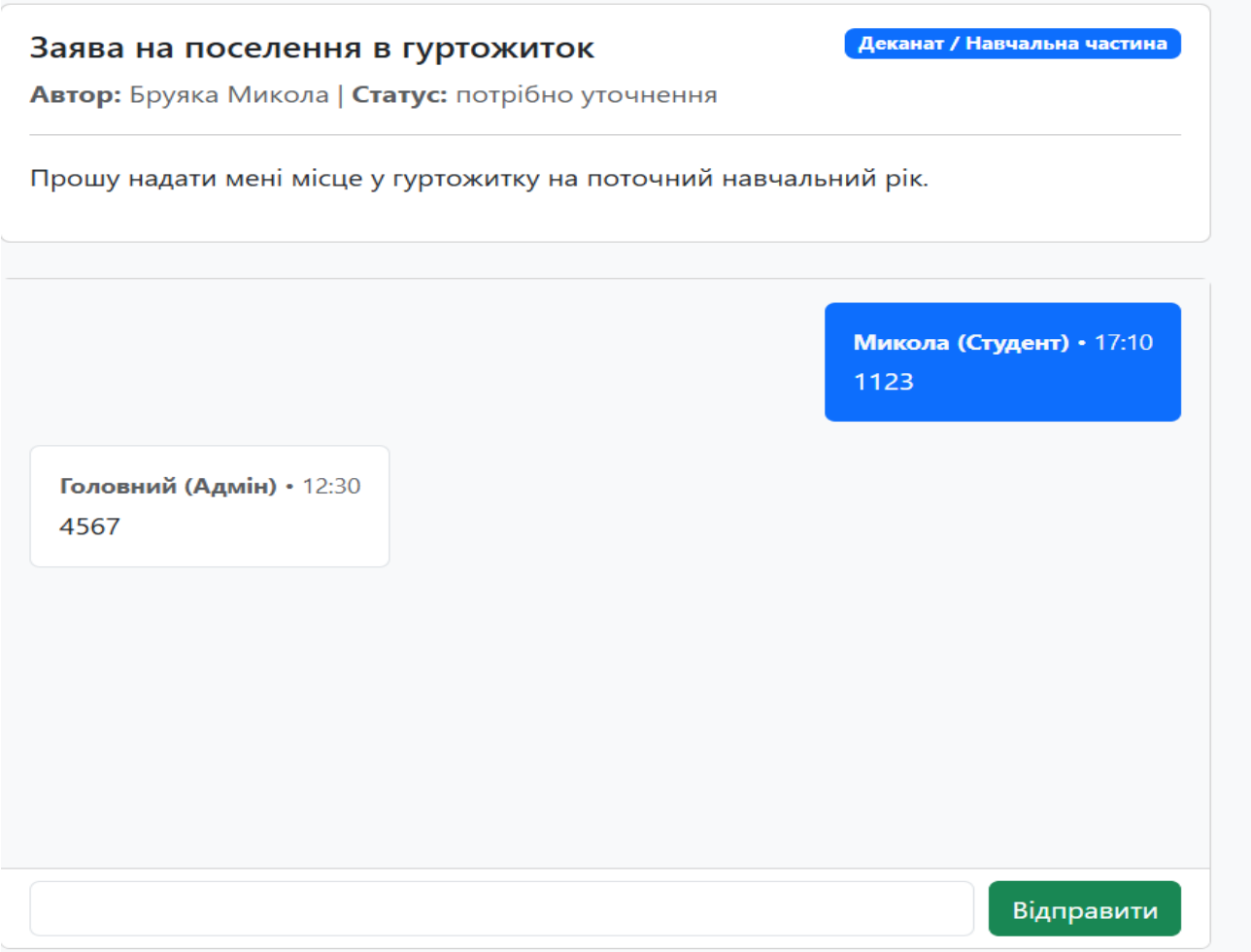


Рисунок 3.5. Інтерфейс системи тікетів

3.4 Реалізація адміністративної панелі

Панель обробки заявок «requests.php» дає адміністраторам оновлювати статус звернень. Технічна частина цього процесу реалізується через приховані поля (request_id та status). Скрипт виконує запит «UPDATE requests SET status = ? WHERE id = ?», і паралельно створює запис у таблицю аудиту

«request_processing», що дає змогу користувачу бачити коментарі та час, коли змінився статус.

