НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО Декан факультету	ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ Завідувач кафедри комп'ютерних наук
_____ Ігор БОЛБОТ	_____ Белла ГОЛУБ
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система обліку мешканців гуртожитку»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми

д.е.н., професор _____ Роман РУДЕНСЬКИЙ

(підпис)

Керівник бакалаврської кваліфікаційної роботи

ст. викладач _____ Віктор ПАНКРАТЬЄВ

(підпис)

Виконав _____ Марія КАРПОВА

(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
к.т.н., доцент _____ Белла ГОЛУБ
(підпис)
«_06_» _____січня_____ 2026 р.

ЗАВДАННЯ

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ЗДОБУВАЧУ

Карповій Марії Олександрівні

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Інформаційна система обліку мешканців гуртожитку»

затверджена наказом від «06» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи: результати дослідження літературних джерел.

Перелік питань, що підлягають дослідженню: системний аналіз сфери обліку поселення; інформаційне забезпечення для розробки;

Перелік графічних документів: діаграми, скріншоти прикладу сторінок.

Дата видачі завдання «06» січня 2026 р.

Керівник бакалаврської

кваліфікаційної роботи _____ Віктор ПАНКРАТЬЄВ

(підпис)

Завдання взяв до виконання _____ Марія КАРПОВА

(підпис)

ЗМІСТ

ВСТУП.....5
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....8
1.1 Постановка завдання.....8
1.2 Огляд інформаційних джерел та існуючих рішень.....11
1.3 Моделювання предметної області.....21
2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ.....23
2.1 Логічна модель даних.....23
2.2 Вибір системи управління інформаційною базою.....29
2.3 Створення інформаційної бази.....35
3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.....42
3.1 Організаційна структура програмного забезпечення.....42
3.2 Вибір інструментарію для створення ППЗ.....43
3.3. Алгоритмізація та програмування програмних модулів.....46
4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ.....53
4.1 Тестування системи.....53
4.2 Вимоги до апаратного та програмного забезпечення.....61
4.3 Склад інсталяційного пакету.....61
ВИСНОВКИ.....63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....65
Додаток А.....68
Додаток Б.....74
Додаток В.....80

Перелік умовних позначень

ІС – інформаційна система.

ІЗ – інформаційне забезпечення.

БД – база даних.

СУБД – система управління базою даних.

UML - Unified Modeling Language - уніфікована мова моделювання.

ВСТУП

Актуальність. Актуальність полягає у необхідності автоматизації процесів обліку мешканців гуртожитків закладів вищої освіти. Щороку до закладів вищої освіти вступають тисячі студентів з різних міст, містечок, сіл та селищ України, а також значна кількість студентів з усього світу. І зазвичай вони не мають постійного місця проживання на період навчання. Тому університети зазвичай забезпечують студентів місцями в гуртожитках. Також з кожним роком в Україну приїжджають студенти зі 127 країн світу. Набільша кількість іноземних студентів прибуває з Китаю, Азербайджану, Грузії, Індії, Марокко, Туреччини, Туркменістану, Польщі, Вірменії. Для прикладу кількість студентів першокурсників по роках іноземних та українських[1,2]:

- 2025 рік - 24 718/ 178 720;
- 2024 рік - 27 226/ 187 801;
- 2023 рік - 51 676/ 267 700.

Відповідно до цих факторів збільшується і навантаження не лише на приймальні комісії закладів вищої освіти, але й на адміністрацію гуртожитків, яка здійснює контроль поселення та виселення студентів, розподіл місця у кімнатах, ведення документації, контроль оплати, заяв, квитанцій та інших документів.

У багатьох гуртожитках значна частина таких процесів досі контролюється та виконується за допомогою паперової документації та електронних таблиць Excel. І це може призводити до проблем та помилок в ході обліку студентів гуртожитку, таких як: плутаність у паспортних даних, дублювання кімнат при поселенні або помилок при виселенні, довгого пошуку даних про студентів, також до збільшення часу формування звітності. У зв’язку з цим виникає необхідність для створення автоматизованої інформаційної системи обліку мешканців гуртожитку, яка дозволить

оптимізувати процеси управління, зменшити кількість помилок та підвищити ефективність роботи адміністрації гуртожитку.

Мета роботи. Метою бакалаврської кваліфікаційної роботи є розробка інформаційної системи обліку мешканців гуртожитку - щоб автоматизувати процеси поселення та виселення студентів, забезпечення контролю наявності вільних місць, ведення обліку мешканців, документів та оплати проживання, та формування статистичної та звітної інформації, мінімізувати помилки.

Методи та технології, які використовуються при розробці програмного додатку.

- Visual Studio Code - середовище розробки програмного забезпечення;
- MySQL - СУБД управління даними в базі даних;
- HTML - мова розмітки створення інтерфейсу користувача;
- CSS - мова стилізації інтерфейсу користувача;
- Python - мова програмування для реалізації серверної логіки програмного додатку;
- Java Script - мова програмування для реалізації динамічної взаємодії з елементами інтерфейсу користувача;
- Bootstrap - фреймворк для створення адаптивних інтерфейсів.

Апробація програмного додатку. У ході виконання бакалаврської роботи також було підготовлено тези для участі у VII ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ СТУДЕНТІВ ТА АСПІРАНТІВ «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП’ЮТЕРНИХ СИСТЕМ '2026». У секції «Теоретичні аспекти у галузі інформаційних технологій».

Структура записки.

1. Системний аналіз предметної області - детальний огляд предметної області, аналогів до розроблюваної системи та визначення вимог.
2. Огляд інформаційного забезпечення - розробка інформаційної бази та даних які будуть міститись у базі даних.

3. Огляд прикладного програмного забезпечення - опис структури прикладного програмного забезпечення представленої у вигляді діаграми пакетів та архітектури.
4. Рекомендації щодо впровадження - завершальний етап, в якому виконується системне тестування ІС, описуються вимоги до користування системою та опис інсталяційного пакету.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Завданням розробки інформаційної системи є автоматизація процесів поселення та виселення студентів гуртожитку, облік документів (заяв на поселення, квитанцій про оплату на проживання, документів для пільгових категорій), а також обліку інформації про студентів, які проживають у гуртожитку. Система також повинна забезпечувати доступ до інформації про заселеність кімнат, наявність вільних місць та поточний стан проживання мешканців.

Основними завданнями системи є:

- ведення обліку мешканців гуртожитку;
- зберігання персональних даних студентів та інформації про їх поселення;
- автоматизація процесів поселення та виселення;
- облік вільних та зайнятих місць у кімнатах;
- формування звітності щодо заселеності гуртожитку;
- надання можливості адміністрації редагувати дані мешканців;
- забезпечення авторизації користувачів (адміністратор, комендант, вахтер);
- облік оплат за проживання;
- збереження історії проживання мешканців;
    - відслідкування стану проживання мешканців;
    - формування звітних документів.

Основна інформація яка обробляється в системі

Інформаційна система забезпечує збереження та обробку інформації про мешканців гуртожитку, персональної інформації - імена, прізвища по-батькові, контактні данні, факультети.

Також система обробляє інформацію про кімнати гуртожитку - на скільки заповнені кімнати, інформацію про мешканців, які проживають у кімнаті, кількість вільних місць. Це дозволяє здійснювати контроль заселеності гуртожитку.

Для контролю процесу проживання система зберігає інформацію про поселення: дата поселення та виселення, закріплену кімнату та інформацію про студента.

Окремо система забезпечує облік фінансових операцій, зокрема інформації про оплату проживання: суми оплати для кожного студента, також період оплати та платежів, статус оплати, наявність заборгованості.

Система також підтримує електронний облік, включаючи назву документа, дату його додавання та електронну копію документа.

Для студентів спеціальної категорії надається інформація про вид пільги, відсоток знижки та автоматичний розрахунок вартості проживання з урахуванням пільги.

На основі оброблених даних система публікує статистичну інформацію та звіти про наповнюваність гуртожитків, кількість мешканців, вартість проживання та інші показники, необхідні для управління.

Критерії класифікації розроблюваної системи

Розроблювана інформаційна система обліку мешканців гуртожитку є автоматизованою ІС - дана система призначена для обробки, зберігання та аналізу даних, зокрема про студентів, оплату, поселення та кімнати.

За характером використання - оскільки система подає інформацію та обчислення даних, то належить до інформаційно-довідкових і облікових систем.

За масштабом - оскільки система призначена для використання в межах гуртожитку, то система є локальною.

За ступенем автоматизації - автоматизована система управління, оскільки дозволяє автоматизувати процес реєстрації мешканців, контроль платежів, управління поселенням та формування звітності.

За типом архітектури система розробляється як клієнт-серверна веб-орієнтована із використанням бази даних MySQL. Для взаємодії між користувачем та системою використовується мова програмування Python та вебтехнології HTML, CSS.

За способом обробки даних дана ІС підтримує централізоване зберігання інформації та багатокористувацький доступ, із розмежуванням прав доступу користувачів відповідно до їх ролей.

Формування вимог до інформаційної системи

Функціональні вимоги

У системі повинна бути можливість:

- авторизуватись у системі (адміністратор, комендант, вахтер);
- додавати нового мешканця;
- редагувати особисті дані мешканця (ПІБ, дата народження, факультет, група, номер телефону, email, паспортні дані);
- видаляти записи (при виселенні);
- здійснювати поселення мешканця у конкретну кімнату;
- вести облік кімнат (номер кімнати, поверх, кількість місць, статус - вільна/зайнята);
- переглядати список мешканців за кімнатами;
- здійснювати пошук мешканців за ПІБ, групою, факультетом або номером кімнати;
- контролювати оплату проживання (сума, дата оплати, заборгованість);
- формувати звіти (заселеність по поверхах, кількість вільних місць, боржники);
- зберігати історію проживання мешканця;
- здійснювати облік відсутності мешканців;
- ведення обліку та зберігання необхідних документів мешканців;
- експортувати звіти у форматах .pdf або .doc, .exel.

1.2 Огляд інформаційних джерел та існуючих рішень

Аналіз схожих існуючих рішень та інформаційних джерел дає змогу дослідити та проаналізувати підходи до організації інформаційних систем обліку мешканців гуртожитків. Також визначити основний функціонал подібних систем, оцінити переваги та недоліки. Також проведений аналіз

дозволяє сформувати основні вимоги до інформаційної системи яка розробляється, визначити функціонал, який бажано реалізувати.

У процесі аналізу було визначено основний функціонал:

- реєстрація/авторизація;
- облік мешканців гуртожитку;
- перегляд та контроль процесу поселення і виселення;
- контроль інформації про гуртожиток;
- ведення звітності.

Основними існуючими джерелами для аналізу стали - вебсайти університетів, мобільні застосунки, а також інформаційні системи для управління гуртожитками та студентським житлом. Було проаналізовані як системи націлені на діяльність студентських гуртожитків так і сервіси призначені для управління житловими приміщеннями загального призначення.

Під час огляду оцінювався не лише тільки функціонал існуючих рішень, але й зручність користування системою, зрозумілість навігації, організація інтерфейсу користувача, подання інформації, яка інформація висвітлювалась, швидкість доступу до інформації та загальний стиль оформлення.

У результаті проведеного аналізу виокремлено сучасні інформаційні системи та мобільні застосунки для управління гуртожитками та студентським житлом, функціональні можливості які можуть бути використані під час розробки інформаційної системи. Також проаналізовані переваги та недоліки існуючих рішень, що враховано для проєктування власної системи.

Інформаційні джерела та існуючі рішення

1. Студмістечко КПІ ім. Ігоря Сікорського [24] - даний сайт, приклад зображено на рис.1-4, надає основну інформацію для поселення студентів (необхідні документи, дати поселення, адреси гуртожитків), також перелік

гуртожитків для поселення, контакти, основну інформацію про гуртожитки та студмістечко у цілому. Недолік: дана система є більш інформативною та не призначена для обліку. Оскільки до систем гуртожитків окремих університетів для обліку немає відкритого доступу.

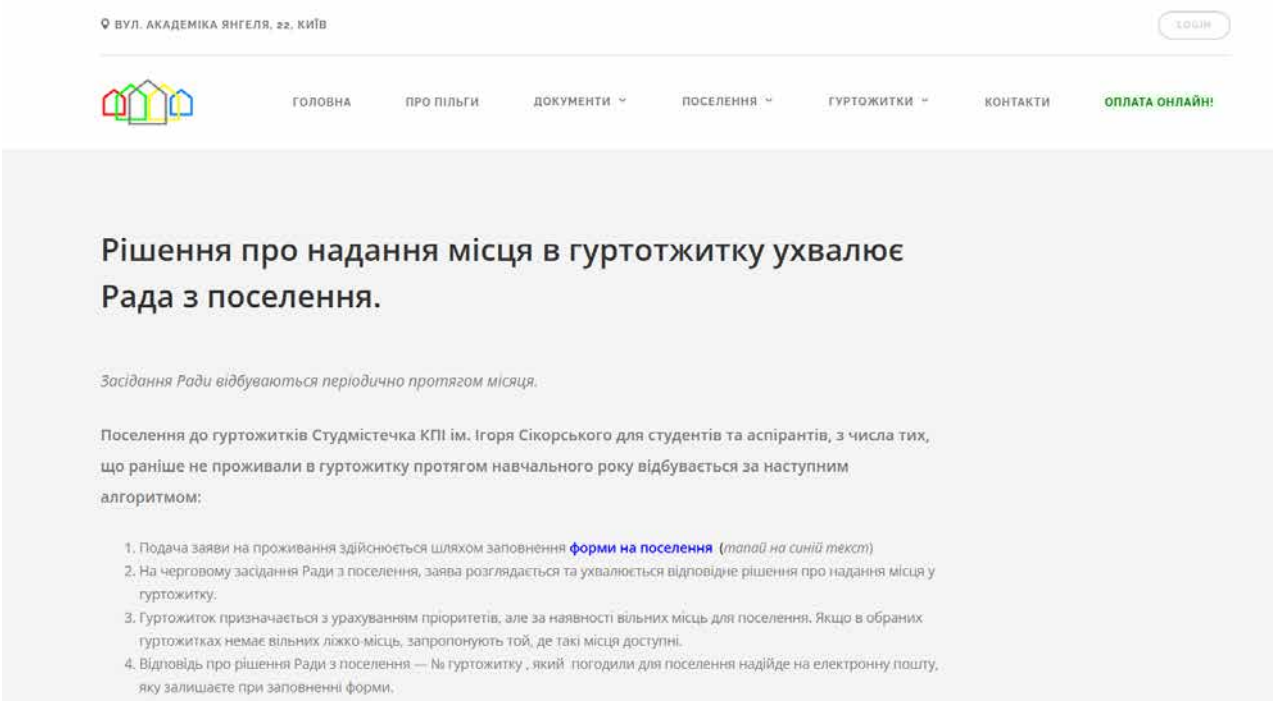


Рис.1 Головна сторінка

Рис.2 Перелік документів

Рис.3 Про студмістечко

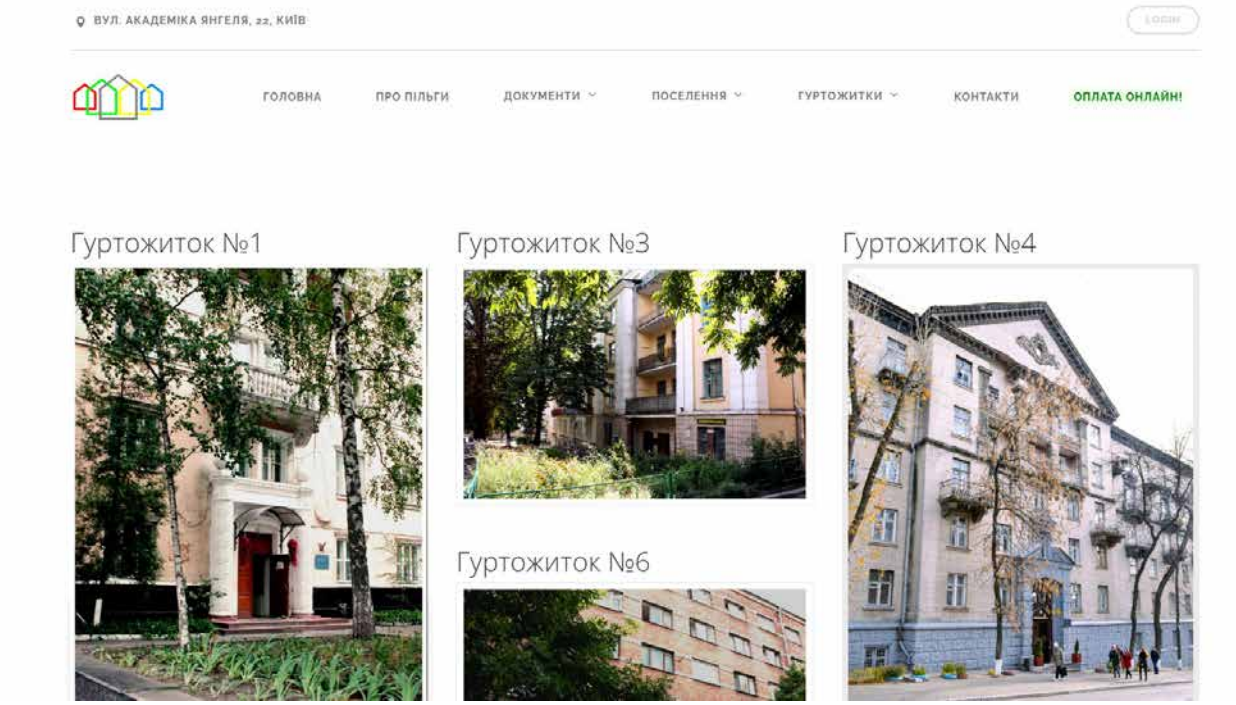


Рис.4 Перелік гуртожитків

2. Одеська міжнародна академія [23] - сайт університету, приклад зображено на рис.5-7, з інформацією для студентів: необхідні документи для поселення, пільгові категорії, умови проживання, плата для різних категорій. На жаль також немає доступу до самої системи поселення.

