НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
__інформаційних систем і технологій_
(назва кафедри)

_____ / Швиденко М. З. /
(підпис) (ПІБ)

“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Інформаційна система автоматизованого робочого місця фахівця призовної комісії»

Спеціальність 122 – «Комп’ютерні науки»

Гарант освітньої програми
_____д. ек. н, професор_____ _____ _____ Руденський Р.А_____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
_____ к. ек. н._____ _____ __Стариченко Є. М._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Кондратюк Назар Васильович _____
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
__інформаційних систем і технологій __
(назва кафедри)

___к.ек. н, доцент _____ Швиденко М. З.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
“___” _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Кондратюк Назар Васильович

Спеціальність 122 – «Комп’ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система автоматизованого робочого місця фахівця призовної комісії» затверджена наказом ректора НУБіП України від 25.04.2025 № 699 “С”
2. Термін подання завершеної роботи на кафедру_____02.06.2025 р._____
(рік, місяць, число)
3. Вихідні дані до роботи: надання інформації про облік військовозобов'язаних на території України у вигляді списків та таблиць.
4. Перелік питань, що розглядаються:

- Аналіз проблемної області
- Моделювання предметної області
- Проєктування програмної системи
- Впровадження та експлуатація системи

Дата видачі завдання “25”_квітня_ 2024 р.

Керівник бакалаврської кваліфікаційної роботи _____/ Стариченко Є. М./
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/ Кондратюк Н.В. /
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота присвячена розробці інформаційної системи автоматизованого робочого місця фахівця призивної комісії. Основна мета проєкту – автоматизувати та спростити процес обліку військовозобов'язаних, ведення їх медичної документації та проведення медичних оглядів для визначення придатності до військової служби.

Пропоноване рішення є настільною програмою, розробленою з використанням мови програмування **C#** та фреймворку **WinForms**. **Microsoft SQL Server** використовується як система керування базами даних для забезпечення безпечного та структурованого зберігання інформації.

Розроблена система містить модулі для керування даними призовників, створення та редагування електронних медичних карток, планування та реєстрації результатів медичних оглядів, формування звітів за результатами оглядів. Програмне забезпечення забезпечує зручний інтерфейс і надійну роботу для повсякденного використання працівниками призовної комісії.

Впровадження даного програмного забезпечення підвищить точність та ефективність процесу медичного огляду, забезпечить безпечне та організоване зберігання медичної та персональної документації, зменшить навантаження на персонал комісії.

ABSTRACT

The bachelor's thesis is devoted to the development of software for employees of a conscription commission. The main goal of the project is to automate and simplify the process of registering conscripts, maintaining their medical records, and conducting medical examinations to determine their fitness for military service.

The proposed solution is desktop application developed using the C# programming language and the WinForms framework. Microsoft SQL Server is used as the database management system to ensure secure and structured storage of information.

The developed system includes modules for managing conscript data, creating and editing electronic medical cards, scheduling and recording results of medical examinations, and generating reports based on examination outcomes. The software provides user-friendly interfaces and reliable performance for everyday use by conscription commission staff.

The implementation of this software will improve the accuracy and efficiency of the medical examination process, ensure secure and organized storage of medical and personal records, and reduce the workload of commission personnel.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Отримання завдання	25 квітня 2025	
2	Аналіз предметної області	квітень 2025	
3	Моделювання предметної області	квітень 2025	
4	Проєктування програмної системи	травень 2025	
5	Розгортання та експлуатація програмного забезпечення	травень 2025	
6	Економічне дослідження розробки та експлуатації програмної системи	квітень- травень 2025	
7	Оформлення записки	травень 2025	
8	Перевірка на плагіат	травень 2025	
9	Проходження нормо контроля	травень 2025	
10	Проходження передзахисту	2 червня 2025	
11	Захист роботи	червень 2025	

Студент _____ Кондратюк Н.В.

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

к.ек.н., доцент _____ Стариченко Є. М.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

ЗМІСТ

ВСТУП

У сучасному світі автоматизація адміністративних та медичних процесів стала невід’ємною частиною забезпечення ефективності та точності роботи закладів. Однією з таких сфер, що потребують технологічного прогресу, є робота призовних комісій, які відповідають за реєстрацію та медичне обстеження призовників для визначення їх придатності до військової служби.

Зараз багато призовних комісій все ще покладаються на паперові записи та застарілі програмні засоби, що часто призводить до дублювання даних, трудомісткої ручної роботи та збільшення ймовірності людських помилок. Така неефективність може вплинути як на якість прийняття рішень, так і на загальну швидкість процесу призову. Тому існує очевидна потреба в інтегрованому цифровому рішенні, яке дозволить працівникам призовної комісії безпечно керувати даними, ефективно проводити медичне обстеження та формувати точні звіти.

Метою даної дипломної роботи є покращення поточних способів обліку призовників для працівників призовної комісії, розробивши зручне настільне програмне забезпечення, яке забезпечує комплексний функціонал для працівників призовної комісії. Система забезпечуватиме облік та ведення військовозобов’язаних, створення цифрових медичних карток, документування результатів медичного огляду та присвоєння категорій придатності згідно з типовими положеннями.

Програма розроблена з використанням мови програмування C# у середовищі Windows Forms (WinForms). Microsoft SQL Server використовується для безпечного та ефективного зберігання та керування даними. Програмне забезпечення розроблено таким чином, щоб бути інтуїтивно зрозумілим, надійним і адаптованим до реальних умов роботи персоналу призовної комісії.

Актуальність даної роботи полягає в практичному застосуванні сучасних засобів розробки програмного забезпечення для оптимізації однієї з найважливіших адміністративних функцій військової системи. Завдяки автоматизації рутинних операцій і забезпеченню узгодженості медичних і особистих записів запропоноване рішення має на меті підвищити продуктивність і надійність процесу призову.

Об’єктом дослідження є призовна комісія як організаційна структура, а **предметом** – процес автоматизації її основних функцій, а саме обліку, медичного огляду призовників. У бакалаврській роботі наголошується на інтеграції сучасних практик програмування та технологій баз даних для забезпечення стабільного та масштабованого рішення, адаптованого до потреб персоналу призовної комісії.

Для досягнення цілей проєкту було визначено кілька завдань:

- аналіз поточного робочого процесу призовних комісій;
- визначення вимог користувачів;
- проєктування структури бази даних;
- розробка інтерфейсу та логіки програми;
- проведення тестування для перевірки функціональності та зручності використання.

Результатом є функціональний прототип програмного забезпечення, який може стати основою для подальшого розвитку та потенційного впровадження в реальні призовні комісії.

Структура дипломної роботи включає аналітичний огляд існуючих систем, формулювання вимог до програмного забезпечення, проєктування та впровадження архітектури системи, тестування та валідацію, а також остаточні висновки щодо ефективності системи та можливостей майбутнього розвитку.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Призов — це складна багатоетапна адміністративно-медична процедура, яка включає облік громадян, придатних до військової служби, оформлення особових і медичних книжок, остаточну оцінку їх придатності до служби. Цей процес здійснюється спеціалізованими державними установами — призовними комісіями, які відповідають за дотримання законодавства та належне оцінювання призовників.

Незважаючи на важливість і формальність цієї процедури, багато призовних комісій все ще покладаються на ручне введення даних і фізичне зберігання інформації, що створює ряд проблем. Ведення записів на паперових носіях вразливе до пошкоджень, втрат і людських помилок. Повторювані завдання, такі як заповнення форм, пошук в архівах і підготовка звітів, забирають багато часу та схильні до невідповідностей. За відсутності інтегрованих систем дані не можна легко перевірити, обмінюватися чи аналізувати, що впливає на ефективність і надійність усього процесу призову.

На додаток до адміністративної неефективності, етап медичного огляду представляє свої власні проблеми. Медична документація має бути повною, точною та доступною для лікарів та посадових осіб під час огляду. Проте рукописні нотатки та фрагментарна документація ускладнюють отримання повної картини історії здоров’я призовника. Це може призвести до неправильної класифікації придатності до служби, судових спорів і затримок у обробці [1].

Існуючі програмні рішення, якщо вони взагалі є, часто є застарілими, обмеженими у функціональності або не адаптованими до чинних законодавчих та процедурних потреб призовної системи. Багато з цих систем не мають сучасних інтерфейсів користувача, інтеграції з медичними стандартами або гнучких інструментів звітності. Крім того, такі системи

зазвичай не забезпечують централізоване зберігання чи контроль доступу користувачів, що є критично важливим для забезпечення захисту та конфіденційності даних.

Тому зростає потреба в модернізації процесу призову через розробку спеціалізованої програмної системи, яка б поєднувала адміністративні та медичні функції в одній платформі. Ця система має дозволити працівникам призовної комісії керувати даними призовників, створювати та оновлювати медичні картки, фіксувати результати обстеження та формувати детальні звіти з мінімальними ручними зусиллями. Вирішуючи ці проблеми, програмне забезпечення значно зменшить навантаження на персонал комісії, підвищить точність і послідовність, а також забезпечить кращу відповідність нормативним актам.

Призовна комісія (призовна комісія) є спеціалізованим органом управління, який відповідно до національного законодавства здійснює організацію та контроль за призовом громадян на строкову військову службу. До його основних функцій входить ведення в актуальному стані обліку громадян, придатних до служби, організація та проведення медичних оглядів, присвоєння категорії придатності, прийняття остаточних рішень щодо призову або звільнення.

Робочий процес типової призовної комісії включає кілька основних учасників: працівників комісії, медичних працівників та адміністративний персонал. Кожен із цих учасників відіграє вирішальну роль у процесі оцінювання. На працівника комісії покладається облік військовозобов’язаних, збір та перевірка їх анкетних даних, ведення медичної документації, узгодження прийому до лікарів. Медичні спеціалісти проводять фізичне та психологічне обстеження та видають висновки щодо придатності призовника до служби. За цими результатами комісія визначає остаточну категорію придатності особи.

