Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка програмного засобу оптимізації обліку житла на мові C# в середовищі MVS»

Виконав(ла): студент(ка) IV курсу, групи СП-41
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

	(підпис)	Гаврилюк І.О. (прізвище та ініціали)
Керівник	(підпис)	Цуприк Г.Б. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	Мацюк О.В (прізвище та ініціали)

Тернопіль 2023

РЕФЕРАТ

Кваліфікаційна робота бакалавра містить: с., рис., табл., джер.

предмента область, мета, завдання, об’єкт, розробка, програмна система, вимоги, середовище програмування, Microsoft Visual Studio, C#, сервер, база даних

За мету кваліфікаційної роботи взято розробку програмного засобу основне призначення якого полягає в оптимізації процесу обліку житла на мові C# в середовищі MVS.

Методи розробки запропоновано реалізувати використавши середовище програмування Microsoft Visual Studio, мову програмування обрано C#, а за сервер бази даних – MySQL.

ABSTRACT

subject field, purpose, task, object, development, software system, requirements, programming environment, Microsoft Visual Studio, C#, server, database

The purpose of the qualification paper is the development of a software tool whose main purpose is to optimize the process of housing accounting and in the C# language in the MVS environment.

The development methods are proposed to be implemented using the Microsoft Visual Studio programming environment, the programming language is C#, and MySQL is the database server.

ЗМІСТ

ВСТУП

Відповідно до Положення про кваліфікаційні роботи студентів ТНТУ ім.І.Пулюя на здобуття першого (бакалаврського) рівня вищої освіти я повинен вирішити якусь спеціалізовану задачу та запропонувати вирішення практичних проблем у котрійсь із галузей професійної діяльності. Обов'язковим моментом є необхідність застосувати певні теорії та методи відповідних наук. Моя робота повинна характеризуватись в цілому комплексністю та невизначеністю умов, відповідати компенетціям та вимогам Стандарту вищої освіти відповідного освітнього рівня.

Першою трудністю з якою я стикнувся на самому початку написання кваліфікаційної роботи було вибір предметної області та окреслення об'єкту дослідження. На жаль, все що планував раніше, сьогодні вважаю не актуальним та навіть недоречним. Отож, добре подумавши і проконсультувавшись з керівником я вирішив, що нічого кращого немає, як працювати на допомогу людям які потрапили в скрутне становище в умовах війни.

Втрата матеріальних цінностей, рухомого та нерухомого майна та реальна загроза позбутися життя та здоров'я примусила людей до міграції буквально із перших днів початку повномасштабного вторгнення. Люди тікали в чому були, без речей, часто без документів та засобів зв'язку з рідними. В мирний час ми плануємо поїздки, а в таких умовах головне було просто виїхати, а місце призначення фактично не мало значення, просто це мала бути відносно спокійна територія. Таким чином люди прибували в стані шоку, розгублені, голодні, в незнайому місцевість, без будь який знайомих чи рідних. Одним словом виїзджали в нікуди. Хоча на місцях їх і зустрічали волонтери та психологи, які разом допомагали, проте, кардинально це ситуацію не міняло. В таких випадках, коли тривалість подорожі могла легко бути не одну добу в умовах напруженості та стресу, в постійній небезпеці обстрілів, важливо було з хаосу людей розмістити в більш-менш спокійну обстановку, погодувати та дати відпочити. Зокрема в

Тернопіль людей прибували сотні, тисячі щодня. В результаті майже одразу виникло питання реального стану справ по вільному житлу. І оскільки квартирний фонд дуже швидко закінчився, а кількість людей лише зростала і тенденція не змінювалась, людей почали розселяти в тому числі і по гуртожиткам. Оскільки динаміка змін була просто шалена. Одні прибували, кілька годин відпочивали і рушали далі на захід, інші зупинялись на довше, ще комусь потрібні були спеціалізовані приміщення з відповідним обладнанням і завжди невідомою була складова кількості людей, періоду проживання, необхідності обміну тимчасового житла на постійне та подібне. Одним словом всіх потреб і проблем не перелічиш, вони видозмінювались та постійно лише доповнювались. Таким чином, в мене вималювалась мета мого дослідження використати сучасні інформаційні технологій на поміч людям, а саме при розробці програмного засобу оптимізації обліку та обміну житла.

Для себе я виокреслив конкретний напрямок дослідження – тимчасове проживання, а об’єктом, який розглянув детально я обрав проблемні питання пов’язані з обліком, обміном та реєстрацією проживаючих у гуртожитку. Не дивлячись на те, що на дворі 21 століття, війна внесла все таки свої корективи і часто такий облік ведеться вручну, письмово, а це, зрозуміло, що однозначно призведе до некоректностей при оформленні та своєчасності заповнення журналів обліку. Саме проблеми такого типу і пропонуються до вирішення шляхом інформатизації та автоматизації даного процесу.

Таким чином за мету кваліфікаційної роботи є створення програмного засобу, який буде достатньо простим та зрозумілим при експлуатації і, що найголовніше, який буде повністю буде вирішувати задачу обліку, обміну та реєстрації проживаючих у гуртожитку, а також пришвидшення та спрощення процесу обліку та поселення.

Реалізація такої програмної системи змогла б значно полегшити працю волонтерів, які відповідають за розселення. Вона дасть змогу і допоможе при зберіганні інформації про мешканців та статус кімнат в одному місці, без необхідності ведення паперових чи будь-яких інших форм чи носіїв.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналізування вимог до програмної системи

В результаті огляду стану справ обраного напрямку дослідження було виявлено проблемні питання та сформульовано завдання, яке полягає в розробці програмної системи з використанням сучасних інформаційних технологій, основною метою якої є облік, реєстрація та контроль на прикладі мешканців гуртожитку [1-7].

Система має передбачати введення нової інформацію, редагування та видалення вже існуючих даних.

З описаної інформації з системою можуть працювати наступні користувачі:

- відповідальні працівники;
- волонтер;
- комендант гуртожитку.

Програмний засіб повинен володіти таким параметром як висока надійність. До всього програмного засобу потрібно застосовувати обмеження стосовно введення данних до полів у таблицях.

Якщо говорити про коректність роботи програми у цілому потрібно, щоби комп’ютер відповідав таким мінімальним вимогам до системи:

- до операційної системи: Microsoft WindowsXP/Vista[8];
- до процесору: AMD Athlon x64 із частотою не меншою 2,8 Ггц, або ж Intel Pentium із частотою не монше ніж 1,8 Ггц[9-11];
- до оперативної пам’яті(ОЗП), не менше ніж512 Мб;
- HDD: 100 Мб вільний дискового простір, щоби програма працювала коректно;
- наявність принтеру, клавіатури та мишки.

Також необхідно, щоби на комп’ютері було передбачено наявність необхідного мінімуму програмних продуктів ,зокрема таких як:

- MicrosoftSQL Server 2012 і вище [13-15];
- Adobe Acrobat Reader (або інша програма для читання файлів з розширенням *.pdf).

В цілому ж програмний продукт має підтримувати коректність та швидке функціонування під управлінням самої операційної системи MicrosoftWindows XP/Vista. Розроблюваний програмний продукт має назву «Розробка програмного засобу оптимізації обліку житла на мові C# в середовищі MVS»» та повинен бути реалізований в середовищі Microsoft Visio Studio [17] на мові C# [18], в якості вибраного СУБД виступає MicrosoftSQL Server 2012 та вище [13-15].

1.1.1 Аналіз предметної області

Програма основною метою якої є фактично збір та зберігання з можливістю доповнення та іншого редагування даних про тимчасово проживаючих осіб є фактично програмною системою призначеноюа для ведення обліку мешканців у гуртожитку. Спосіб ведення обліку може бути доволі різним, наприклад це зберігання інформації в паперовому вигляді, електронні таблиці та програми з прикріпленими БД(базами даних) із зручним графічним інтерфейсом.

Інформаційні технології та й системи в цілому в наш час стали, по суті, чи не єдиним із робочих інструментів призначених саме для вирішення багатьох із складних завдань управління та організації ведення будь-якої діяльності, особливо в наш складний час коли ціною будь-якої помилки, навіть не знаюної, може бути життя та здоров’я і навіть не однієї людини. Саме тому будь-яка серйозна комерційна організація чи волонтерська максимально намагається комп’ютеризувати всі процеси на різних етапах своєї роботи.

Інформаційна база є ні чим іншим як сукупністю даних про проживаючих та кількість мешканців кімнат в гуртожитку. Вся ця інформаційна база є достатньо динамічною і складною в організації: люди поселяються та виселяються,

переміщуються в межах будови, змінюються їхні персональні дані, склад та кількість в кімнатах змінюється кількість ліжок всього та зайнятих, кількість ліжок (тобто вмістимість) в одній і тій самій кімнаті в залежності від інтенсивності прибування тощо.

Таким чином, інформаційна система повинна містити та надавати повну інформацію про мешканців, інформацію про кімнати, мати зручну та зрозумілу навігацію та пошук [19-24].

1.1.2 Формулювання завдань, окреслення та постановка задачі

Насамперед при занесенні нових данних рекомендовано проводити перевірку стосовно введеної інформації на її відповідність. Якісний програмний засіб повинен характеризуватися високою швидкодією, а саме мається на увазі обробка усіх операцій за достатньо короткий проміжок часу.

При створенні обраного програмного засобу потрібно передбачити можливість:

- заносити до БД дані про нових тимчасових мешканців;
- редагувати інформацію про вже існуючих в базі проживаючих;
- видаляти з бази неактуальних осіб;
- здійснювати пошук по базі;
- додавати нові кімнати;
- здійснювати друк інформації про проживаючих та про кімнати у *.pdf файлі;
- видаляти порожні кімнати.

А також підтримувати інший проте не менш важливий функціонал:

- здійснення навігації у рандомному порядку;
- інтерфейс має бути зрозумілим на інтуїтивному рівні;

– важливим моментом є і універсальність коду програми (чим більше код буде універсальним там краще);

– інформація повинна подаватись системою зрозумілою та зручною формою, сприйматись легко;

– в метою побудови інтерфейса користувача повинна застосовуватися бібліотека по типу WindowsForm;

– для інтерфейсу зі сторони користувача повинен характеризуватись побудовою з врахуванням особливостей ергономіки, а також сприйняття інформації людиною, в тому числі без відповідної освіти та потреби в особливих знаннях;

– робота програми функціонувати без збоїв, а також повинна бути передбачена коректна реакція на будь-які дії, будь якого рівня освіти та навиків користувача.

1.1.3 Опис пошуку актантів та варіантів використання

На даний час не існує хорошого та зручного програмного продукту для обліку мешканців гуртожитку, який би відповідам жорстким вимогам сьогодення. Найчастіше гуртожитки в цією метою використовують електронні таблиці, текстові документи та паперові носії для ведення обліку мешканців. Зазвичай список попереднього розселення по кімнатах проєктується заступником(комендантом), директору передається у текстовому форматі документу, та передається для якщо гуртожиток від навчального закладу то у студмістечко, якщо від підприємства – то у відповідний відділ. Видаючи ордери на поселення, вповноважений заносить дані у електронні таблиці. Комендант веде облік на паперових носіях, інформацію про мешканців та в які кімнати їх розселено він отримує із ордерів, які видав вповноважений підрозділ чи особа.

Щоб спростити цей процес обміну даними розробляється програмна система для обліку мешканців у гуртожитку оминаючи всі ці часозатратні проміжні етапи.

Перевагою програми є ведення єдиної бази (даних), що дає змогу користувачам одразу помітити зміни внесені іншими користувачами, та швидкий доступ до інформації про мешканців та кімнати.

Проведемо пошук та виявлення основних акторів, які можуть взаємодіяти з системою:

З системою мають можливість працювати відразу три групи користувачі:

- волонтери;
- відповідальні працівники;
- комендант гуртожитку.

Наступним кроком буде виявлення варіантів використання розробленої системи.

З описаної інформації з системою повинні працювати волонтер, відповідальний працівник і комендант.

Волонтер, при роботі з програмною системою має можливість заносити основну інформацію про особу, що потребує тимчасового прихистку та в яку кімнату його рекомендовано поселити.

Відповідальний працівник при роботі з даною програмною системою повинен мати можливість вирішувати такі види завдань:

- Вносити всі дані про особу;
- Присвоювати потребуючому номер кімнати, в якій він буде мешкати.

Комендант при роботі з програмною системою може редагувати дані про особу та переглядати іншу інформацію.

Результат виконання побудови діаграми варіантів використання представлено на рисунку 1.1.

Рисунок 1.1 – Представлення діаграми варіантів використання

1.2 Проєктування програмної системи

1.2.1 Побудова схеми для бази даних

Для того, щоби була можливість коректної взаємодії із користувачем, варта буде сформувати певну структуру даних, яка б забезпечувала функціональність високого рівня. Та незважаючи на те, що не передбачено, щоби користувач зміг бачити структуру бази даних, проте його робота все рівно буде пов’язана із деякими певними атрибутами [25-26].

Використовуючи вимоги, які було визначено майбутніми користувачами я можу сформулювати різні компоненти концептуальної моделі даних [27-29].