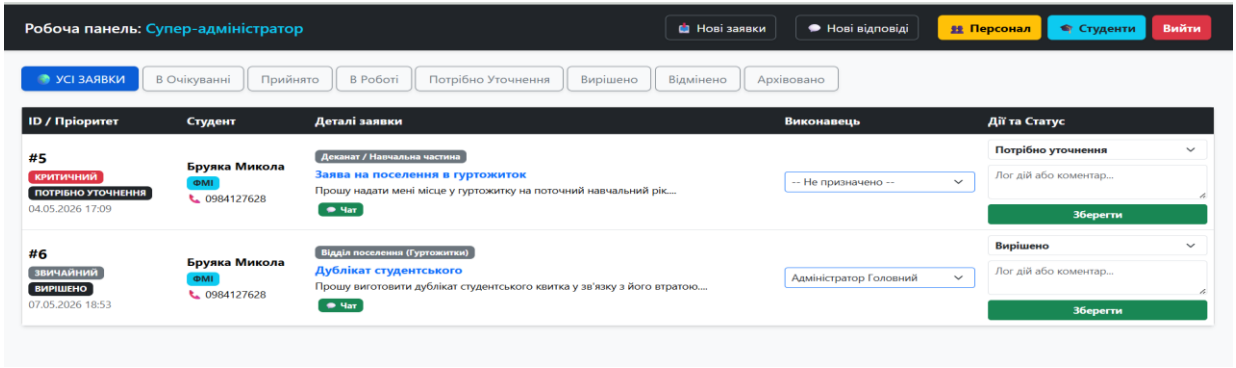


Рисунок 3.6. Інтерфейс адмін-панелі

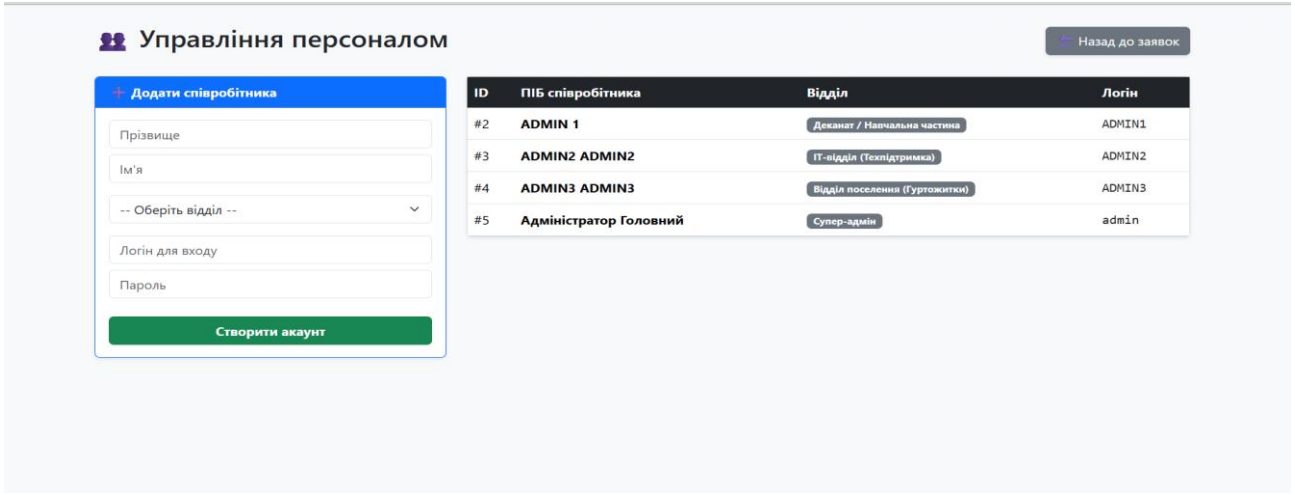


Рисунок 3.7. Інтерфейс управління персоналом

3.5 Забезпечення коректності та стабільності роботи системи

Для забезпечення цілісності робочого процесу в систему впроваджено скрипти самоочищення. На початку файлу «requests.php» виконується фоновий SQL-запит: «UPDATE requests SET status = 'архівовано' WHERE status IN ('вирішено', 'відмінено') AND date_submitted < DATE_SUB(NOW(), INTERVAL 5 DAY);» Він гарантує, що неактуальні звернення автоматично зникають з головного робочого екрана методиста, запобігаючи перевантаженню інтерфейсу.