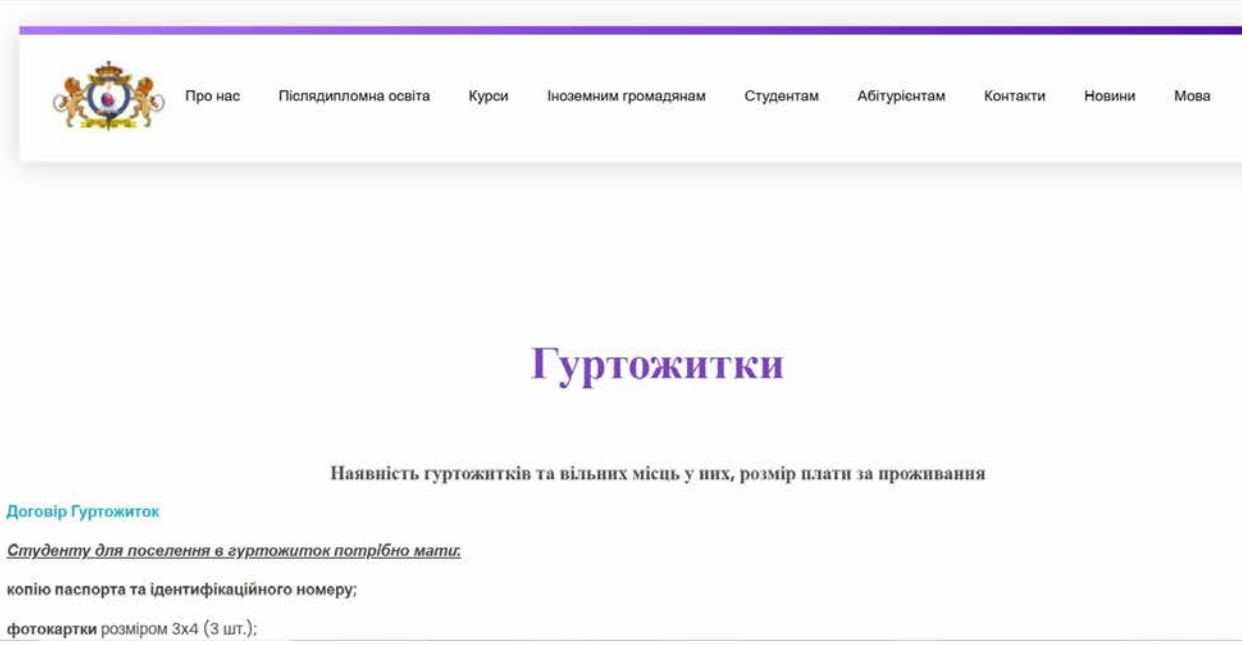


Рис.5 Головна сторінка про гуртожиток

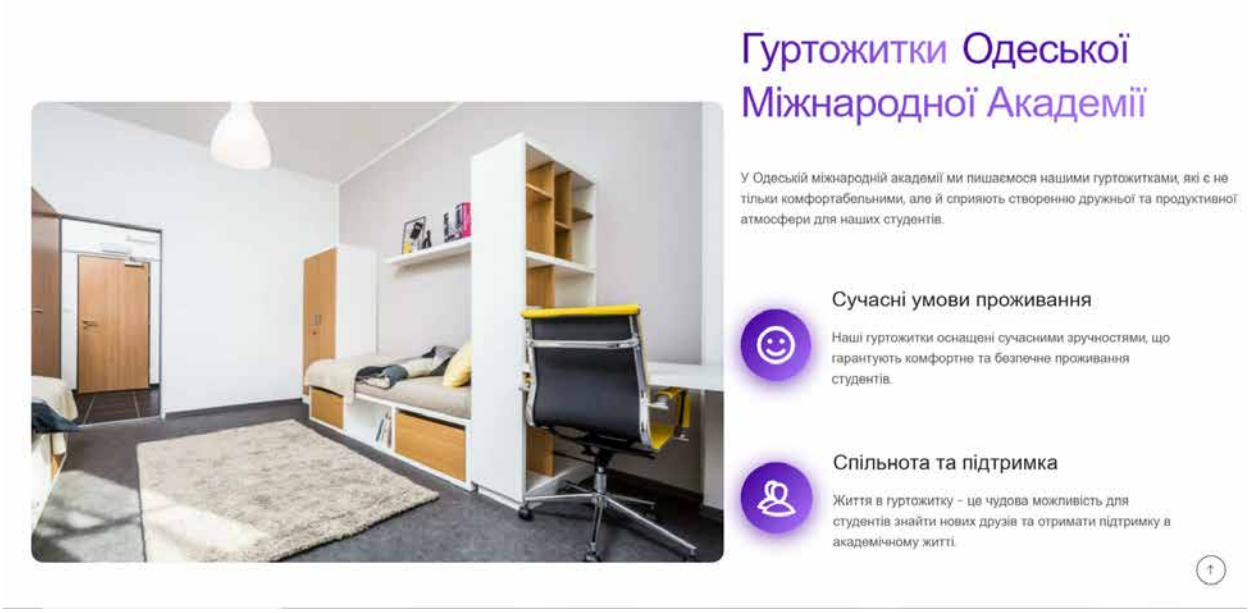
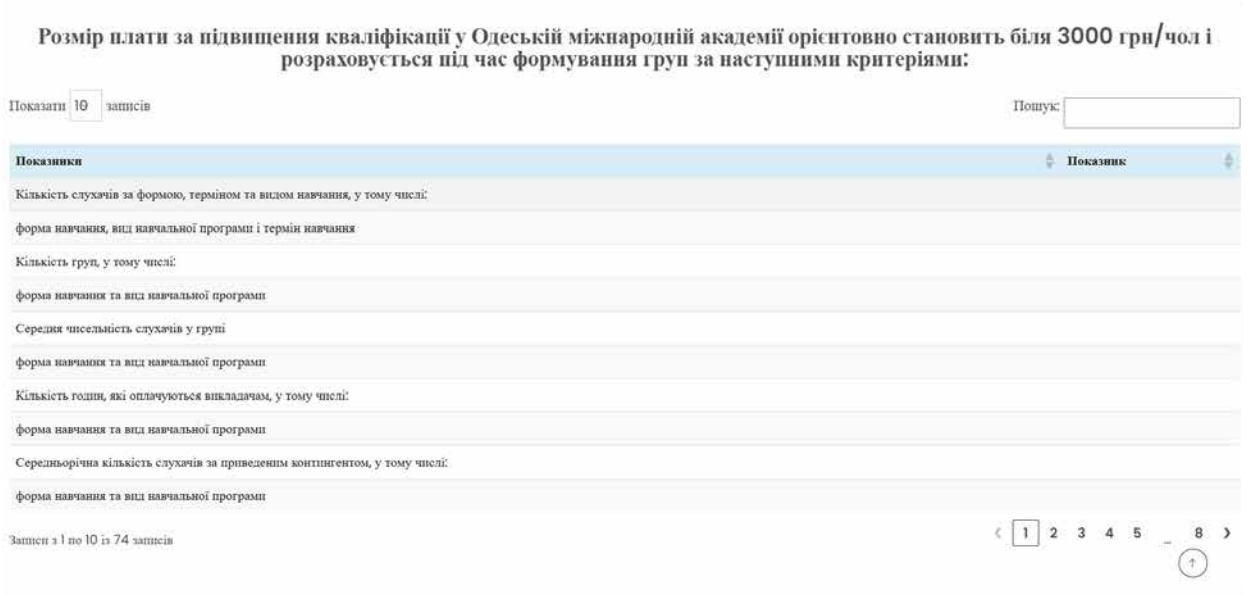


Рис.6 Інформація про гуртожиток



Розмір плати за підвищення кваліфікації у Одеській міжнародній академії орієнтовно становить біля 3000 грн/чол і розраховується під час формування груп за наступними критеріями:

Показати 10 записів Пошук:

Показники	Показник
Кількість слухачів за формою, терміном та видом навчання, у тому числі:	
форма навчання, вид навчальної програми і термін навчання	
Кількість груп, у тому числі:	
форма навчання та вид навчальної програми	
Середня чисельність слухачів у групі	
форма навчання та вид навчальної програми	
Кількість годин, які оплачуються викладачам, у тому числі:	
форма навчання та вид навчальної програми	
Середньорічна кількість слухачів за приведеним контингентом, у тому числі:	
форма навчання та вид навчальної програми	

Записи з 1 по 10 із 74 записів ‹ 1 2 3 4 5 … 8 ›

Рис.7 Приклад формування оплати за групами

3. StarRez [26] - інформаційна система для управління студентським житлом та гуртожитками, приклад зображено на рис. 8-10. Використовується у більш ніж 20 країнах світу. Забезпечує автоматизацію процесів поселення студентів, управління кімнатами та майном студентів. Також підтримкою студентів, та комунікацією між адміністрацією та студентами.

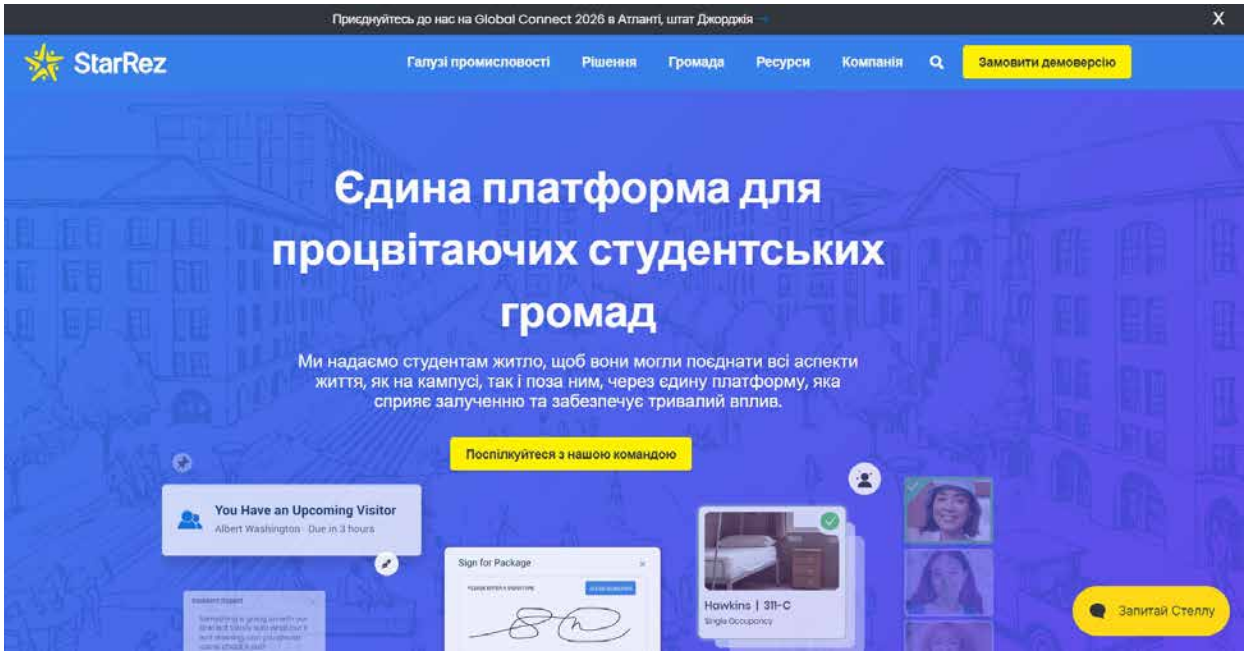


Рис.8 Головна сторінка

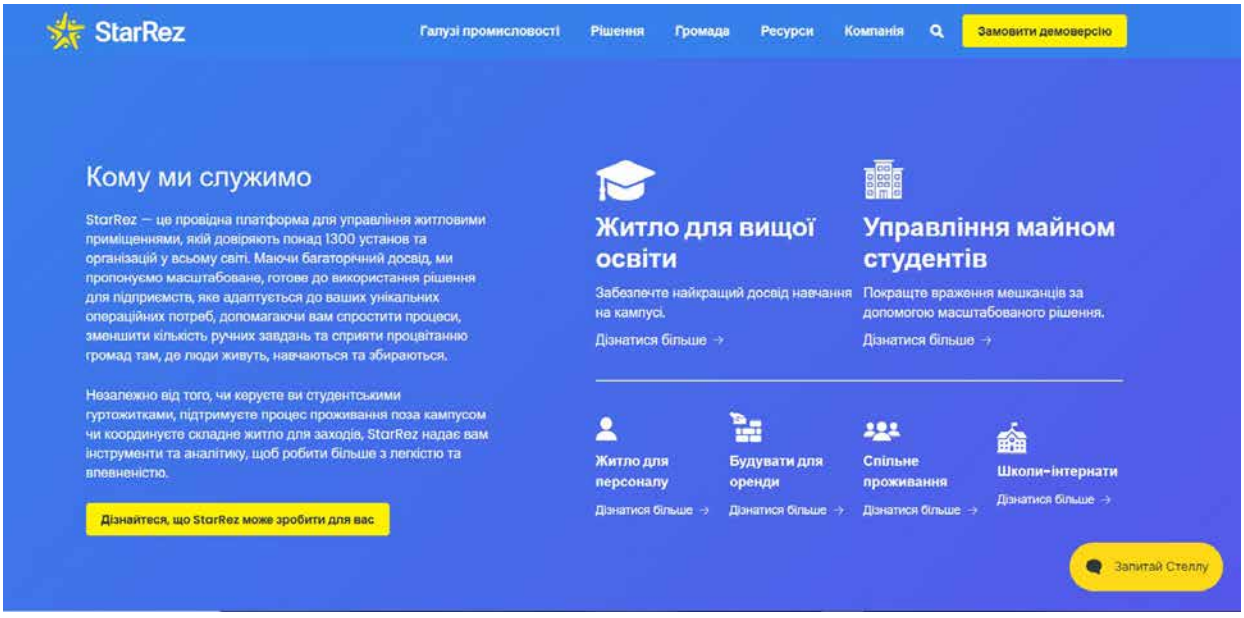


Рис.9 Сфера застосування

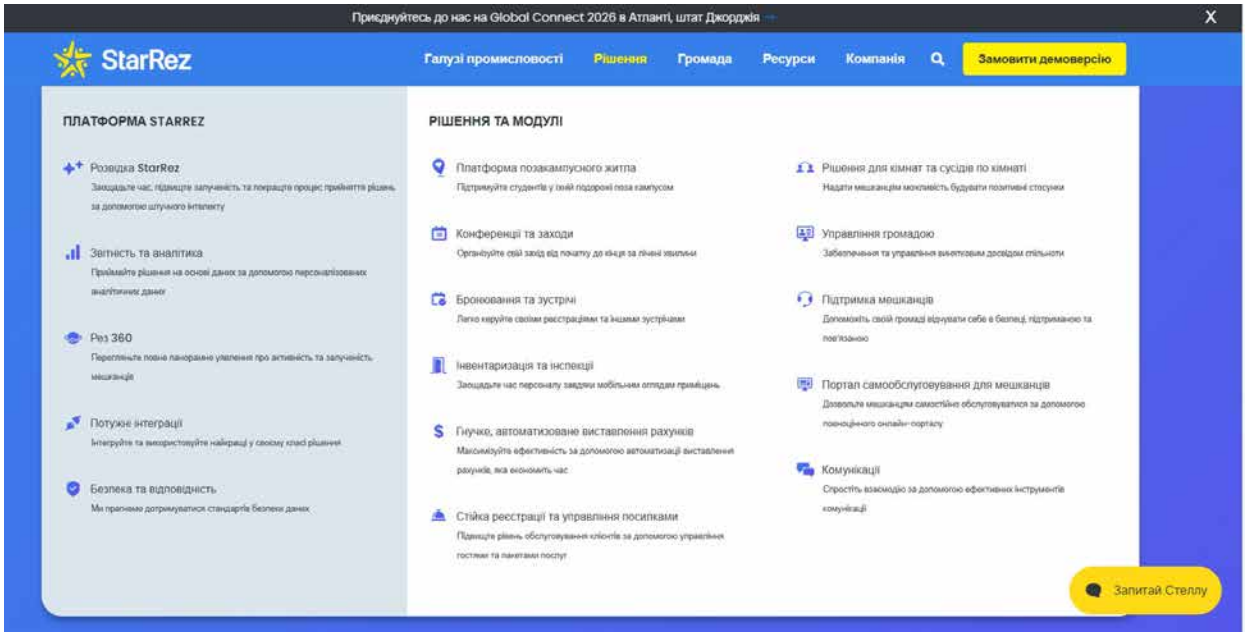


Рис.10 Приклад функціоналу системи StarRez.

4. HostelBuddy [25] - система для управління гуртожитками та контролю проживання мешканцями, приклад функціоналу зображено на рис.11-13. Відстеження платежів, управління кімнатами, управління персоналом. Основною метою є автоматизація основних процесів обліку та взаємодія між адміністрацією гуртожитку.

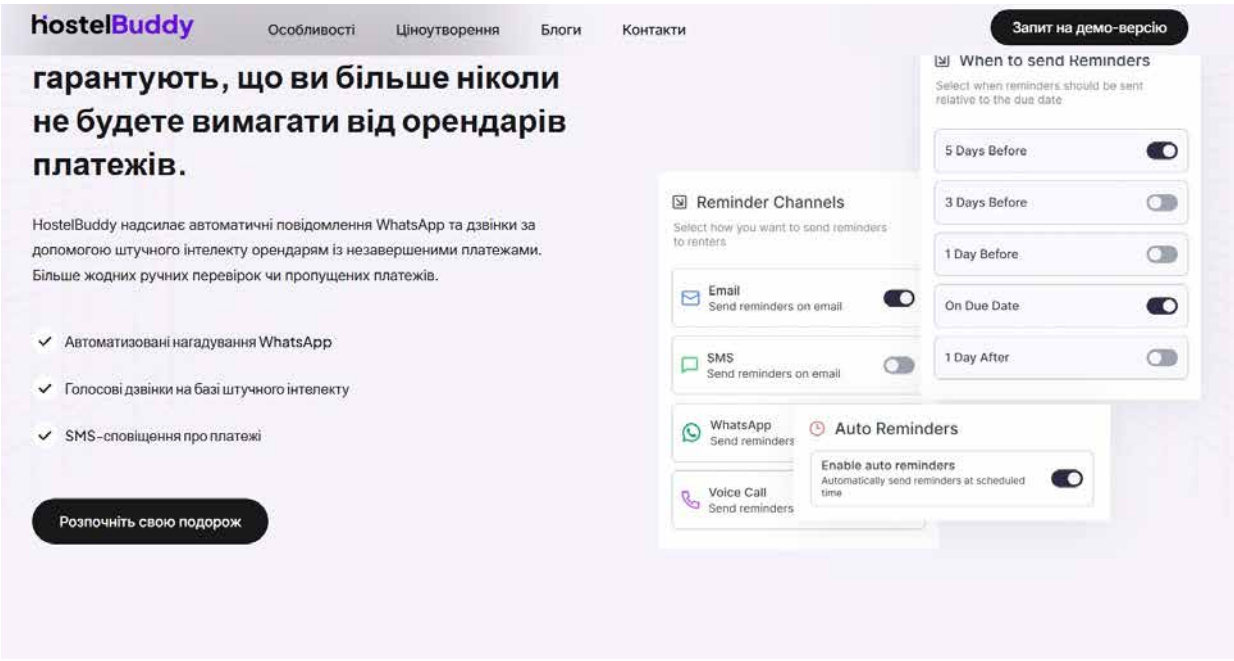


Рис.11 Головна сторінка

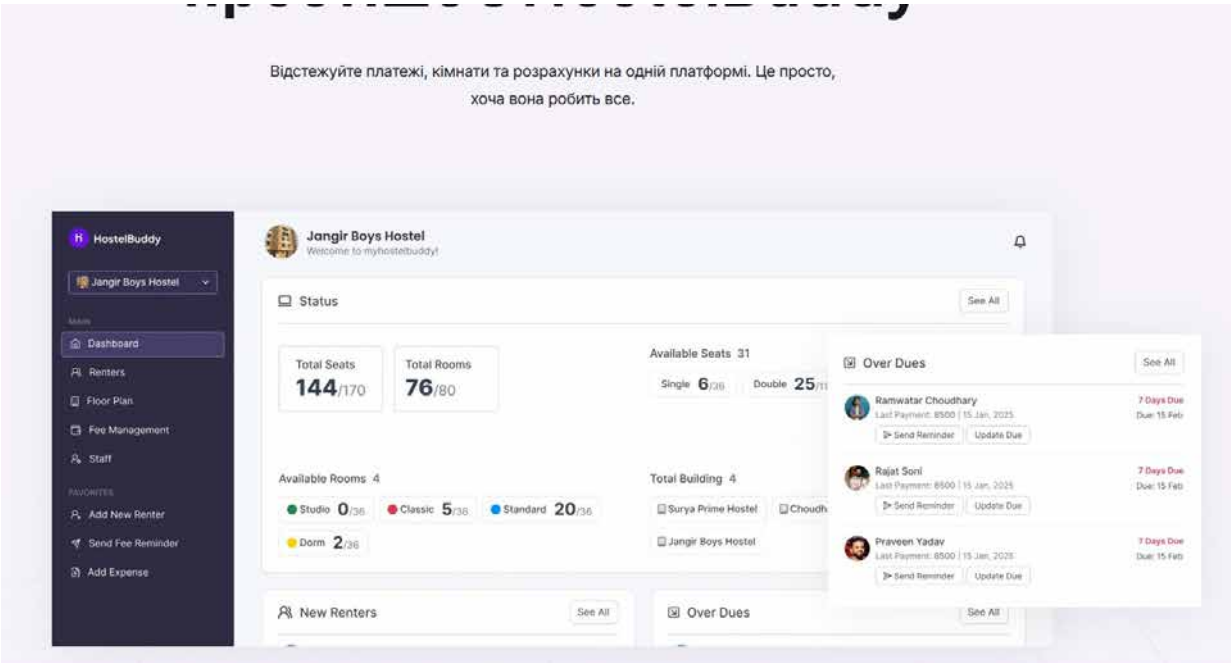


Рис.12 Приклад функціоналу

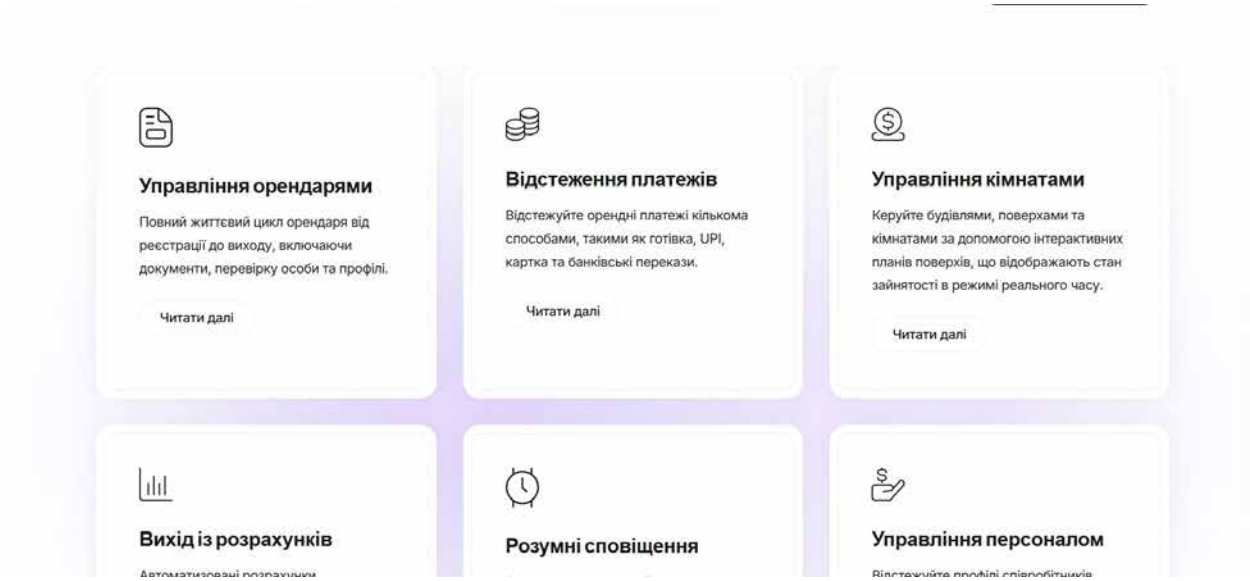


Рис.13 Основні функції системи

5. E-Hostel Management [27] - це цифрова система управління гуртожитком, призначене для спрощення роботи гуртожитку та підвищення прозорості для студентів, персоналу та адміністраторів. Приклад сторінок зображено на рис.14-15. Незалежно від того, чи це відстеження відвідуваності, планування тижневого меню, керування інвентарем або обмін оновленими стравами, ця програма спрощує керування гуртожитком, ніж будь-коли. Система спрощує адміністративні завдання гуртожитку та покращує взаємодію між студентами, персоналом та керівництвом.