Отже, наступним кроком буде визначення основних типів сутностей та їх зв’язків. В результаті аналізу предметної області та постановки технічного завдання виявлено основні сутності, а саме:

- кімнати;
- поселенці;
- особисті дані;
- контактна особа.

Інформацію про типи сутностей наведено у таблиці 1.1.

Враховуючи всі вимоги висунуті користувачем для кожної зі сутностей, я визначився щодо конкретного переліку атрибутів щодо виявлених головних сутностей у системі. Усі сутності разом із атрибутами і складають опис предметної області. Інформацію щодо атрибутів та сутностей, до яких вони приналежні, наведено у наступній таблиці (див.табл.1.2).

Таблиця 1.1 – Інформація щодо типів сутностей

Ім’я конкретної сутності	Опис сутності	Її псевдонім	Особливості її використання
Кімнати	Описується інформація про кімнати	room	Містить інформацію про кількість місць в кімнаті
Поселенці	Записуються дані про мешканців гуртожитку	occupiers	Містить всі потрібні дані про поселенця
Особисті дані	Інформація про особисті дані мешканця гуртожитку, звідки прибув, за якою спеціальністю працював, освіта,з ким прибув тощо	personal information	Можна дізнатися особисту інформацію про тимчасово переміщеного, щоб поселити його в той блок (кімнату), де йому буде найкомфортніше
Контактна особа	Записується інформація щодо контактної особи	contact_info	Зазвичай контактною особою являється хтось із рідних чи знайомих

Таблиця 1.2 – Описування атрибутів сутностей

Тип конкретної сутності	Атрибут	Тип даних	Обмеже-ння	Псевдонім	Значення NULL
Кімнати	Ключ Автоінкремент	Числовий	Первинний ключ	Id	Ні
	Номер Кімнати	Числовий		Room_number	Ні
	Стан Кімнати	Текстовий		Status	Так
	Номер Блоку	Числовий		Block_noomber	Ні
	Кількість Місць	Числовий		Space	Ні
	Поверх	Числовий		Floor	Ні
Пожильці	Ключ Автоінкремент	Числовий	Первинний ключ	Id	Ні

Продовження таблиці 1.2

Тип сутності	Атрибут	Тип даних	Обмеже-ння	Псевдонім	Значення NULL
	Ім'я пожильця	Текстовий		Name	Ні
	Прізвище пожильця	Текстовий		Surname	Ні
	По Батькові	Текстовий		Full_name	Ні
	Стать	Текстовий		Sex	Ні
	Рік Народження	Дата		Birth_date	Ні
	Ключ Кімнати	Числовий		Room_id	Ні
Особисті дані	Ключ Автоінкремент	Числовий	Первинний ключ	Id	Ні
	Ключ пожильця	Числовий		Occupier_id	Ні
	Вік пожильця	Числовий		Age	Ні
	Статус	Числовий		Status	Ні
	Початок Проживання	Числовий		Start_year	Ні
	Звідки прибув	Текстовий		From	Ні
	Професія/спеціальність	Текстовий		Speciality	Ні
Контак-тна особа	Ключ Автоінкремент	Числовий	Первинний ключ	Id	Ні
	Ключ пожильця	Числовий		Occupier_id	Ні
	Електронна Пошта	Текстовий		Mail	Так
	Номер Мобільного	Числовий		Main_phone	Ні
	Ім’я Контактної Особи	Текстовий		Contact_person __name	Ні
	Номер Контактної Особи	Числовий		Contact_person _ _phone	Ні
	Місце Проживання	Текстовий		Location	Ні

Далі я більш детально описав про призначення окремого атрибута для кожної конкретно визначеної сутності.

Далі варта приділити увагу розгляду атрибутів сутності Кімната, яка є головною сутністю для всієї бази даних. Атрибут «КлючАвтоінкремент» – кожна кімната має свій ідентифікаційний номер, при допомозі якого можна

ідентифікувати кімнати лише за номером Id, без потреби вводу додаткової інформації. Атрибут «Номер кімнати» дозволяє отримати інформацію про фізичний номер кімнати гуртожитку. Атрибут «СтанКімнати» відображає чи є вільні місця в кімнаті, і в разі якщо є, то стан (статус) кімнати буде «вільна», в іншому випадку – «зайнята». Атрибут «НомерБлоку» містить інформацію, в якому блоці розташована кімната. Атрибут «Кількість місць» вказує скільки всього може мешкати пожильців у кімнаті. І останній атрибут цієї сутності «Поверх» відображає, на якому поверсі розташована кімната.

Далі розглянемо опис атрибутів сутності Пожильці. Атрибут «КлючАвтоінкремент» – кожний пожилець має свій ідентифікаційний номер, що дозволяє ідентифікувати його лише за номером Id, без потреби вводу додаткової інформації. Атрибут «Ім'яПожильця» містить ім'я мешканця гуртожитку. Атрибут «ПрізвищеПожильця» відображає прізвище мешканця гуртожитку. Атрибут «ПоБатькові» містить по батькові пожильця. Атрибут «Стать» відображає, до якої статі належить пожилець. Атрибут «РікНародження» вказує дату народження мешканця гуртожитку. Атрибут «КлючКімнати» точно вказує, в яку кімнату поселений пожилець.

Наступна сутність, атрибути, якої потрібно описати це «Особисті дані». Атрибут «КлючАвтоінкремент» присвоює кожному рядку таблиці свій унікальний номер, що запобігає зміщенню інформації в таблиці. Атрибут «КлючПожильця» пов'язує сутність «Особисті дані» із сутністю «Пожильці». Атрибут «ВікПожильця» відображає важливу інформацію, яка дасть змогу комфортно осолити людину, враховуючи її психологічний стан. Атрибут «Статус» показує та визначає пріоритетність у рейтингу на поселення. Атрибут «ПочатокПроживання» містить інформацію, з якого часу пожилець почав жити у даному гуртожитку. Атрибут «Звідки прибув» вказує звідки прибув пожилець (важлива інформація для можливості поселення з краянами, це дасть змогу покращити емоційний та психологічний статус особи). Останній атрибут описуваний у сутності «Особисті дані» це «Професія/спеціальність». Він відображає, яку спеціальність чи де попередньо працював (професія) поселенець. На мою думку це один із найважливіших атрибутів, оскільки володіючи цією

інформацією буде легше адаптувати людину і лишній раз не турбуючи її є можливість допомогти знайти заняття для душі чи роботу при необхідності.

І остання сутність, яка існує в базі даних, це сутність «КонтактнаОсоба». Атрибути цієї сутності наступні: «КлючАвтоінкремент» присвоює кожному рядку таблиці свій унікальний номер, що запобігає зміщенню інформації в таблиці; «Ключ Пожильця» пов’язує сутність «КонтактнаОсоба» із сутністю «Пожильці»; «ЕлектроннаПошта» містить адресу електронної скриньки мешканця гуртожитку; «НомерМобільного» відображає номер мобільного телефону пожильця. Атрибут «Ім’яКонтактноїОсоби» містить інформацію про контактну особу (зазвичай це хтось із рідних чи знайомих). Атрибут «НомерКонтактноїОсоби» відображає номер мобільного телефону контактної особи. І завершую опис сутності атрибутом «МісцеПроживання» - це домашня адреса мешканця гуртожитку.

Таким чином, в результаті я виявив усі основні сутності та описав їх атрибути. Таким чином отримавши чітку картину, який саме вигляд повинні мати основні таблиці в інформаційній системі. Деталізація представлена на рисунку 1.2).

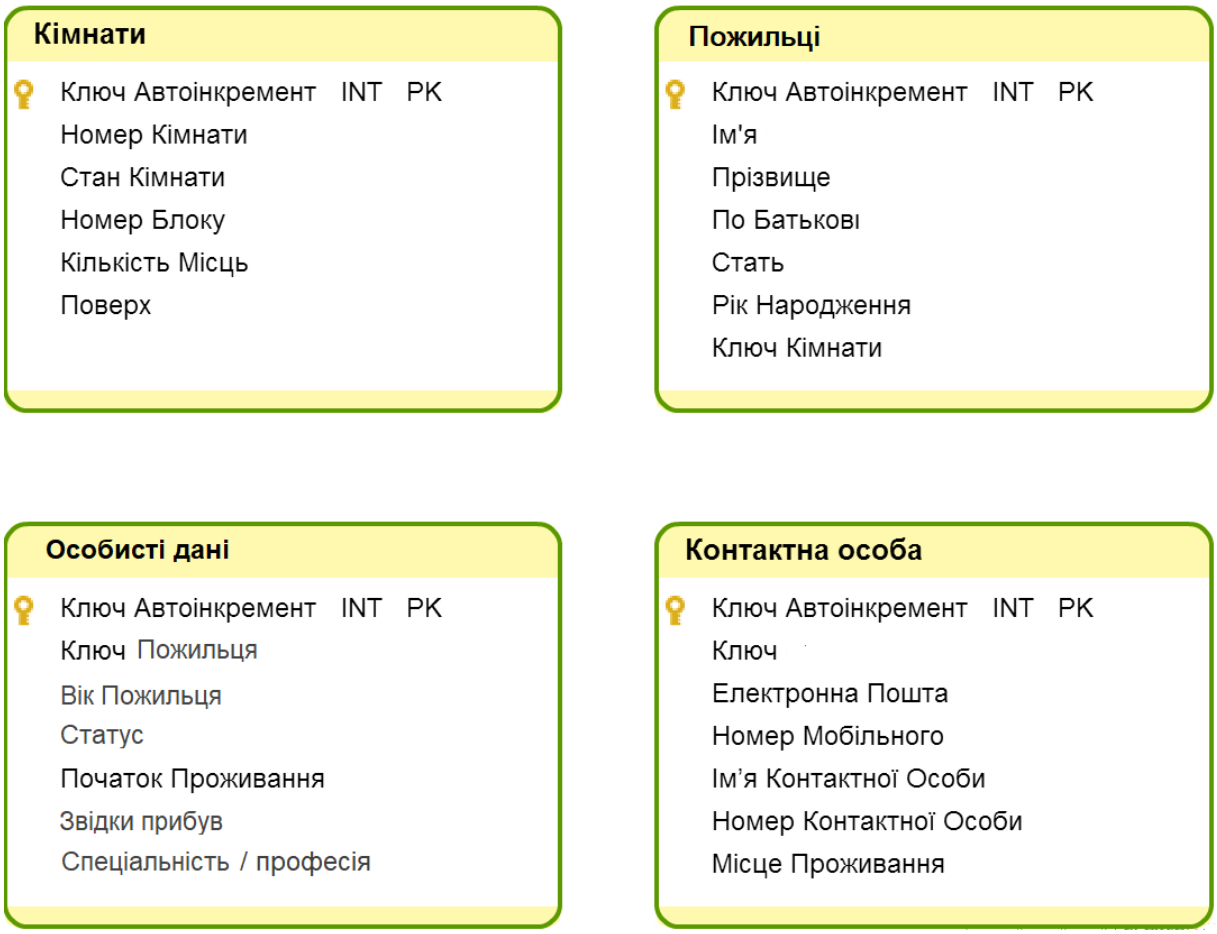

Рисунок 1.2 – Виявлені сутності бази даних після аналізу

Так як всі головні сутності, так само як і їх атрибути у вигляді модулі реляційного типу, було описано та створено наступною дією, яку потрібно здійснити, а саме йдеться про встановленням взаємозв'язків поміж існуючими сутностями системи [4,5].

Першим вибраним зв'язком буде взаємозв'язок між сутностями «Пожильці» та «Кімнати», а саме один до багатьох, оскільки в одній кімнаті може жити стільки мешканців, скільки є місць в кімнаті. Наступний зв'язок має знову ж таки сутність «пожильці» але уже з іншою сутністю «Вік» і це зв'язок один до одного, тому що одному мешканцю належать одна частина інформації про вік. Сутність «Контактна особа» має зв'язок з сутністю «Пожильці», що дає можливість дізнатися дані про одну контактну особу одного конкретного пожильця, тому взаємозв'язок між цими сутностями відповідно один до одного.

В результаті виконаного опису і утворення виявлених головних сутностей, а також їхніх атрибутів, та встановлення зв’язків залежності поміж ними, я отримав схему даних виду як представлено на рисунку 1.3.