3.6 Забезпечення інформаційної безпеки та захист персональних даних

Оскільки розроблена система отримує критичні конфіденційні дані користувачів (прізвище, ім’я, номер телефону), важливою частиною проєкту є інформаційна безпека [11].

1. Захист від SQL-ін'єкцій. У системі захист від ін’єкцій проводиться 2 кроками [15]. Перший – всі текстові дані, які проходять крізь $_POST та $_GET, перевіряються через процедуру екранування спеціальних символів у методі «real_escape_string()». Другий – у важливих вузлах мережі використовуються підготовлені запити. Механізм bind_param ("iis", $req_id, $uid, $message) гарантує, що база даних отримуватиме дані користувача тільки як тест, а не як команду для виконання [17].
2. Безпека завантаження файлів. У проєкті є механізм прикріплення студентами файлів у заявках. Це створює потенційну небезпеку. Для зменшення ризику завантаження PHP-скрипту (Web Shell) під виглядом іншого файлу, було реалізовано алгоритм перейменування, кожен новий файл отримує випадкову назву-рядок завдяки функції «uniqid()». І ці файли зберігаються в окремій директорії «uploads/».

3.7 Модель життєвого циклу звернення

У розробленій системі заявка виступає об’єктом, на якому базується алгоритм зміни станів. Для запобігання логічних помилок, було створено бізнес-логіку системи на основі «Finite State Machine».

У логіці даного проєкту модель може перебувати лише в одному стані, перехід між станами відбувається виключно за тригерами.

Життєвий цикл складається з шести станів:

1. Стан «Нове»

- Умова переходу. Студент заповнив форму (create_request.php), і відправив її. Система призначила заявку на виконавця (assigned_staff_id).
- Характеристика. Заявка перебуває в активній черзі. Студент вже немає змоги змінювати її, але може переглянути.

2. Стан «В роботі»
    - Умова переходу. 1.Виконавець взяв заявку в роботу (requests.php), і вибрав підходящий статус. 2.Студент дав відповідь на уточнення.
    - Характеристика. Студент отримує сповіщення про зміну статусу. Заявка блокується для автоматичного перепризначення.
3. Стан «Потрібно уточнення»
    - Умова переходу. Виконавець не має змоги вирішити проблему, через недостатню кількість інформації або через її низьку якість. Він змінює статус і обов’язково додає коментар або пише в чат.
    - Характеристика. Обробка заморожується, до моменту відповіді студента.
4. Стан «Вирішено»
    - Умова переходу. Запит успішно виконано. Виконавець успішно зафіксував це у адмін-панелі.
    - Характеристика. Заявка вважається успішно виконаною, обробка зупиняється. Студент отримує фінальне сповіщення.
5. Стан «Відмінено»
    - Умова переходу. Заявка вважається не прийнятою.
    - Характеристика. Заявка закривається без виконання. У таблицю журналу (request_processing) обов’язково записується причина відмови.
6. Стан «Архівовано»

- Умова переходу. Скрипт у системі проводить аналіз бази даних і переміщає зі статусами «вирішено» або «відмінено», дата яких більше 5 днів.
- Характеристика. Заявка зникає з екранів всіх користувачів, залишається лише в історії логування.