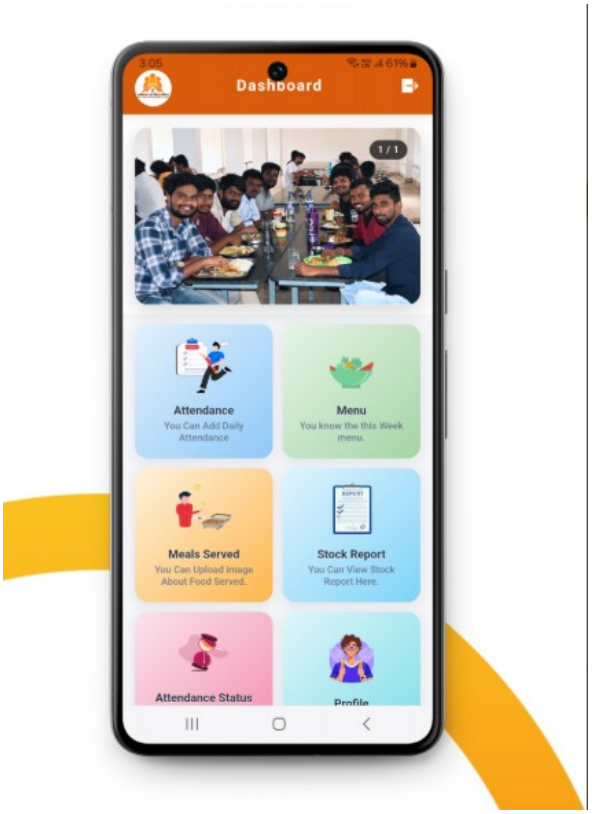


Рис.14 Приклад сторінки системи E-Hostel Management

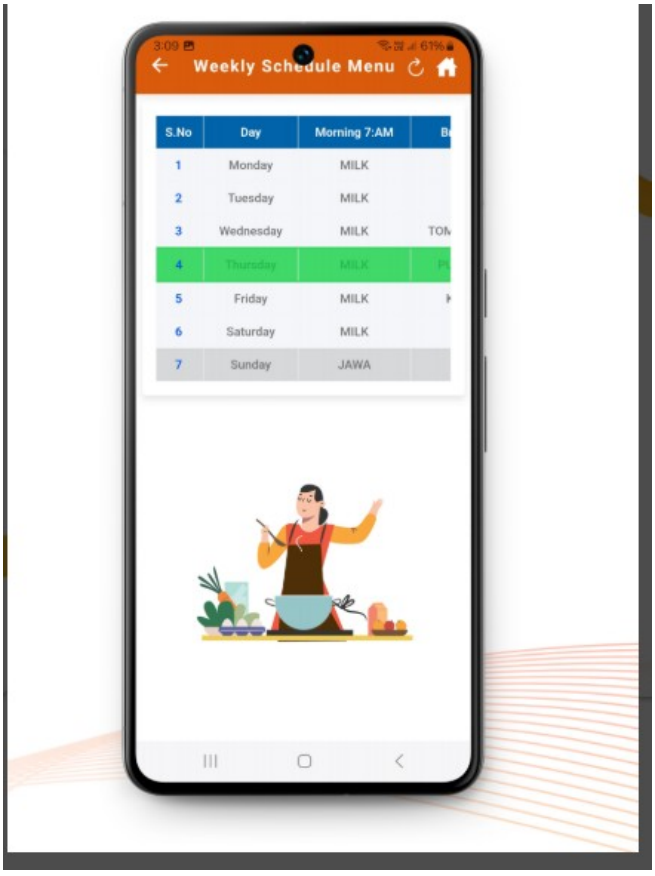


Рис.15 Приклад сторінки системи «Меню»

1.3 Моделювання предметної області

Предметна область - це сфера впливу, навколо якої будується або розроблюється інформаційна система. Моделювання предметної області необхідне для того, щоб зрозуміти її структуру, основні процеси, взаємодії, основні ролі.

Модель предметної області створюється для:

- більшого розуміння предметної області для розробників, що б виявити ключові особливості та обмеження сфери, у якій розробляється система;
- виокремлення та формалізації функціональних і нефункціональних вимог до системи;
- передачі знань про предметну область між усіма учасниками проєкту.

У даній роботі предметна область охоплює процеси управління гуртожитками та поселенням мешканців. Для аналізу основних функцій системи та її користувачів було побудовано діаграму прецедентів.

На діаграмі рис.16 відображено взаємодію двох основних акторів системи (адміністратор, комендант) із системою.

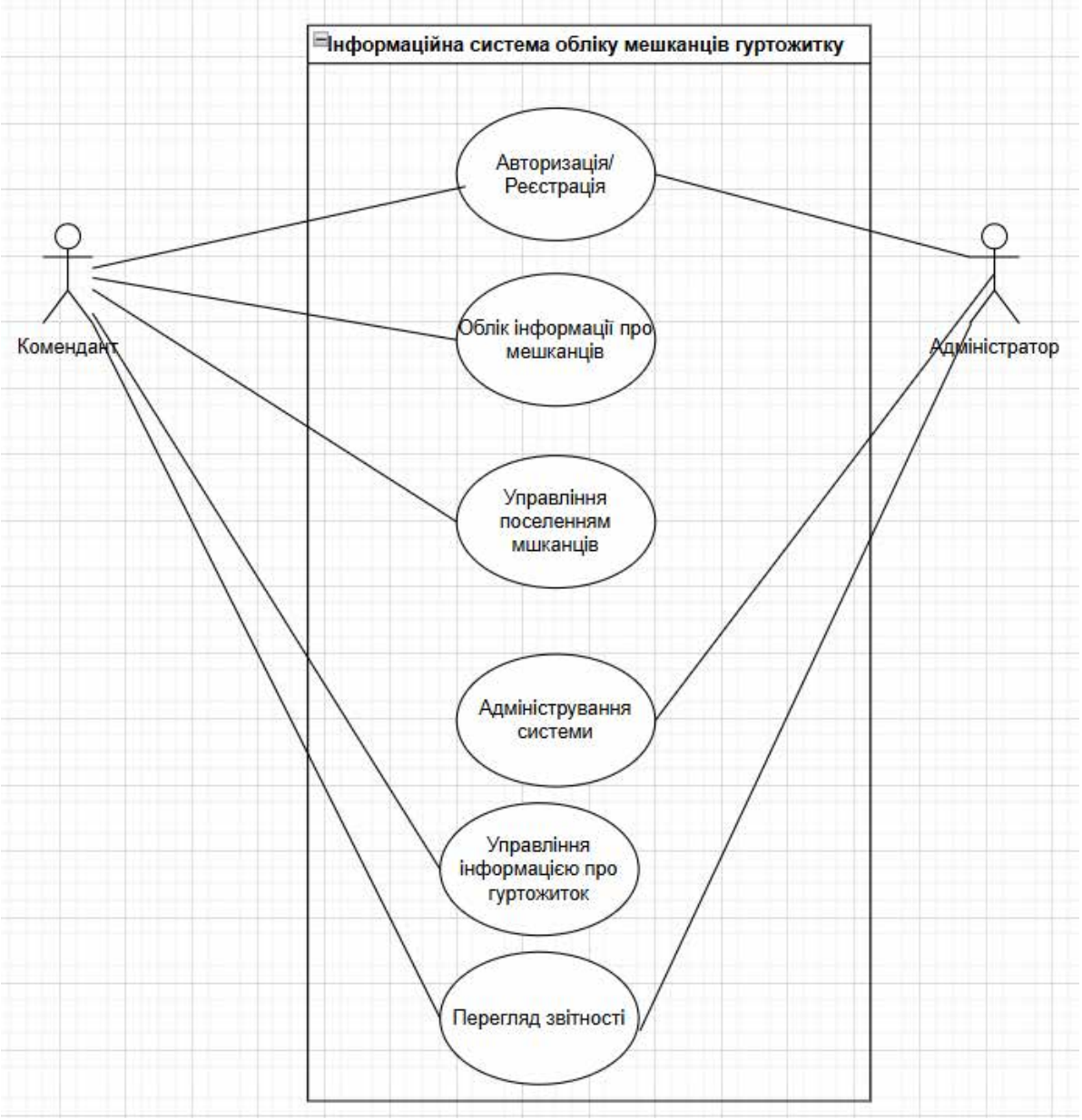


Рис.16 Діаграма прецедентів інформаційної системи

Адміністратор має повний доступ до системи та виконує такі функції: авторизація або реєстрація, адміністрування системи(керування користувачами, правами доступу, налаштуваннями), перегляд звітності про мешканців та систему.

Комендант взаємодіє з системою для виконання основних операцій, пов’язаних із щоденною роботою: авторизація, облік мешканців гуртожитку, управління поселенням та виселенням, перегляд інформацією про гуртожиток, формування або перегляд звітів.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Інформаційна система [5] - це системи в яких відбуваються такі процеси:

- введення інформації, отриманої з джерел інформації;
- опрацювання (перетворення) інформації;
- зберігання вхідної і опрацьованої інформації;
- виведення інформації, призначеної для користувача;
- відправка / отримання інформації мережею.

Інформаційне забезпечення - це сукупність методів, дій, необхідної інформації, документів, засобів розміщення і організації інформації. Від якості інформаційного забезпечення значною мірою залежить достовірність і якість прийнятих управлінських рішень [8].

2.1 Логічна модель даних

Логічна модель даних повинна описувати дані детально, незважаючи як вони будуть фізично реалізовані у базі даних [9] та незалежно від системи керування БД. Основна мета розробки логічної моделі є визначення структури майбутньої бази даних, зв’язків між сутностями та правил збереження інформації. Така модель дозволяє зберегти цілісність, коректність даних, які будуть використовуватись у системі. Та забезпечити збереження та обробку інформації.

Під час побудови логічної моделі даних визначаються основні сутності предметної області, атрибути, первинні та зовнішні ключі, і зв’язки між таблицями.

Логічна модель включає:

• сутності;

• атрибути кожної сутності;

• первинний ключ кожної сутності;

• зовнішні ключі (визначають зв’язок між сутностями);

• також відбувається нормалізація до третьої нормальної форми.

Нормалізація дозволяє уникнути дублювання інформації, надлишковості даних, забезпечення коректності інформації, оптимізувати зберігання даних та підвищити ефективність роботи системи.

Основна інформація у базі даних:

•основна інформація про користувача (ім’я, прізвище, по-батькові, посада, пароль, email, роль);

•інформація про кімнати гуртожитку (поверх, номер, кількість місць, кількість вільних місць);

•інформація про мешканців гуртожитку(ім’я, прізвище, по-батькові, пароль, email, дата народження, стать, факультет, курс, номер телефону, номер студентського квитка);

•інформація про документи(назва, скан);

•інформація про оплату (сума, період, дата оплати, статус);

•інформація про поселення(дата заселення, дата виселення);

•інформація про відсутність студента - якщо відсутній більше доби(дата виїзду, дата приїзду).

На основі логічної моделі (наведена на рис.17) даних формується фізична модель бази даних та реалізуються таблиці у середовищі MySQL Workbench.

Опис сутностей:

1. Сутність Студент

Сутність Студент призначена для збереження основної інформації про мешканців гуртожитку.

Атрибути:

- #Студент;
- Ім’я ;
- Прізвище;

- По_батькові;
- Дата_народження;
- Стать;
- Факультет;
- Курс;
- Група;
- Номер_телефону;
- Email;
- Номер_студ_квитка;
- #Пільга (FK).

2. Сутність Кімната

Сутність Кімната призначена для збереження інформації про кімнати гуртожитку.

Атрибути:

- #Кімната;
- Номер_кімнати;
- Поверх;
- Кількість_місць.

3. Сутність Поселення

Сутність Поселення використовується для збереження інформації про факт проживання студента у певній кімнаті.

Атрибути:

- #Поселення;
- #Студент (FK);
- #Кімната (FK);
- Дата_поселення;
- Дата_виселення.

4. Сутність Оплата

Сутність Оплата призначена для обліку оплат за проживання у гуртожитку.

Дозволяє контролювати оплату за певний період та визначати наявність заборгованості.

Атрибути:

- #Оплата;
- #Студент (FK);
- Період_оплати - кількість місяців, за скільки здійснюється оплата;
- Дата_оплати;
- Сума_нарахована - сума, яку необхідно сплатити;
- Сума_сплачена - фактично сплачена сума.

5. Сутність Документ

Сутність Документ використовується для збереження інформації про документи мешканців гуртожитку.

Атрибути:

- #Документ;
- #Студент (FK);
- Назва;
- Файл - файл або скан документа;
- Дата_додавання.

6. Сутність Відсутність

Сутність Відсутність призначена для фіксації випадків тимчасової (більше 24 годин) відсутності студента у гуртожитку.

Атрибути:

- #Відсутність;
- Дата_виїзду;
- Дата_повернення;
- #Студент (FK).

7. Сутність Пільга

Сутність Пільга використовується для збереження інформації про пільги, які можуть надаватися студентам.

Атрибути:

- #Пільга;
- Назва;
- Відсоток_знижки;
- Документ;
- Термін_дії.

8. Сутність Інвентар

Сутність Інвентар призначена для обліку речей, які видаються студентам під час проживання у гуртожитку.

Наприклад: ключі, постільна білизна, меблі.

Атрибути:

- #Інвентар;
- Назва;
- Стан_при_видачі;
- Дата_видачі;
- Дата_повернення;
- #Студент (FK).

9. Користувач

Атрибути:

- #Користувач;
- Логін;
- Пароль;
- Ім’я;
- Email;
- Роль.

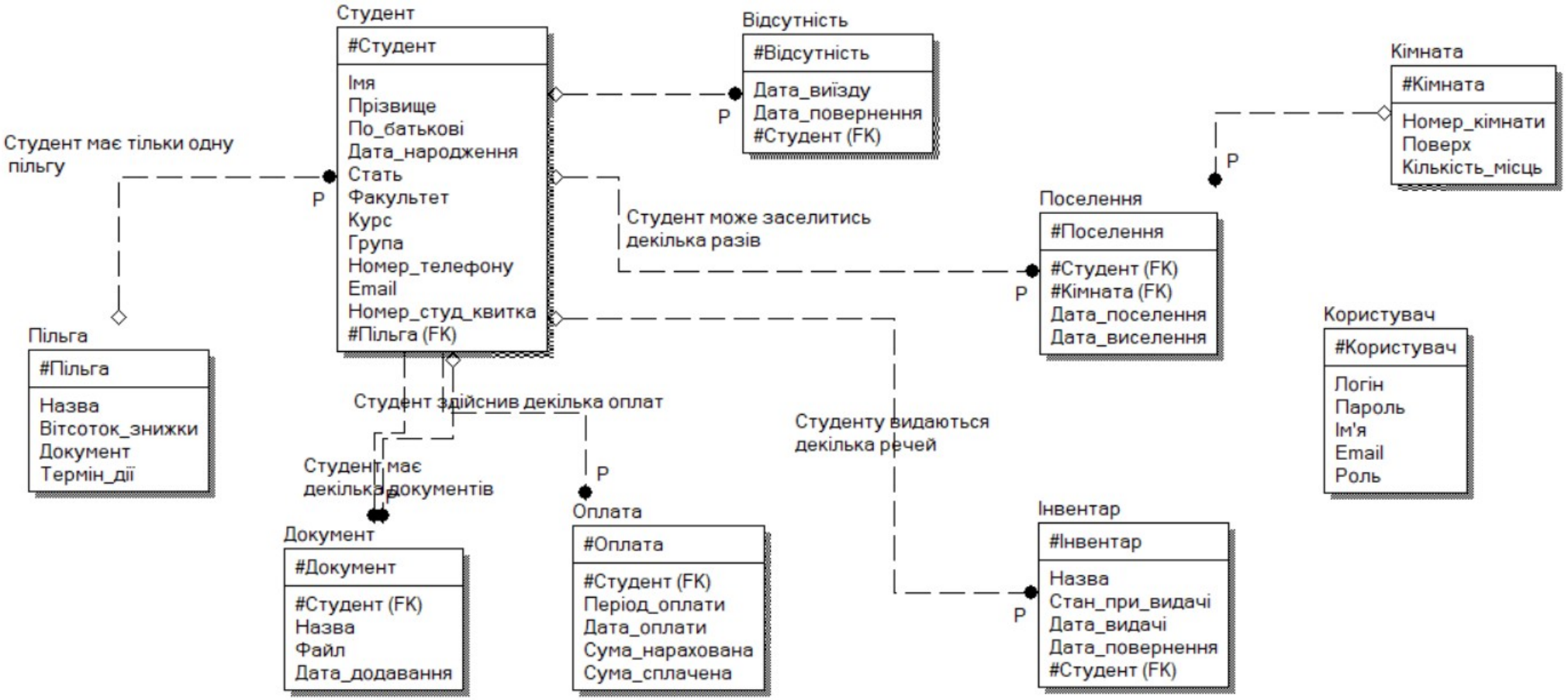


Рис.17 Зображено схему опису структуризації даних в БД «bdstudent».

2.2 Вибір системи управління інформаційною базою

Для того що б було можливо ефективно керувати даними потрібно вибрати систему управління базою даних. СУБД повинна відповідати типу розроблюваної системи, вимогам структурування даних та логічній моделі. Також бути простою якщо вперше використання.

Система управління базою даних [10] - це комплекс програмного забезпечення, що використовується, для створення, організації, збереження, адмініструванням даних. Також дозволяє взаємодіяти з ними. Дані системи використовуються для керуваннями великих обсягів даних.

На прикладі нашої системи обліку студентів гуртожитку коли користувач переходить на вкладку «Студенти», сайт надсилає запит у СУБД, а та, своєю чергою, шукає потрібну інформацію в БД, оновлює дані (список студентів – сутність студент) і повертає результат. Якщо б такої системи не було – то доводилося б вручну шукати і виводити інформацію з більше ніж 1000 студентів, як для нормального гуртожитка.