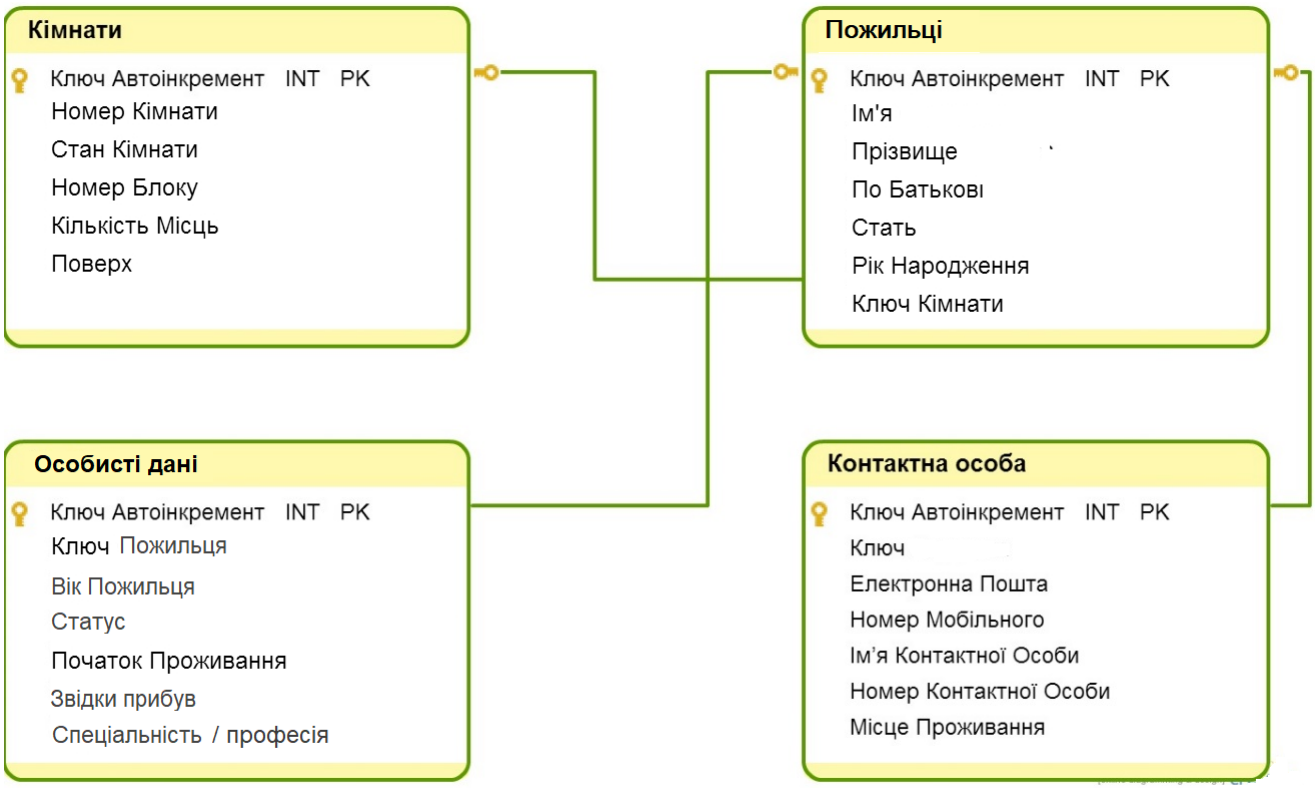

Рисунок 1.3 – Схема даних таблиць в базі даних

Основною сутністю бази даних є сутність Кімнати, вона зберігає інформацію про кімнати, які є у гуртожитку. Структура таблиці Кімнати наведена на рисунку 1.4.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
room_number	int(11)	NO		*NULL*	
status	varchar(45)	YES		*NULL*	
block_noomber	int(11)	NO		*NULL*	
space	int(11)	NO		*NULL*	
floor	int(11)	NO		*NULL*	

Рисунок 1.4 – Таблиця Кімнати

Атрибут КлючАвтоінкремент (id) – тип даних є цілим числом, яке є більшим нуля, ключ, значення NULL – ні. У атрибуту НомерКімнати (room_number) – тип даних ціле число, значення NULL – ні. Атрибут СтанКімнати (status) поле текстового типу довжиною символів 45, значення NULL допускається. Атрибут НомерБлоку (block_number) тип даних ціле число, значення NULL для цього поля не допускається. Атрибут КількістьМісць (space) поле числового типу, відображає кількість місць в кімнаті. Атрибут Поверх (floor) – тип даних є цілим числом, а значення NULL для такого поля також є відсутнім.

Наступна сутність, яка буде розглянута це сутність Пожильці, є теж не менш важливою структурою даних в системі обліку мешканців гуртожитку, оскільки містить інформацію про пожильців, які в ньому мешкають. На рисунку 1.5 зображено поля таблиці Пожильці.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
name	varchar(45)	NO		*NULL*	
surname	varchar(45)	NO		*NULL*	
full_name	varchar(80)	YES		*NULL*	
sex	varchar(10)	NO		*NULL*	
birth_date	date	NO		*NULL*	
room_id	int(11)	NO	MUL	*NULL*	

Рисунок 1.5 – Таблиця Пожильці

Атрибут КлючАвтоінкремент (id) – тип представлених даних є цілим числом, більшим за нуль, ключ, значення NULL – ні. А для Атрибуту Ім’яПожильця (name) – поле текстового типу довжиною 45 символів, значення NULL немає. Атрибут ПрізвищеПожильця (surname)– поле текстового типу довжиною 45 символів, значення NULL немає. Атрибут ПоБатькові (Full_name) поле з текстовим типом довжиною в 80 символів, допускає значення NULL. Атрибут Стать (sex) - поле текстового типу, яке не допускає значення NULL. Атрибут РікНародження (birth_date) – поле типу дата, містить дату народження

пожильця, значення NULL не допускає. Атрибут КлючКімнати (room_id) поле числового типу, є зовнішнім ключем до сутності Кімнати, NULL не допускає.

Далі наведена сутність Особисті дані (див. рисунок 1.6)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
occupier_id	int(11)	NO	MUL	*NULL*	
age	t(7)	NO		*NULL*	
status	double	NO		*NULL*	
start_year	int(11)	NO		*NULL*	
from	varchar(45)	NO		*NULL*	
speciality	varchar(45)	NO		*NULL*	

Рисунок 1.6 – Таблиця Особисті дані

Атрибут КлючАвтоінкремент (id), що є ключовим полем з типом даних числовим, значення NULL не допускає. Атрибут КлючПожильця (occupier_id) – числового типу значення NULL відсутнє, за допомогою нього сутність Особисті дані пов’язується з сутністю Пожильці. Атрибут ВікПожильця (age) – поле з числовим типом даних, значення NULL не допускає. Атрибут Статус (status) – поле з числовим типом подвійної точності даних, значення NULL не допускає. Атрибут ПочатокПроживання (start_year) – поле з числовим типом даних, значення NULL не допускає. Атрибут З відки (прибув) (from) – поле з текстовим типом даних довжиною 45 символів, значення NULL не допускає. Атрибут Спеціальність / професія (Speciality) – поле з текстовим типом даних довжиною 45 символів, значення NULL не допускає.

Наступною сутністю для системи є сутність Контактна Особа таблиця якої представлена на рисунок 1.7).

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
occipier_id	int(11)	NO	MUL	*NULL*	
mail	varchar(45)	YES		*NULL*	
main_phone	int(13)	NO		*NULL*	
contact_person_name	varchar(45)	NO		*NULL*	
contact_person_phone	int(13)	NO		*NULL*	
location	varchar(100)	NO		*NULL*	

Рисунок 1.7 – Таблиця Контактна Особа

Сутність Контактна Особа містить такі атрибути:

- атрибут КлючАвтоінкремент (id) поле цілого типу, первинний ключ, значення NULL відсутнє;
- атрибут КлючПожильця (occupier_id) це поле також містить цілий тип та являється як зовнішній ключ таблиці Пожильці, значення NULL не допускає;
- атрибут ЕлектроннаПошта (mail) поле текстового типу довжиною 45 символів, зберігає електронні адреси мешканців гуртожитку, допускає значення NULL;
- атрибут НомерМобільного (main_phone) поле, яке має числовий тип і містить номер телефону пожильця, значення NULL не допускає;
- атрибут Ім’яКонтактноїОсоби (contact_person_name) поле текстового типу довжиною 45 символів, не допускає значення NULL;
- атрибут НомерКонтактноїОсоби (contact_person_phone) поле текстового типу довжиною 13 символів, зберігає номер телефону контактної особи, значення NULL не допускає;
- атрибут МісцеПроживання (location) поле текстового типу довжиною 100 символів, містить домашню адресу мешканця гуртожитку, не допускає значення NULL.

1.2.2 Будова (створення)UML-діаграми класів [30]

В системі обліку пожильців у гуртожитку мною було визначено основні класи програми, такими виявились дев’ять основних класів, зокрема:

- основний(головний) інтерфейс програми;
- створення нового пожильця;
- «створення(додавання)» нової кімнати;
- Вилучення(видалення) пожильця;
- Вилучення(видалення) порожньої кімнати;
- перегляд блоків;
- перегляд кімнат;
- перегляд пожильців;
- функція пошуку.

На даному представленому етапі вважаю, що на часі розгляд більш детального опису класів, а саме:

– Класу «основного(головного) інтерфейсу програми»,який містить весь основний програмний функціонал. Крім цього у ньому відображаються похідні класи.

– Класу «створення нового пожильця» призначений для додавання інформації про пожильця в базу даних.

– Класу «додавання нової кімнати» заносить до бази даних інформацію про кімнати.

– Класу «вилучення пожильця», яким надається можливість видаляти дані про пожильця із бази гуртожитку.

– Класу «вилучення порожньої кімнати» дає можливість витирати кімнату із бази гуртожитку, але лише якщо вона порожня, тобто у кімнаті немає поселених пожильців.

– Класу «перегляду блоків», за допомогою якого відображається таблиця даних про блоки кімнат у гуртожитку.

– Класу «перегляду кімнат», яким відображається таблиця даних про кімнати гуртожитку.

– Класу «перегляду пожильців», який відображається таблицею даних про мешканців гуртожитку.

Всі звертання від програми і до таблиць, що належать до одної бази даних сервера здійснюються через використання одного з'єднання, яким замикаються увсі компоненти по доступу до даних, які володіють відповідними значеннями властивості DatabaseName.

Все управління одиночним з'єднанням з якоюсь-небудь базою даних здійснюється за допомогою компонента ADO Connection. У загальному ж з випадків використання цієї компоненти в програмах до баз даних не є обов'язковою складовою. Проте при роботі із серверами SQL він є необхідним, оскільки:

– дозволяє керування з'єднанням із базою даних;

– дозволяє керування транзакціями значно простішим шляхом, аніж при допомозі операторів SQL;

– забезпечує можливість реєстрації користувачеві на сервері.

1.3 Конструювання програмної системи

1.3.1 Загальний опис програмної системи

Загальним описом системи являється високорівневий опис програмних компонентів, які були використані та створені при розробці програмної системи, характеризація рис системи як засобу поліпшення життєдіяльності людини. Облік

житла є невід’ємною частиною справи багатьох агенцій з нерухомості. Отож, загально описати програмну систему можна як зручний засіб обліку житла створений на основі мови C# та бібліотек написанх на цій мові.

1.3.2. Обґрунтування вибору мови програмування

За результатами детального аналізу я прийняв рішення в розробці програмного забезпечення зупинитись на мові програмування C, оскільки її синтаксис є достатньо близьким саме до C++ та мови програмування Java. C# є мовою зі строгою статичною типізацією, з підтримкою поліморфізму, можливістю перевантаження операторів, вказівниками на функції, які є членами класів, атрибутів, подій, властивостей, з винятками, коментарями форматування типу XML. Крім цього, в даному випадку можна говорити про те, що С C# «перейняла» багато чого від своїх «попередників», зокрема мови C++,Delphi, Модула, а також Smalltalk. В C#, саме опираючись на практику їх використання, виключаються деякі моделі, які «зарекомендували» себе проблемними саме при розробці програмних систем, для прикладу можна назвати множинне спадкування класів (що є відмінним від, наприклад, C++).

C# є мовою програмування, яку можна вважати за близького родича з мовою програмування Java, створеною відомою світовою корпорацією Сан Мікросистемс, саме тоді, коли на той час вже глобальний розвиток інтернета зумовив необхідність вирішення задачі роззосереджених обчислень. Взявши за базу популярну мову C++, в мові Java було виключено потенційно небезпечні речі, такі як вказівники без контролю за виходом за межі, тощо. Саме для роззосереджених обчислень було створено концепцію віртуальної машини, так само як і машинно-незалежний байт-код, який свого роду є посередником зпоміж вихідним текстом програми й апаратним комп’ютерним інструктуванням або ж якогось іншого пристрою, інтелектуального.

Java – мова, яка стала набувати дедалі більшої популярності, і, крім власника SunMicrosystems була ліцензованою і компанією Майкрософт. Проте там виникла суперечка з приводу того, що Microsoft, при створенні свого клону Джава робить її сумісною лише виключно із платформою Windows, що суперечило парадигмам самої концепції для машинно-незалежного середовища та, крім цього, ще й порушує ліцензійну угоду. В судовому порядку позиція SunMicrosystems була визнана справедливою, і рішенням суду було зобов'язання Microsoft відмовитись від використання Java без дотримання ліцензійних вимог.

Ситуація, що виникла залишалась досить складною і потребувала рішення, тому Microsoft вирішила, користуючись своїм авторитетом на ринку, що найкращим варіантом буде створення свого власного аналогу мови Java. Цей аналог отримав назву C#. Вона у спадок від мови Java залишила концепції віртуальної машини (у вигляді середовища .NET), байт-код(MSIL) і більшу безпеку вихідного коду програм, і, крім переліченого було враховано також і попередній досвід при використанні програм на мові Java.