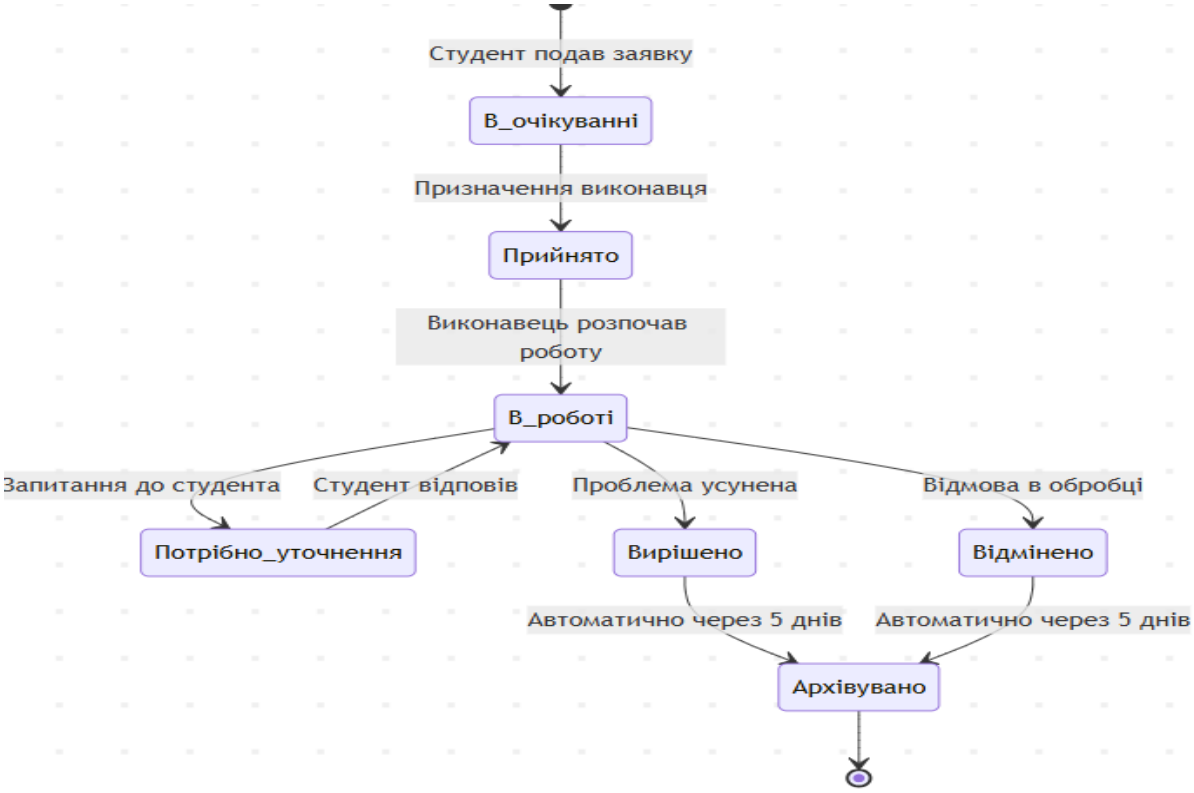


Рисунок 3.8. Діаграма станів

Для підтримки стабільності системи створені обмеження:

- неможливо перемістити напряму зі стану «нове» в «архівовано»;
- неможливо перемістити зі стану «архівовано» в стан «в роботі»;
- при переміщенні у стан «потрібно уточнення» обов’язково потрібен текстовий тригер у таблиці «ticket_messages».

РОЗДІЛ 4
ТЕСТУВАННЯ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

4.1 Методика та стратегія тестування програмного забезпечення

Мета тестування – перевірка фінального програмного продукту початковим функціональним вимогам, пошук багів, та випробування на стабільність.

Для імітації реальної поведінки студентів була обрана стратегія «чорного ящика». Тестувальник використовує лише зовнішній користувацький інтерфейс, при цьому не має доступу до внутрішнього коду.

Процес поділився на 3 рівні.

1. Функціональне тестування. Перевірка коректності основних бізнес процесів.
2. Тестування безпеки. Перевірка аутентифікації, стійкість до SQL-ін'єкцій та захист від несанкціонованого доступу.
3. Тестування інтерфейсу. Перевірка адаптивності на різних пристроях.