Види СУБД [11]:

1. Реляційна база даних - інформація у таких базах даних упорядковується та нормалізується, має чітки зв’язки між собою. Подається у вигляді таблиць. Наприклад списки кімнат як є у гуртожитку.
2. Нереляційними базами даних (NoSQL) - призначені для неструктурованої інформації, що не має чітких зв’язків між собою. Наприклад якщо в системі є відгуки чи заявки.
3. Об’єктно-орієнтовні (ODBMS) - зберігають дані у виді об’єктів на основі класів, включають дані, атрибути та методи обробки. Підтримують успадкування і поліморфізм.
4. Ієрархічні - у даній системі управління дані організовані у деревоподібну структуру, де кожен елемент може мати кілька підлеглих.

Функції системи управління базами даних [11, 21]:

1. Управління даними - основна роль це зберігання інформації, вилучати та оновлювати, змінювати, що дає змогу ефективно керувати даними в базі. Оптимізація гарантує швидкість роботи системи, що важливо у великих системах де потрібен моментальний доступ. Для цього використовуються спеціальні програми для бази даних і мови запитів.

2. Зберігання та організація даних - забезпечення впорядкованого зберігання інформації будь-якому потрібному форматі - текст, цифри, відео, зображення. Але дані розташовані логічно і доступно.

3. Забезпечення безпеки даних - у СУБД надають механізми для захисту інформації і важливими є аутентифікація та авторизація. Також шифрування - забезпечує безпеку чутливої інформації. Система прав доступу, контроль цілісності, надаються інструменти моніторингу (виявлення потенційних загроз і відновлюватись після збоїв).

4. Резервне копіювання і відновлення - функції СУБД включають систематичне створення резервних копій і можливість швидкого відновлення бази даних після збоїв або втрати - забезпечення відмовостійкості.

5. Оптимізація і продуктивність - СУБД забезпечують оптимізацію виконання запитів, що дозволяє максимально швидко отримувати потрібну інформацію, навіть при значному навантаженні на систему. Завдяки чому забезпечується висока швидкість обробки даних в БД та стабільна робота у режимі реального часу.

6. Конкурентний доступ - системи управління базами даних надають можливість одночасної роботи з однією базою даних багатьом користувачам або програмним продуктам. Ця функція дозволяє уникати конфліктів під час роботи з даними, забезпечуючи їх цілісність і узгодженість.

Також система управління базами даних має свої складники для того що б база даних працювала ефективно та злагоджено, і відповідала всім процесам.

Основні компоненти середовища СУБД [22]:

1. Програмне забезпечення - набір програм, які використовуються для управління базою даних. Включає такий набір для керування: програмне

забезпечення СУБД, операційну систему, мережеве ПЗ (використовується для обміну даних між користувачами), прикладні програми (використовуються для доступу до даних).

2. Апаратне забезпечення - набір фізичних електронних пристроїв, включає: комп’ютери, пристрої введення/виведення, пристрої для зберігання даних та інше. Такі пристрої забезпечують інтерфейс між комп'ютерами та реальними системами.

3. Система управління базами даних для даних – керування збором, зберіганням, обробки, оновленням та доступу до збережених даних.

4. Процедури - інструкції та правила, вони допомагають користувачам у використанні СУБД, а також проектувати та запускати БД з використанням документованих процедур.

5. Мова доступу до бази даних - використовується користувачами для доступу до БД та даних, керування даними(введення, виведення, оновлення вже існуючих, та отримання необхідних даних). Для цього користувачу необхідно написати набір команд відповідною мовою доступу до БД, надсилає їх до СУБД, яка обробляє цей запит, генерує і відображає відповідні результати у зручному для користувача форматі.

6. Процесор запитів - один із ключових компонентів СУБД, його основним призначенням є обробка запитів від користувача та перетворення їх у послідовність внутрішніх команд. Після отримання SQL-запиту процесор аналізує його - структур, правильність - формує план виконання операцій. Також він забезпечує оптимізацію запитів, що і зменшує час для доступу і зберігає продуктивність. Після обробки запит передається менеджеру даних.

7. Менеджер бази даних - відповідає за обробку даних у БД, і дозволяє у разі збою відновити дані.

8. Механізм обробки даних - основний компонент для зберігання, обробки та захисту даних, яка забезпечує контрольований доступ і швидку обробку транзакцій.

9. Програма генератора звітів - спеціальна програма, яка призначена для формування, обробки та відображення інформації у зручному для користувача вигляді. Отримати дані можна з тільки однієї або декількох таблиць БД. В інформаційній системі обліку мешканців гуртожитку засоби формування звітів можуть використовувати для створення статистики заселеності, звітів про оплату проживання, списки мешканців за критеріями та інформації про наявність вільних місць у гуртожитку.

Ці компоненти забезпечують ефективну роботу всієї бази даних.

І зважаючи на те що СУБД є різні типи для окремих систем та інформації вибір є дуже важливим, для того що б інформаційна система працювала злагоджено, дані виводились та вводились, запити працювали і швидко знаходилась потрібна інформація.

Таблиця 1

Порівняння основних СУБД

СУБД	**Тип**	**Переваги**	**Недоліки**	**Сфера використання**
MySQL	Реляційна	Висока швидкість, простий у використанні, підтримка вебтехнологій, відкритий код, підтримка SQL-запитів	Малі можливості для складної аналітики	Вебсайти, інформаційні системи, онлайн-сервіси
PostgreSQL	Об’єктно-реляційна	Надійність, підтримка SQL-запитів, висока цілісність даних	Складніше налаштування	Великі інформаційні системи, аналітичні платформи

Продовження таблиці 1

Microsoft SQL Server	Реляційна	Інструменти адміністрування, інтеграція з продуктами Microsoft, підтримка аналітики	Платна ліцензія та високі системні вимоги	Корпоративні системи, бізнес-додатки
Oracle Database	Об’єктно-реляційна	Висока продуктивність, безпека, масштабованість	Платна ліцензія та складність адміністрування	Корпоративні та банківські системи
MongoDB	Документна	Гнучка структура даних, масштабованість, висока швидкість роботи з неструктурованими даними	Менша підтримка складних транзакцій	Вебзастосунки, мобільні додатки
SQLite	Реляційна	Компактність, не потребує сервера, проста у встановленні	Не підходить для великих багатокористувацьких систем	Мобільні застосунки, невеликі програми

Після аналізу існуючих систем управління базами даних для розробки інформаційної системи обліку було обрано MySQL. Оскільки база даних для ІС є реляційною і буде підтримка, дана система забезпечить швидкість роботи процесів, має зрозумілий інтерфейс, добре інтегрується з мовою програмування Python і вебтехнологіями, підтримує SQL-запити. Також обрана СУБД є безкоштовною та широко використовується саме для веборієнтованих ІС.

Для візуального проєктування структури бази даних, створення таблиць та виконання SQL-запитів, під час розробки ІС використовується

середовище MySQL Workbench приклади функціоналу зображено на рис.18-19.

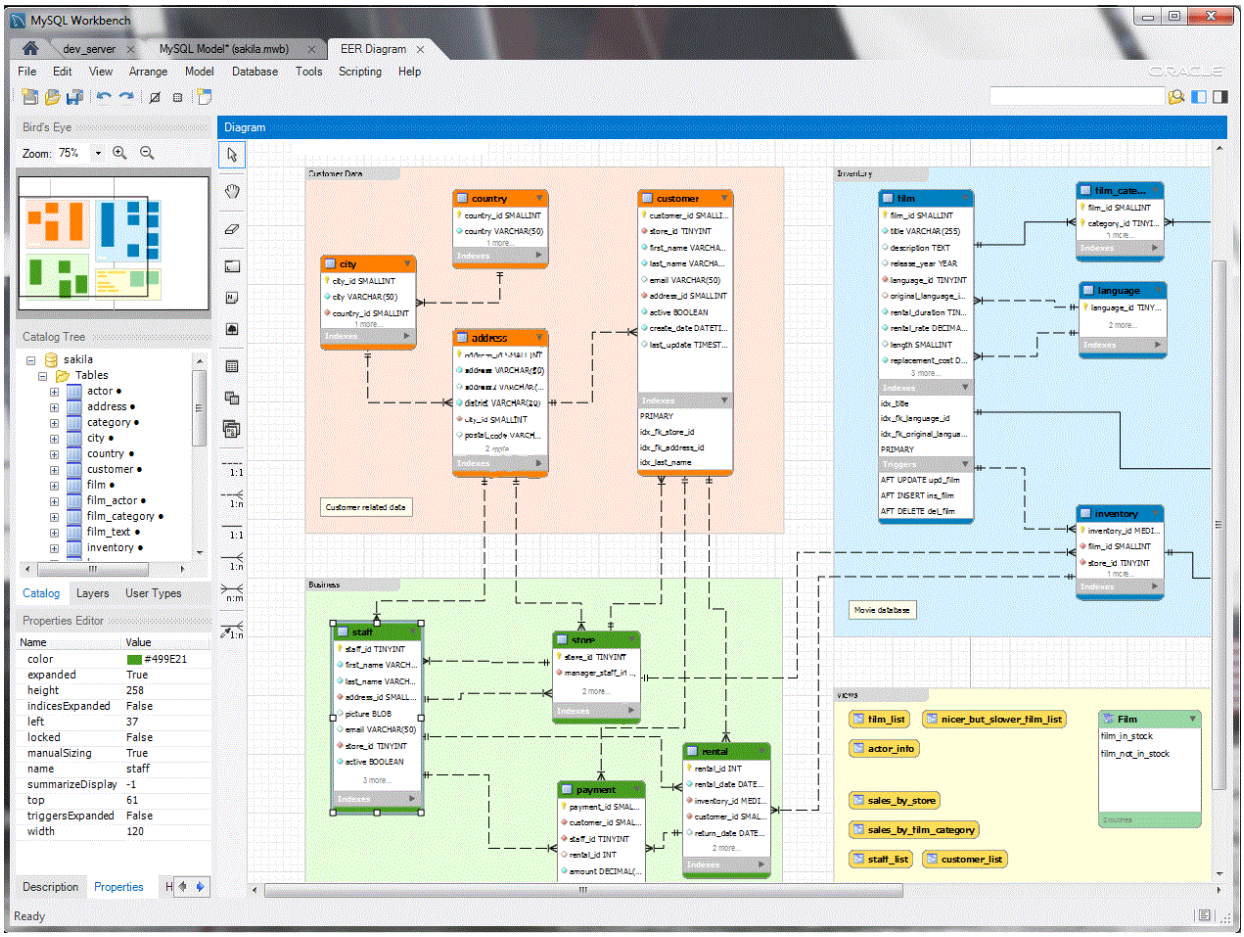


Рис.18 Структура БД

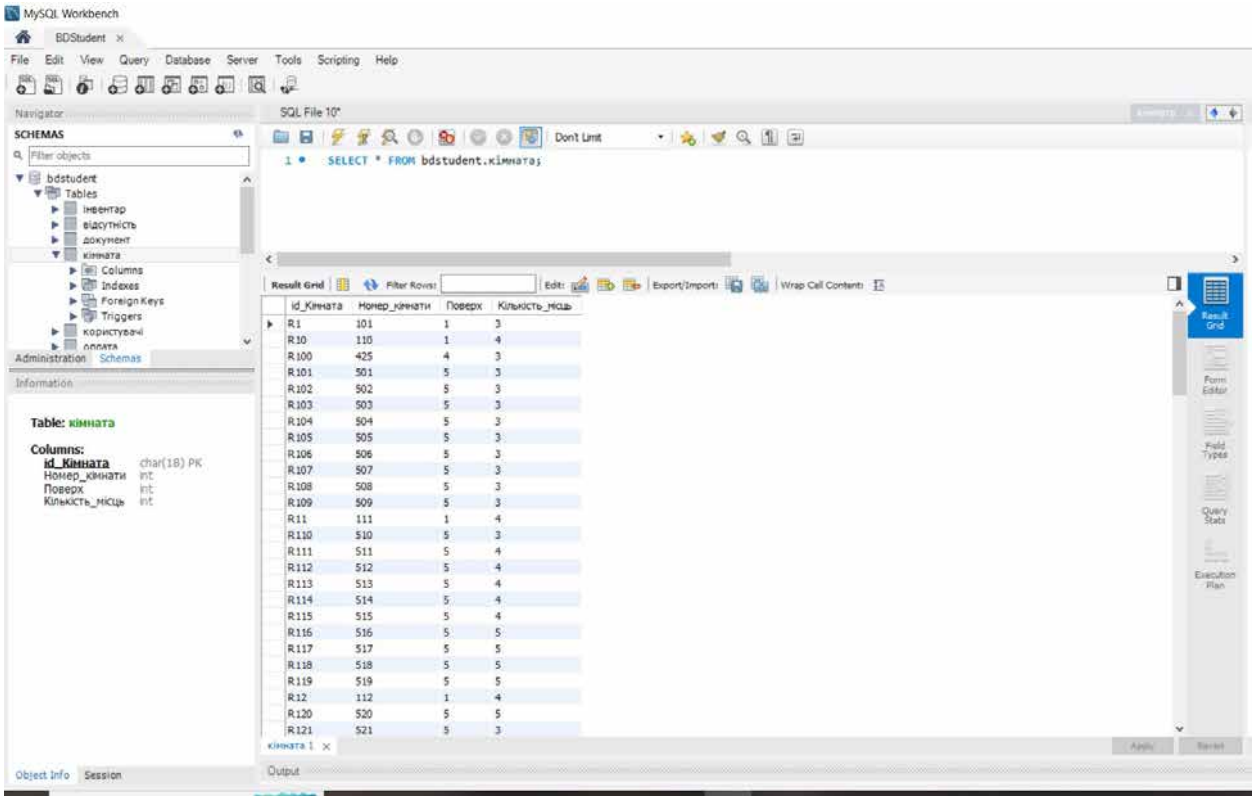


Рис.19 Приклад вікна MySQL Workbench.

2.3 Створення інформаційної бази

Для створення інформаційної бази - бази даних «bdstudent» - було за допомогою запитів SQL. Схема БД зображена на рис.21.

CREATE SCHEMA dbstudent;

Приклад створення таблиці на рис.20 куди заноситься інформація про те, який інвентар якому студенту буде виданий.

```
USE BDStudent;
CREATE TABLE інвентар (
   id_інвентар CHAR(18) PRIMARY KEY,
   Назва VARCHAR(100),         -- Назва (Ліжко, Постільна білизна)
   Стан VARCHAR(50),      -- Стан (Нове, Вживане)
   Дата_видачі DATETIME,
   Дата_повернення DATETIME DEFAULT NULL, (NULL якщо ще у студента)
   id_Студент CHAR(18),            -- Хто взяв
   FOREIGN KEY (id_Студент) REFERENCES студент(id_Студент)
);
```

Рис.20 Код для створення таблиці «Інвентар»

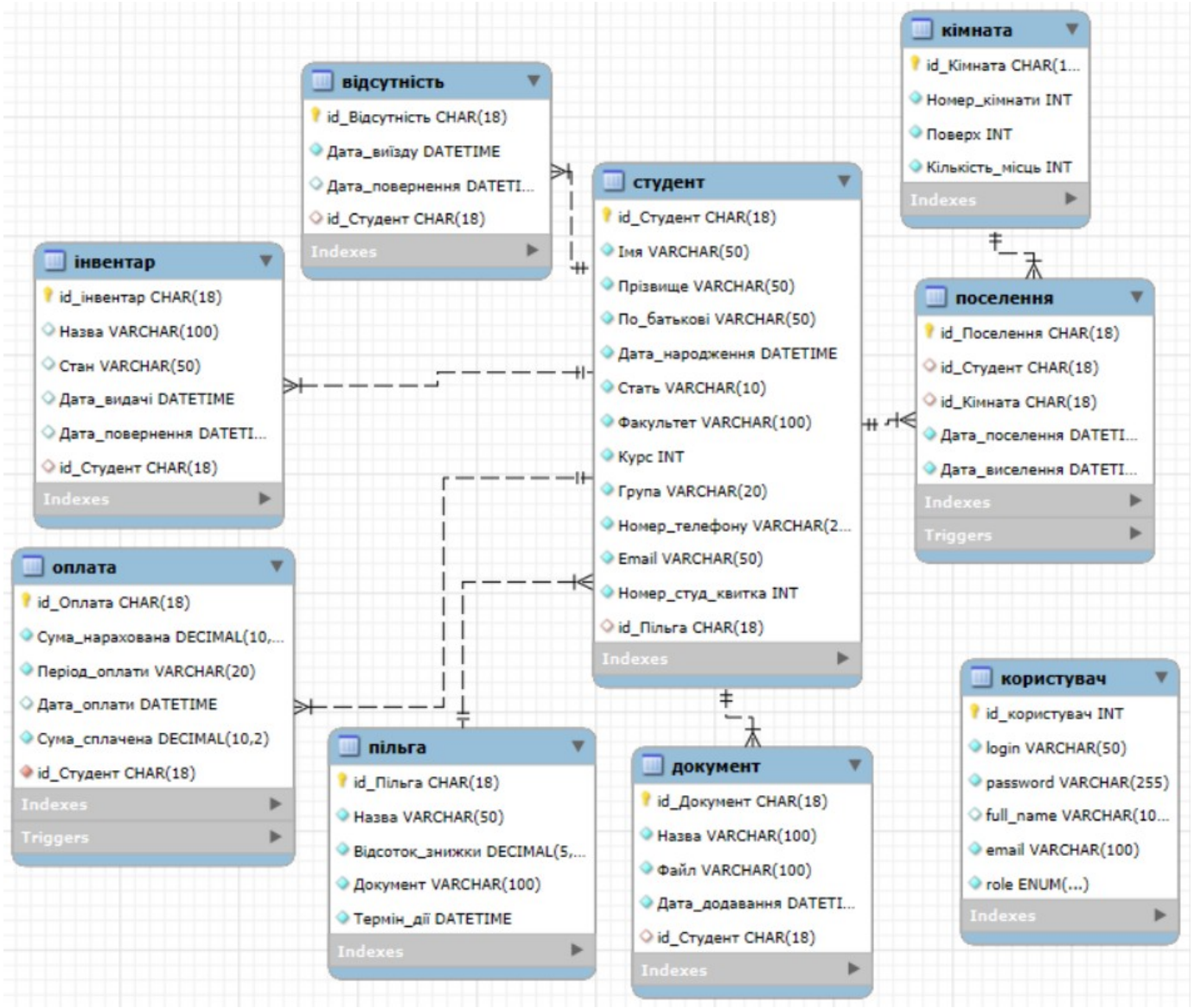


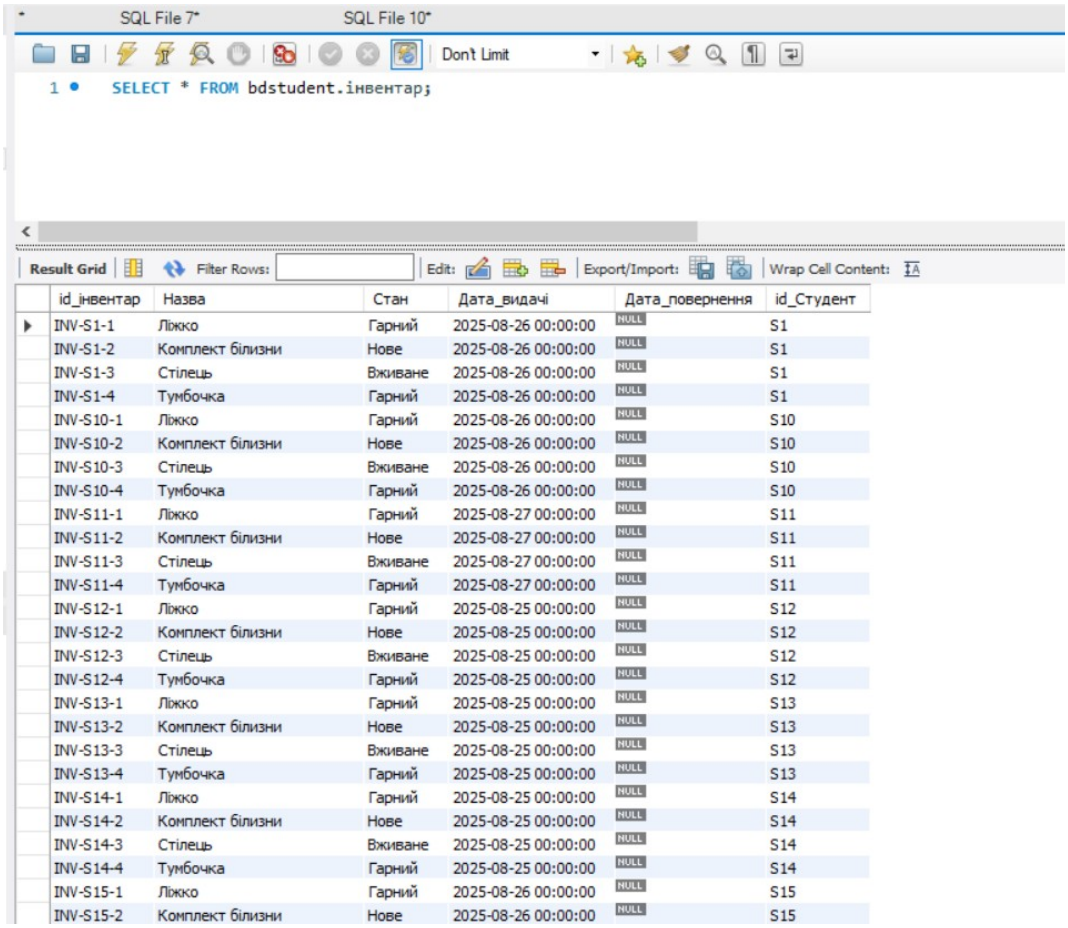
Рис.21 Фізична модель БД „bdstudent”

Також заповнення умовно-постійною інформацію для тестування та перевірки чи все коректно працює чи всі зв’язки виконано правильно. Приклади заповнених таблиць зображено на рис. 23-24.