Предметна область охоплює різні типи даних, включаючи особисту інформацію (повне ім’я, дата народження, адреса, контактні дані), рівень

освіти, статус зайнятості та медичні записи. Медична документація містить інформацію про хронічні захворювання, попередні діагнози, результати обстеження та історію щеплень. Ці набори даних підпадають під дію суворих правил конфіденційності, які вимагають безпечних і добре керованих систем зберігання.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
Призовник	Прізвище, ім’я, по батькові	Повне ім’я особи, яка підлягає військовому обліку
	Дата народження	Число, місяць і рік народження призовника
	Адреса проживання	Поточне місце проживання
	Контактний телефон	Номер телефону для зв’язку
	Освіта	Рівень та назва закладу освіти
	Сімейний стан	Інформація про шлюб, наявність дітей
Медична картка	Група крові	Встановлена група крові призовника
	Хронічні захворювання	Список наявних тривалих захворювань
	Алергії	Наявність алергічних реакцій
	Перенесені операції	Інформація про проведені хірургічні втручання
	Вакцинації	Дані про зроблені щеплення
	Зір, слух, тиск тощо	Основні фізіологічні показники стану здоров’я
Медичний огляд	Дата огляду	Коли було проведено медичне обстеження
	Лікарі, що проводили огляд	Прізвища та спеціалізації лікарів
	Висновок лікарської комісії	Рішення щодо придатності до військової служби
	Категорія придатності	Наприклад: «А – придатний», «Б – обмежено придатний», тощо

Зараз більшість інформації збирається вручну та зберігається в паперовому вигляді або в ізольованих базах даних, що перешкоджає ефективному доступу, оновленню та синхронізації. Це ускладнює швидке отримання інформації про призовника, відстеження змін у медичному стані чи

формування звітів для вищих інстанцій. Це також збільшує ймовірність помилок, дублювання записів або втрати важливих даних [1].

Модернізація цієї сфери програмним рішенням, адаптованим до конкретних потреб призовних комісій, допоможе вирішити ці завдання. Завдяки оцифровці записів і автоматизації ключових процесів програмне забезпечення може значно підвищити швидкість, точність і прозорість операцій призовної комісії. Крім того, це забезпечить дотримання стандартів захисту даних, спростить прийняття рішень медичним персоналом та забезпечить загальне покращення якості послуг, які надає система призову.

1.2 Огляд існуючих рішень

На сьогоднішній день в Україні та за її межами існує обмежена кількість спеціалізованих програмних рішень, орієнтованих саме на автоматизацію роботи працівників військкомату. Більшість військових установ користуються або застарілими внутрішніми програмами, або загальними інформаційними системами, які не враховують специфіку процесу призову.

Одним із небагатьох програмних продуктів, спрямованих саме на забезпечення роботи працівників призовних комісій, є автоматизована система АРМ «Призовник». Це рішення було створено для оцифрування та оптимізації управління призовниками, зменшення залежності від ручної роботи з документами та спрощення доступу до особистих записів. Програмне забезпечення дозволяє користувачам зберігати та керувати вичерпною інформацією про призовників, включаючи особисті дані, результати медичних оглядів та рішення щодо їх придатності до служби [3].

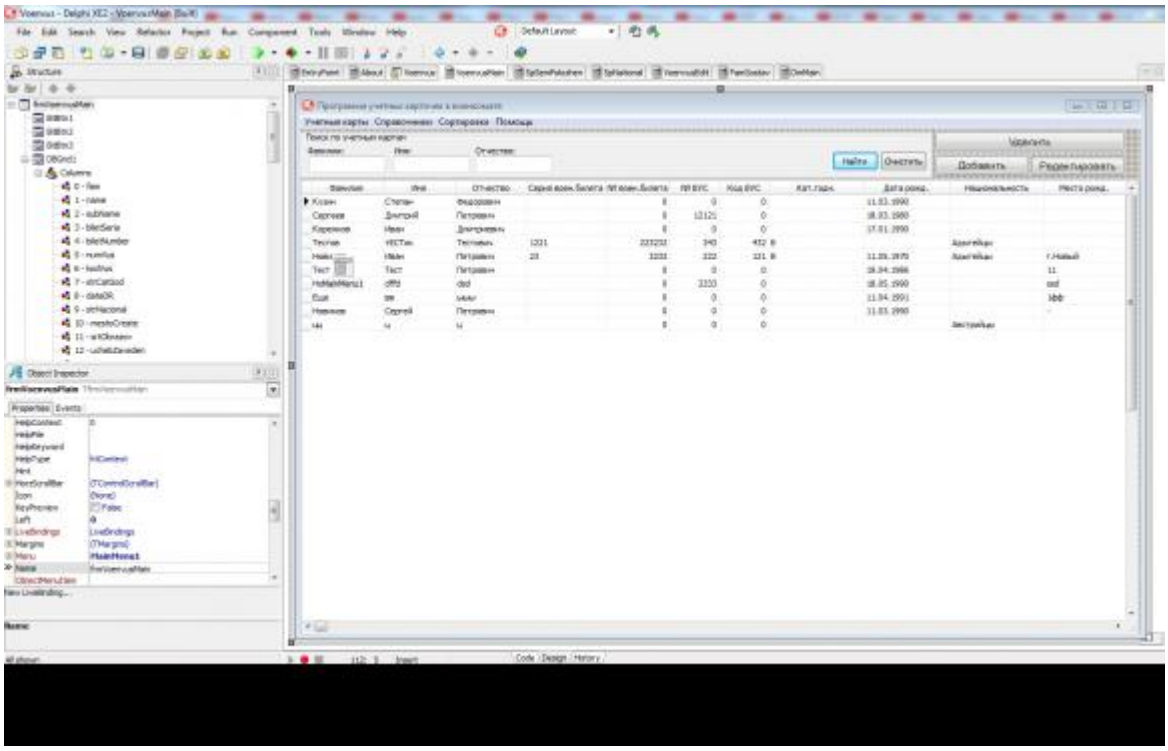


Рис. 1.1 – Автоматизоване робоче місце “Призовник”

АРМ «Призовник» оснащений інструментами для формування стандартних документів, таких як призовні, медичні висновки, зведені листи. Він пропонує параметри фільтрації на основі різних критеріїв, таких як категорія здоров’я чи призовна кампанія, що допомагає організувати й обробляти дані ефективніше. При цьому система була розроблена вузьконаправлена, лише на базові потреби призовних ділянок.

Незважаючи на ці переваги, програмне забезпечення має кілька суттєвих недоліків. Його інтерфейс здається застарілим за сучасними стандартами і може бути не інтуїтивно зрозумілим для нових користувачів. Жорстка архітектура обмежує гнучкість, що ускладнює зміну або розширення функціональності системи. Крім того, йому бракує надійних функцій безпеки для захисту конфіденційної особистої інформації, а його обмежена підтримка багатокористувацьких середовищ може перешкоджати одночасному доступу співробітників. Примітно, що АРМ «Призовник» не пропонує хмарної або веб-версії, що обмежує його використання в більш динамічних або децентралізованих середовищах.

Враховуючи ці обмеження, хоча система все ще може служити базовим інструментом для керування даними про призов, вона більше не відповідає вимогам сучасної цифрової інфраструктури. Зростає потреба у більш досконалому та зручному для користувача рішенні, яке поєднує в собі сучасний дизайн, покращений захист даних і гнучкі можливості інтеграції. Додаток наступного покоління також має відповідати мінливим юридичним і технічним вимогам до обробки даних у державних установах, пропонуючи більш безпечне та ефективне середовище для роботи призовної комісії [3].

Розглянемо інше програмне рішення на ринку.

Мегаполіс Документообіг — це одна з популярних систем для автоматизації документообігу, яка використовується в організаціях різних рівнів, включаючи державні установи. Вона створена для оптимізації процесів зберігання, обробки та передачі документів у рамках підприємства або установи. Система дозволяє керувати внутрішніми та зовнішніми документами, а також забезпечує зручний доступ до інформації для всіх користувачів [2].

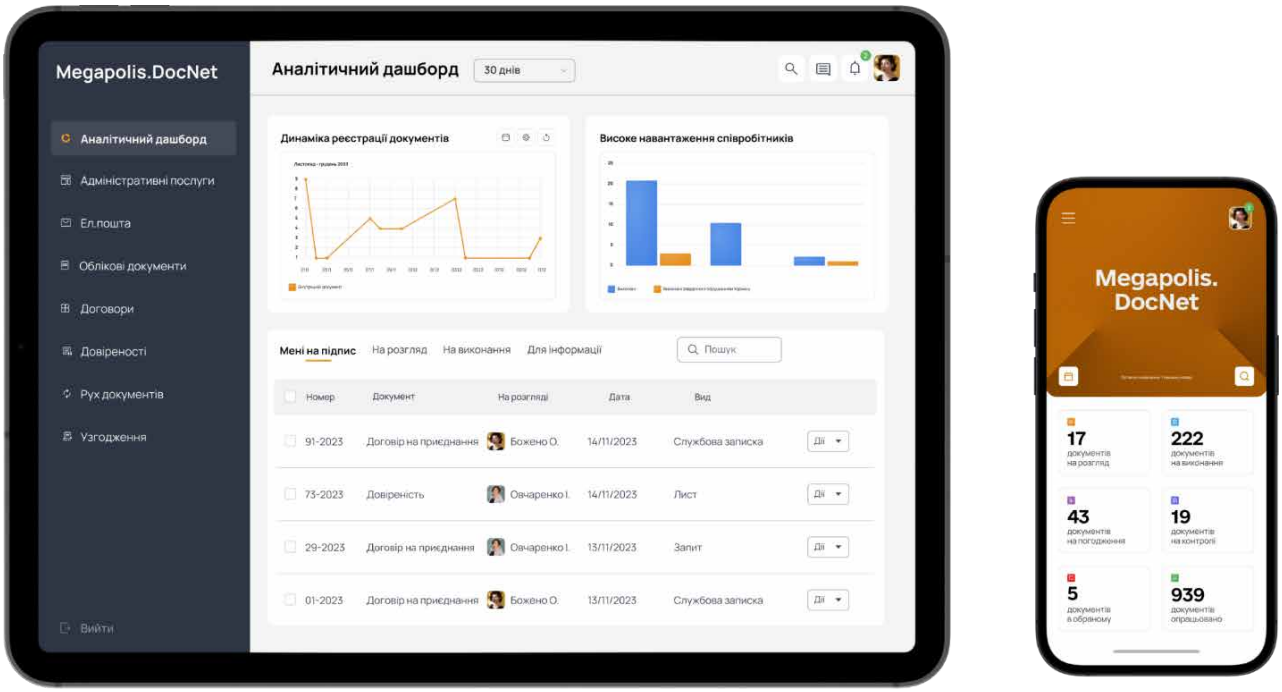


Рис. 2.1 – Інформаційна система “Мегаполіс. Документообіг”

Основною особливістю Мегаполіс. Документообіг є його здатність централізовано зберігати документи в електронному вигляді та надавати доступ до них через зручний інтерфейс. Користувачі можуть створювати, редагувати, підписувати та погоджувати документи, що суттєво знижує час на обробку інформації. Система також включає модулі для автоматизації процесів узгодження та контролю, що дозволяє знизити кількість помилок і зберігати в повному обсязі всю історію змін документів.