Ще одним із нововведень у мові C# стало отримання можливості спрощеної взаємодії, порівняно із попередніми мовами, з кодом програм, які написані іншими мовами, що є важливим моментом в процесі створення великих проєктів. Якщо програми написані різними мовами виконуються на платформі .NET, то сумісність програм (їх типів даних при кінцевому рахунку) приймає на себе саме бере на себе .NET.

Таким чином, сьогодні мова програмування C# для корпорації Microsoft є флаґманською мовою, оскільки найповніше застосовує нові можливості.NET. Про решту мов програмування можна сказати, що вони може й підтримуються, проте визнаються такими, що володіють спадковими недоліками при застосуванні на платформі .NET. Таким чином мій вибір є очевидним та актуальним.

Якщо в загальному, то варта зазначити, що мову C# було розроблено в якості такої мови програмування до уваги беремо саме прикладний рівень для CLR і, у в якості такої, що є залежною, перш за все, від можливостей безпосередньо самої CLR, зокрема варта згадати про систему типів мови C#. Сама

наявність або ж відсутність саме тєї або ж іншої виразної особливості мови спонукається можливістю конкретної мовної особливості до трансльованості до відповідних конструкцій CLR. Саме CLR надає і C#,інші .NET-орієнтовані мови програмування достатньо багато можливостей якими не володіють, просто позбавлені так звані «класичні» мови програмування. Зокрема, якщо говорити про збирання сміття – в самій мові C# функція не реалізується, проте проводиться CLR для програм, які написані на C# точно так саму, як це і робиться для програм написаних на VB . NET ,J# та інших.

1.3.3. Обґрунтування вибору системи управління базами даних

В своїй роботі при проектуванні системи управління БД я зупинив свій вибір на MySQL системі вільного керування призначеної для баз даних (СУБД).

MySQL , на мою думку, є вдалим вирішенням питань саме додатків малого та середнього розміру, яка входить до комплектсу серверного програмного забезпечення. Часто MySQL використовують в якості серверу, на який відправляють звернення як локальні так і видалені клієнти, однак до дистрибутиву входить і бібліотека для внутрішнього серверу,, і це дає змогу включити MySQL до автономних програм. Гнучкість обраної мною системи управління забезпечується насамперед можливістю підтримки великої кількості різнотипових таблиць, зокрема, користувач може обрати як таблицю типу MyISAM, яка підтримує повнотекстовий пошук, так і таблицю типу InnoDB, яка підтримує транзакцію і на рівнях окремих записів. Заслуговує уваги і те, що MySQL надається із спеціальними таблицями типу EXAMPLE, і це свідчить про принцип створення різновиду таблиць вже нових типів. Також варта звернути увагу і на те, що завдячуючи відкритій архітектурі та GPL-ліцензуванню, для системи MySQL регулярно появляються таблиці нових типів.

Ці особливості були вирішальними під час того, як я визначався яку систему управління базами даних та мову програмування обрати.

1.3.4 Технології проєктування та створення програмного засобу

Таким чином, я визначився, що моя система розробляється власне для обліку мешканців у гуртожитку при навчальних закладах, наприклад Тернополя, чи на базі підприємств чи організацій, основне призначення якої є оптимізація роботи.

Даний програмний продукт є універсальним і його можна редагувати для будь-яких інших гуртожитків. Система типу «Гуртожиток» легка і комфортна в користуванні, інтерфейс буде зрозумілий користувачам без будь-якої спеціальної освіти.

Для розробки реляційної бази даних вибрано СУБД SQL Server [14,15].

Для проєктування та створення програмного засобу пропоную в якості середовища для програмування використати -Microsoft Visual Studio.

Причинами вибору даного СУБД які вплинули на мій вибір:

- безкоштовність обраної СУБД із відкритим кодом;
- висока ступінь надійності щодо захисту даних;
- гарантія безперебійності в роботі;
- прискореність процесу при розробці БД;
- збереження та використання різнотипових даних.
- можливість збереження, виведення запитів створення та наповненості бази даних;
- ну і те, що я вже працював, а отже краще знаю особливості обраної СУБД.

Таким чином проєктування мого програмного засобу та його розробка на мові програмування C# в середовищі програмування та створення застосувань MicrosoftVisualStudio. Особливостями надання мною переваги даного середовища та мови програмування стали функції [31-33]:

- високопродуктивного компілятора (в машинний код);
- об’єктно-орієнтованої моделі компоненту;
- візуальної (а, отже, і швидкісної) побудови додатків із програмних прототипів;
- засобів, які масштабуються, з метою побудови баз даних;
- компілятора в машинний код;
- легкого та швидкого проектування та розробки користувацьких програм;
- інтелектуального управління кодом;
- розширених можливостей відладки;
- підсвітки синтаксису;
- досвіду роботи у вибраному середовищі;

В загальному розроблений програмний продукт призначений для обліку мешканців гуртожитку. Розроблена система оптимізує та суттєво полегшує роботу, також економить час працівників чи осіб які можуть нею користуватись (волонтерів, працівників, відповідальних осіб).

Для коректної роботи програми комп’ютер повинен забезпечувати можливість функціонування наступними мінімальними вимогами системи до:

- операційної системи: Microsoft Windows XP/Vista;
- процесору типу: AMD Athlon x64 із частотою не менш ніж 2,8 Ггц, або ж Intel Pentium із частотою, яка була би не менш ніж 1,8 Ггц;
- оперативної пам’яті (ОЗП) 512 Мб;
- HDD: 100 Мб розмір для вільного простору диску для коректності в роботі самої програми;
- Необхідна певна кількість принтерів, клавіатур та мишок.

Також потрібно передбачити на машині можливість певних програмних продуктів, зокрема таких як:

– Майкрософт SQL Server 2008 та вище;
– Adobe FineReader або інша програма для читання *.pdf формату.

1.3.5 Опис інтерфейсу програмної системи [34]

Програмна система для обліку мешканців у гуртожитку, має на меті полегшити процес ведення обліку тимчасових мешканців та розселенню по кімнатах.

Після запуску програми відкривається головне вікно програми, загальний вигляд якого представлено на рисунок 1.8.

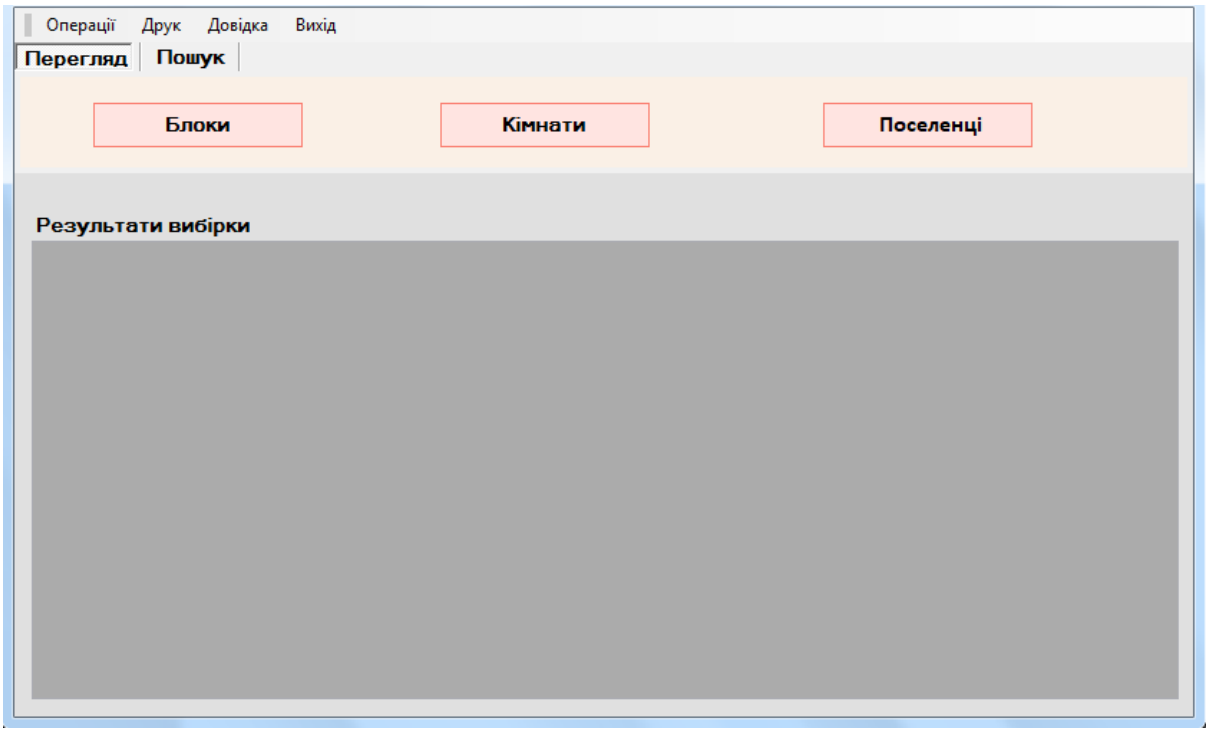

Рисунок 1.8 – Головне вікно програми

Обравши в меню Операції – > Додати – > Нового мешканця, з’являється форма для запису даних про нового поселенця.

Обравши в меню Операції – > Додати – > Нову кімнату, з’являється форма для запису даних про кімнату (див рисунок 1.10).

При коректному вводі інформації про нового поселенця після натискання кнопки Зберегти виводиться повідомлення з порядковим номером (id) вже нового мешканця у базі як це представлено на рисунку 1.11).

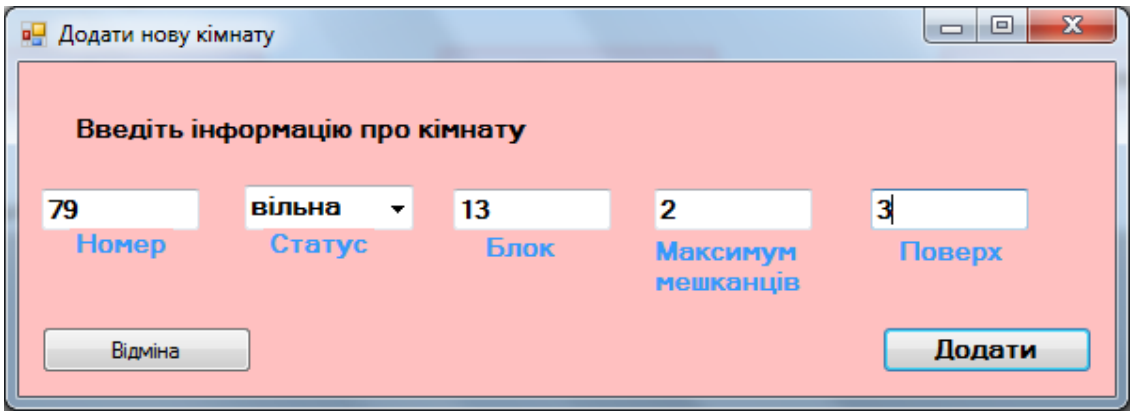

Рисунок 1.10 – Форма додання нової кімнати (вже заповнена)

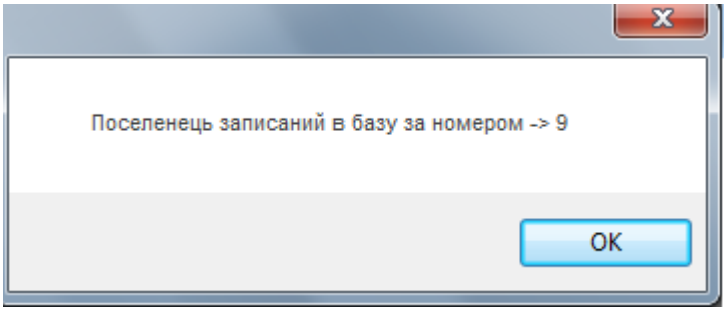

Рисунок 1.11 – Вікно успішного додання поселенця

На рисунку 1.12 зображено вікно повідомлення про успішне додання кімнати у базу.

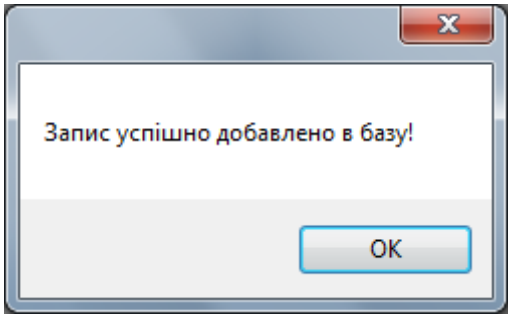

Рисунок 1.12 – Повідомлення про успішне додання кімнати у базу

Вікно пошуку вже тепер мешканця у базі для перегляду чи редагування зображено на рисунку 1.13.