У системі 40 студентів – і для кожного студента по 4 інвентаря, тому що б не витрачати час на заповнення то можна автоматично, приклад коду на рис.22.

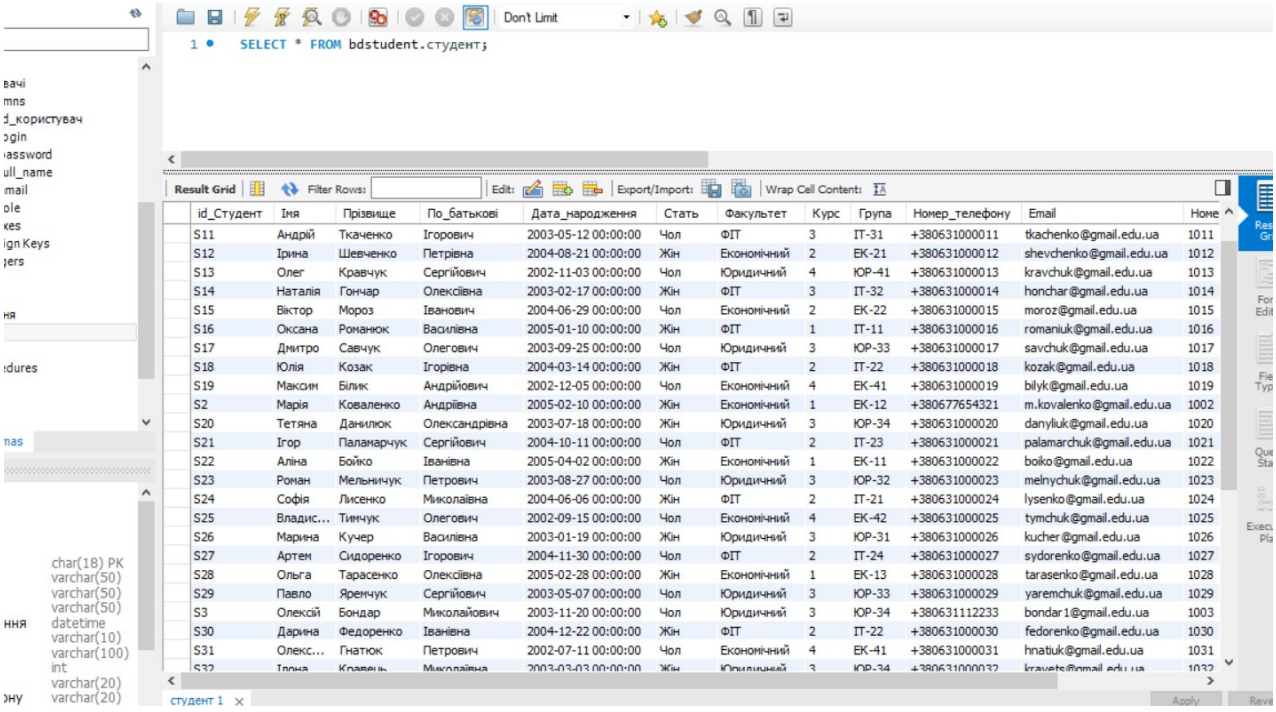
```
USE BDStudent;
-- 1. Видаємо Ліжка (40 штук)
INSERT INTO інвентар (id_інвентар, Назва, Стан, Дата_видачі, id_Студент)
SELECT CONCAT('INV-', id_Студент, '-1'), 'Ліжко', 'Гарний', '2025-08-25',
id_Студент
FROM студент;
-- 2. Видаємо Комплекти білизни (40 штук)
INSERT INTO інвентар (id_інвентар, Назва, Стан, Дата_видачі, id_Студент)
SELECT CONCAT('INV-', id_Студент, '-2'), 'Комплект білизни', 'Нове', '2025-
08-25', id_Студент
FROM студент;
-- 3. Видаємо Стільці (40 штук)
INSERT INTO інвентар (id_інвентар, Назва, Стан, Дата_видачі, id_Студент)
SELECT CONCAT('INV-', id_Студент, '-3'), 'Стілець', 'Вживане', '2025-08-25',
id_Студент
FROM студент;
-- 4. Видаємо Тумбочки (40 штук)
INSERT INTO інвентар (id_інвентар, Назва, Стан, Дата_видачі, id_Студент)
SELECT CONCAT('INV-', id_Студент, '-4'), 'Тумбочка', 'Гарний', '2025-08-25',
id_Студент
FROM студент;
```

Рис.22 Код заповнення таблиці «Інвентар»



id_інвентар	Назва	Стан	Дата_видачі	Дата_повернення	id_Студент
INV-S1-1	Ліжко	Гарний	2025-08-26 00:00:00	NULL	S1
INV-S1-2	Комплект білизни	Нове	2025-08-26 00:00:00	NULL	S1
INV-S1-3	Стілець	Вживане	2025-08-26 00:00:00	NULL	S1
INV-S1-4	Тумбочка	Гарний	2025-08-26 00:00:00	NULL	S1
INV-S10-1	Ліжко	Гарний	2025-08-26 00:00:00	NULL	S10
INV-S10-2	Комплект білизни	Нове	2025-08-26 00:00:00	NULL	S10
INV-S10-3	Стілець	Вживане	2025-08-26 00:00:00	NULL	S10
INV-S10-4	Тумбочка	Гарний	2025-08-26 00:00:00	NULL	S10
INV-S11-1	Ліжко	Гарний	2025-08-27 00:00:00	NULL	S11
INV-S11-2	Комплект білизни	Нове	2025-08-27 00:00:00	NULL	S11
INV-S11-3	Стілець	Вживане	2025-08-27 00:00:00	NULL	S11
INV-S11-4	Тумбочка	Гарний	2025-08-27 00:00:00	NULL	S11
INV-S12-1	Ліжко	Гарний	2025-08-25 00:00:00	NULL	S12
INV-S12-2	Комплект білизни	Нове	2025-08-25 00:00:00	NULL	S12
INV-S12-3	Стілець	Вживане	2025-08-25 00:00:00	NULL	S12
INV-S12-4	Тумбочка	Гарний	2025-08-25 00:00:00	NULL	S12
INV-S13-1	Ліжко	Гарний	2025-08-25 00:00:00	NULL	S13
INV-S13-2	Комплект білизни	Нове	2025-08-25 00:00:00	NULL	S13
INV-S13-3	Стілець	Вживане	2025-08-25 00:00:00	NULL	S13
INV-S13-4	Тумбочка	Гарний	2025-08-25 00:00:00	NULL	S13
INV-S14-1	Ліжко	Гарний	2025-08-25 00:00:00	NULL	S14
INV-S14-2	Комплект білизни	Нове	2025-08-25 00:00:00	NULL	S14
INV-S14-3	Стілець	Вживане	2025-08-25 00:00:00	NULL	S14
INV-S14-4	Тумбочка	Гарний	2025-08-25 00:00:00	NULL	S14
INV-S15-1	Ліжко	Гарний	2025-08-26 00:00:00	NULL	S15
INV-S15-2	Комплект білизни	Нове	2025-08-26 00:00:00	NULL	S15

Рис.23 Таблиця - інвентар



id_Студент	Ім'я	Прізвище	По_батькові	Дата_народження	Стать	Факультет	Курс	Група	Номер_телефону	Email	Номе
S11	Андрій	Ткаченко	Ігорович	2003-05-12 00:00:00	Чол	ФІТ	3	ІТ-31	+380631000011	tkachenko@gmail.edu.ua	1011
S12	Ірина	Шевченко	Петрівна	2004-08-21 00:00:00	Жін	Економічний	2	ЕК-21	+380631000012	shevchenko@gmail.edu.ua	1012
S13	Олег	Кравчук	Сергійович	2002-11-03 00:00:00	Чол	Юридичний	4	ЮР-41	+380631000013	kravchuk@gmail.edu.ua	1013
S14	Наталія	Гончар	Олексіївна	2003-02-17 00:00:00	Жін	ФІТ	3	ІТ-32	+380631000014	honchar@gmail.edu.ua	1014
S15	Віктор	Мороз	Іванович	2004-06-29 00:00:00	Чол	Економічний	2	ЕК-22	+380631000015	moroz@gmail.edu.ua	1015
S16	Оксана	Романюк	Василівна	2005-01-10 00:00:00	Жін	ФІТ	1	ІТ-11	+380631000016	romaniuk@gmail.edu.ua	1016
S17	Дмитро	Савчук	Олегович	2003-09-25 00:00:00	Чол	Юридичний	3	ЮР-33	+380631000017	savchuk@gmail.edu.ua	1017
S18	Юлія	Козак	Ігорівна	2004-03-14 00:00:00	Жін	ФІТ	2	ІТ-22	+380631000018	kozak@gmail.edu.ua	1018
S19	Максим	Білик	Андрійович	2002-12-05 00:00:00	Чол	Економічний	4	ЕК-41	+380631000019	bilyk@gmail.edu.ua	1019
S2	Марія	Коваленко	Андріївна	2005-02-10 00:00:00	Жін	Економічний	1	ЕК-12	+380677654321	m.kovalenko@gmail.edu.ua	1002
S20	Тетяна	Данилюк	Олександрівна	2003-07-18 00:00:00	Жін	Юридичний	3	ЮР-34	+380631000020	danyliuk@gmail.edu.ua	1020
S21	Ігор	Паламарчук	Сергійович	2004-10-11 00:00:00	Чол	ФІТ	2	ІТ-23	+380631000021	palamarchuk@gmail.edu.ua	1021
S22	Аліна	Бойко	Іванівна	2005-04-02 00:00:00	Жін	Економічний	1	ЕК-11	+380631000022	boiko@gmail.edu.ua	1022
S23	Роман	Мельничук	Петрович	2003-08-27 00:00:00	Чол	Юридичний	3	ЮР-32	+380631000023	melnychuk@gmail.edu.ua	1023
S24	Софія	Лисенко	Миколаївна	2004-06-06 00:00:00	Жін	ФІТ	2	ІТ-21	+380631000024	lysenko@gmail.edu.ua	1024
S25	Владис...	Тимчук	Олегович	2002-09-15 00:00:00	Чол	Економічний	4	ЕК-42	+380631000025	tymchuk@gmail.edu.ua	1025
S26	Марина	Кучер	Василівна	2003-01-19 00:00:00	Жін	Юридичний	3	ЮР-31	+380631000026	kucher@gmail.edu.ua	1026
S27	Артем	Сидоренко	Ігорович	2004-11-30 00:00:00	Чол	ФІТ	2	ІТ-24	+380631000027	sydorenko@gmail.edu.ua	1027
S28	Ольга	Тарасенко	Олексіївна	2005-02-28 00:00:00	Жін	Економічний	1	ЕК-13	+380631000028	tarasenko@gmail.edu.ua	1028
S29	Павло	Яремчук	Сергійович	2003-05-07 00:00:00	Чол	Юридичний	3	ЮР-33	+380631000029	yaremchuk@gmail.edu.ua	1029
S3	Олексій	Бондар	Миколайович	2003-11-20 00:00:00	Чол	Юридичний	3	ЮР-34	+380631112233	bondar1@gmail.edu.ua	1003
S30	Дарина	Федоренко	Іванівна	2004-12-22 00:00:00	Жін	ФІТ	2	ІТ-22	+380631000030	fedorenko@gmail.edu.ua	1030
S31	Олекс...	Гнатюк	Петрович	2002-07-11 00:00:00	Чол	Економічний	4	ЕК-41	+380631000031	hnatiuk@gmail.edu.ua	1031

Рис.24 Таблиця – студент

Для автоматизації обробки даних у базі даних було створено декілька збережених процедур. Одна з яких «БоргСтудента», дозволяє виконувати автоматичний підрахунок заборгованості конкректного студента за проживання в гуртожитку.

Процедура підрахунок боргу студента, приклад коду на рис.25.

```
USE bdstudent;
DELIMITER //
CREATE PROCEDURE БоргСтудента(IN p_id CHAR(18))
BEGIN
   SELECT
      s.Імя,
      s.Прізвище,
      SUM(o.Сума_нарахована - o.Сума_сплачена) AS Борг
   FROM Студент s
   JOIN Оплата o ON s.id_Студент = o.id_Студент
   WHERE s.id_Студент = p_id
   GROUP BY s.id_Студент;
END //
DELIMITER ;
```

Рис.25 Код для підрахунку боргу студента

У процесі виконання процедура:

- отримує ім’я та прізвище студента;
- обчислює різницю між сумою нарахованих коштів та сумою сплачених платежів;
- формує загальну суму боргу студента;
- групує результати;

Результатом процедури є інформація про студента та його поточну заборгованість.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Для описання та представлення загальної структури розроблюваної інформаційної системи обліку мешканців гуртожитку, було побудовано діаграму пакетів UML. На даній діаграма рис.26 представлені основні модулі системи, структура та взаємозв'язки між компонентами.

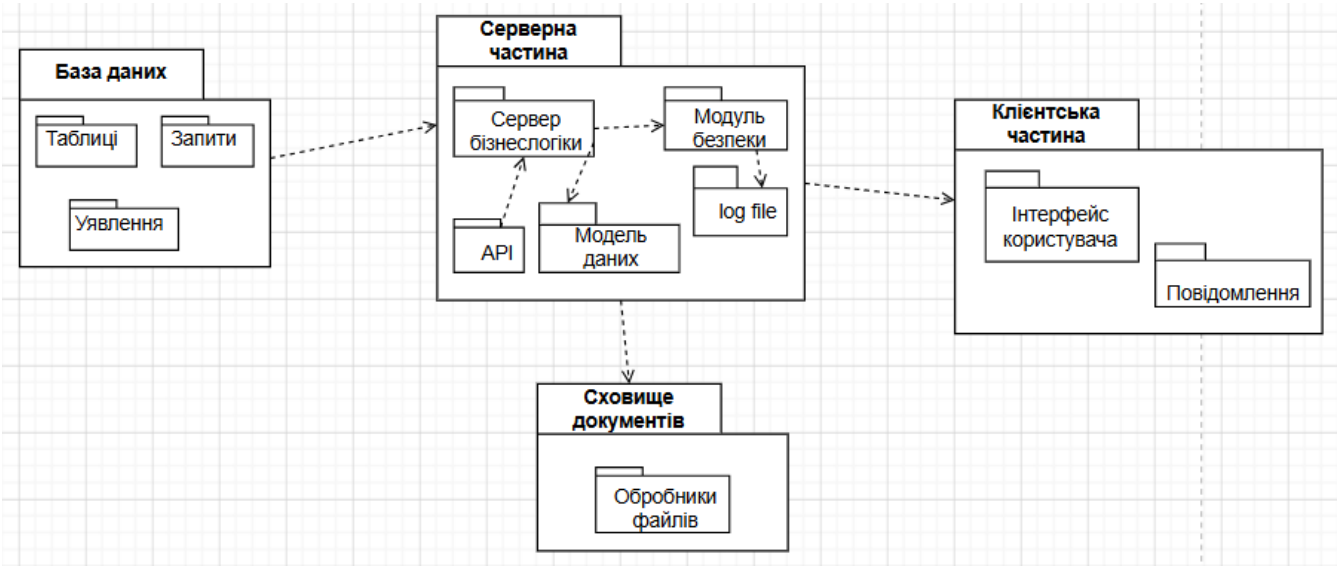


Рис.26 Діаграма пакетів ІС

Між пакетами встановлено залежності, що відображають процес обміну даними між компонентами системи.

Основні пакети системи:

•база даних – містить SQL-запити та модулі управління даними, відповідає за збереження інформації про мешканців гуртожитку, кімнати, оплату;

•серверна частина – реалізує основну бізнес-логіку системи. Забезпечує обробку запитів користувачів, перевірку даних, взаємодію з БД, авторизацію. До його складу входять: сервер бізнес-логіки, API для взаємодії між клієнтською частиною та базою даних, модуль безпеки, модель даних, журнал подій;

•клієнтська частина – відповідає за взаємодію користувача з системою. Входять: інтерфейс користувача та модулі відображення повідомлень;

•сховище документів – зберігання електронних копій документів мешканців гуртожитку.

3.2 Вибір інструментарію для створення ПЗ

Для розробки інформаційної системи було проаналізовано сучасні програмні засоби та технології, які забезпечують зручність створення саме веборієнтованих інформаційних систем. Також критеріями для відбору було – підтримка роботи з обраною СУБД та базами даних, та які дозволяють реалізувати функціональний і зрозумілий інтерфейс для користувачів. У результаті аналізу програмних засобів було обрано інструментарій, який забезпечує швидкість розробки, можливість масштабування системи, підтримку сучасних вебтехнологій.

Середовище розробки Visual Studio Code [12] – безкоштовний, відкритий, крос-платформний редактор коду. Випущений у квітні 2015 року, популярний серед розробників. Легкий і добре настроюваний, що робить його ідеальним для розробників, які хочуть швидко та ефективно кодувати без зайвого роздуття. Підтримці декількох мов програмування, інтегрованому Git-контролю, можливостям налагодження та великій бібліотеці розширень.

Особливості:

1. Крос-платформна сумісність – програмний продукт доступний для Windows, macOS та Linux.

2. Вбудована інтеграція з GitHub – інтеграція безкоштовна, дозволяє клонувати репозиторії, фіксувати зміни, надсилати оновлення.

3. Інтелектуальні пропозиції коду - підсвічування коду, виявлення помилок, що пришвидшує кодування та зменшує кількість потенційних помилок.

4. Розширення - VS Code має для налаштування бібліотеку розширень. Встановлення тем, мовної підтримки, інструментів налагодження.

5. Вбудований термінал – розробники мають змогу виконувати команди, не виходячи з редактора.

6. Можливості налагодження - відладчик, який підтримує точки зупинки, покрокове налагодження та перевірку змінних.

7. Підтримка декількох мов програмування - VS Code підтримує такі мови, як JavaScript, Python, C++, C#, HTML, CSS та PHP.

Для налагодження зв'язку з базою даних, налаштування інтерфейсу користувача було обрано такі мови.

1. Мова програмування Python [13] – високорівнева мова програмування. Використовується для розробки вебзастосунків, програмного забезпечення, машинного навчання. Обрана мова має досить простий синтаксис.

Можливості:

- розробка програмних застосунків;
- розробка серверної частини мобільних застосунків;
- розробка ігор;
- розробка вбудованих систем для різних пристроїв;
- написання скриптів та плагінів для автоматизації процесів чи створення інших рішень;
- тестування;
- мова для написання алгоритмів у сфері Машинного навчання.

2. Мова розмітки сторінки HTML [14] – використовується для опису зовнішнього вигляду та структури сторінки.

3. CSS [16] – стилізація. Спеціальна мова, за допомогою якої описують зовнішній вигляд сторінок – як відображати елементи сторінки,

кольори, розташування, навіть анімацію. Найчастіше CSS використовується для документів, котрі написані мовами HTML, XHTML та XML.

4. Інтерактив Java Script [17, 18] – високорівнева скриптова мова програмування. Використовується саме для створення інтерактивних застосунків і вебсайтів. За допомогою Java Script можна додати анімацію, спливаючі повідомлення, оновлення контенту, реагування інтерфейсу на дії користувачів. Наприклад, якщо користувач наведе курсор на кнопку, то колір світлішає, змінюється, що робить сторінки функціональними.