Тестування проводилось на локальному серверному середовищі «XAMPP»

4.2 Розробка та виконання тестових сценаріїв

Для кращого тестування було створено набір тестових сценаріїв

Таблиця 4.1

Протокол тестування модуля автентифікації та контролю доступ

№	Опис тестового сценарію (Дія)	Очікуваний результат	Фактичний результат	Статус
1	Введення коректного логіна та пароля студента у форму login.php.	Успішна авторизація, редирект на dashboard.php, створення сесії.	Редирект виконано коректно, дані студента завантажено.	Успішно
2	Введення невірного пароля.	Відмова в доступі, виведення повідомлення "Невірний пароль!".	Виведено червоний alert, доступ заблоковано.	Успішно
3	Спроба доступу до adminstudents.php з акаунта студента (через URL).	Система блокує доступ і перенаправляє на сторінку login.php або видає помилку доступу.	Виконано примусовий редирект (спрацювала перевірка ролі).	Успішно
4	SQL-ін'єкція у поле логіна (введення admin' OR '1'='1).	Відхилення запиту завдяки функції real_escape_string()..	Спроба входу відхилена, помилка авторизації.	Успішно

Таблиця 4.2

Протокол тестування модуля обробки заявок та авторозподілу

№	Опис тестового сценарію (Дія)	Очікуваний результат	Фактичний результат	Статус
1	Створення звернення без прикріплення файлу (лише текст).	Запис успішно додається в БД (таблиця requests), статус "нове".	Заявка з'явилася в кабінеті студента і в панелі адміна.	Успішно
2	Прикріплення файлу (наприклад, .pdf) до нової заяви.	Файл отримує унікальне ім'я (uniqid) і зберігається в папку uploads/.	Файл завантажено, посилання коректно збережено в БД.	Успішно
3	Перевірка авторозподілу (Load Balancing) на найменш завантаженого працівника.	Система автоматично призначає assigned_staff_id тому адміну, у якого найменше відкритих заяв.	Заява закріпилася за вільним працівником згідно з SQL-логікою.	Успішно
4	Зміна статусу заявки адміністратором на "Потрібно уточнення" та додавання коментаря.	Статус оновлюється, коментар записується в request_processing.	Студент миттєво побачив новий статус і коментар адміна.	Успішно

Таблиця 4.3

Протокол тестування системи тікетів (чату) та Telegram-бота

№	**Опис тестового сценарію (Дія)**	**Очікуваний результат**	**Фактичний результат**	**Статус**
1	Відправка повідомлення в чат заявки зі сторони студента.	Повідомлення зберігається в ticket_messages та відображається в інтерфейсі (вирівнювання праворуч).	Повідомлення збережено і коректно відрендерено..	Успішно
2	Синхронізація Telegram: відправка боту свого контакту (номера телефону).	Бот знаходить номер у таблиці students і записує telegram_id.	Бот знаходить номер у таблиці students і записує telegram_id.	Успішно
3	Запит "Мої заявки" у Telegram-боті.	Бот виводить список із 5 останніх заяв студента та їхні статуси.	Коректно сформовано список активних звернень з БД.	Успішно

4. 3 Оцінка організаційної ефективності впровадження системи

Оцінка ефективності розподіляється на 3 аспекти.

1. Оптимізація часу.

При використані паперової систем процес подачі заявок займає від 20 до 40 хвилин. Студенту треба самостійно прийти в деканат, отримати зразки, написати все вручну.

Завдяки готовим шаблонам і вебзастосунку, час який затрачається на подачу заявок, займає в середньому не більше 3 хвилин. Також плюсом виступає сповіщення як для виконавців, так і для студентів.

2. Розподілення навантаження.

Завдяки впровадженню алгоритму автоматичного роутингу заявок, людська робота зводиться до мінімуму. Програма сама знаходить «ідеального» виконавця для тої чи іншої задачі.

3. Комунікація

Вбудована система чату, дає можливість комунікувати в цифровому просторі. Якщо раніше це проводилось напряму, через додаткові джерела, то зараз це все можна написати прямо в заявці. Інтеграція з Телеграм ботом гарантує, що студент у будь-який момент зможе отримати повідомлення і відреагувати на нього.

4.4 Специфікація прецедентів та моделювання поведінки системи

Прецедент 1: Формування та подача нового студентського звернення.

Актор: Здобувач освіти (Студент).

Мета: Створити нове звернення до обраного структурного підрозділу ЗВО, прикріпити супровідні документи та ініціювати процес його обробки.

Передумови: Користувач успішно пройшов процедуру автентифікації (login.php) та має активну сесію з роллю student.