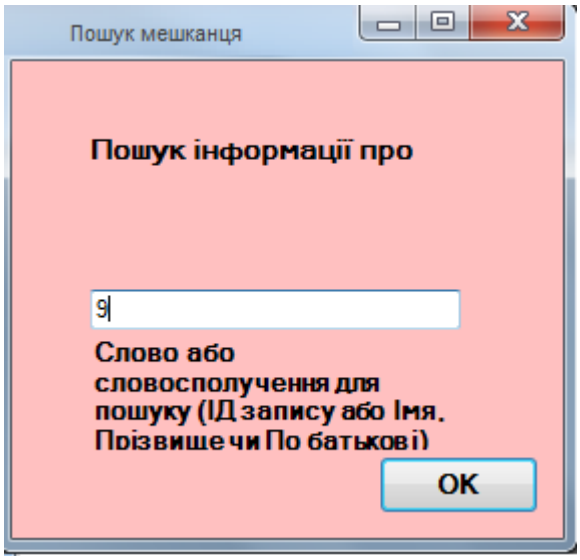

Рисунок 1.13 – Вікно пошуку мешканця у базі

Після редагуванні даних про уже на цьому етапі мешканця виводиться повідомлення про успішне завершення операції (див. рисунок 1.14).

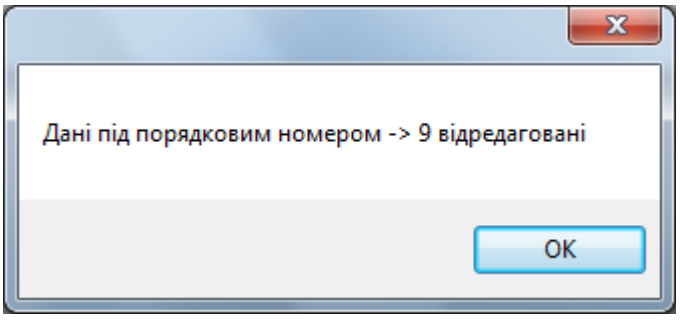

Рисунок 1.14 – Повідомлення про успішне завершення операції

Якщо при додаванні нового майбутнього пожильця чи кімнати було заповнено не всі поля, то видається повідомлення про помилку, подібно як представлено на рисунок 1.15.

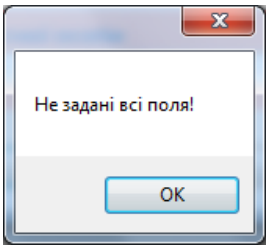

Рисунок 1.15 – Повідомлення про помилку

Під час заповнення інформації про нового пожильця потрібно вказати до якої кімнати він поселяється. Приклад вікна вибору кімнати з списком вільних кімнат представлено на рисунку 1.16.

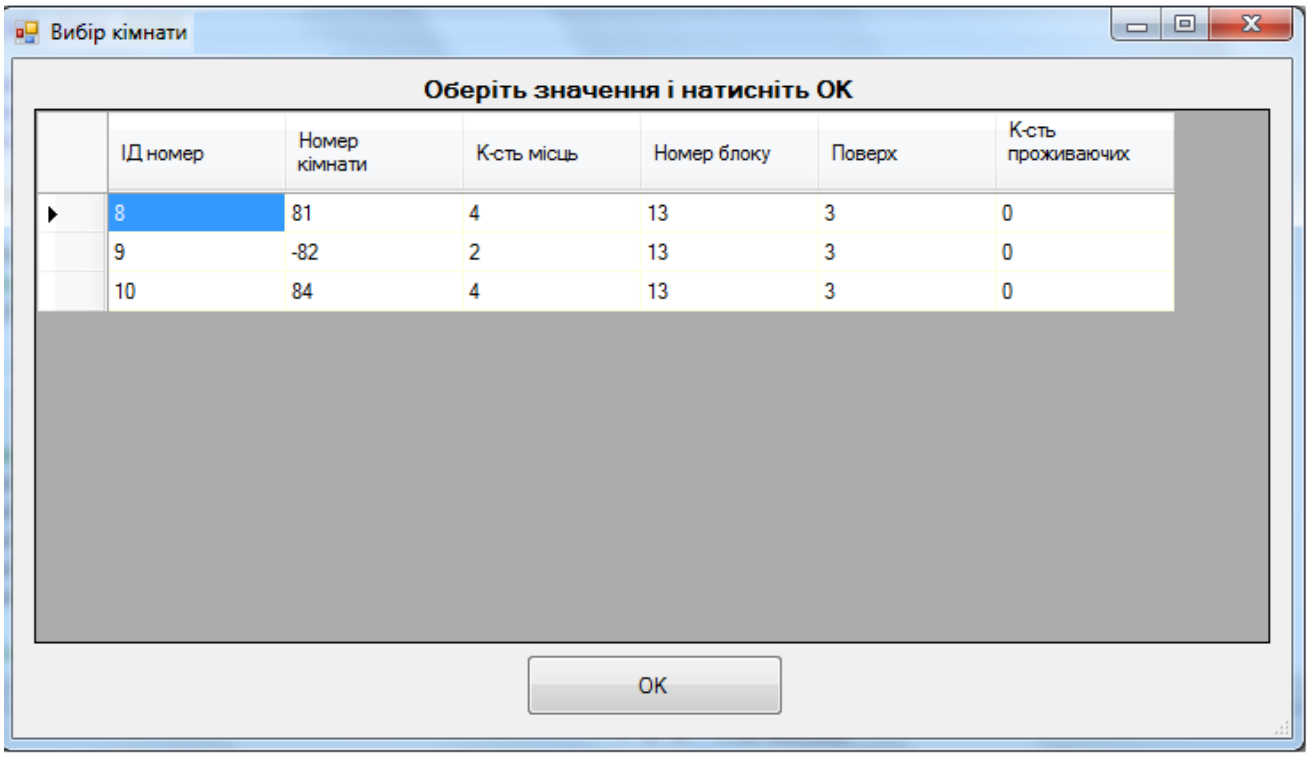

Рисунок 1.16 – Вікно вибору вільної кімнати

На рисунках 1.17, 1.18, 1.19 зображено вікна перегляду інформації про блоки, кімнати і мешканців відповідно.

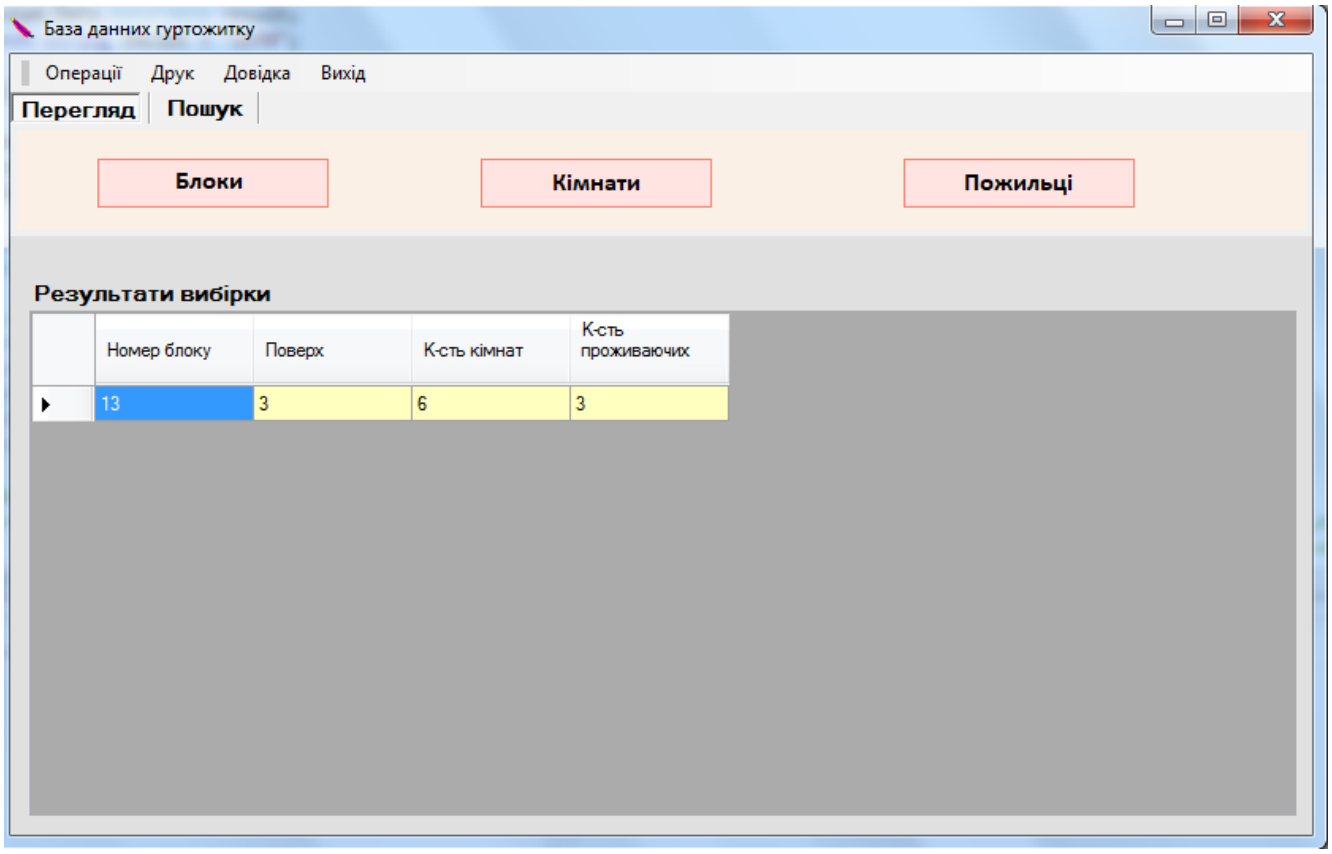

Рисунок 1.17 – Вікно перегляду інформації про блоки гуртожитку

Головною особливістю сторінки перегляду пожильців є така, що відображення представлене трьома пов’язаних поміж собою таблицями. Така реалізація дає можливість отримати детальну інформацію про кожного мешканця.

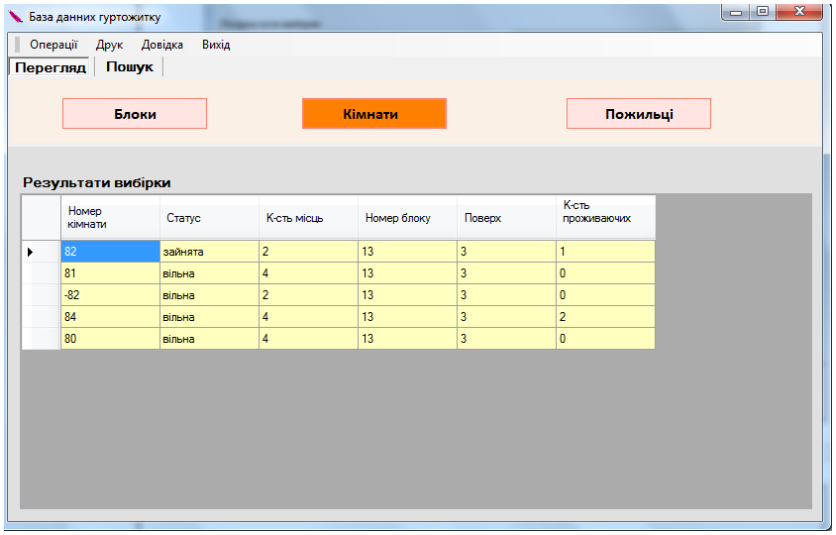

Рисунок 1.18 – Вікно перегляду інформації про кімнати гуртожитку

Таким чином, проілюструвавши рисунки візуального інтерфейсу програми я описав засоби з допомогою яких користувач може використовувати весь потенціал програмних компонентів та засобів візуального інтерфейсу та обчислювальних потужностей гаджету користувача.

2 ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Розробка плану тестування

Тестування програмного забезпечення (з англ.SoftwareTesting) – є ні чим іншим ніж процесом саме технічного дослідження, призначеного для отримання інформації стосовно якості програмного продукту по відношенню до його контексту (в якому його призначено до використання) . Варта зазначити, що технічно тестування включає в себе процеси пошуку помилок чи якихось інших дефектів, оцінку через випробування програмних складових [35].

Про вид робіт чи навіть етап пов’язаний з тестуванням програмного забезпечення варта говорити як про один із найважливіших процесів, що є однією з основних частин при розробці програмного забезпечення. Руйнування репутації – це найменше, що чекає будь-яку компанію, яка нехтує цим процесом. Наявність плана тестування є обов’язковою складовою, і його варта обов’язково зазначити безпосередньо в проєктній документації. В плані варта передбачити означені цілі та призначення даного конкретного програмного продукту. Документація по тестуванню програмного продукту створюється таким чином щоби в процесі тестування можна було би перевірити роботу програми при різних ситуаціях, і з цією метою створюють так звані реалістичні сценарії тестування [36].

Проведення етапного тестування дають можливість побачити в програмному забезпеченні проблеми і подивитися на програму під дещо «іншим кутом зору», оскільки перевірку здійснюють при використанні реалістичних сценаріїв, які є далекими від формальних перевірок.

Для того щоби із впевненістю врахувати та відповідати усім вимогам сучасного ІТ ринку програмних продуктів, є різні види тестувань, які все ж таки було б варта доручати професійним спеціалістам-тестувальникам. Адже, оскільки це їх як правило основний напрямок діяльності, то вони в курсі тенденцій сьогодення які пред’являє бізнес, постійно розвиваються в цьому напрямку та вдосконалюються [37-39].