Проте, незважаючи на безліч переваг, система має й певні недоліки. Основним обмеженням є те, що вона розрахована на широке використання в комерційних і державних структурах загального призначення, тому для специфічних задач, таких як автоматизація роботи військових комісаріатів, її функціональність може виявитися не достатньою. Система не включає в себе можливості для ведення медичних карток, реєстрації призовників або визначення придатності до служби, що є важливими функціями для військових установ.

Окрім того, для повної інтеграції Мегаполіс. Документообіг з іншими системами, наприклад, з медичними базами даних чи реєстрами призовників, необхідна додаткова настройка та розробка інтерфейсів. Крім того, існуюча архітектура системи не дозволяє легко адаптувати її до потреб певних відомств без істотних змін у програмному коді.

Таким чином, хоча Мегаполіс. Документообіг є потужним інструментом для загального документообігу, для специфічних потреб, таких як облік призовників та їх медичних оглядів, необхідне створення спеціалізованого програмного забезпечення, яке враховує всі специфічні вимоги та функції.

1.3 Постановка завдання

Основна мета програмного забезпечення для працівника призовної комісії – спростити та автоматизувати завдання, пов’язані з веденням даних

призовників та медичних оглядів. Система сприятиме створенню, зберіганню та пошуку облікових записів призовників, відстежуватиме їх медичне обстеження та допоможе працівникам призовної комісії приймати обґрунтовані рішення щодо придатності призовника до військової служби.

Щоб досягти цього, програма потребуватиме добре структурованої бази даних, яка зберігає повну інформацію про призовників, включаючи особисті дані, історію хвороби та результати обстеження. Ця база даних забезпечить легкий доступ до важливих даних і генеруватиме персоналізовані сповіщення на основі медичних оцінок, наприклад, коли призовника визнають непридатним до служби або потребує подальшої медичної допомоги.

Програмне забезпечення буде зосереджено на таких ключових функціях:

- програма дозволить працівнику призовної комісії зберігати та оновлювати персональні дані призовників, такі як ім’я, вік, адреса та контактна інформація;
- система відстежуватиме медичні записи кожного призовника, включаючи історію хвороби, результати медичних оглядів і остаточне медичне рішення (придатний, частково придатний або непридатний);
- на основі медичного огляду система повідомить користувача, якщо призовник визнано придатним або непридатним до військової служби, і надасть додаткові відомості, якщо це необхідно (наприклад, рекомендоване лікування або подальші тести);
- програмне забезпечення автоматично створюватиме звіти про статус призовника та надсилатиме сповіщення про необхідні дії, такі як оновлення медичних оцінок або подальше спостереження за призовниками, які не склали іспити.

Основна програма буде зручною для користувача та працюватиме в простому діалоговому режимі на основі системи меню. Користувачі можуть легко переміщатися між параметрами додавання нових військовозобов’язаних, оновлення існуючих записів, проведення медичних

оглядів і створення звітів. Крім того, користувачі зможуть при необхідності скасувати або відмінити раніше зареєстровані дії.

Програмна система підвищить ефективність роботи призовних комісій, оптимізує робочий процес і забезпечить точне та своєчасне управління призовниками. Завдяки інтеграції цих функціональних можливостей в єдину програмну платформу призовна комісія буде краще оснащена для управління процесом призову та дотримання нормативних актів.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. програмне забезпечення має дозволяти працівнику призовної комісії безпечно входити в систему за допомогою імені користувача та пароля;
2. необхідно підтримувати різні ролі користувачів (наприклад, адміністратора, медичного працівника, військовослужбовця), кожна з яких має різні рівні доступу до функцій системи;
3. система має дозволяти створення, оновлення та видалення військової служби;
4. кожен призовник має містити особисту інформацію, а саме: ім’я, дату народження, адресу та контактні дані;
5. система повинна зберігати та відображати медичну документацію призовників, включаючи результати медичних оглядів, діагнози та остаточний стан здоров’я (придатний, частково придатний, непридатний);
6. програмне забезпечення має дозволяти медичним працівникам вводити результати медичних оглядів, наприклад фізичні тести, оцінки стану здоров’я та результати діагностики;
7. програма має передбачати можливість класифікації військовозобов’язаних за станом здоров’я (наприклад, придатний до служби, непридатний до служби, потребує додаткового обстеження);
8. система сповіщень повинна сповіщати користувачів про призовників, які потребують повторних медичних оглядів або не пройшли обстеження;

9. система повинна автоматично розраховувати класифікацію військовозобов’язаного за придатністю на основі результатів медичного огляду та попередньо визначених критеріїв (наприклад, придатний, непридатний, тимчасово непридатний).

Нефункціональні вимоги:

1. система повинна мати можливість обробляти велику кількість записів про призовників (наприклад, тисячі) без істотного зниження продуктивності;
2. система повинна забезпечувати обробку даних у режимі реального часу, зокрема при формуванні звітів або поновленні військової служби;
3. інтерфейс користувача має бути інтуїтивно зрозумілим і легким для навігації для всіх типів користувачів, включаючи офіцерів призовної комісії, медичного персоналу та системних адміністраторів;
4. новим користувачам має знадобитися мінімальне навчання для виконання основних функцій, таких як введення даних призовників або створення звітів;
5. система має бути масштабованою для майбутнього зростання, наприклад, додаткових призовників, нових ролей користувачів або інтеграції з іншими системами (наприклад, національними медичними записами чи іншими державними базами даних);
6. це має дозволяти легке оновлення та доповнення до функціональних можливостей системи без значного перепроєктування чи простою системи;
7. система повинна бути високонадійною і працювати з мінімальними простоями;
8. він повинен містити механізми відновлення після помилок, щоб гарантувати, що дані не будуть втрачені у разі збою або несподіваного завершення роботи.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача (UI) програмного забезпечення, призначеного для працівників призовної комісії, має віддавати перевагу зрозумілості, простоті та зручності використання. З огляду на те, що основні користувачі

можуть не мати великого досвіду роботи зі складними системами, інтерфейс повинен сприяти інтуїтивно зрозумілій взаємодії та оптимізувати повсякденні завдання. Послідовна мова дизайну на всіх екранах, що підтримується логічним макетом і чіткою візуальною ієрархією, допомагає підтримувати зручність роботи.

В основі системи лежить екран входу, який забезпечує швидкий доступ через мінімалістичну форму для введення імені користувача та пароля з посиланнями підтримки для відновлення пароля або зв'язку з адміністратором. Після автентифікації користувача вітає централізована інформаційна панель. Ця область діє як центр управління, пропонуючи огляд ключових статистичних даних, таких як кількість призовників, заплановані медичні огляди та останні дії. Панель навігації забезпечує плавний доступ до всіх розділів системи, включаючи призовні записи, медичні дані, результати обстеження, звіти та налаштування.

Модуль для керування призовниками представляє список з можливістю пошуку та сортування, який дозволяє швидко фільтрувати за різними критеріями, такими як ім'я чи статус іспиту. Вибір призовника відкриває детальний профіль, розділений на інтуїтивно зрозумілі вкладки, що охоплюють особисті дані, медичні записи, історію обстежень та остаточну оцінку. Форми структуровані логічно, що забезпечує зрозумілість введення даних. Там, де це можливо, поля введення доповнюються спадними меню та прапорцями для забезпечення стандартизації та зменшення помилок. Інтерфейс також дозволяє завантажувати важливі документи, такі як скановані документи або медичні довідки.

Управління медичною карткою здійснюється за допомогою структурованої форми, яка містить життєво важливу інформацію про здоров'я, результати тестів, оцінки психічного здоров'я та примітки лікаря. Ці записи призначені для легкого редагування та збереження з опціями очищення або друку форми. Крім того, інтерфейс медичного огляду дозволяє персоналу

переглядати графіки та ефективно вводити результати. На основі введених даних система може запропонувати пропозиції щодо класифікації призовника.

Створення звітів і керування ними спрощено завдяки спеціальному розділу, який містить попередньо визначені шаблони для типових запитів. Користувачі також можуть створювати власні звіти за допомогою гнучких фільтрів і експортувати результати в такі формати, як PDF або Excel. Сповіщення в режимі реального часу відіграють важливу роль у продуктивності користувачів, висвітлюючи незавершені дії, такі як незавершені записи або майбутні обстеження, за допомогою сповіщень, позначених кольором, щоб вказати пріоритет [5].

Системні налаштування пропонують інструменти для керування обліковими записами користувачів, призначення ролей і безпеки даних, включаючи функції резервного копіювання та відновлення. Параметри мови та функції доступності також можна налаштувати відповідно до індивідуальних потреб. Для подальшої підтримки користувачів інтегрований розділ довідки містить цифровий посібник, поширені запитання та контактну форму для технічної допомоги.