Основний потік подій:

- Студент у своєму особистому кабінеті (dashboard.php) натискає кнопку «Створити нову заявку».
- Система генерує інтерфейс форми створення заявки, завантажуючи з бази даних (request_templates) список доступних шаблонів та список відділів (departments).
- Студент обирає необхідний шаблон (наприклад, «Довідка про навчання»).
- Система за допомогою JavaScript-обробника автоматично підставляє стандартний текст заяви у поле опису.
- Студент обирає цільовий відділ («Деканат») та рівень пріоритету заявки.
- (Опціонально) Студент завантажує скан-копію документа через поле input type="file".
- Студент ініціює відправку даних натисканням кнопки «Відправити на розгляд».
- Система приймає POST-запит. Якщо прикріплено файл, сервер генерує для нього унікальне ім'я (функція uniqid) і зберігає у директорію uploads/.
- Система виконує алгоритм авторозподілу (Load Balancing): SQL-запит аналізує таблицю staff і знаходить працівника обраного відділу з найменшою кількістю активних заявок.
- Система створює новий запис у таблиці requests, записуючи туди ID студента, ID обраного працівника (assigned_staff_id) та встановлює статус «нове».
- Система перенаправляє студента на головну сторінку кабінету та виводить повідомлення про успішне створення звернення.

  Альтернативні потоки (Винятки):

- А1. Помилка валідації форми: Якщо студент не заповнив обов'язкові поля або не обрав відділ, система (на рівні HTML5-атрибута required) блокує відправку і підсвічує пропущені поля.
- А2. Помилка завантаження файлу: Якщо розмір файлу перевищує ліміти сервера (upload_max_filesize), система перериває запит і виводить повідомлення «Помилка: файл занадто великий».

Постумови: Заява збережена в базі даних. Вона з'являється в кабінеті студента та в робочій панелі призначеного працівника деканату.

Прецедент 2: Обробка звернення та взаємодія через систему тікетів.

Актори: Працівник підрозділу (Адміністратор), Здобувач освіти (Студент).

Мета: Розглянути подану заяву, змінити її статус та, за необхідності, отримати додаткову інформацію від студента через внутрішній чат.

Передумови: У системі існує активна заява. Адміністратор та студент авторизовані у своїх кабінетах.

Основний потік подій:

- Адміністратор відкриває робочу панель (requests.php) і бачить список заяв, призначених особисто йому.
- Адміністратор натискає на кнопку «Відкрити чат за заявкою» для детального ознайомлення (ticket.php).
- Система формує інтерфейс тікету: виводить текст заяви, посилання на завантажений документ та панель внутрішнього чату (вибірка з таблиці ticket_messages).
- Адміністратор виявляє, що в заяві бракує даних (наприклад, нечітке фото документа).
- Адміністратор у текстовому полі чату пише повідомлення: «Будь ласка, завантажте якісніше фото паспорта» і натискає «Відправити».
- Система зберігає повідомлення в таблиці ticket_messages із прив'язкою до поточного request_id.

- Адміністратор змінює статус заяви у випадаючому списку на «потрібно уточнення» і натискає «Зберегти».
- Система фіксує зміну статусу та додає запис у лог-журнал request_processing.
- Студент під час оновлення свого кабінету бачить новий статус заявки та повідомлення від адміністратора.
- Студент відповідає у чаті заявки та прикріплює посилання на новий документ.
- Адміністратор, отримавши потрібні дані, змінює статус заявки на «вирішено».

Постумови: Заявка набуває фінального статусу. Цикл обробки документа вважається завершеним. Після закінчення 5-денного терміну система автоматично переведе цю заявку в статус «архівовано».

Прецедент 3: Синхронізація профілю користувача з Telegram-ботом

Актор: Здобувач освіти (Студент), Telegram API.

Мета: Налаштувати інтеграцію між вебзастосунком та месенджером Telegram для швидкого доступу до статусу заявок.

Передумови: Студент має зареєстрований профіль на сайті з коректно вказаним номером телефону.

Основний потік подій:

- Студент знаходить Telegram-бота системи за посиланням або нікнеймом і натискає кнопку «/start».
- Сервер системи (bot_sync.php) отримує вхідний запит через Telegram Bot API (метод getUpdates).
- Бот надсилає користувачу вітальне повідомлення та генерує клавіатуру з кнопкою «Поділитися контактом» (request_contact => true).
- Студент натискає кнопку і підтверджує відправку свого номера телефону месенджером.