Існує певна кількість ознак, за допомогою яких визначають та зазвичай роблять класифікацію на види тестування. Як правило виділяють наступні такі ознаки:

1. По тестувальному об’єкту:
   - функціональне(англ.functional) тестування (англ.testing);
   - тестування (англ.performance) продуктивності (англ.testing):
     a. навантажувальне (англ.load) тестування(англ.testing);
     b. стрес(англ.stress) -тестування (англ.testing);
     c. стабільності(англ.stability /endurance/ soak);
   - за зручністю використання(англ. usability);
   - інтерфейсне (ui) користувача;
   - тестування безпеки(англ.security);
   - локалізації(англ.localization testing);
   - сумісності(англ.compatibility testing).
2. По системним знанням:
   - тестування за методикою типу «чорний ящик» (англ.black box);
   - тестування за методикою типу «білий ящик» (англ.white box);
   - тестування за методикою типу «сірий ящик» (англ.gray box).
3. По ступеню автоматизації:
   - ручне(англ.manual) тестування (англ.testing);
   - автоматизоване(англ.automated) тестування (англ.testing);
   - напівавтоматизоване (англ.semiautomated) тестування (англ.testing).
4. По ступеню ізольованості компонент:
   - компонентне (модульне) (англ.component / unit) тестування (англ.testing);
   - інтеграційне (англ.integration) тестування (англ.testing);
   - системне (англ.system / end-to-end) тестування (англ.testing).
5. По часу здійснення тестування:

- альфа(англ.alpha) -тестування (testing);
- бета-тестування(англ.beta testing).

6. По ознаці позитивності сценаріїв:
   - позитивне (англ.positive) тестування (англ.testing);
   - негативне (англ.negative) тестування (англ.testing).

2.2 Розробка тестів для тестування програмної системи

Загалом тестом називають процедуру при застосуванні якої є можливість чи підтвердити, або ж навіть спростувати без перебільшення працездатність всього коду. Коли я, як програміст, перевіряю «життєздатнясть» розробленого мною коду, то, зазвичай, виконую тестування в ручному режимі. В представленому контексті тест можна розділити на два етапи: стимулювання кода та перевірка результату його роботи. Якщо говорити про автоматичне виконання тестування, то в такому випадку замість програміста стимуляцією кода так само як і перевіркою результатів займається безпосередньо комп'ютер, на екрані якого відображається кінцевий результат роботи тесту, а саме працездатний код або ж не працездатний. В цілому ж за допомогою методики розробки через тестування я можу отримати відповіді на запитання стосовно організації автоматичних тестів, а також вироблення деяких певних навиків в тестуванні [36-39].

Далі, для розробленої програмної системи, пропоную розглянути три види тестів:

- ad hoc-тестування та складання Bug report - різновид ручного тестування;
- тестування функціонального типу;
- тестування модульного типу.

Тестування виду ad hoc та складання Bug report (англ.ad hoc testing) – тестування без тест-плана та документації, яка базується на методиці передбачення помилок та на особистому досвіді самого тестувальника.

Десктопний варіант програмної системи основною функцією є швидкий пошук вільного тимчасового житла в традиційному стилі (на прикладі гуртожитку будь-якої фори власності, з умовою, що власники готові поселяти тимчасово осіб з відповідним статусом), всі звичні елементи розташовані в очікуваних місця. (див. рисунок 2.1).

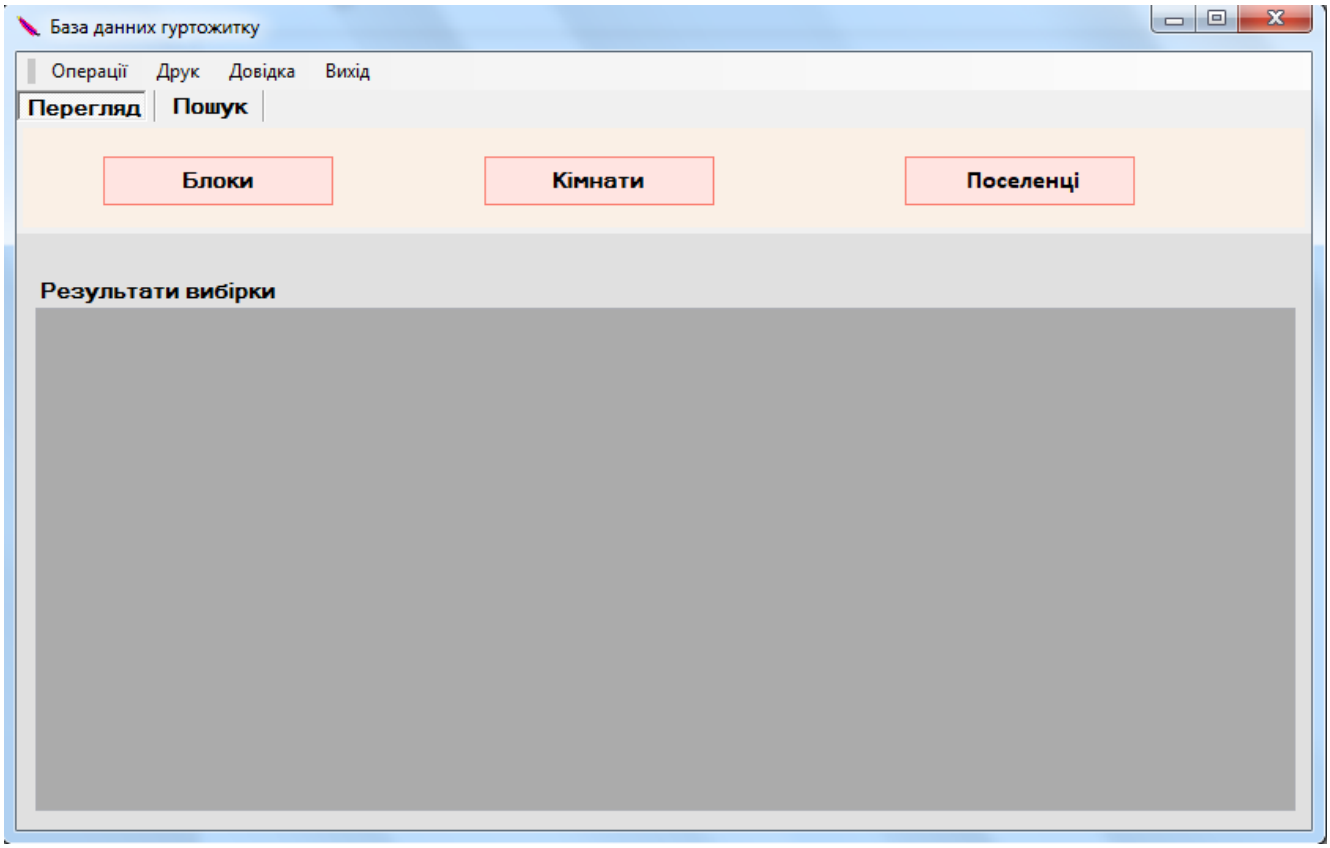

Рисунок 2.1 – Головна форма програмної системи

Слідуючим і не меншим по важливості аспектом у проєктуванні графічного користувацького інтерфейсу є співпадіння іконки у заголоці вікна та у пункті головного меню «Про програму» як це показано на рисунку 2.2. В представленому нижче варіанті і те і те співпадає (іконка у заголовку вікна з вікном «Про програму»). Таким чином можна зробити висновки про те, що враховано правила побудови графічного інтерфейсу користувача [5], і представлений інтерфейс користувача програмної системи побудований коректно із врахуванням вимог та правил.

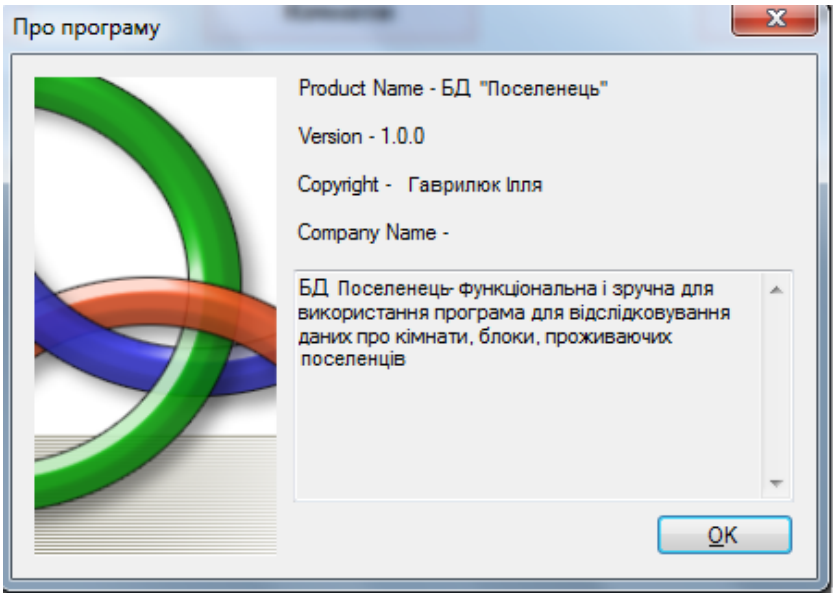

Рисунок 2.2 – Форма Про програму програмної системи «Поселенець»

Далі розглянемо Головне меню робочої області програми «Поселенець». Як видно з представленого рисунку 2.3 складові елементи для головного меню розміщені коректно, і, що є найголовніше, візуально зручно та у «правильній» послідовності, вони легкі для сприйняття простого, без спеціальної освіти користувача. Підменю для кожного елементу з головного меню оформлене також коректно, і це я показав на рисунку2.3.

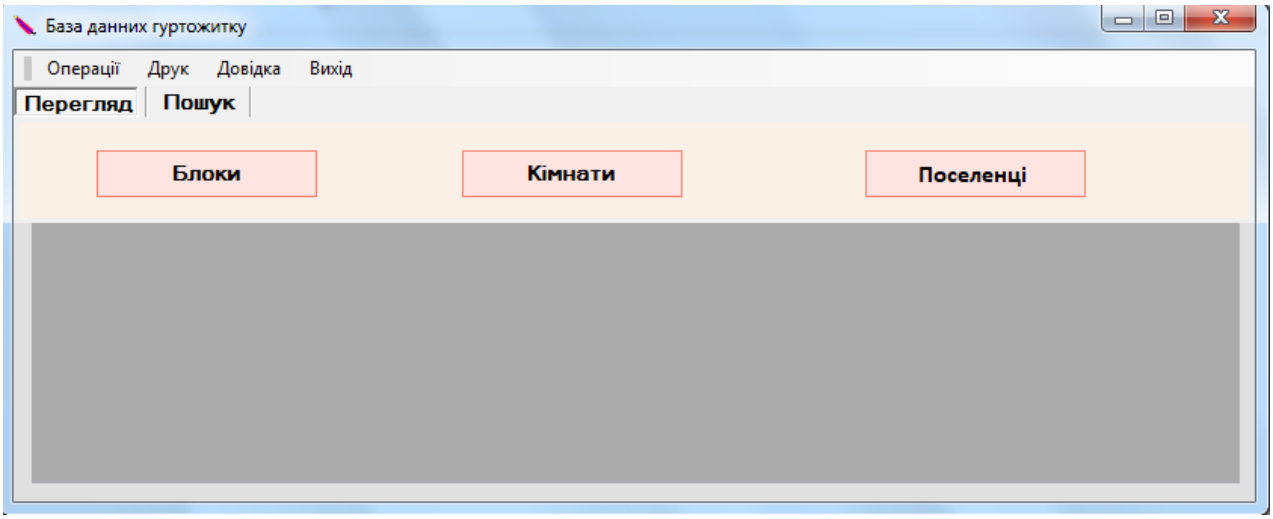

Рисунок 2.3 – Представлення головного меню розроблюваної програмної системи

В результаті, я виконав ручне тестування, за його підсумками отримав так званий Bug-report, структуру звіту я представив нижче у таблиці 2.1.

Таблиця 2.1 – Створення Bug-report

Help Desk Service	
Короткий опис (Summary)	При введенні текстової інформації у поля для вводу (Edit, TextBox, ComboBox) зникає активність курсору і подальше введення тексту неможливе, поки знову не буде вибране поле для вводу
Проект (Project)	Help Desk Service
Компонент додатку (Component)	GUI
Номер версії(Version)	Версія..v 1.1.0.0
Серйозність (Severity)	S3 Значний (Major)
Пріоритет (Priority)	P2 Середній (Medium)
Автор (Author)	Гаврилюк Ілля
Призначення (Assigned To)	Developer
Середовище	
ОС /Сервіс Пак і т.д./ Браузер + версія / ...	Windows 7 Max, програмна система «Поселенець»
...	
Опис	
Кроки відтворення (Steps to Reproduce)	1. Запуск програмної системи 2. Перехід на форму додання поселенця 3. Спроба введення даних в текстові поля
Фактичний результат (Result)	При введенні інформації в текстові поля зникає активність з текстового поля, зникає курсор.
Очікуваний результат (ExpectedResult)	При введені інформації в текстові поля активність поля не зникає.
Додатково	
Прикріплений файл (Attachment)	Додача нового поселенця (форма)

Проблема виникла в результаті роботи, і яку показано у звіті пов’язана із неправильним відображенням для модальних форм у програмній системі. Я детально проглянув програмний код системи та виявив відсутність одного методу, який несе відповідальність як за виклик так і за знищення вказаної модальної

форми але уже після закриття вікна. Така помилка висвічувалась по всім формам, яких було потрібно ввести текстові дані по всьому проєкту.