Щоб покращити загальний досвід, інтерфейс містить запити підтвердження, щоб уникнути ненавмисних дій, функції автоматичного збереження, щоб запобігти втраті даних, і чіткі індикатори прогресу під час роботи з багатоетапними процесами або великими завантаженнями даних. Загалом дизайн і функціональність інтерфейсу користувача забезпечують доступність, надійність та оптимізованість програмного забезпечення для роботи працівників призовної комісії.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) — це стандартизована візуальна мова, яка використовується для моделювання та проєктування систем програмного забезпечення. Він служить потужним інструментом для розробників, аналітиків і зацікавлених сторін, щоб повідомити структуру та поведінку системи до її впровадження. Діаграми UML забезпечують чіткий, структурований спосіб представлення різних аспектів програмної системи, від загальної архітектури до окремих компонентів та їх взаємодії.

UML допомагає подолати розрив між технічними та нетехнічними членами команди, використовуючи легко зрозумілі візуальні символи для представлення складної системної логіки. Він підтримує різні методології розробки, включаючи об’єктно-орієнтовані, гнучкі та ітераційні моделі, що робить його гнучким вибором для різноманітних програмних проєктів. [6]

У контексті розробки програмного рішення для працівника призовної комісії, UML має важливе значення для планування та організації системи. Це дозволяє візуалізувати такі сутності, як призовники, медичні записи, процедури обстеження та ролі користувачів, а також те, як вони взаємодіють у системі. Такі діаграми, як діаграми варіантів використання, пояснюють, як користувачі взаємодіятимуть із програмою, а діаграми класів показують структуру бази даних і зв’язок між даними. Діаграми діяльності та діаграми послідовності дають змогу зрозуміти робочі процеси та потік даних, дозволяючи групі розробників розробляти логіку, яка відображає реальні операції.

Використовуючи UML на етапі планування, розробники можуть завчасно виявити потенційні проблеми, зменшити непорозуміння та переконатися, що функціональні можливості системи відповідають потребам користувачів. Крім того, діаграми UML служать документацією, яка

залишається корисною протягом життєвого циклу програмного забезпечення, від початкової розробки до майбутніх оновлень і обслуговування. Таким чином, включення UML у процес розробки значно сприяє створенню надійної та добре організованої програми.

У розробці будь-якої програмної системи діаграми відіграють вирішальну роль у візуалізації архітектури, процесів і взаємодії між різними компонентами. Вони служать важливим інструментом спілкування між розробниками, аналітиками та зацікавленими сторонами, дозволяючи кожному зрозуміти, як структурована система та як вона поводиться в різних умовах.

Для програмного забезпечення, призначеного для допомоги працівникам призовних комісій, особливо актуальні кілька типів діаграм. Кожен служить певній меті, допомагаючи визначити як структурні, так і поведінкові аспекти системи [7].

Діаграми випадків використання ілюструють взаємодію між користувачами (такими як працівники призовної комісії, медичний персонал і системні адміністратори) і самою системою. Вони допомагають визначити ключові функціональні можливості з точки зору користувача, такі як керування записами про призовників, проведення медичних оглядів або створення звітів.

Діаграми класів забезпечують детальне уявлення про модель даних системи. Вони представляють основні сутності, як-от призовники, медичні картки, користувачі та іспити, і показують, як ці сутності пов’язані одна з одною через атрибути та асоціації. Ця діаграма особливо корисна для проєктування бази даних і розуміння об’єктно-орієнтованої структури.

Діаграми діяльності описують робочий процес різних процесів у системі. Наприклад, вони можуть проілюструвати, як медичне обстеження проводиться від початку до кінця, показуючи умовні гілки, паралельні завдання та моменти прийняття рішень у чіткому, покроковому форматі.

Діаграми послідовності зосереджені на взаємодії в часі між різними компонентами системи. Вони корисні для зображення того, як об’єкти спілкуються у відповідь на конкретні події, наприклад, для отримання історії хвороби призовника під час підготовки до обстеження.

Діаграми компонентів і розгортання також можна використовувати для представлення фізичної та логічної архітектури системи, що особливо корисно на наступних етапах розробки або під час планування розгортання програмного забезпечення на реальному обладнанні.

Загалом, ці діаграми не тільки покращують розуміння та співпрацю під час розробки, але також служать цінною документацією для майбутнього обслуговування та оновлень. Вони забезпечують чітке візуальне представлення як технічної, так і функціональної частин системи, забезпечуючи добре організоване та ефективне програмне рішення.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання є одним із ключових інструментів моделювання в UML, який зосереджується на охопленні функціональних вимог системи з точки зору її користувачів. Замість детального опису того, як реалізована система, він показує, що система робить і хто з нею взаємодіє. Це робить діаграми варіантів використання особливо цінними на ранніх стадіях розробки програмного забезпечення, коли розуміння потреб користувачів і функціональності системи має вирішальне значення [8].

У контексті програмного забезпечення для працівника призовної комісії діаграма варіантів використання надає загальнорівневий огляд основних функцій системи та ролей, які з ними взаємодіють. Він візуально представляє акторів (користувачів або зовнішні системи) і варіанти використання (конкретні завдання чи операції), які вони можуть виконувати.

Звичайні учасники такої системи можуть включати працівника призовної комісії, медичного працівника та адміністратора.

Кожен актор пов’язаний із варіантами використання, за які він відповідає, показуючи їхню взаємодію з системою. Наприклад, медичний офіцер може бути пов’язаний із випадками використання, як-от «Заповнити медичну картку» або «Оцінити призовника», тоді як адміністратор може отримати доступ до «Керування користувачами» або «Налаштування параметрів системи».

Метою діаграми варіантів використання є не просто документація — вона допомагає всім зацікавленим сторонам (розробникам, тестувальникам і особам, які приймають рішення) зрозуміти, як користувачі взаємодіятимуть із програмним забезпеченням, гарантуючи, що всі необхідні функції визначено та враховано в системі. Він закладає міцну основу для подальшого моделювання та розвитку, зосереджуючись на ефективному задоволенні реальних потреб користувачів.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

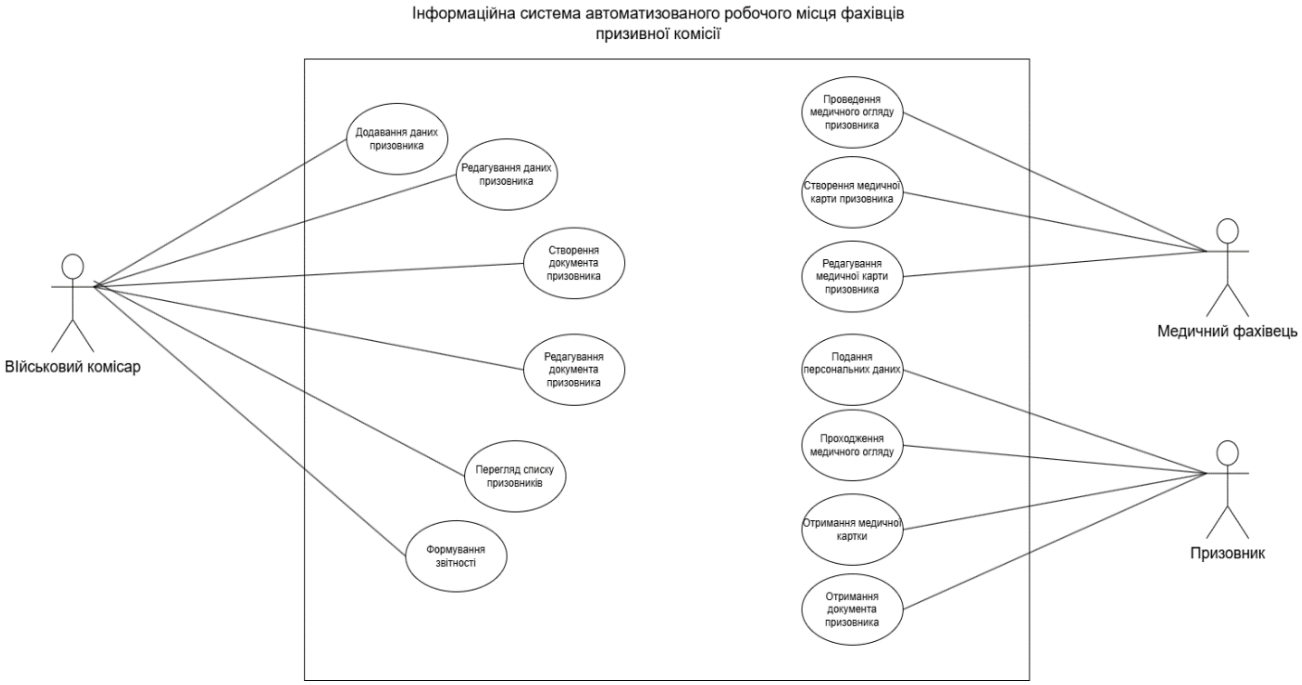


Рис. 2.1 – Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Військовий комісар”;
- “Медичний фахівець”;
- “Призовник”.

Актор «Військовий комісар» включає такі прецеденти:

- додавання даних призовника;
- редагування даних призовника;
- створення документа призовника;
- редагування документа призовника;
- перегляд списку призовників;
- формування звітності.

Актор «Медичний фахівець» включає такі прецеденти:

- проведення медичного огляду призовника;
- створення медичної карти призовника;
- редагування медичної карти призовника.

Актор «Призовник» включає такі прецеденти:

- подання персональних даних;
- проходження медичного огляду;
- отримання медичної картки;
- отримання документа призовника.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Сценарій використання: Створення документа призовника

Актор: військовий комісар

Мета: Зареєструвати новоспеченого призовника в системі, сформувавши офіційний обліковий запис (документ).