- Сервер отримує масив JSON із даними користувача (зокрема його номер телефону та унікальний chat_id).
- PHP-скрипт виконує підготовлений запит SELECT до таблиці students, шукаючи збіг за отриманим номером телефону.
- При знаходженні збігу система виконує запит UPDATE, записуючи отриманий chat_id у поле telegram_id знайденого студента.
- Система (через метод sendMessage) відправляє в Telegram повідомлення: «Профіль успішно синхронізовано!» та відображає головне меню бота.
- Студент натискає кнопку «Мої заявки».
- Система ідентифікує студента за його telegram_id, витягує з БД 5 останніх його заявок та надсилає їх у чат у вигляді форматованого тексту.

Альтернативні потоки (Винятки):

А1. Номер не знайдено: Якщо номер телефону, відправлений з Telegram, не збігається з жодним номером у базі students (наприклад, студент ще не зареєструвався на сайті або вказав інший номер), система відхиляє синхронізацію та надсилає повідомлення: «Помилка: ваш номер не знайдено в базі даних університету. Зареєструйтесь на сайті».

Постумови: Акаунт у Telegram безпечно та надійно пов'язаний з вебпрофілем студента. Користувач може отримувати актуальні дані без необхідності вебавторизації.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було успішно спроєктовано, розроблено та впроваджено автоматизовану інформаційну систему для подачі, обробки та моніторингу студентських звернень у закладах вищої освіти. Розроблений продукт повністю відповідає поставленим меті та завданням, а також було сформовано загальну структуру системи.

Проведено системний аналіз предметної області та вже існуючих рішень. Досліджено, що використання паперової системи звернень призводить до значних витрат часу, ризику втрати документів та нерівномірного розподілу навантаження. Доведено, що використання універсальних рішень є недоцільним через їх складність та відсутність спеціалізованого функціоналу для студентів. Це обґрунтувало створення власного вебзастосунку.

Спроєктовано найкращу архітектуру та модель бази даних. На основі зібраних функціональних вимог сформовано багаторівневу клієнт-серверну архітектуру. Спроєктовано реляційну базу даних під управлінням СКБД MySQL (MariaDB), яка містить взаємопов'язані таблиці. Завдяки нормалізації бази даних до третьої нормальної форми (3NF) та використанню зовнішніх ключів (Foreign Keys) з каскадним оновленням і видаленням (ON DELETE CASCADE),гарантовано абсолютну цілісність даних та усунено проблему їх дублювання. Логіку переходів статусів заявки спроєктовано на базі математичної моделі скінченного автомату з шістьма термінальними та транзитними станами.

Обґрунтовано вибір технологічного стеку та реалізовано серверну логіку.

Для реалізації бекенду обрано мову програмування PHP (версії 8.x) завдяки її нативній інтеграції з вебсерверами та широкому інструментарію. Розроблено модульну серверну частину, яка виконує обробку запитів,

маршрутизацію та взаємодію з базою даних через об'єктно-орієнтований інтерфейс розширення MySQLi. Для генерації клієнтського інтерфейсу (фронтенду) застосовано фреймворк Bootstrap 5, що дозволило реалізувати стратегію «Mobile-First» та забезпечити коректне відображення системи на пристроях із будь-якою роздільною здатністю екрана:

- механізм динамічних шаблонів, який дозволяє студентам генерувати типові заяви в один клік;
- інтелектуальний алгоритм балансування навантаження (Load Balancing), який за допомогою складних SQL-запитів автоматично призначає нову заяву на найменш навантаженого працівника відповідного відділу;
- модуль внутрішньої тікет-системи, що переводить комунікацію щодо проблемних заяв у формат зручного цифрового чату;
- підсистему автоматичного архівування застарілих записів для підтримки швидкодії БД.

Успішно впроваджено модуль кросплатформної взаємодії через Telegram Bot API.