Основні можливості JavaScript:

- динамічна зміна HTML-коду сторінки та стилів елементів без перевантаження сторінки;
- опрацювання дій користувача, таких як натискання кнопок, переміщення вказівника, вибір елементів меню, взаємодія з елементами сторінки;
- перевірка коректності введених даних у поля, форми перед їх відправленням;
- відображення модальних вікон, сповіщень;
- надсилання асинхронних запитів мережею до серверів, за допомогою технологій AJAX і COMET;
- отримання та збереження даних на стороні клієнта за допомогою Local Storage та Cookies;
- оновлення даних таблиць, списків мешканців або статистики у режимі реального часу.

У розроблювальній системі використовується для реалізації інтерактивного інтерфейсу користувача, роботи модальних вікон для додавання та редагування, перевірка введених даних, динамічного оновлення інформації про студентів, оплату, кімнати, поселення.

5. Bootstrap [19] - безкоштовний фреймворк із відкритим вихідним кодом, призначений для створення сучасних вебсайтів та вебзастосунки. Bootstrap містить шаблони CSS, HTML, JavaScript для реалізації

адаптивного та зручного користувацького інтерфейсу. Використання даного фреймворку значно спрощує процес розробки вебінтерфейсів. Основна перевага Bootstrap є наявність великої кількості готових елементів інтерфейсу.

Основні інструменти Bootstrap:

- Grid-система – адаптивна сітка для побудови структури сторінок;
- Templates - готові шаблони сторінок та компонентів;
- Typography – стилізація тексту, заголовків та інших текстових елементів;
- Media - засоби для роботи з мультимедійним контентом;
- Tables – засоби стилізації та відображення таблиць;
- Forms – оформлення форм;
- Navigation (Navbar) – створення меню навігації, вкладок, сторінок;
- Icons - набір іконок з 500 компонентів.

Bootstrap – також містить інтерактивні JavaScript-компоненти:

- Modal - модальні вікна;
- Dropdown - випадні списки;
- Scrollspy - плагін, який автоматично змінює активний пункт у меню залежно від позиції прокручування сторінки;
- Tab – вкладки;
- Tooltip - спливні підказки;
- Alert – інтерактивні повідомлення;
- Button – керування станами кнопок.

3.3. Алгоритмізація та програмування програмних модулів

Під час проєктування інформаційної системи важливим етапом є розробка алгоритмів функціонування програми. Побудова чіткої послідовності виконання операцій та дій дозволяє наочно продемонструвати,

як саме окремий модуль буде приймати, обробляти та виводити дані. Програмування модулів здійснюється мовою Python. На даному етапі важливо визначити структуру вхідної та вихідної інформації, а також встановити точну послідовність подій.

Алгоритмізація застосовується для вирішення масштабних завдань, спрощення процесу розробки та оптимізації коду. Створення алгоритмів дає змогу ефективно аналізувати складні процеси, знаходити відповідні рішення та керувати іншими процесами написання програмного забезпечення.

Для прикладу показано декілька модулів:

1. Алгоритм додавання нового студента, блок-схема зображена на рис.27.

- запуск процесу;
- користувач заповнює форми відповідною інформацією;
- система зчитує введені дані для подальшої обробки;
- далі йде перевірка чи всі поля заповнені:

– Ні: Система виводить повідомлення про помилку, і повертає користувача до етапу заповнення форми;

– Так: Алгоритм рухається далі;

- створення об’єкта класу Student програма ініціалізує новий об’єкт;
- об’єкт готується до запису у відповідну таблицю БД;
- спроба збереження мін;
- далі перевірка чи коректно збережені зміни:

– Ні: Система виводить повідомлення про помилку, і повертає користувача до етапу заповнення форми;

– Так: Якщо БД успішно прийняла дані, система виконує «Оновлення сторінки»;

- процес завершено.

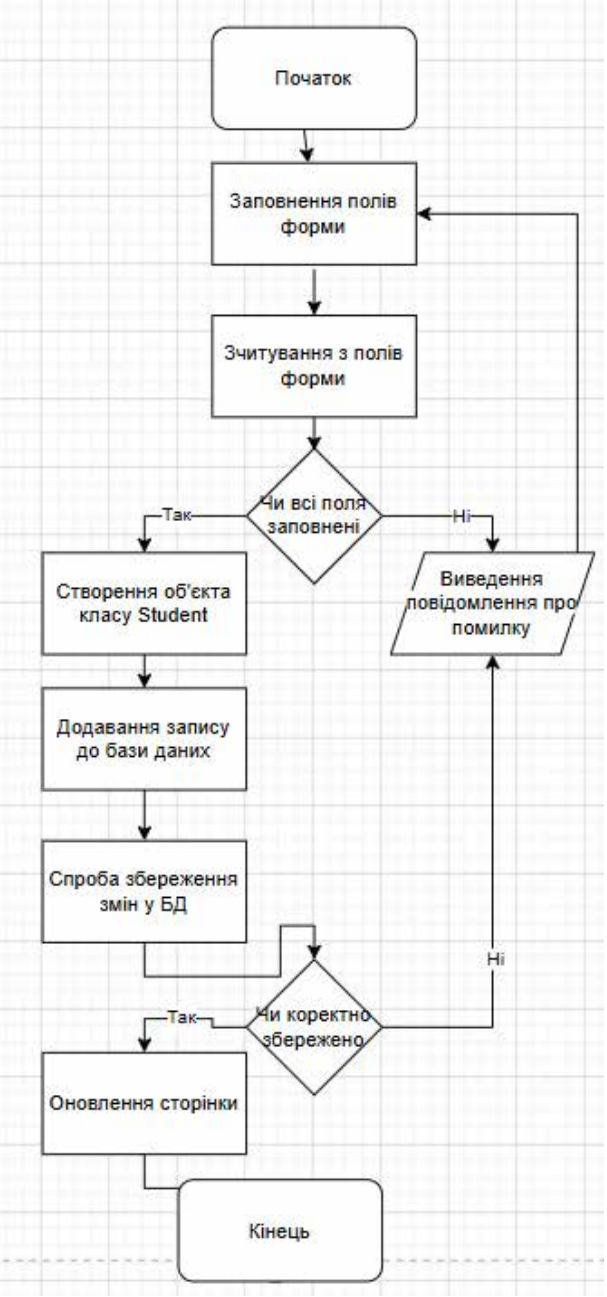


Рис.27 Блок-схема додавання нового студента

Приклад коду

Опис моделі для вибірки даних з БД класу Student за допомогою SQLAlchemy ORM зображеного на рис.28.

```
class Student(db.Model):
    __tablename__ = 'Студент'

    id_Студент = db.Column(db.Integer, primary_key=True)
    Імя = db.Column(db.String(50), nullable=False)
    Прізвище = db.Column(db.String(50), nullable=False)
    По_батькові = db.Column(db.String(50))
```

```
    Дата_народження = db.Column(db.Date)
    Стать = db.Column(db.String(10))
    Факультет = db.Column(db.String(100))
    Курс = db.Column(db.Integer)
    Група = db.Column(db.String(20))
    Номер_телефону = db.Column(db.String(20))
    Email = db.Column(db.String(100))
    Номер_студ_квитка = db.Column(db.String(20))
    id_Пільга = db.Column(db.Integer)

    def __repr__(self):
        return f'<Student {self.Прізвище} {self.Імя}>'
```

Рис.28

Логіка збереження, запит – приймає дані з форми, робляться перевірки та запис у БД зображеного на рис.29

```
db.init_app(app)

@app.route('/students/add', methods=['POST'])
def add_student():
    """Маршрут для обробки форми та додавання студента до БД"""
    try:
        # Зчитування даних
        last_name = request.form.get('Прізвище')
        first_name = request.form.get('Імя')
        middle_name = request.form.get('По_батькові')
        faculty = request.form.get('Факультет')
        course = request.form.get('Курс')
        group = request.form.get('Група')
        phone = request.form.get('Номер_телефону')

       Перевірка
        if not last_name or not first_name:

            return "Помилка: Прізвище та Ім'я є обов'язковими для
заповнення!", 400

        # Створення об'єкта класу Student
        new_student = Student(
            Прізвище=last_name,
            Імя=first_name,
            По_батькові=middle_name,
            Факультет=faculty,
            Курс=int(course) if course else None,
            Група=group,
            Номер_телефону=phone
        )
```

```
        # Додавання запису
        db.session.add(new_student)

        # збереження змін у БД
        db.session.commit()


        return redirect('/students')

    except Exception as e:
        # Якщо сталася помилка
        db.session.rollback()
        return f"Сталася помилка при збереженні в базу даних: {str(e)}",
500
```

Рис.29

2. Алгоритм авторизації, блок-схема зображена на рис.30

- початок – користувач заходить на сторінку входу;
- користувач вводить логін та пароль;
- зчитуються введені данні з полів;
- перевірка чи всі поля заповнені:

– Ні: Виведення повідомлення про помилку;

– Так: обидва поля заповнені, система переходить до роботи з базою даних;

- пошук користувачі в БД – програма робить запит;
- перевірка пароля;
- перевірка чи користувач є:

– Ні: Виведення повідомлення про помилку;

– Так: користувача знайдено і пароль правильний;

- перехід на початкову сторінку системи
- процес завершено.

Рис.30 Блок-схема авторизації користувача

Маршрут авторизації зображеного на рис.31 – програма отримує дані з полів, перевірка системою користувача, і якщо все правильно, то йде перенаправлення на відповідну сторінку.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        # Отримуємо дані з полів name="login" та name="password"
        login_val = request.form.get('login')
        password_val = request.form.get('password')

        user = User.query.filter_by(login=login_val).first()
```

```
        if user and user.password == password_val:
            login_user(user)

            if user.role == 'guard':
                return redirect(url_for('index'))
            else:
                return redirect(url_for('index_vaht'))
        else:
            flash('Невірний логін або пароль', 'danger')

    return render_template('login.html')
```

Рис.31

Опис моделі для вибірки даних з БД класу User за допомогою SQLAlchemy ORM зображеного на рис.32.

```
class User(db.Model, UserMixin):
    __tablename__ = 'користувачі'

    id = db.Column('id_користувач', db.Integer, primary_key=True,
autoincrement=True)
    login = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(255), nullable=False)
    full_name = db.Column(db.String(100))
    email = db.Column(db.String(100), nullable=False)
    role = db.Column(db.Enum('warden', 'guard', name='user_roles'),
nullable=False)