Передумови: Військовий комісар аутентифікований у системі та має відповідні права доступу.

Опис сценарію:

1. військовий комісар входить до системи програмного забезпечення призовної комісії за допомогою захищених облікових даних. З головної інформаційної панелі вони переходять до розділу «Записи призовників». Увійшовши до цього розділу, комісар натискає кнопку «Додати», після чого відкривається чиста форма приписки призовника;
2. форма містить поля для введення персональних даних, таких як ПІБ, дата народження, місце проживання, контактна інформація, паспорт або ідентифікаційний номер, освіта та статус роботи. Додаткові поля дозволяють вводити сімейний стан, попередню військову службу (за наявності) та будь-які відповідні примітки чи зауваження;
3. військовий комісар заповнює обов'язкову інформацію, перевіряючи заповнення всіх важливих полів. У відповідних випадках система може запропонувати стандартні параметри за допомогою розкривних списків (наприклад, для рівня освіти чи регіону). Програмне забезпечення автоматично перевіряє дані, щоб запобігти дублікатам або неправильним записам;
4. після перевірки достовірності наданої інформації комісіонер натискає «Зберегти». Система підтверджує успішне створення призовного документа та присвоює новододаній особі унікальний ідентифікаційний номер. Цей документ тепер видно в журналі призовників і доступний для подальших дій, таких як медичне обстеження та оцінка придатності до служби;
5. якщо під час введення даних будуть виявлені помилки, система запропонує користувачеві їх виправити, перш ніж продовжити. Крім того, комісар може завантажити відскановані документи (наприклад, копії посвідчень або довідок) до профілю призовника.

Альтернативні потоки:

1. якщо призовник уже існує в системі (на основі збігу ідентифікатора або імені), програмне забезпечення попереджає комісара та запобігає створенню дублікатів запису;

2. якщо сеанс закінчився, незбережені дані видаляються, і комісару пропонується увійти знову.

Цей сценарій ілюструє типову реальну дію військового комісара та відображає практичний процес підготовки до призову. Він забезпечує систематичний збір і зберігання всієї важливої інформації для подальшого використання в оцінках, звітності та юридичній документації.

Розглянемо ще один сценарій використання.

Сценарій використання: проходження медичного огляду

Актор: призовник

Мета: Пройти плановий медичний огляд в рамках призову.

Умови: Призовник зареєстрований в системі та має медичну картку, оформлену працівником призовної комісії. Медична комісія готова прийняти призовників на зазначену дату.

Опис сценарію:

1. призовник прибуває до медпункту призовної комісії в призначений день і ідентифікується співробітником або за унікальним номером призовника, або за персональними даними, які зберігаються в системі. Після перевірки військовозобов’язаний додається до списку осіб, яким заплановано огляд у цей день;
2. медичне обстеження складається з кількох етапів, кожен з яких проводять лікарі-спеціалісти (наприклад, терапевт, психіатр, хірург, офтальмолог). Коли призовник просувається по кожній станції, лікарі отримують доступ до його медичної картки через програмний інтерфейс і вводять результати своїх оцінок. Це включає результати тестів, діагнози, нотатки щодо фізичного та психічного здоров’я та будь-які відповідні спостереження;
3. кожен лікар на підставі своєї оцінки присвоює попередню категорію здоров'я. Після завершення всіх обстежень збирається медична комісія, яка розглядає зведені результати та приймає остаточне рішення щодо придатності призовника до військової служби. Програмне забезпечення розраховує або відображає запропоновану загальну категорію (наприклад, придатний,

обмежено придатний, непридатний) на основі попередньо встановлених правил і даних лікарів;

4. остаточний вердикт вноситься в систему і зберігається в цифровій медичній картці призовника. За потреби можна створити друковане резюме. Після цього призовник повідомляється про результат, і результат стає доступним для подальшої обробки працівниками призовної комісії.

Альтернативні потоки:

1. якщо призовник відмовляється або не може пройти певне обстеження, система позначає етап як незавершений і позначає його для повторного розгляду;
2. якщо потрібні додаткові обстеження (наприклад, рентген, психологічне тестування), система призначає інший прийом і фіксує це в анкеті призовника;
3. у разі помилки під час введення даних система дозволяє уповноваженому персоналу виправити або оновити інформацію з належним протоколюванням.

Цей сценарій демонструє, як процес медичного огляду управляється цифровим способом, забезпечуючи прозорість, точність і послідовність оцінки придатності призовників. Інтеграція всіх етапів у централізовану програмну систему мінімізує ризик помилок та сприяє ефективному прийняттю рішень комісією.

2.2.2 Діаграма послідовності. Діаграма послідовності — це інструмент, який використовується в розробці програмного забезпечення для візуального представлення взаємодії між різними частинами системи з часом. Він зосереджується на тому, як об’єкти або учасники спілкуються один з одним через обмін повідомленнями, ілюструючи логічний хід конкретного процесу чи сценарію. Замість того, щоб описувати статичні структури, діаграми послідовності підкреслюють часовий порядок, у якому відбуваються дії, допомагаючи розробникам краще зрозуміти поведінку системи [7].

У контексті програмного рішення, розробленого для працівників призовної комісії, діаграми послідовності можуть відігравати вирішальну роль у моделюванні ключових взаємодій. Наприклад, під час медичного огляду призовника на схемі може бути показано, як система взаємодіє з

користувачами, такими як медичні працівники, як інформація отримується з бази даних або записується в базу даних, а також як результати обробляються та зберігаються. Це забезпечує прозоре уявлення про внутрішній робочий процес, що стоїть за, здавалося б, простими операціями.

Кожен учасник взаємодії, наприклад користувач, програмний модуль або база даних, представлений вертикальною лінією, яку часто називають лінією життя. Повідомлення, якими обмінюються лінії життя, намальовані у вигляді стрілок, і кожна стрілка відображає операцію, запит або відповідь. Над цими життєвими лініями смужки активації підсвічують періоди, коли об’єкт активно виконує дію або чекає на відповідь.

Наприклад, у сценарії призову, коли призовник прибуває на медичний огляд, система спочатку перевіряє його особу. Потім інтерфейс користувача може надіслати запит до серверної служби, щоб отримати медичну карту призовника. По ходу експертизи різні спеціалісти додають свої висновки до цього протоколу. Після завершення система аналізує дані для визначення остаточної категорії придатності, зберігає її та відповідно оновлює профіль призовника. Усі ці кроки можна чітко показати за допомогою добре структурованої діаграми послідовності.

Такі діаграми — це більше, ніж просто наочна допомога — вони служать важливою документацією, яка сполучає спілкування між розробниками, аналітиками та експертами в області. Відстежуючи, як системні компоненти взаємодіють у реальному часі, вони допомагають переконатися, що кінцевий програмний продукт працює належним чином, відповідає потребам користувача та є стійким до помилок.

Діаграма послідовності, зображена на рис. 2.2, включає наступні об'єкти:

- “Військовий комісар”;
- “Медичний фахівець”;
- “Призовник”.

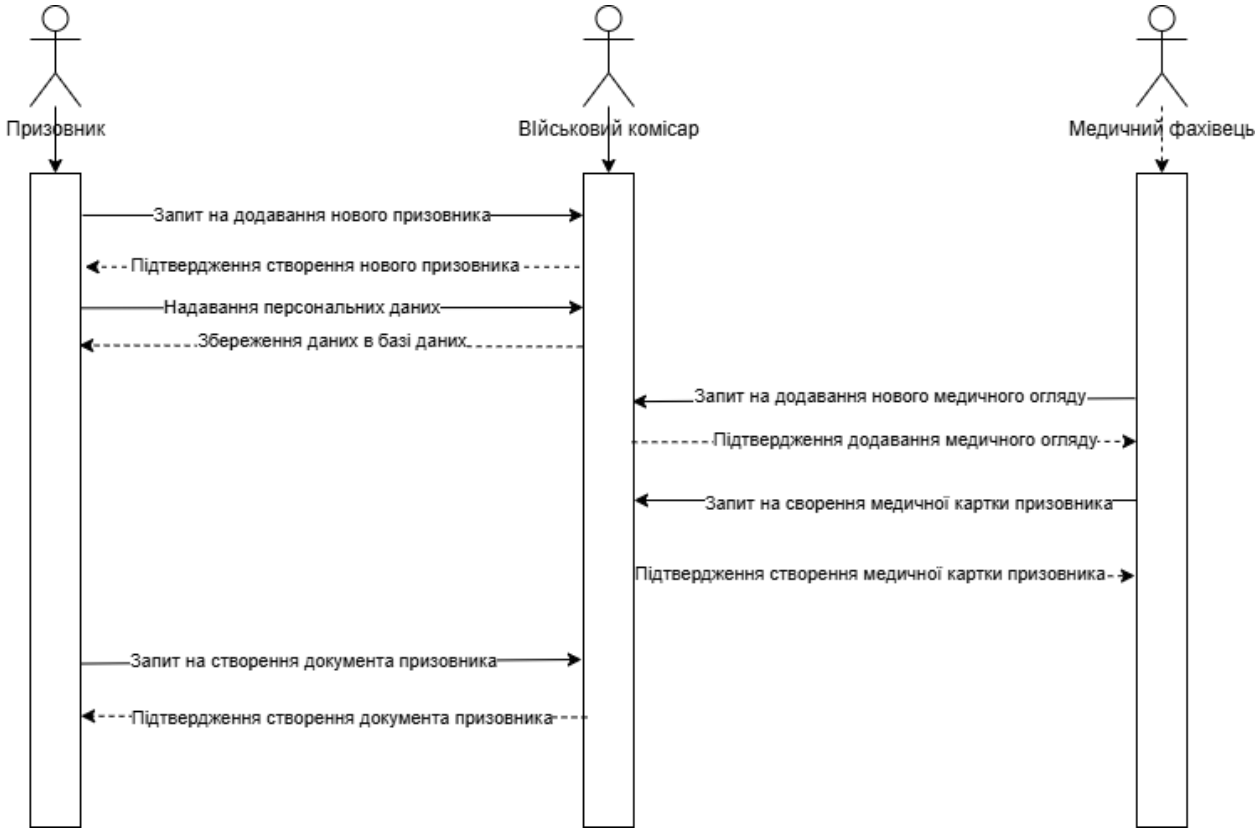


Рис. 2.2 - Діаграма послідовності

На схемі показано взаємодію трьох ключових учасників — призовника, військового комісара та медичного працівника, підкреслюючи послідовність їхніх дій. Призовник ініціює процес, запитуючи документацію або надаючи нові особисті дані для бази даних. Цей крок передбачає надсилання індивідуальної інформації, яка надійно зберігається та супроводжується підтвердженням успішного завершення цих операцій.