Задля покращення якості користувацького досвіду (UX) та вирішення проблеми ігнорування E-mail сповіщень, систему інтегровано з популярним месенджером Telegram. Написано PHP-скрипт, який використовує технологію Long Polling (метод getUpdates) для обміну JSON-пакетами із серверами Telegram. Реалізовано алгоритм безпечної ідентифікації студента за номером телефону з подальшим записом його telegram_id до бази даних. Це дозволило користувачам миттєво отримувати інформацію про стан своїх запитів прямо у смартфоні.

Практичне значення одержаних результатів полягає в тому, що розроблена автоматизована система є повністю функціональним, надійним та готовим до масштабування продуктом. Її впровадження у діяльність закладу вищої освіти дозволить перейти до концепції без паперового офісу (мінімізувати бюрократичні затримки, підвищити прозорість взаємодії між

студентами та адміністрацією, а також суттєво оптимізувати робочий час працівників деканатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Соммервілл І. Інженерія програмного забезпечення / Іан Соммервілл ; пер. з англ. – 10-те вид. – Київ : Видавництво «Діалектика», 2018. – 816 с.
2. Ніксон Р. Створюємо динамічні веб-сайти з допомогою PHP, MySQL, JavaScript, CSS і HTML5 / Робін Ніксон ; пер. з англ. – 6-те вид. – Київ : Форс Україна, 2022. – 832 с.
3. Локвуд К. Проєктування програмного забезпечення та архітектура інформаційних систем. – Харків : Фоліо, 2020. – 340 с.
4. Коннолліт., Бегг К. Бази даних: проєктування, реалізація та супровід. Теорія і практика. – Київ : ВД «Вільямс», 2017. – 1440 с.
5. PHP: Hypertext Preprocessor. Офіційна документація мови програмування PHP [Електронний ресурс]. — Режим доступу: https://www.php.net/manual/ru/ (дата звернення: 10.05.2026).
6. MySQL 8.0 Reference Manual. Офіційна документація СУБД MySQL [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 12.05.2026).
7. Bootstrap 5 Documentation. The most popular HTML, CSS, and JS library in the world [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 14.05.2026).
8. Telegram Bot API. Офіційна документація для розробників Telegram [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 15.05.2026).
9. Chart.js Documentation. Simple yet flexible JavaScript charting for designers & developers [Електронний ресурс]. – Режим доступу: https://www.chartjs.org/docs/latest/ (дата звернення: 16.05.2026).

10. Грицюк Ю. І. Основи проєктування інформаційних систем : навчальний посібник / Ю. І. Грицюк. – Львів : Львівський державний університет безпеки життєдіяльності, 2019. – 404 с.
11. Захист персональних даних: Закон України від 01.06.2010 № 2297-VI. База даних «Законодавство України» / Верховна Рада України. URL: https://zakon.rada.gov.ua/laws/show/2297-17 (дата звернення: 10.05.2026).
12. Пасічник В. В., Резніченко В. А. Організація баз даних та знань : підручник. – Київ : Видавнича група BHV, 2018. – 384 с.
13. ДСТУ ISO/IEC/IEEE 29148:2018. Інженерія систем і програмного забезпечення. Процеси життєвого циклу. Вимоги до інженерії (ISO/IEC/IEEE 29148:2018, IDT). – Київ : ДП «УкрНДНЦ», 2019. – 94 с.
14. Фляйшман А. Основи веб-дизайну та розробки користувацьких інтерфейсів (UI/UX). – Київ : КНТУ, 2021. – 215 с.
15. Довідник з веб-безпеки OWASP Top 10 [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/ (дата звернення: 05.05.2026).
16. Сурмін Ю. П. Аналіз і проєктування систем документообігу в державних установах та закладах освіти. – Київ : Академія, 2018. – 190 с.
17. Даніель Г., Швенг К. Розробка безпечних веб-додатків. Відкриті технології та методи захисту. – Харків : Ранок, 2020. – 256 с.
18. XAMPP Apache + MariaDB + PHP + Perl [Електронний ресурс]. – Режим доступу: https://www.apachefriends.org/ (дата звернення: 08.05.2026).
19. Гільберт Д. Створення сучасних REST API на PHP. – Київ : Олді-плюс, 2021. – 310 с.
20. Мельникова Н. І. Інформаційні технології в управлінні навчальним процесом закладу вищої освіти // Вісник інноваційних технологій. – 2022. – № 4. – С. 45–52.

ДОДАТКИ