За допомогою функціонального тестування є змога перевірити чи повною мірою вдалось реалізувати функціональні вимоги, тобто можливостями програмного забезпечення при деяких певних умовах до вирішення поставленого завдання, потрібного користувачам. Таким чином функціональними вимогами визначється, що саме здійснює програмний продукт, які конкретно завдання вирішуються ним.

До функціональних вимог можна включити:

- функціональну придатність
- точність
- можливості до взаємодії
- відповідність до стандартів та правил
- захищеність.

При тестуванні програмної системи «Поселенець» я розглянув безпосередньо на відповідність функції, методи, компоненти, які було реалізованов програмі, результат тестування із використанням функціонального методу наведено в таблиці 2.2.

Окрім ручного тестування я ще додатково провів DUnit Testing. Так зване тестування модульного типу, яке є різновидністю тестів, за допомогою якого є можливість перевірити функціональність деякого певного розділу коду, і як правило на функціональному рівні. Для об’єктно-орієнтованого середовища, це, зазвичай, тестування на рівні класу, а в мінімальних модульних тестах містяться конструктори, а також деструктори.

Таблиця 2.2 – Функціональне тестування програмної системи Help Desk Service

ID	Тип	Опис	Очікувано	Реально	Pass/ Fail
01_All	positive	Перевірка відображення на різних розширеннях монітора	Усі компоненти знаходяться на своїх місцях, немає ніяких спотворень інтерфейсу	Усі компоненти знаходяться на своїх місцях, немає ніяких спотворень інтерфейсу	PASS
02_All	positive	Функція переходу по компонентах клавішею Tab (function button Tab)	Повинен здійснюватися перехід курсора між полями для вводу та кнопками	При переході функція клавіші Tab працює у всіх випадках	PASS
03_All	negative	При доданні нового пожильця заповнено не усі поля	Поле з помилкою	Поле з помилкою	PASS
04_All	positive	Усі поля при доданні нового пожильця коректно заповнені	Повідомлення про успішне додання	Повідомлення про успішне додання	PASS

Як правило, тести такого типу пишуть розробники коли проводять роботу над кодом (такий стиль називають «біла скринька»). Це роблять для того, щоби переконатись, що функція працює очікувано. Часто характерно для однієї функції наявність декількох тестів. Це потрібно для того, щоби переглянути всі можливі з випадків використання коду. Для модульного тестування не притаманна перевірка функціонування частини програмного забезпечення. Зазвичай, його використовують для гарантування незалежності роботи один від одного основних блоків програмного забезпечення [38].

Модульне тестування є процесом розробки ПЗ, для його характерне включення синхронізованих застосувань широкого спектру. Це потрібно для

передбачення/запобігання дефектів, а також щоб виявити стратегії для пониження ризиків при розробці програмного забезпечення, так само як і оптимізації часу та витратної частини. Його виконує розробник програмного продукту або інженер, вже під час фази життєвого циклу розробки програмного продукту, яка називається будівельна. Основна роль модульного тестування спрямована на усунення помилок при проектуванні. Цю стратегію спрямовано на підвищення якості розроблюваного програмного продукту, до того рівня, як то передбачається та вимагається процесом контролю за якістю.

В залежності від очікуваної організації при розробці програмного продукту, до модульного тестування можне входити статичний аналіз коду та аналіз потоку даних,аналізу метрик, експертна оцінка коду, аналіз покриття коду, так само як і інші методи для перевірки програмногопродукту [36].

Для цього потрібно вибрати команду File/New/Other і в списку, що з’явиться вибрати пункт Unit Test та вибрати Test Project .

Після створення проєкту тестування потрібно обрати перелік з основних модулів системи, над яким потрібно здійснити тестування. Для цього потрібно здійснити аналогічні як і у попередньому випадку дії, але обрати в діалоговому вікні пункт Test Case.

Клас,в якому містяться тести, потрібно анотувати атрибутом Test Class. Якщо йдеться про окремі тести, які представляють собою методи, то їх потрібно анотувати атрибутом Test Case.

В лістингу 2.1 нижче представлений фрагмент програмного коду, яким безпосередньо перевіряється модуль додавання студенту у базу.

Лістинг 2.1 – Код введення даних про пожильця

```
query = "INSERT INTO (name, surname, full_name, sex, birth_date, room_id) VALUES ('" + this.text B ox1.Text. To
String() + "' , '" + this.textBox2.Text. To String() + "' , '" + this.textB ox3 .Text. ToString() + "' , '" + this.textB ox4.Text.
ToString() + " ' ,'" + this.textB ox5.Text. To String() + " ' ,'" + this.textBox6.Text. To String() + "')";
        long insert_id = MySqlLib.Database.getSingleton.Write(query);
```

Продовження лістингу 2.1

```
            if (insert_id > 0)
            {
//Додаткова інформація про особисті дані і контакти
                query = "INSERT INTO status  VALUES ('" + insert_id.ToString() + "','" + this.textBoxEdu1.Text.ToString()
+ "','" + this.textBoxEdu2.Text.ToString() + "','" + this.textBoxEdu3.Text.ToString() + "','" +
this.textBoxEdu4.Text.ToString() + "','" + this.textBoxEdu5.Text.ToString() + "')";
                MySqlLib.Database.getSingleton.Write(query);
                query = "INSERT INTO contact_info (id, mail, main_phone, contact_person_name, contact_person_phone,
location) VALUES ('" + insert_id.ToString() + "','" + this.textBoxCon1.Text.ToString() + "','" +
this.textBoxCon2.Text.ToString() + "','" + this.textBoxCon3.Text.ToString() + "','" + this.textBoxCon4.Text.ToString() +
"','" + this.textBoxCon5.Text.ToString() + "')";
                MySqlLib.Database.getSingleton.Write(query);
                //Перевіряємо чи зайнята кімната, якщо так то змінюємо статус кімнати на "зайнята"
                query = "SELECT r.status AS 'Статус',r.space AS 'К-сть місць',count(s.room_id) AS 'К-сть проживаючих
пожильців' FROM room AS r LEFT JOIN AS s ON r.id=s.room_id WHERE r.id=" + this.textBox6.Text.ToString() + "
GROUP BY r.id";
                result = MySqlLib.Database.getSingleton.Read(query);
                if (Convert.ToInt16(result.DefaultView.Table.Rows[0]["К-сть місць"]) <=
Convert.ToInt16(result.DefaultView.Table.Rows[0]["К-сть проживаючих пожильців"]))
                {
                  query = "UPDATE room SET status='зайнята' WHERE id=" + this.textBox6.Text.ToString();
                  MySqlLib.Database.getSingleton.Write(query);
                  MessageBox.Show("Кімната з ІД " + this.textBoxStud6.Text.ToString() + " тепер зайнята");
                }
                        this.ReturnValue = Convert.ToString(insert_id);
```

Unit тесту метод додавання пожильця до бази успішно пройшов тест.

Після здійснення перевірки програмної системи у вигляді кількох різних видів тестів дозволю собі зробити висновок про те, що в цілому програми без помилок напевне не існує. На практиці доведено, що часто винуватцями помилки в програмі найчастіше бувають безпоседеньо ті що їх пишуть – програмісти. А одним з законів для практичного програмування є твердження про те, що що жодна з програм не дасть бажаного результату вже під час першої спроби трансляції і виконання.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Загальні вимоги безпеки з охорони праці для користувачів ПК

При використанні ПЗ, яке є результатом даної розробки, як і при використанні будь-якого іншого ПЗ, необхідно дотримуватися вимог з охорони праці при роботі з ПК. Розглянемо основні нормативні документи, в яких зазначені вимоги до робочих місць та приміщень при використанні ПК.

У відповідності до вимог, площа для одного робочого місця при використанні розробленого ПЗ повинна бути не менше 6,0 м^2, а об'єм – не менше 20,0 м^3, при цьому робочі місця не можуть бути розміщені у підвальних приміщеннях чи на цокольних поверхах. З метою збереження функції зору, в приміщенні повинне бути рівномірне комбіноване освітлення. Згідно ДБН В, природне освітлення повинно здійснюватися через світлові прорізи, які орієнтовані на північ чи північний схід, і забезпечувати коефіцієнт природного освітлення не менше 1,5%. Прорізи повинні мати жалюзі або завіски. Штучне освітлення мають забезпечувати люмінесцентні лампи. Покриття підлоги повинне мати незначний коефіцієнт відбиття, який знаходиться в межах 0,3-0,5. Також у цих приміщеннях потрібно щодня робити вологе прибирання.

Щодо організації робочого місця, то воно також повинне відповідати вимогам. Насамперед, конструкція робочого місця повинна забезпечувати підтримання оптимальної робочої пози, а конструкція робочого столу має забезпечувати оптимальне розміщення ПК та периферійних пристроїв.

Висота робочого столу, на якому розміщений ПК має знаходитися в межах 680...800 мм, а ширина і глибина – 600...1400 мм і 800..1000 мм відповідно. Стіл також повинен мати достатній простір для ніг, що забезпечить зручну осанку користувача ПК [40-42].

Робочий стілець повинен бути підйомно-поворотним, регульованим за висотою, за кутом і за нахилом сидіння та спинки. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Висота

поверхні сидіння стільця має регулюватися в межах 400...500 мм, а ширина і глибина становити не менше ніж 400 мм. Кут нахилу сидіння – до 15 град. вперед і до 5 град. назад. Висота спинки стільця має становити 300..320 мм, ширина – не менше ніж 380 мм, радіус кривизни горизонтальної площини - 400 мм. Кут нахилу спинки має регулюватися в межах 1...30 град. від вертикального положення. Відстань від спинки до переднього краю сидіння має регулюватися в межах 260...400 мм. Для зниження статичного напруження м'язів верхніх кінцівок слід використовувати стаціонарні або змінні підлокітники. Поверхня сидіння і спинки стільця має бути напівм'якою та не слизькою, з повітронепроникним покриттям.

Монітор ПК має розташовуватися на відстані 600...700 мм від очей користувача. Розташування монітору має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 град. до нормальної лінії погляду працівника [44].

Пристрої вводу (клавіатуру та мишку) слід розташовувати на поверхні столу на відстані 100...300 мм від краю поверхні робочого столу. У конструкції клавіатури має бути опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5...15 град. Висота середнього рядка клавіш має не перевищувати 30 мм. Поверхня клавіатури має бути матовою з коефіцієнтом відбиття 0,4.

Вимоги безпеки при роботі з ПК визначено в НПАОП 0.00-1.28-10. Згідно вимог електробезпеки, ПК повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань. Не допускається підключати ПК до звичайної двопровідної електромережі, в тому числі з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення ПК потрібно виконувати за магістральною схемою. При організації робочих місць електромережу штепсельних розеток для живлення ПК у центрі приміщення прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах.

Щодо безпеки при роботі з ПК, щодня перед початком роботи необхідно очищати монітор від пилу та інших забруднень. Після закінчення роботи з ПК, він та периферійні пристрої повинні бути відключені від електричної мережі. У разі виникнення певної аварійної ситуації необхідно негайно відключити ПК від електричної мережі. Не допускається виконувати обслуговування, ремонт та налагодження ПК безпосередньо на робочому місці.

Під час користування розробленим ПЗ, як і при будь-якій іншій роботі за ПК, необхідно робити перерви по 15 хвилин через кожну годину. Якщо такі перерви неможливо проводити, то тривалість роботи з ПК повинна бути не більше, ніж чотири години.

Основні вимоги до пожежної безпеки вказані в НАПБ «Правила пожежної безпеки в Україні». Згідно цих вимог, у приміщеннях, де знаходиться ПК, обов’язково повинні бути переносні вуглекислотні або аерозольно водопінні вогнегасники. Підходи до засобів пожежогасіння повинні бути вільними. Також у приміщенні, де розміщені ПК, робочі місця повинні бути обладнанні системою автоматичної пожежної сигналізації з димовим пожежним сповіщувачем.

В дипломній роботі виконано дослідження оптимальності REST сервісів для веб-систем створених на різних мовах програмування. Проведена розробка сервісів та тестів проводилась з урахуванням всіх етапів життєвого циклу ПЗ, тому важливо було розглянути основні вимоги до приміщень та робочих місць, де використовують ПК, що й було зроблено в цьому підрозділі. Також були наведені правила електробезпеки під час роботи з ПК та вимоги до пожежної безпеки в приміщенні. Щоб гарантувати безпечні умови праці під час розробки програмного забезпечення, необхідно дотримуватися наведених вимог.