Водночас військовий комісар виконує організаційну роль, забезпечуючи проходження призовником медичного огляду та стежачи за оформленням медичної книжки. Цей процес включає перевірку виконання цих завдань. Медичний працівник, який є невід’ємною частиною процесу, проводить фізичний огляд і гарантує, що медична карта створена належним чином і перевірена на точність.

Діаграма ефективно відображає структуровану взаємодію між цими ролями та підкреслює, як кожна дія взаємопов’язана з потоком операцій. Він

дає чіткість щодо обов’язків кожного учасника, пропонуючи зрозуміти динаміку їхньої співпраці.

2.2.3 Діаграма активності. Діаграма діяльності — це тип діаграми UML (Unified Modeling Language), який використовується для моделювання динамічного потоку керування в системі. Він візуально представляє послідовність дій або кроків у процесі, показуючи, як дії виконуються від початку до кінця, і як керуються різними рішеннями, циклами та паралельними процесами [9].

Діаграми діяльності зазвичай використовуються для моделювання бізнес-процесів, робочих процесів або будь-якої ситуації, коли потік контролю проходить через ряд дій або дій. На відміну від діаграм послідовності, які зосереджені на взаємодії між об’єктами, діаграми діяльності висвітлюють потік контролю та те, як здійснюються різні процеси, часто допомагаючи проілюструвати складну поведінку більш зрозумілим способом.

На діаграмі діяльності дії або завдання представлені у вигляді заокруглених прямокутників, а потік керування позначається стрілками, що з’єднують ці завдання. Вузли прийняття рішень використовуються для представлення точок розгалуження, де процес може слідувати одним із кількох можливих шляхів залежно від умови. Вузли злиття об’єднують різні потоки, тоді як розгалуження та об’єднання використовуються для демонстрації паралельної обробки. Початкова точка (позначена зафарбованим

колом) і кінцева точка (показана у вигляді кола з рамкою) допомагають позначити початок і завершення дії.

Розроблена діаграма активності представлена на рис. 2.3

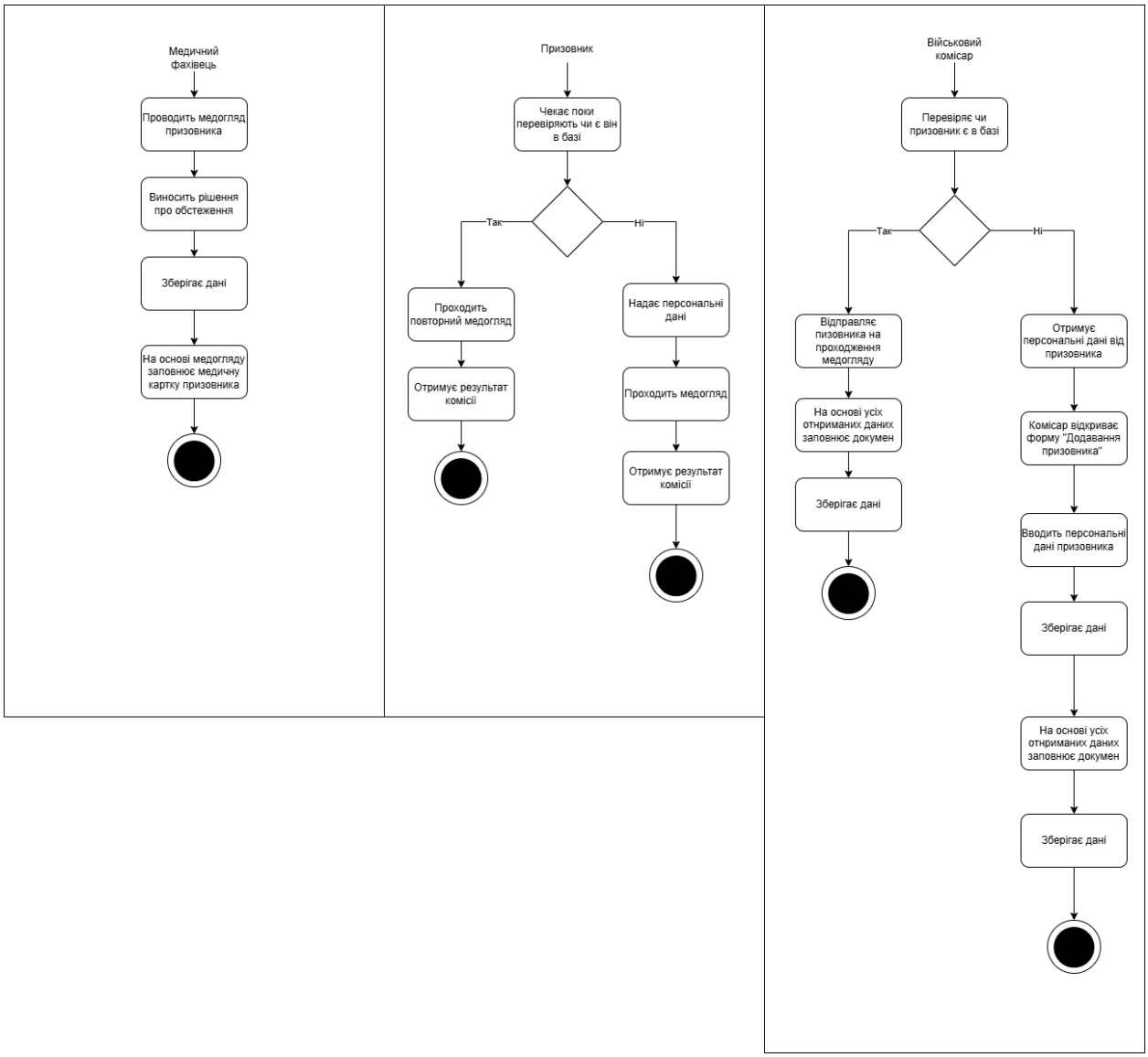


Рис. 2.3 – Діаграма активності

Ця діаграма активності ілюструє процеси, пов'язані з медичним обстеженням та адмініструванням призовників. Вона складається з трьох основних учасників: медичного закладу, призовника та військового комісара, кожен з яких виконує свої специфічні дії.

Медичний заклад бере на себе завдання проведення медичних оглядів, прийняття рішень про подальше обстеження, заповнення медичних карт та збереження отриманих даних. Призовник, зі свого боку, перевіряє свою

наявність у базі даних, надає особисті дані у разі необхідності, проходить медичний огляд і отримує результати від комісії. Військовий комісар перевіряє базу даних на наявність призовника, організовує процес медичного огляду, координує створення документів на основі зібраної інформації та забезпечує збереження даних.

Ця діаграма чудово демонструє взаємозв'язок між цими ролями, забезпечуючи чітке уявлення про процес і його послідовність.

2.3 Абстракції предметної області

У контексті проєктування програмного забезпечення абстракції являють собою процес спрощення складних систем шляхом зосередження на основних функціях, ігноруючи менш критичні деталі. При застосуванні до предметної області програмного забезпечення призовної комісії абстракції дозволяють розробникам моделювати процеси реального світу більш керованим і зрозумілим способом. Ці абстракції зосереджені на ключових сутностях і взаємодіях, забезпечуючи високорівневе уявлення про функціональність системи, не занурюючись у особливості реалізації.

Для програмної системи призовної комісії абстракції предметної області зазвичай передбачають ідентифікацію основних елементів або сутностей, які мають вирішальне значення для роботи системи. Це можуть бути учасники, такі як призовники, військові комісари та медичні працівники, а також системні компоненти, такі як процес медичного обстеження, створення документів та керування базами даних. Абстрагуючи ці компоненти, можна спроєктувати програмне забезпечення для простого та ефективного виконання таких завдань, як реєстрація призовників, керування медичною документацією, оцінка придатності та створення документів.

Основною метою абстракції в цьому контексті є зменшення складності, полегшення взаємодії користувачів і розробників із системою. Наприклад, замість того, щоб безпосередньо керувати кожною частиною

історії хвороби призовника, програмне забезпечення абстрагує ці деталі в медичну картку або профіль, який можна легко оновити та отримати доступ. Подібним чином система абстрагує робочий процес призову на логічні етапи, такі як введення даних, медичне обстеження, прийняття рішень і створення сповіщень, які можуть бути виконані стандартизованим способом без необхідності зосереджуватися на кожній незначній деталі [10].

Таким чином, абстракції в предметній області програмної системи призовної комісії спрощують проєктування та роботу програмного забезпечення, дозволяючи розробникам зосередитися на основних функціях і функціях, одночасно полегшуючи підтримку та розширення системи. Цей підхід покращує взаємодію з користувачем і продуктивність системи, зберігаючи зрозумілість і доступність інтерфейсу та основних процесів.