    # Метод для перевірки ролі
    def is_admin(self):
        return self.role == 'guard'
    def is_warden(self):
        return self.role == 'warden'
```

Рис.32

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Системне тестування – виконується для первірки програмного забезпечення в цілому [20]. Таке тестування проводиться на всій системі. Використовується для тестування функціональних і нефункціональних аспектів.

Для перевірки чи коректно працює система зв’язки з базою даних потрібно проводити тестування системи. Та перевірки чи відповідає система вимогам. Виявляються помилки- в логіці, безпеці, або продуктивності(довго завантажується сторінка). Тестуються зовнішні функції програмного забезпечення. Тобто саме з точки зору користувача.

Що перевіряється під час такого тестування:

1. Функціональність

Перевіряється правильність роботи всіх основних функцій програмного забезпечення. Основною метою є визначення того, чи коректно працює система в цілому та чи відповідає функціонал поставленим вимогам.

На відмінно від попереднього тестування де перевіряється логіка внутрішнього коду і того, як різні модулі інтегруються між собою, системне тестування дозволяє оцінити роботу всієї інформаційної системи вцілому.

2. Інтеграція

Системне тестування також використовується для перевірки взаємодії між окремими компонентами програмної системи. Під час тестування оцінюється коректність обміну даними між модулями та стабільність роботи. Також можна перевіряти взаємодію системи із зовнішніми сервісами, базою даних, програмними компонентами.

3. Перевірка очікуваних результатів

У процесі тестування програмне забезпечення використовується так як і під час роботи з користувачами. Це дозволяє оцінити, чи відповідають результати роботи системи очікуваним значенням.

4.Баги та помилки

Системне тестування дозволяє виявити помилки, збої та проблеми продуктивності, які можуть виникати під час роботи програмного забезпечення. Також перевіряється стабільність роботи розроблюваної системи в різних середовищах і умовах експлуатації.

1. Авторизація та ролі

Успішний сценарій, зображений на рис.33-36:

- користувач вводить логін та пароль;
- виконується перевірка – все правильно, користувач знайдений;
- користувач перенаправляється на головну сторінку системи.

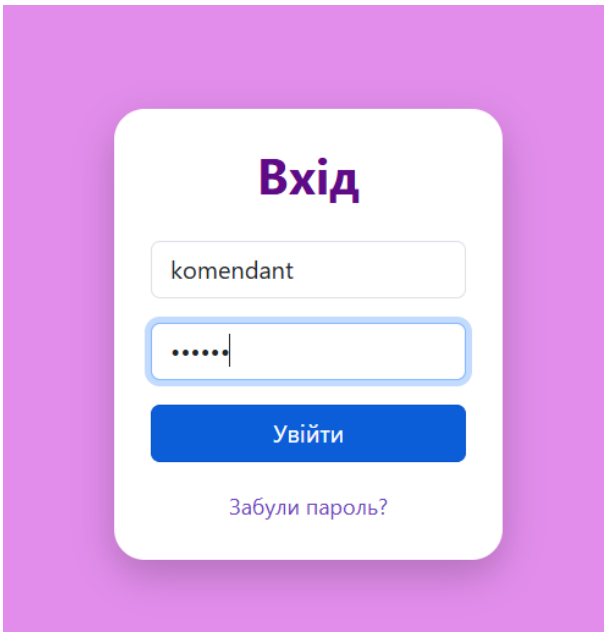


Рис.33 Вікно авторизації

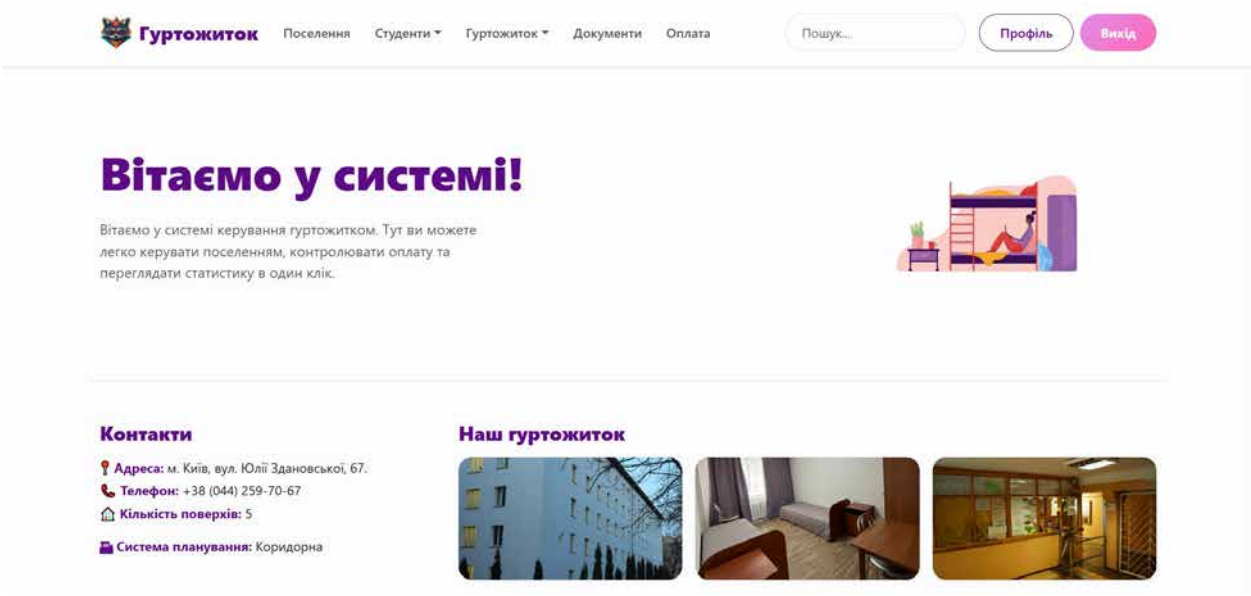


Рис.34 Вікно головної сторінки для коменданта

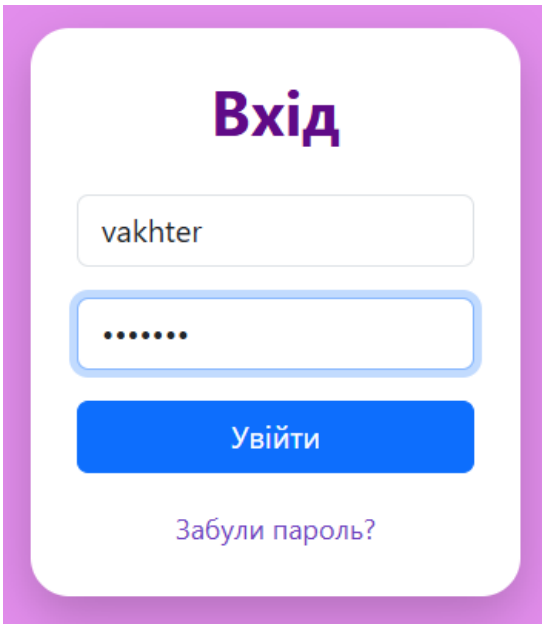


Рис.35 Вікно авторизації вахтера

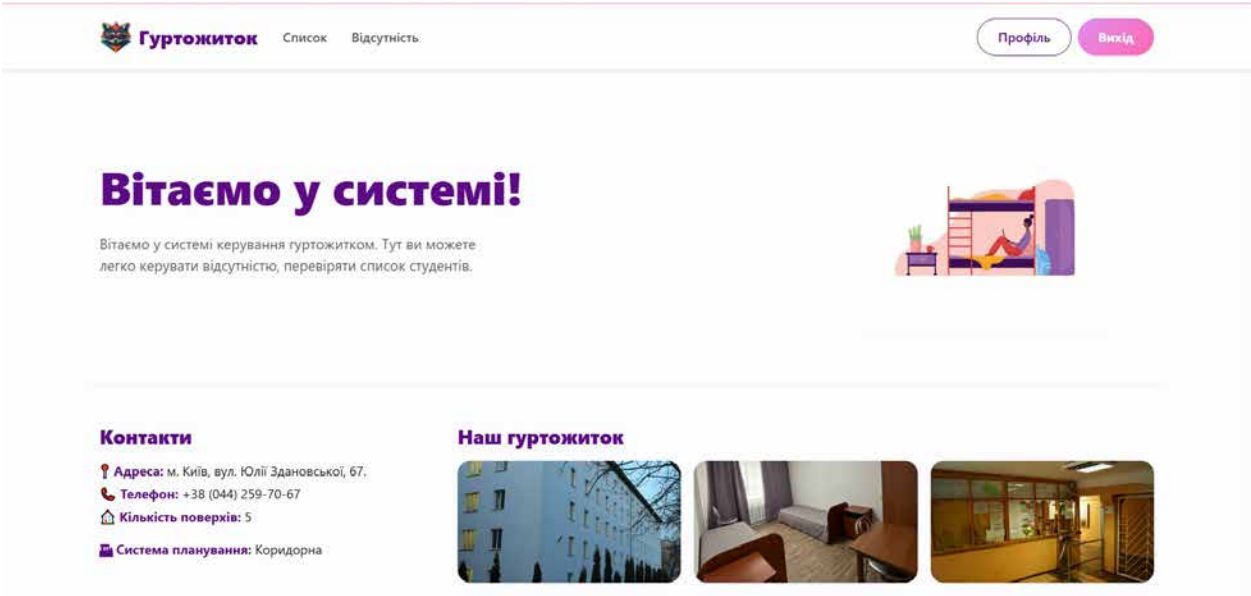


Рис.36 Вікно головної сторінки для коменданта

Якщо користувач ввів неправильний логін або пароль то висвічується повідомлення про помилку, приклад зображено на рис.37.

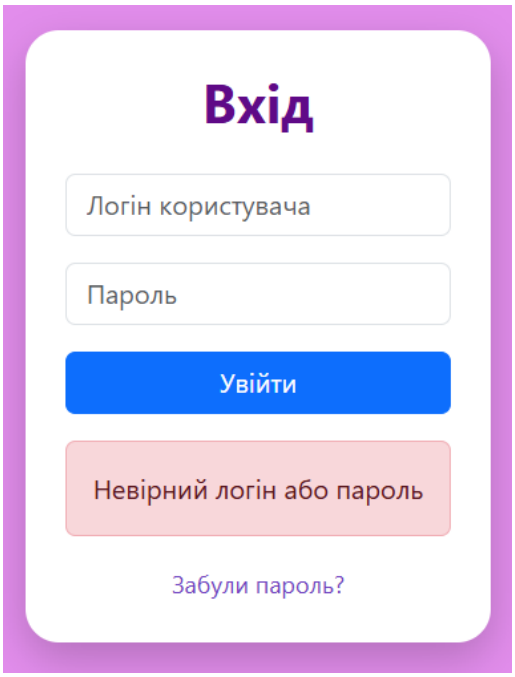


Рис.37 Вікно авторизації з помилкою

2. Валідація даних

На сторінці вахтера – фіксує відсутність студентів.

Успішний сценарій, приклад зображено на рис.38-40:

- вводяться данні у форму;

- у системі фіксується відсутність і одразу з’являється у табличці на сторінці після оновлення.

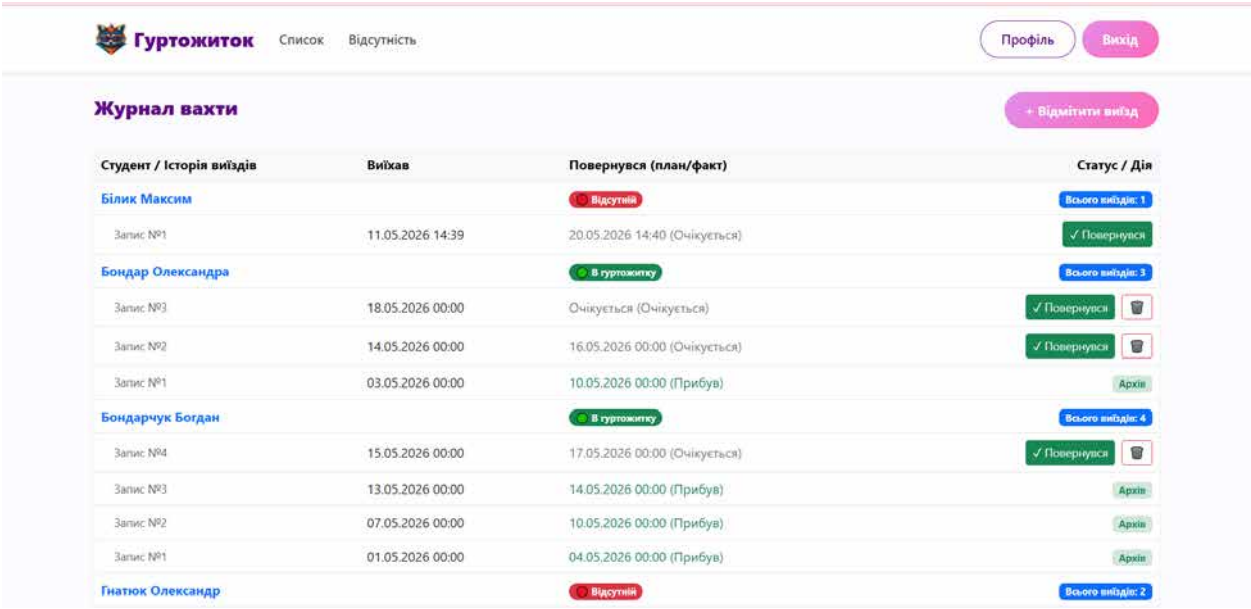


Рис.38 Вікно запису відсутності студента

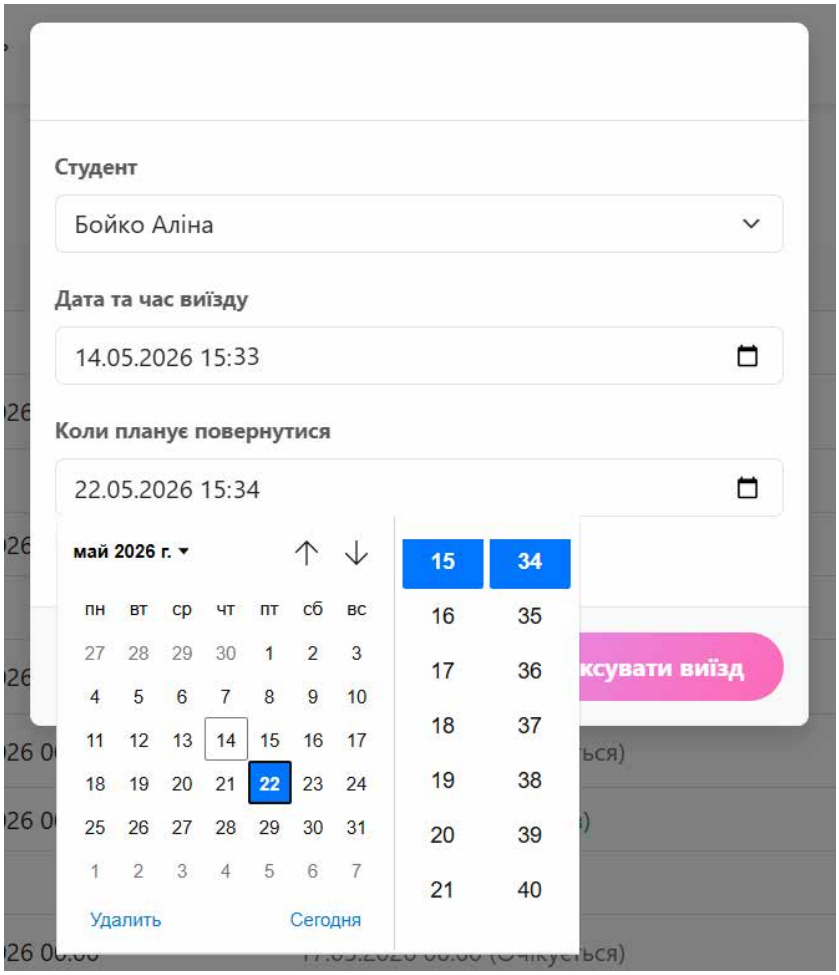


Рис.39 Форма для запису відсутності мешканця

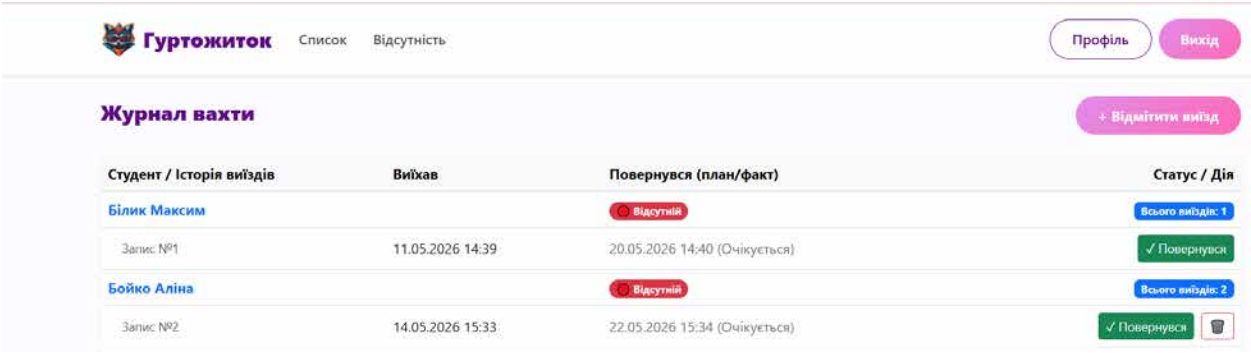


Рис.40 Вікно після запису відсутності студента

Якщо некоректно введені дані то висвічується відповідне повідомлення, приклад зображено на рис.41.

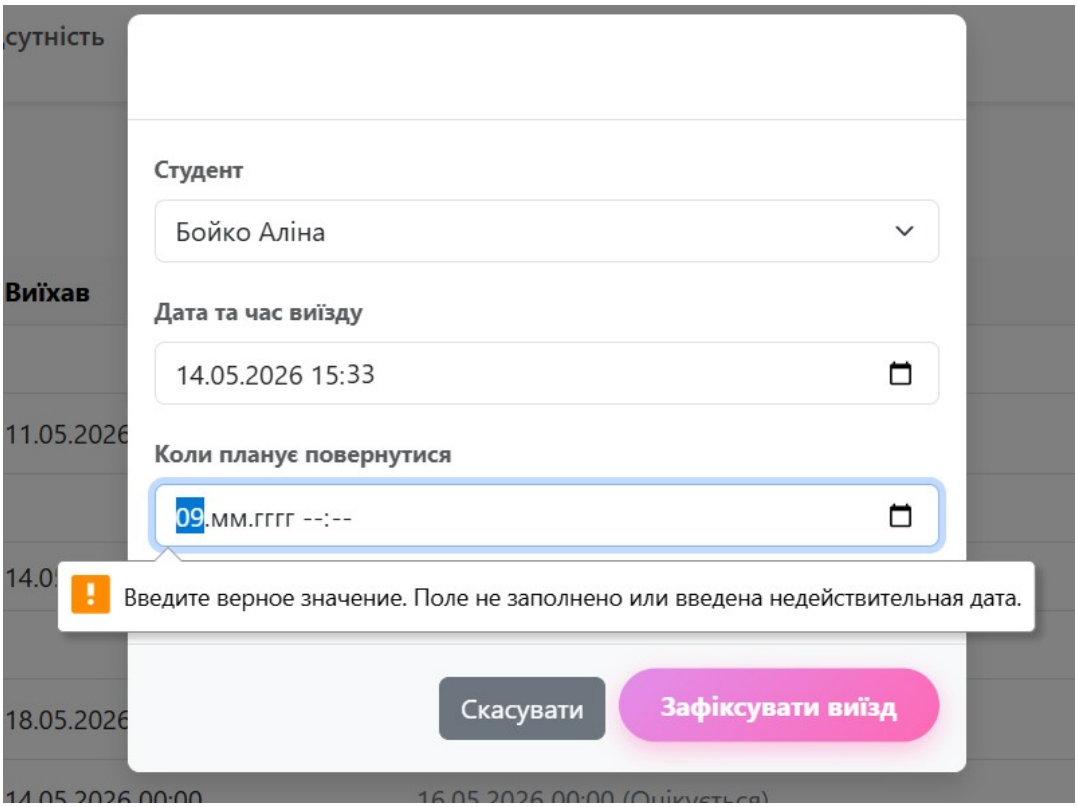


Рис.41 Помилка під час введення даних у форму

3. Перевірка сторінки Статистика

Приклад зображено на рис.42-43 – чи вся статистика виводиться коректно. При успішному сценарії виводиться – кількість поселених студентів, кількість вільних місць, діаграма розподілу студентів за факультетами, діаграма заповненості, та динаміка відсутності.

Якщо виникає помилка – то виводиться відповідне повідомлення – про відсутність зв’язку або даних в БД.

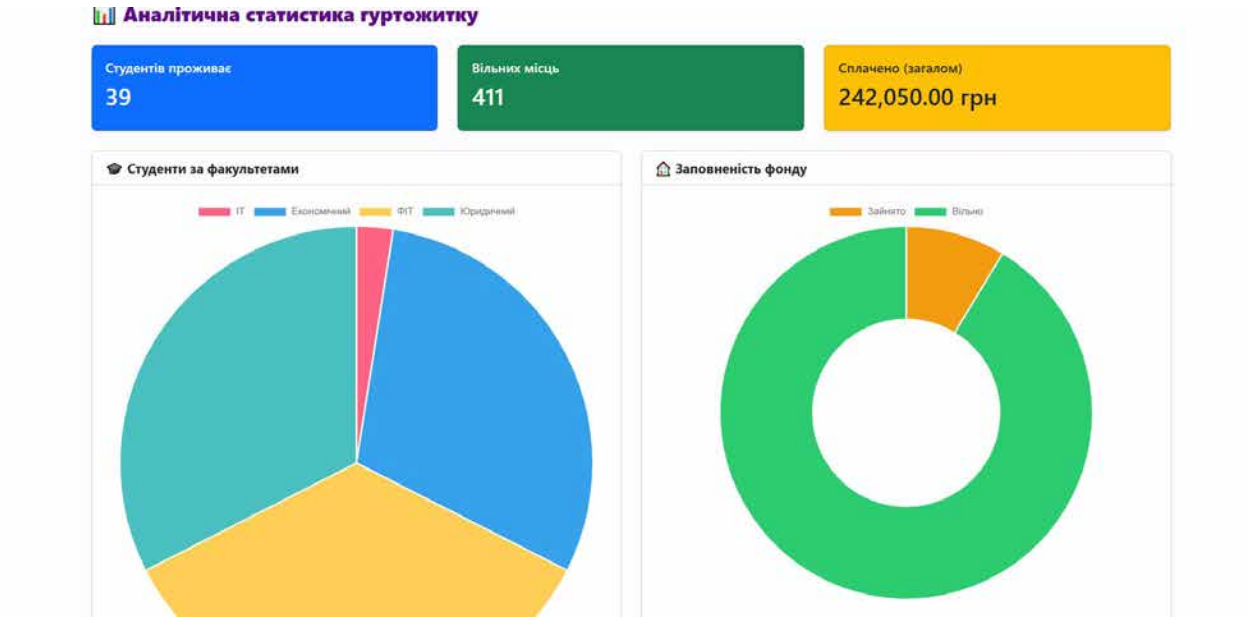


Рис.42 Сторінка статистики

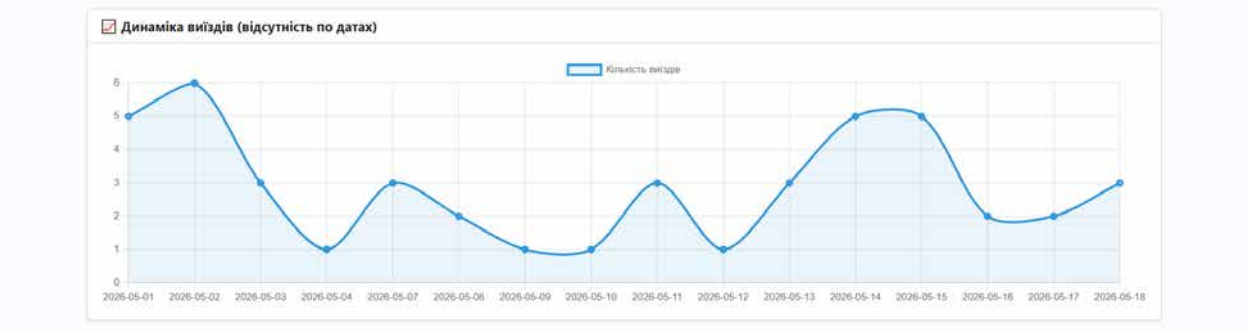


Рис.43 Діаграма відсутності студентів

4. Правильність виводу даних про оплату

При успішному сценарії виводиться таблиця з студентами, сумою оплати, чи сплачено, пільги, скільки повинен сплатити студент з урахуванням пільги. Чи є борг. Приклад сторінки зображено на рис.44.

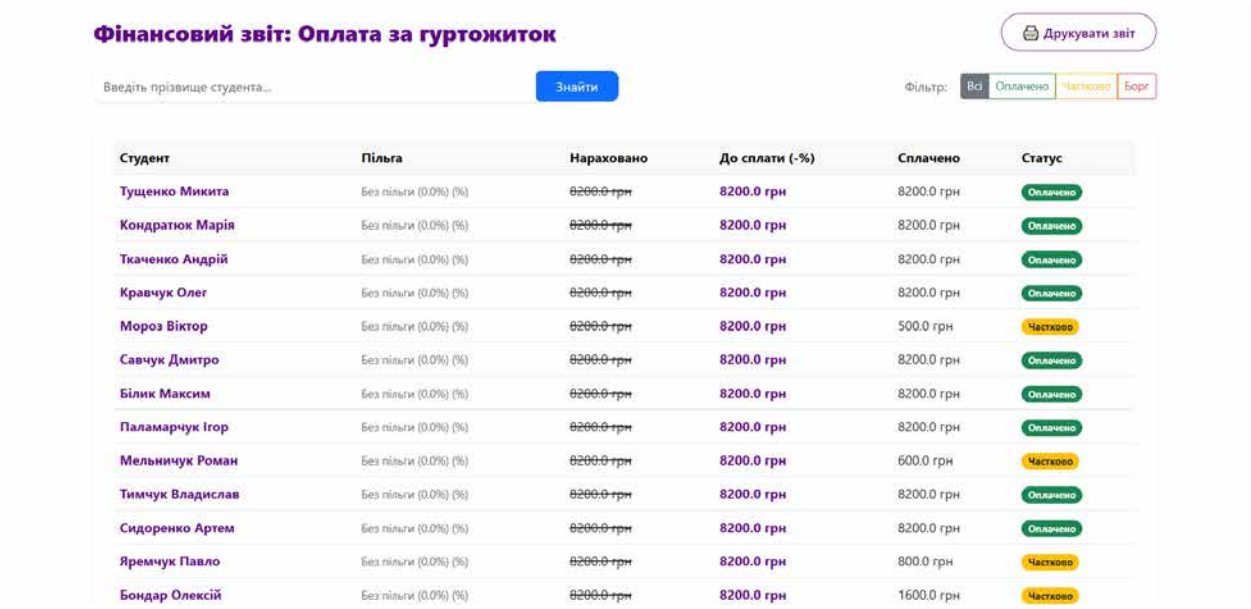


Рис.44 Вікно даних оплати

Також перевірка фільтрів на сторінці. Приклади функціоналу зображено на рис.45-46.

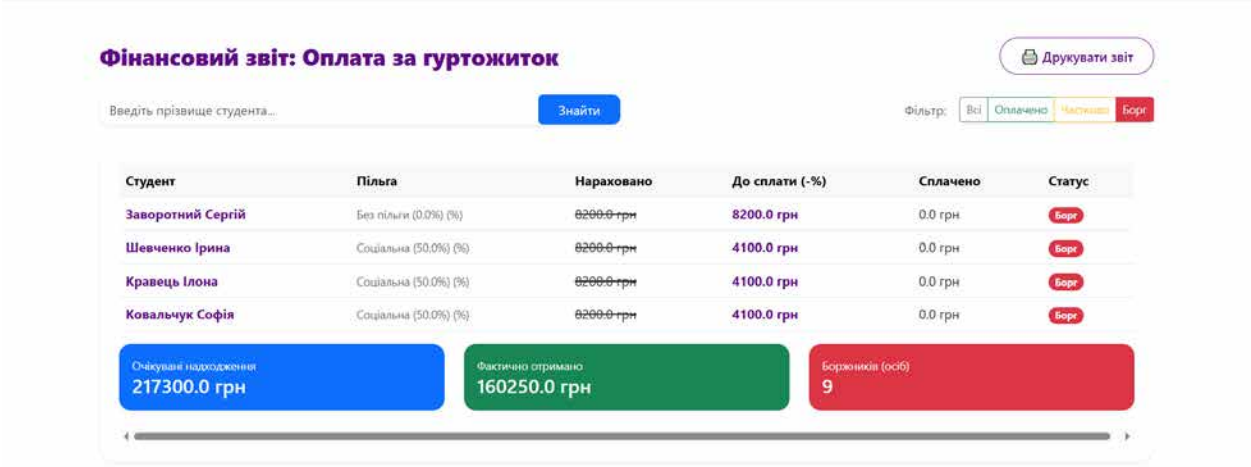


Рис.45 Фільтрація у кого є борг

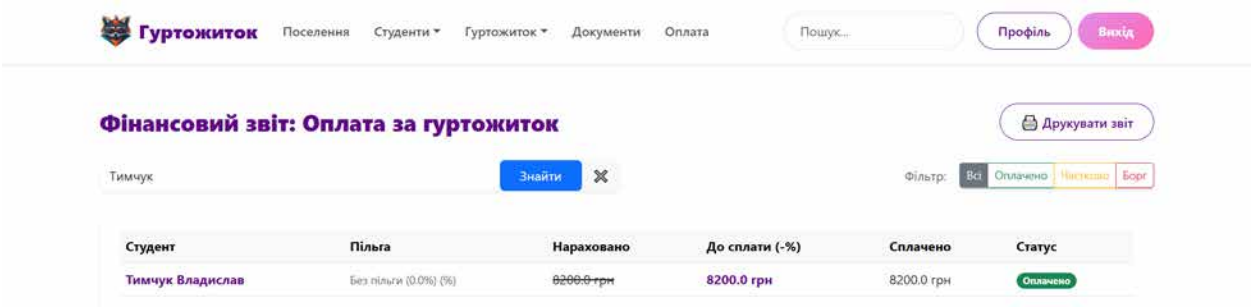


Рис.46 Результат пошуку за прізвищем

4.2 Вимоги до апаратного та програмного забезпечення

Апаратні та програмні вимоги

Апаратні:

1. Будь-який пристрій з доступом до мережі: комп’ютер, планшет або смартфон.
2. Операційна система – Windows 11/10, macOS(дві останні версії), Android(дві останні версії).
3. Процесор (CPU): мінімум 1 ГГц (або 1 core для VPS).
4. Оперативна пам'ять (RAM): для клієнта достатньо 2 ГБ, але для сервера краще вказати хоча б 512 МБ – 1 ГБ

Програмні:

1. Операційна система: починаючи з Windows 10, MacOS, або дистрибутиви Linux.
2. Браузер: Google Chrome, Edge.

4.3 Склад інсталяційного пакету

Для коректного встановлення та роботи з програмним продуктом потрібні певні файли та пакети які містять весь необхідний програмний код та бібліотеки для коректної роботи розроблюваної системи.

Таблиця 2.

Опис складу інсталяційного пакету

Назва	**Тип**	**Опис**
app.py	Python Script	Ініціалізація додатка, містить логіку маршрутизації, обробку запитів та бізнес логіку
models.py	SQLAlchemy ORM	Опис структури бази даних у вигляді класів. Забезпечує зв’язок з таблицями БД
templates/	HTML	Шаблони сторінок – динамічне відображення даних користувачу
static/	CSS, JS	Папка з стилями (дизайн, верстка) та скрипти для інтерактивності
image/	PNG, JPG	Папка з графічними елементами
requirements.txt	Config File	Перелік усіх необхідних бібліотек
Bootstrap 5	UI Library	Фронтенд-фреймворк
.venv/	Virtual Environment	Папка віртуального середовища
__pycache__/	Cache Directory	Папка з скомпільованими файлами

ВИСНОВКИ

За результатами виконання бакалаврської кваліфікаційної роботи було виконано розробку інформаційної системи обліку мешканців гуртожитку. Розроблена система призначена для автоматизації обліку студентів, кімнат, документації для поселення, процесів поселення та виселення студентів, контролю оплати за проживання, формування звітності та статистики.

Під час виконання роботи було проведено аналіз предметної області – за результатами визначено основні функції для системи яка розроблялась. Також проведено дослідження аналогів до інформаційної системи: StarRez, HostelBuddy, E-Hostel Management та інформаційні системи університетських гуртожитків. Проведено аналіз функціональних можливостей аналогів, інтерфейсу користувача та особливостей навігації. На основі аналізу визначено основні функціональні вимоги до системи яка розробляється.

У ході розробки було створено логічну модель бази даних, реалізовано програмний продукт з використанням сучасних вебтехнологій. Для реалізації програмного забезпечення використано мову програмування Python, систему управління базами даних MySQL, для створення користувацького університету використано HTML, CSS, JavaScript та Bootstrap. Для моделювання та алгоритмізації предметної області та роботи окремих модулів системи було створено ряд UML-діаграм.

У результаті система забезпечує авторизацію користувачів, ведення обліку мешканців гуртожитку, управління поселення та виселенням студентів, контроль кімнат та вільних місць, облік оплати за проживання, облік документів. Розроблена система пройшла тестування основних функціональних можливостей. У результаті тестування система і всі програмні модулі працюють коректно.

Практично дану систему можна використовувати в гуртожитках закладів освіти для зберігання інформації та автоматизації процесів. У подальшому можливе розширення функціоналу – розробка особистого кабінету студента, впровадження функцій для контролю діяльності в гуртожитках – чергування, інформаційні оголошення, організація системи обліку користуванням пральними машинами, реалізація функції надсилання повідомлень мешканцям гуртожитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Понад 21 тисяча іноземців здобувають освіту в українських ЗВО: https://www.education.ua/news/2026/02/18/ponad-21-tysiacha-inozemtsiv-zdobuvaiut-osvitu-v-ukrainskykh-zvo/

2. Радіо Трек: НОВИНИ. Джерело: https://radiotrek.rv.ua/news/studentiv-skoro-ne-zalishitsya-u-mon-nazvali-realni-cifri-nedoboru_351088.html

3. Процес моделювання предметної області. Гайд для бізнес-аналітиків: https://dou.ua/forums/topic/42366/

4. Побудова та розуміння алгоритмів: крок за кроком для новачків: https://cloud.itstep.org/blog/building-and-understanding-algorithms-a-step-by-step-guide-for-beginners

5. інформаційні системи, їх види; апаратне та програмне забезпечення інформаційної системи. : https://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/95/95.html

6. Поняття та зміст інформаційного забезпечення (ІЗ) ІС обліку: https://buklib.net/books/23290/

7. МЕТОДИКА ФОРМУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ОПЛАТИ ПРАЦІ У АГРАРНИХ ПІДПРИЄМСТВАХ: http://www.economy.nayka.com.ua/?op=1&z=5578

8. Інформаційне забезпечення: https://studfile.net/preview/7789985/page:6/

9. Логічна модель даних: https://data-life-ua.com/db/lohichna-model-danykh/

10. Системи управління базами даних для малого та середнього бізнесу: як вибрати Джерело: https://hub.kyivstar.ua/articles/sistemi-upravlinnya-bazami-danih-dlya-malogo-ta-serednogo-biznesu-yak-vibrati

11. СУБД: що це таке й навіщо вона потрібна: https://goit.global/ua/articles/subd-shcho-tse-y-navishcho-potribna/

12. Що таке Visual Studio Code: https://top-keis.com.ua/shho-take-visual-studio-code/

13. Що таке мова програмування Python?: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/

14. Основні HTML теги: https://training.qatestlab.com/blog/technical-articles/basic-html-tags/

15. Базові поняття мови розмітки гіпертексту, структура HTML-документа, форматування символів і тексту.: https://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/132/132.html

16. Що таке CSS: https://css.in.ua/article/shcho-take-html_10

17. Вступ до JavaScript: https://uk.javascript.info/intro

18. Що таке JavaScript і для чого він потрібен: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/

19. Bootstrap: https://uk.wikipedia.org/wiki/Bootstrap

20. Що таке системне тестування? Глибоке занурення в підходи, типи, інструменти, поради та підказки і багато іншого!: https://www.zaptest.com/uk/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B5-%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B3%D0%BB%D0%B8%D0%B1%D0%BE%D0%BA%D0%B5-%D0%B7

21. Що таке СУБД і для чого вони потрібні: https://foxminded.ua/systema-upravlinnia-bazamy-danykh/

22. Components of a Database Management System: https://www.dataentryoutsourced.com/blog/components-of-a-database-management-system/

23. Одеській міжнародній академії: https://oiatest.space/studentam/

24. Студмістечко КПІ ім. Ігоря Сікорського: https://studmisto.kpi.ua/

25. HostelBuddy: https://www.hostelbuddy.in/

26. StarRez: https://www.starrez.com/

27. Hostel Management System: https://play.google.com/store/apps/details?hl=uk&id=com.EHostel.app&utm_source=chatgpt.com

28. Голуб Б. , Руденський Р. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ПІДГОТОВКИ ТА ОФОРМЛЕННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ для студентів, що навчаються за освітньо-професійною програмою «КОМП'ЮТЕРНІ НАУКИ» (СПЕЦІАЛЬНІСТЬ 122 КОМП'ЮТЕРНІ НАУКИ») - 2026. – С.36

Додаток А

Структура бази даних SQL-запити