3.2 Медичний захист і забезпечення санітарного та епідемічного благополуччя населення

Враховуючи вибрану мною тему, питання медичного захисту і забезпечення санітарного та епідемічного благополуччя населення Під час роботи за ПК які зазвичай розташовані в приміщеннях з великою кількістю людей всередині при недотриманні санітарного та епідемічного благополуччя можуть виникнути різні типи захворювань з різним рівнем небезпеки для життя людини [44]. Медичний захист і забезпечення санітарного та епідемічного благополуччя населення включає: надання медичної допомоги постраждалим внаслідок надзвичайних ситуацій, рятувальникам та іншим особам, які залучалися до виконання аварійно-рятувальних та інших невідкладних робіт, гасіння пожеж, проведення їх медико-психологічної реабілітації [41]. Медична допомога населенню забезпечується службою медицини катастроф, керівництво якою здійснює центральний орган виконавчої влади, який забезпечує формування та реалізує державну політику у сфері охорони здоров’я; планування і використання сил та засобів закладів охорони здоров’я незалежно від форми власності; своєчасне застосування профілактичних медичних препаратів та своєчасне проведення санітарно-протиепідемічних заходів; контроль за якістю та безпекою харчових продуктів і продовольчої сировини, питної води та джерелами водопостачання; завчасне створення і підготовку спеціальних медичних формувань; утворення в умовах надзвичайних ситуацій необхідної кількості додаткових тимчасових мобільних медичних підрозділів або залучення додаткових закладів охорони здоров’я; накопичення медичного та спеціального майна і техніки; підготовку та перепідготовку медичних працівників з надання екстреної медичної допомоги; навчання населення способам надання домедичної допомоги та правилам

дотримання особистої гігієни; здійснення заходів з метою недопущення негативного впливу на здоров’я населення шкідливих факторів навколишнього природного середовища та наслідків надзвичайних ситуацій, а також умов для виникнення і поширення інфекційних захворювань; проведення моніторингу стану навколишнього природного середовища, санітарно-гігієнічної та епідемічної ситуації; санітарну охорону територій та суб’єктів господарювання в зоні надзвичайної ситуації; здійснення інших заходів, пов’язаних з медичним захистом населення, залежно від ситуації, що склалася [43-45]. Здійснення заходів медичного захисту населення покладається на суб’єктів забезпечення цивільного захисту [45].

ВИСНОВКИ

Зазвичай чомусь так складається що реальну глибину проблеми починаєш усвідомлювати вже як повністю вник в суть проблеми та реально побачив актуальність та необхідність вирішення такої конкретної специфічної задачі. І може до початку повномасштабного вторгнення було достатньо існуючих інструментів, сьогодні вважаю кожне навіть на перший погляд не значне вдосконалення, яке спростить чи автоматизує вирішення поставлених задач, але яке врахує настільки не стандартну ситуацію – це чиєсь життя, здоров’я та спокій. Не стало виключенням в моя робота, зокрема необхідність у впровадженні програмного засобу я повністю усвідомив, коли я сам стикнувся із цим питанням.

Запропонована система є зручною та комфортною у користуванні, її можна оптимізовано використовувати на будь-якому комп’ютері та на платформах операційних систем сімейства Windows.

Найбільш важливою перевагою, якої було досягнуто, вважаю, що під час використання системи, роботи з нею досягається заощадження часу, оскільки програма містить зручний пошук по базі даних, що є набагато швидше ніж інші види пошуку. Крім цього із системою мають можливість повноцінно працювати одразу три групи користувачів: відповідальні особи(реєстратори), волонтери, комендант.

Метою представленої на розгляд екзаменаційної комісії кваліфікаційної роботи на здобуття освітнього рівня «бакалавра» було створення програмного засобу, який буде простим і зрозумілим у експлуатації і, що вважаю найголовнішим, який повністю дасть змогу вирішити проблему обліку тимчасово проживаючих мешканців у гуртожитках та в цілому спрощення всієї процедури розселення.

Розроблена база даних є універсальною, що дає змогу застосовувати її як конкретно для гуртожитку, так і для інших гуртожитків у яких передбачено так звану блокову систему розташування кімнат. Створена база даних дає можливість знаходити потрібну інформацію без необхідності застосування довго тривалого пошуку, що цілком відповідає завданню в його частині з заощадження часу.

Окрім вища наведеного реалізовано можливість внесення змін у дані чи додавання інформації, без втручання та руйнування практичної її будови.

Спроєктована база даних спрощує роботу осіб, які можуть нею користуватись. За допомогою неї можна зберігати інформацію про тимчасово проживаючих в одному місці, без необхідності складання інших видів реєстраційних документів чи ведення паперових журналів.

Вважаю, що цілі, які були поставлені переді мною, як розробника програмного засобу та бази даних були досягнутими повноцінно в повній мірі. Розроблена система має широкий спектр застосувань завдяки можливостям її підлаштовувати під нові виклики.

Таким чином, роботу завершено, проте існує можливість продовження вдосконалення та покращення функціональності реалізованої в програмному засобі. Також базу даних можна розмістити на сервері для спільного доступу із кількох комп’ютерів. Крім цього передбачено можливість вдосконалити інтерфейс програми для більшої зручності програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки до виконання дипломної роботи освітнього рівня ―бакалавр студентами усіх форм навчання для напряму підготовки 121 – ―Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

2. М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – “Інженерія програмного забезпечення” для усіх форм навчання [Текст] – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.

3. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, Я.І. Кінах, Г.Б. Цуприк - Тернопіль: ТНТУ, 2017-38 с

4. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

5. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Моделювання та видобуток даних (висопродуктивні обчислення у великих алгебраїчних та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с.

6. Нога В.В., Цуприк Г.Б. Інформаційні технології при створенні автоматизованих систем для задач збору та збереження інформації. Збірник тез доповідей VII Міжнародної науково-технічної конференції молодих учених та студентів. Актуальні задачі сучасних технологій – Тернопіль 28-29 листопада 2018. — Т. : ТНТУ, 2018. — Том 2. — С. 131.

7. І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк. Дискретні структури (Алгебраїчні та числові системи, комбінаторний аналіз). Навчально-методичний посібник для студентів спеціальності 121 «Інженерія програмного забезпечення», аспірантів та викладачів вищих навчальних закладів. Конспект лекцій : Навчальний посібник / І.В. Бойко, М.Р. Петрик, Г.Б. Цуприк – Тернопіль : ТНТУ , 2017. – 64 с.

8. Windows Vista Business - Microsoft Windows XP Professional [Електронний ресурс] – Режим доступу: URL: https://support.microsoft.com/uk-ua/topic/windows-vista-business-%D1%82%D0%B0-windows-xp-professional

9. Inside AMD's Hammer: the 64-bit architecture behind the Opteron and Athlon 64 [Електронний ресурс] – Режим доступу: URL: https://arstechnica.com/features/2005/02/amd-hammer-1/

10. Документація процесорів Pentium [Електронний ресурс] – Режим доступу: URL: https://www.intel.com/content/www/us/en/resources-documentation/developer.html

11. Електротехнічні параметри процесорів, зокрема Intel Pentium [Електронний ресурс] – Режим доступу: URL: http://www.pchardwarelinks.com/

12. [Електронний ресурс] – Режим доступу: URL: https://www.microsoft.com/en-us/sql-server/sql-server-downloads

13. Роберт Виейра. Програмування баз даних Microsoft SQL Server 2005. Базовий курс: «Діалектика», 2007. — С. 832. — ISBN 0-7645-8433-2.

14. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Освоєння Microsoft SQL Server 2005 : «Діалектика», 2007. — С. 2204. — ISBN 0-7822-4380-6.

15. Кен Хендерсон. Професійне керівництво з SQL Server: структура та реалізація : Издательский дом «Вильямс», 2006. — С. 1056. — ISBN 5-8459-0912-0.

16. Маслов В.П. Інформаційні системи і технології в економіці: [Навчальний посібник] / В.П. Маслов. – Київ: “Слово”, 2003 р. – 320с.

17. Розділ «Microsoft Visual Studio» на сайті Майкрософт [Електронний ресурс] – Режим доступу: URL: https://visualstudio.microsoft.com/

18. Visual C# Developer Center [Електронний ресурс] – Режим доступу: URL: https://visualstudio.microsoft.com/

19. Інформаційні системи та технології на підприємстві : конспект лекцій / І. О. Ушакова, Г. О. Плеханова. – Харків : Вид. ХНЕУ, 2009. – 128 с.

20. Береза А. М. Основи створення інформаційних систем: Навч. посібник. – К.: КНЕУ, 2001. – 214 с.

21. Гужва В. М. Інформаційні системи в міжнародному бізнесі: Навч. посібник / В. М. Гужва , А. Г. Постєвой. – К.: КНЕУ, 1999. – 164 с.

22. Пінчук Н. С. Інформаційні системи і технології в маркетингу: Навч. Посібник / Н. С. Пінчук, Г. П. Галузинський, Н. С. Орленко. – К.: КНЕУ, 1999. – 328 с.

23. Терещенко Л. О. Інформаційні системи і технології обліку: Навч. посіб. / Л. О. Терещенко, І. І. Матвієнко-Зубенко. – К.: КНЕУ, 2005. – 187 с.

24. Т.П. Караванова. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. – К.: Форум, 2012 – 271с.

25. Іан Соммервіллем Інженерія програмного забезпечення = Software Engineering. — 6-е вид.: «Вильямс», 2002. — С. 642. — ISBN 5-8459-0330-0

26. Джек Грінфілд, Кіт Шорт, Стів Кук, Стюарт Кент, Джон Крупи Фабрики розробки програм (Software Factories): потокова збірка типових додатків, моделювання, структури та інструменти = Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools: «Діалектика», 2006. — С. 592. — ISBN 978-5-8459-1181-0

27. [Електронний ресурс] – Режим доступу: URL: https://support.microsoft.com/uk-ua/office/посібник-зі-зв-язків-між-таблицями-30446197-4fbe-457b-b992-2f6fb812b58f

28. [Електронний ресурс] – Режим доступу: URL: https://www.wikiwand.com/uk/Модель_«сутність_-_зв'язок»

29. [Електронний ресурс] – Режим доступу: URL: https://uk.wikibooks.org/wiki/SQL/Типи_даних_MySQL

30. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). The unified modeling language reference manual (англ.). Addison Wesley Longman Inc. ISBN 0-201-30998-X.

31. [Електронний ресурс] – Режим доступу: URL: https://wiko.wiki/uk/C_Sharp_(мова_програмування)

32. [Електронний ресурс] – Режим доступу: URL: https://ppc-ntu-khpi.github.io/oop/modules/вступ/Огляд-технології-Java

33. [Електронний ресурс] – Режим доступу: URL: https://support.microsoft.com/uk-ua/office/створення-нової-бази-даних-32a1ea1c-a155-43d6-aa00-f08cd1a8f01e

34. Спепцифікація OMG IDL [Електронний ресурс] – Режим доступу: URL https://www.omg.org/cgi-bin/doc?formal/02-06-39

35. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc (1999). Testing Computer Software, 2nd Ed. New York, et al: John Wiley and Sons, Inc. c. 480. ISBN 0-471-35846-0.

36. Лайза Кріспін, Джанет Грегорі. Гнучке тестування: практичне керівництво для тестувальників ПЗ та гнучких команд = Agile Testing: A Practical Guide for Testers and Agile Teams: «Вільямс», 2010. — 464 с. — (Addison-Wesley Signature Series) — 1000 прим. — ISBN 978-5-8459-1625-9.

37. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестування програмного забезпечення. Фундаментальні концепції менеджменту бізнес-додатків. — Київ: ДіаСофт, 2001. — 544 с. — ISBN 9667393879.

38. Калбертсон Роберт, Браун Кріс, Кобб Гері. Швидке тестування: «Вільямс», 2002. — 374 с. — ISBN 5-8459-0336-X.

38. The Test Management Guide — A to Z and FAQ Knowledgebase (англ.) Бейзер Б. Тестування чорної скриньки. Технології функціонального тестування програмного забезпечення і систем, 2004. — 320 с. — ISBN 5-94723-698-2.

39. [Електронний ресурс] – Режим доступу: URL: https://www.wiki-data.uk-ua.nina.az/Тестування_програмного_забезпечення.html

40. Дистанційний курс «Основи охорони праці» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: http://dl.tntu.edu.ua/index.php

41. Про затвердження Правил охорони праці під час експлуатації ЕОМ [Електронний ресурс]. – Режим доступу: URL: http://zakon4.rada.gov.ua/laws/show/z0382-99

42. Зеркалов Д.В. Охорона праці в галузі: Загальні вимоги. [Текст] / Д.В. Зеркалов: Навчальний посібник. – К.: «Основа». 2011. – 551 с.

43. Михайлюк В.О., Халмурадов Б.Д. Цивільна безпека: Навчальний посібник. – К.: Центр учбової літератури, 2012, - 158 с.

44. Мохняк С.М., Дацько О.С., Козій О.І., Романів А.С., Петрук М.П., Скіра В.В., Васійчук В.О., Безпека життєдіяльності. Навчальний посібник. Львів. Видавництво НУ "Львівська політехніка", 2012.- 264 с.

45. Осипенко С.І., Іванов А.В. "Організація функціонального навчання у сфері цивільного захисту". Навчальний посібник. – К., 2014. – 286с.

ДОДАТКИ

Додаток А Диск