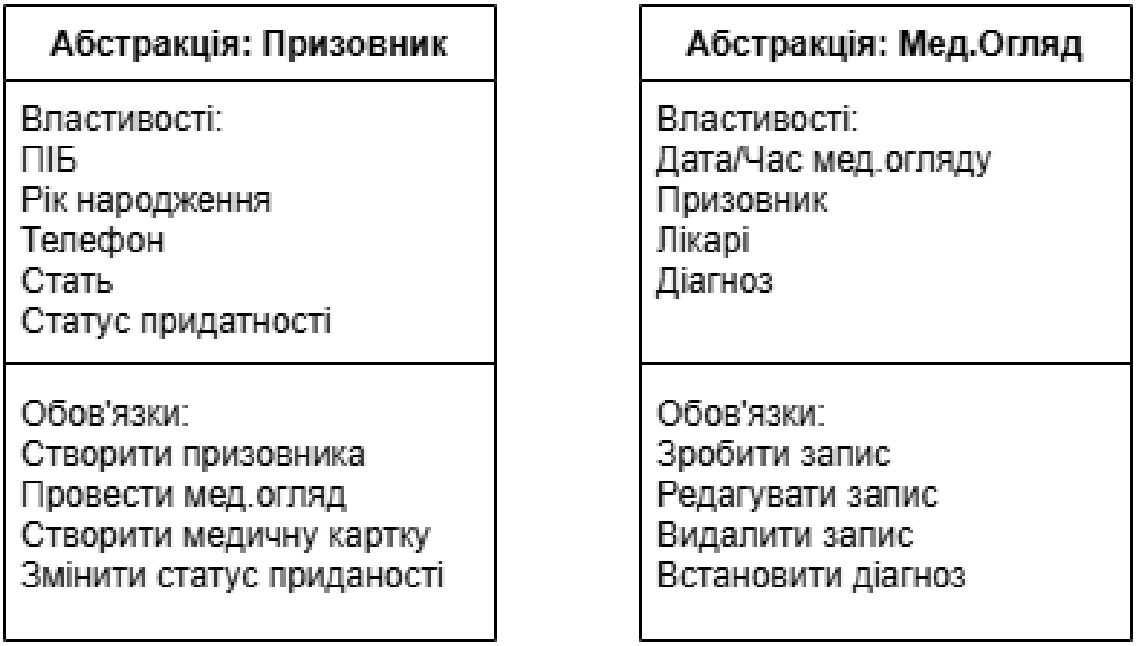


Рис. 2.4 – Абстракції предметної області

2.4 Функціональна модель

Діаграма SADT (Structured Analysis and Design Technique) — це інструмент моделювання, який використовується для представлення систем у структурованому та ієрархічному вигляді. Це особливо корисно для аналізу,

проєктування та документування складних систем і процесів, пропонуючи чітке уявлення про те, як різні частини системи взаємодіють і як через неї проходить інформація. Діаграми SADT часто використовуються в контексті системної інженерії та розробки програмного забезпечення, щоб розбити систему на керовані компоненти для легшого розуміння та проєктування [12].

Основна концепція SADT полягає в тому, щоб розкласти систему на менші, більш керовані частини, кожну з яких можна аналізувати та проєктувати окремо. Цей процес декомпозиції зазвичай представлений у вигляді серії функціональних діаграм або діаграм потоків даних, які ілюструють, як різні компоненти системи виконують завдання та взаємодіють один з одним. Діаграми підкреслюють функціональні аспекти системи, зосереджуючись на процесах, входах, виходах і зв’язках між різними елементами.

Однією з ключових сильних сторін діаграми SADT є її здатність відображати як високорівневі, так і детальні види системи. Це робить його ідеальним як для аналізу існуючих систем, так і для розробки нових. Спочатку діаграма SADT може починатися з загального огляду всієї системи, який потім можна уточнити до більш конкретних процесів. Ця ієрархічна структура полегшує розуміння складних систем і гарантує, що всі компоненти правильно визначені та враховані.

Загалом діаграми SADT є потужним інструментом для візуалізації та організації компонентів системи. Вони пропонують систематичний підхід до

розв’язання складних проблем і допомагають створювати чіткі, структуровані проєкти, які легше зрозуміти, реалізувати та підтримувати.

Рис. 2.5 – Функціональна модель

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних служить абстрактним планом, який описує, як дані організовуються, обробляються та пов’язуються в системі. На відміну від фізичної моделі даних, яка зосереджена на технічних аспектах зберігання та пошуку даних, логічна модель даних стосується структури та зв’язків даних без урахування базової технології бази даних або механізмів зберігання. Основна мета — забезпечити відповідність даних системи бізнес-вимогам і правилам.

В основі логічної моделі даних знаходяться сутності, які представляють реальні об’єкти або концепції в системі. Ці організації можуть бути будь-якими, що стосуються бізнесу, як-от призовник, медична довідка, військовий комісар або експертиза. Кожна сутність містить атрибути, які описують конкретні деталі або властивості. Наприклад, сутність призовника може містити такі атрибути, як ім’я, вік, стать і військовий статус [13].

Відносини між сутностями також є життєво важливим компонентом логічної моделі даних. Ці відносини визначають, як різні сутності взаємодіють один з одним. Наприклад, сутність військовозобов’язаних може мати зв’язок із медичною картою, що свідчить про те, що кожен призовник пов’язаний з однією чи кількома медичними записами. Ці зв’язки також можуть включати кардинальність, яка вказує, скільки екземплярів однієї сутності можна зв’язати з іншою. Наприклад, сутність MedicalRecord, ймовірно, матиме зовнішній ключ, що вказує на сутність призовника, встановлюючи зв’язок «один до одного» або «один до багатьох».

Іншим ключовим аспектом логічної моделі даних є використання первинних ключів, які є унікальними ідентифікаторами для кожної сутності. Наприклад, призовник може мати унікальний ID призовника, який дозволяє його легко відрізнити від інших. Так само зовнішні ключі використовуються

для встановлення зв’язків між сутностями. Ці зовнішні ключі посилаються на первинні ключі інших об’єктів, додатково з’єднуючи структуру даних системи.

Логічна модель даних зазвичай починає використовуватися на початку процесу проєктування. Це допомагає створити спільне розуміння структури даних і забезпечує основу для наступних етапів, включаючи фізичну модель даних. Логічна модель даних не зосереджується на тому, як дані будуть зберігатися, а натомість забезпечує концептуальну структуру для того, як дані будуть перетікати та співвідноситись у системі.

По суті, логічна модель даних є важливим інструментом для проєктування системи, пропонуючи високорівневий огляд взаємозв’язків даних і структур, які формують основу для ефективних і придатних для обслуговування систем баз даних. Це допомагає гарантувати, що система може виконувати вимоги до даних, залишаючись адаптованою до майбутніх змін.

Логічна модель системи представлена на рисунку 9.

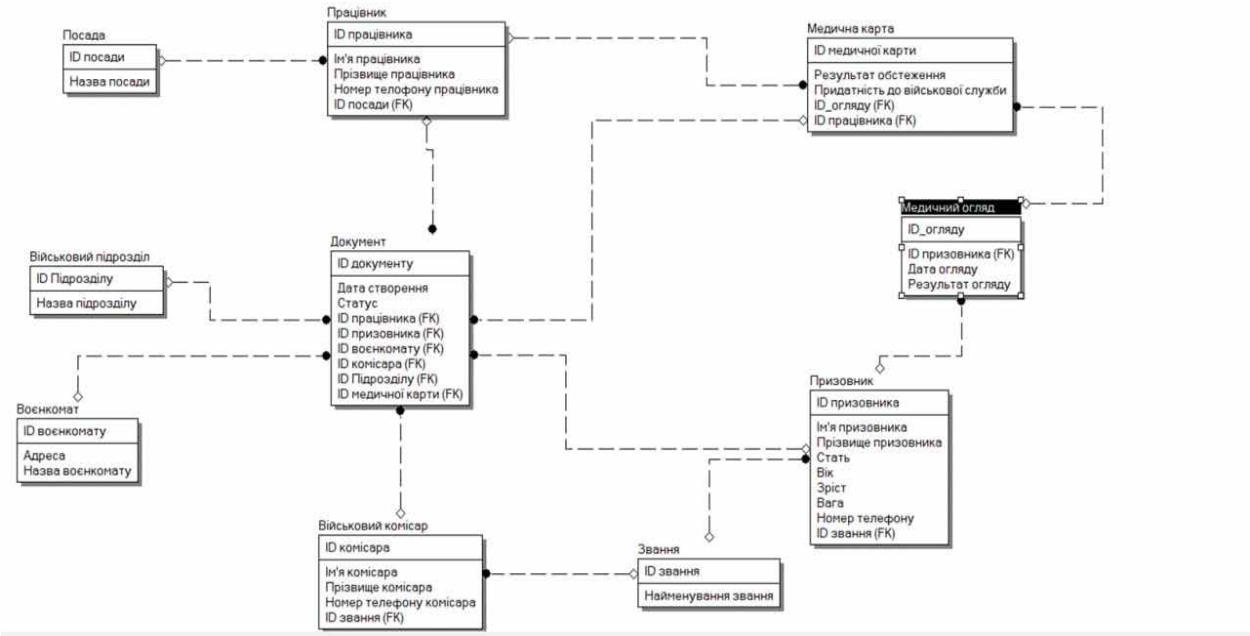


Рис. 3.1 – ER-діаграма

Ця ER-діаграма представляє структуру взаємозв’язків між різними сутностями в базі даних, показуючи їх атрибути та зв’язки. Вона складається з кількох основних сутностей, кожна з яких має свої характеристики.

Сутності та їх атрибути:

1. посада включає назву посади;
2. військовий підрозділ містить назву підрозділу;
3. військомат має адресу та назву військкомату;
4. військовий комісар включає номер, прізвище та звання;
5. документ має дату створення, статус, а також зовнішні ключі до інших сутностей, зокрема працівника, підрозділу, військомату, медичної карти, військового комісара та призовника;
6. працівник включає прізвище, номер та посаду;
7. машина карта містить результати огляду та дату огляду, а також зв’язок із призовником;
8. медичний огляд включає дату огляду та результат;
9. звання містить найменування звання.