```
USE bdstudent;

-- Таблиця Пільга

CREATE TABLE Пільга (

   id_Пільга CHAR(18) NOT NULL,

   Назва VARCHAR(50) NOT NULL,

   Відсоток_знижки DECIMAL(5,2) NOT NULL,

   Документ VARCHAR(100) NOT NULL,

   Термін_дії DATETIME NOT NULL,

   PRIMARY KEY (id_Пільга)

);



-- Таблиця Студент

CREATE TABLE Студент (

   id_Студент CHAR(18) NOT NULL,

   Імя VARCHAR(50) NOT NULL,

   Прізвище VARCHAR(50) NOT NULL,

   По_батькові VARCHAR(50) NOT NULL,

   Дата_народження DATETIME NOT NULL,
```

```
    Стать VARCHAR(10) NOT NULL,
    Факультет VARCHAR(100) NOT NULL,
    Курс INT NOT NULL,
    Група VARCHAR(20) NOT NULL,
    Номер_телефону VARCHAR(20) NOT NULL,
    Email VARCHAR(50) NOT NULL,
    Номер_студ_квитка INT NOT NULL,
    id_Пільга CHAR(18),
    PRIMARY KEY (id_Студент),
    FOREIGN KEY (id_Пільга) REFERENCES Пільга(id_Пільга)
);

-- Таблиця Кімната
CREATE TABLE Кімната (
    id_Кімната CHAR(18) NOT NULL,
    Номер_кімнати INT NOT NULL,
    Поверх INT NOT NULL,
    Кількість_місць INT NOT NULL,
    PRIMARY KEY (id_Кімната)
```

```
);


-- Таблиця Поселення
CREATE TABLE Поселення (
    id_Поселення CHAR(18) NOT NULL,
    id_Студент CHAR(18),
    id_Кімната CHAR(18),
    Дата_поселення DATETIME NOT NULL,
    Дата_виселення DATETIME NOT NULL,
    PRIMARY KEY (id_Поселення),
    FOREIGN KEY (id_Студент) REFERENCES Студент(id_Студент),
    FOREIGN KEY (id_Кімната) REFERENCES Кімната(id_Кімната)
);


-- Таблиця Оплата
CREATE TABLE Оплата (
    id_Оплата CHAR(18) NOT NULL,
    Сума_нарахована DECIMAL(10,2) NOT NULL,
    Період_оплати VARCHAR(20) NOT NULL,
```

```
    Дата_оплати DATETIME NOT NULL,
    Сума_сплачена DECIMAL(10,2) NOT NULL,
    id_Студент CHAR(18) NOT NULL,
    PRIMARY KEY (id_Оплата),
    FOREIGN KEY (id_Студент) REFERENCES Студент(id_Студент)
);

-- Таблиця Документ
CREATE TABLE Документ (
    id_Документ CHAR(18) NOT NULL,
    Назва VARCHAR(100) NOT NULL,
    Файл VARCHAR(100) NOT NULL,
    Дата_додавання DATETIME NOT NULL,
    id_Студент CHAR(18),
    PRIMARY KEY (id_Документ),
    FOREIGN KEY (id_Студент) REFERENCES Студент(id_Студент)
);

-- Таблиця Відсутність
```

```
CREATE TABLE Відсутність (
    id_Відсутність CHAR(18) NOT NULL,
    Дата_виїзду DATETIME NOT NULL,
    Дата_повернення DATETIME NOT NULL,
    id_Студент CHAR(18),
    PRIMARY KEY (id_Відсутність),
    FOREIGN KEY (id_Студент) REFERENCES Студент(id_Студент)
);

-- Таблиця Інвентар
CREATE TABLE Інвентар (
    id_Інвентар CHAR(18) NOT NULL,
    Назва VARCHAR(100) NOT NULL,
    Стан_при_видачі VARCHAR(50) NOT NULL,
    Дата_видачі DATETIME NOT NULL,
    Дата_повернення DATETIME NOT NULL,
    id_Студент CHAR(18),
    PRIMARY KEY (id_Інвентар),
    FOREIGN KEY (id_Студент) REFERENCES Студент(id_Студент),
```

```
);

-- Таблиця Користувач

CREATE TABLE Користувач (

    id_ Користувач CHAR(18) NOT NULL,

    Логін VARCHAR(50) NOT NULL,

    Пароль VARCHAR(50) NOT NULL,

    Ім’я VARCHAR(100) NOT NULL,

    Email VARCHAR(100) NOT NULL,

    Роль VARCHAR(100) NOT NULL,

    PRIMARY KEY (id_Користувач)

);
```

Додаток Б

SQLAlchemy models

```
from flask_sqlalchemy import SQLAlchemy

from flask_login import UserMixin

from datetime import datetime


db = SQLAlchemy()


class Student(db.Model):

    __tablename__ = 'Студент'


    id_Студент = db.Column(db.Integer, primary_key=True)

    Імя = db.Column(db.String(50), nullable=False)

    Прізвище = db.Column(db.String(50), nullable=False)

    По_батькові = db.Column(db.String(50))

    Дата_народження = db.Column(db.Date)

    Стать = db.Column(db.String(10))

    Факультет = db.Column(db.String(100))

    Курс = db.Column(db.Integer)

    Група = db.Column(db.String(20))
```

```
        Номер_телефону = db.Column(db.String(20))
        Email = db.Column(db.String(100))
        Номер_студ_квитка = db.Column(db.String(20))
        id_Пільга = db.Column(db.Integer)

        def __repr__(self):
            return f'<Student {self.Прізвище} {self.Імя}>'

    class Payment(db.Model):
        __tablename__ = 'оплата'

        id_Оплата = db.Column(db.String(18), primary_key=True)
        Сума_нарахована = db.Column(db.Numeric(10, 2))
        Період_оплати = db.Column(db.String(20))
        Дата_оплати = db.Column(db.DateTime)
        Сума_сплачена = db.Column(db.Numeric(10, 2))
        id_Студент = db.Column(db.String(18),
db.ForeignKey('Студент.id_Студент'))
```

```
class Privilege(db.Model):
    __tablename__ = 'пільга'
    id_Пільга = db.Column(db.String(18), primary_key=True)
    Назва = db.Column(db.String(100))
    Відсоток_знижки = db.Column(db.Numeric(5, 2))

class Rooms(db.Model):
    __tablename__ = 'кімната'
    id_Кімната = db.Column(db.String(18), primary_key=True)
    Номер_кімнати = db.Column(db.Integer)
    Поверх = db.Column(db.Integer)
    Кількість_місць = db.Column(db.Integer)

   occupants = db.relationship('Settlement', backref='room_info', lazy=True, overlaps="room,settlements")

class Settlement(db.Model):
    __tablename__ = 'поселення'

    id_Поселення = db.Column(db.String(18), primary_key=True)
```

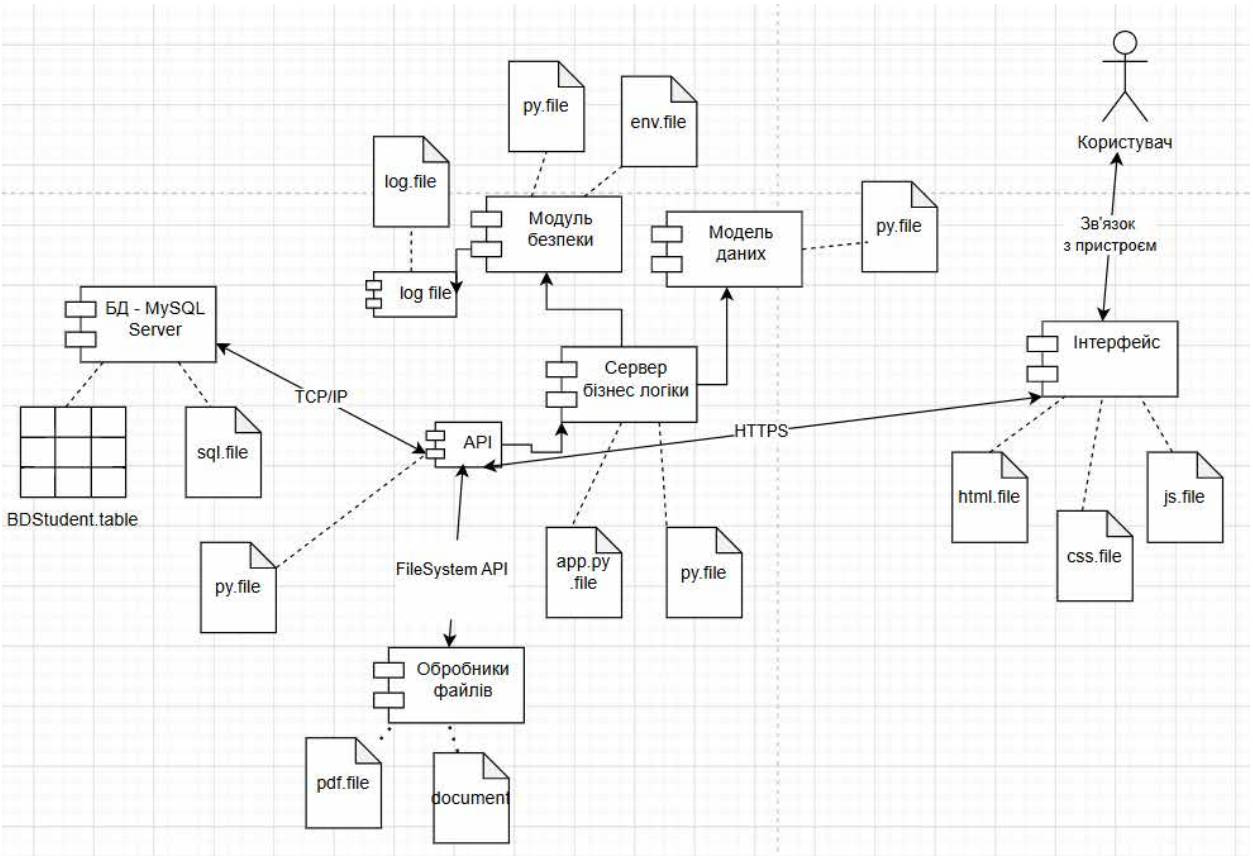
```
        id_Студент = db.Column(db.Integer,
db.ForeignKey('Студент.id_Студент'))

        id_Кімната = db.Column(db.String(18),
db.ForeignKey('кімната.id_Кімната'))

        Дата_поселення = db.Column(db.DateTime, default=datetime.utcnow)

        Дата_виселення = db.Column(db.DateTime, nullable=True)

        room = db.relationship('Rooms', backref='settlements',
overlaps="occupants,room_info")

        student = db.relationship('Student', backref='settlements_list')


    class Absence(db.Model):

        __tablename__ = 'відсутність'


        id_Відсутність = db.Column(db.String(18), primary_key=True)

        Дата_виїзду = db.Column(db.DateTime, nullable=False)

        Дата_повернення = db.Column(db.DateTime, nullable=True)

        id_Студент = db.Column(db.String(18),
db.ForeignKey('Студент.id_Студент'), nullable=False)


        student = db.relationship('Student', backref='absences')
```

```
        def __repr__(self):
            return f'<Absence Student ID: {self.id_Студент} from
{self.Дата_виїзду}>'


    class Inventory(db.Model):
        __tablename__ = 'інвентар'

        id_інвентар = db.Column(db.String(18), primary_key=True)
        Назва = db.Column(db.String(100), nullable=False)
        Стан = db.Column(db.String(50))
        Дата_видачі = db.Column(db.DateTime)
        Дата_повернення = db.Column(db.DateTime, nullable=True)
        id_Студент = db.Column(db.String(18),
db.ForeignKey('Студент.id_Студент'))

        student = db.relationship('Student', backref='items')

        def __repr__(self):
            return f'<Inventory {self.Назва} - {self.Стан}>'
```

```
class User(db.Model, UserMixin):

    __tablename__ = 'користувачі'


    id = db.Column('id_користувач', db.Integer, primary_key=True, autoincrement=True)

    login = db.Column(db.String(50), unique=True, nullable=False)

    password = db.Column(db.String(255), nullable=False)

    full_name = db.Column(db.String(100))

    email = db.Column(db.String(100), nullable=False)

    role = db.Column(db.Enum('warden', 'guard', name='user_roles'), nullable=False)



    def is_admin(self):

        return self.role == 'guard '


    def is_warden(self):

        return self.role == 'warden'
```

Додаток В

Діаграма компонентів



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Карпова Марія Олександрівна, здобувачка НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Інформаційна система обліку мешканців гуртожитку» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [✓] інструменти ШІ ChatGPT, Gemini були використані для:
        - [] перекладу іншомовних джерел;
        - [✓] стилістичного редагування та перевірки граматики;
        - [✓] допомоги у написанні/відлагодженні програмного коду;

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Марія КАРПОВА**
(підпис)