Зв’язки між сутностями:

1. посада пов’язана з Працівником;
2. військомат має зв’язок із Документом;
3. призовник пов’язаний із медичним оглядом та Документом;
4. військовий комісар взаємодіє з Документом;
5. документ включає зв’язки із Призначенням, Медичною картою та Призовником;
6. медичний огляд пов’язане з Медичною картою;
7. медична карта має зв’язок із Призовником;
8. звання взаємодіє з Військовим комісаром.

Ця діаграма створює чітку та структуровану модель даних, яка допомагає зрозуміти логіку зберігання та взаємодії між сутностями.

3.2 Вибір системи управління базою даних та її реалізація

Вибір правильної системи керування базами даних (СУБД) має вирішальне значення для будь-якої програмної системи, оскільки це впливає на загальну ефективність, продуктивність і зручність обслуговування програми. У контексті програмного забезпечення для працівників призовної комісії, де керування інформацією призовників, медичними документами та результатами обстеження є центральним, важливо вибрати систему, яка може легко обробляти складні структуровані дані.

Система управління реляційною базою даних (RDBMS) добре підходить для цього типу програмного забезпечення, оскільки вона організовує дані в структуровані таблиці з чітко визначеними зв’язками. Ця модель добре працює для відстеження таких об’єктів, як призовники, медичні записи та їх взаємодія. Крім того, SQL (Structured Query Language), який використовується для керування реляційними даними, широко підтримується та ідеально підходить для запитів і маніпулювання структурованими наборами даних, такими як ті, що використовуються в роботах проєктної комісії [15].

Серед багатьох доступних варіантів Microsoft SQL Server виділяється як чудовий вибір для цієї програми. Це потужна СУБД корпоративного рівня, яка може ефективно обробляти великі дані. MS SQL Server надає повний набір функцій, які ідеально відповідають вимогам призовної системи.

Під час впровадження бази даних першим кроком є розробка логічної моделі даних на основі вимог системи. Ця модель описує різні сутності, атрибути та їхні зв’язки в системі. Після встановлення моделі створюється схема бази даних, яка включає визначення таблиць, первинних ключів і зв’язків між сутностями. Наприклад, такі сутності, як «Призовник», «Медична карта», «Огляд» і «Військовий комісар», будуть представлені у вигляді таблиць, кожна з яких містить стовпці для відповідних атрибутів.

Для встановлення зв'язків між різними сутностями використовуються зовнішні ключі. Наприклад, таблиця MedicalRecord міститиме зовнішній

ключ, який пов’язує її з таблицею призовника, встановлюючи чіткий зв’язок між призовником та його медичною документацією. Ця реляційна структура забезпечує узгодженість даних і забезпечує ефективне надсилання запитів до пов’язаних даних у кількох таблицях.

Схема бази даних, що задовольняє описані вимоги, подано на рис. 3.2.

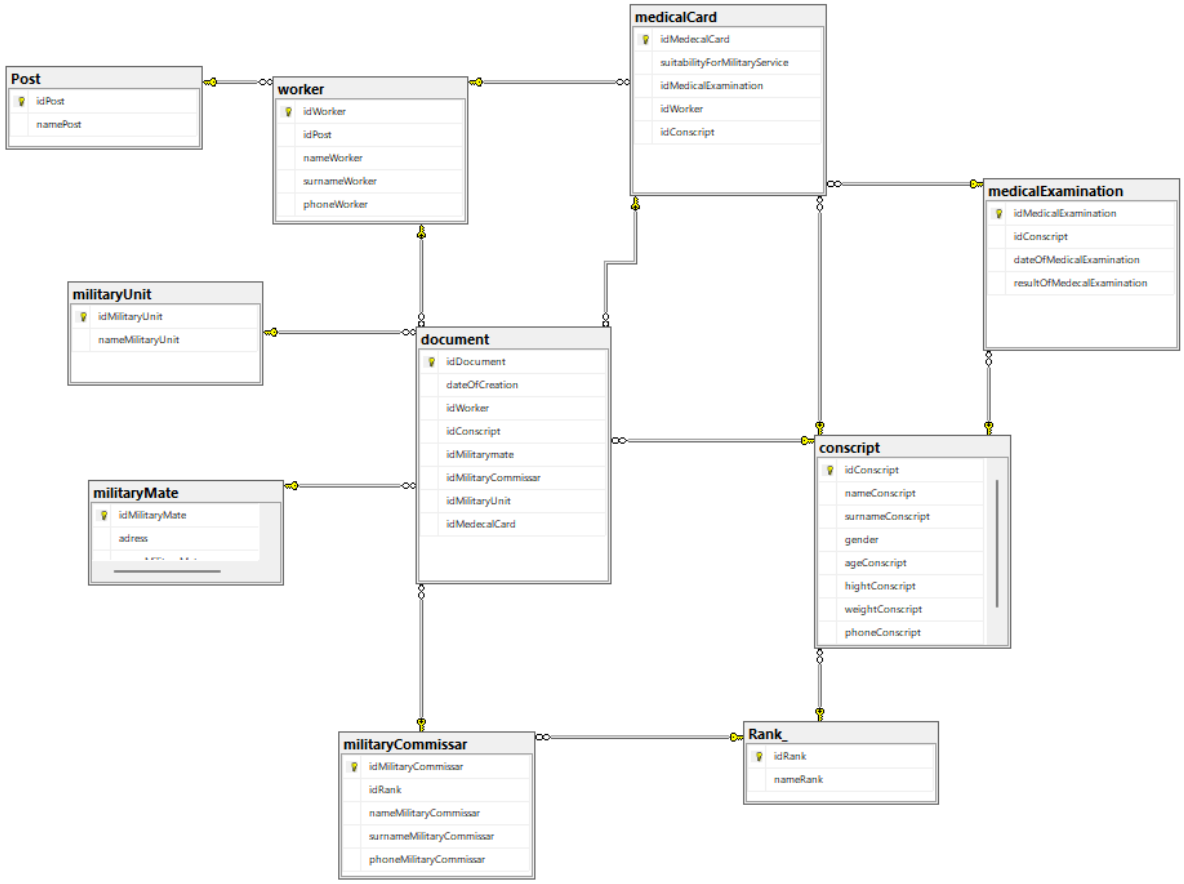


Рис. 3.2 – Схема бази даних системи

Реалізація бази даних також передбачає створення індексів для часто запитуваних стовпців для підвищення продуктивності, особливо при роботі з великими обсягами даних. Методи нормалізації будуть застосовані для мінімізації надмірності даних і забезпечення чистого та ефективного дизайну бази даних. Регулярне резервне копіювання даних має важливе значення для захисту від потенційної втрати даних, а для забезпечення безперервності роботи слід запровадити стратегії аварійного відновлення.

Після створення схеми бази даних її буде інтегровано з програмою.

Програмне забезпечення призовної комісії взаємодіятиме з базою даних через запити SQL, дозволяючи користувачам створювати, оновлювати та отримувати інформацію про призовників та медичні записи. Наприклад, коли військовий комісар додає або оновлює медичну карту призовника, програмне забезпечення генеруватиме SQL-запити, щоб вставити або змінити відповідні дані в базі даних [15].

Рішення вибрати MS SQL Server для реалізації бази даних програмного забезпечення для працівників призовної комісії ґрунтувалося на кількох важливих факторах, усі з яких забезпечують ефективне, безпечне та масштабне функціонування системи.

Перш за все, MS SQL Server — це потужна система керування реляційними базами даних (RDBMS) корпоративного рівня, що робить її придатним вибором для керування структурованими даними з чіткими зв’язками, такими як особиста інформація призовників, медичні записи та результати обстеження. Дані в призовній системі мають чітко визначені зв’язки, де один запис (наприклад, особисті дані призовника) пов’язаний з кількома іншими записами (наприклад, медичні оцінки та результати обстеження). Реляційна природа MS SQL Server дозволяє легко розробляти та впроваджувати ці зв’язки, забезпечуючи цілісність і послідовність даних.

Іншим ключовим фактором у виборі MS SQL Server є його надійні функції безпеки. Програмне забезпечення має справу з конфіденційною інформацією, включаючи особисті дані, медичні записи та інші приватні дані призовників. MS SQL Server пропонує широкий спектр механізмів безпеки, таких як шифрування, автентифікація користувачів і контроль доступу на основі ролей, які необхідні для забезпечення захисту даних від несанкціонованого доступу. Це робить його надійним вибором для додатків, які повинні відповідати суворим нормам захисту даних, гарантуючи безпеку конфіденційної інформації призовників.

На додаток до безпеки, MS SQL Server забезпечує надійну цілісність даних завдяки сумісності з ACID (атомність, узгодженість, ізоляція,

довговічність). Ці властивості гарантують, що всі транзакції бази даних обробляються надійно, і гарантують, що дані не будуть пошкоджені, навіть у випадках системних збоїв. Враховуючи те, що система оброблятиме численні транзакції, пов’язані зі записами військовозобов’язаних, наприклад створення нових записів, оновлення медичних статусів і створення звітів, здатність підтримувати узгодженість даних має першочергове значення. Відповідність SQL Server ACID гарантує, що база даних залишається в дійсному стані навіть під час одночасних операцій.

Масштабованість – ще одна причина, чому був обраний MS SQL Server. Оскільки очікується, що система призовної комісії з часом оброблятиме зростаючий обсяг записів призовників, важливо, щоб база даних могла масштабуватися, щоб відповідати цьому зростанню без шкоди для продуктивності. MS SQL Server добре відомий своєю здатністю ефективно обробляти великі масиви даних і великі обсяги транзакцій, гарантуючи, що система може розвиватися відповідно до потреб призовної комісії [16].

Програмне забезпечення також має переваги від багатого набору вбудованих функцій, які надає MS SQL Server, таких як розширені можливості запитів, збережені процедури та тригери. Ці інструменти спрощують процес розробки, дозволяючи створювати складні операції пошуку даних і автоматизувати певні процеси. Наприклад, збережені процедури можна використовувати для автоматизації створення звітів або оновлення записів про призовників, заощаджуючи час і зменшуючи ймовірність помилок.

Крім того, MS SQL Server легко інтегрується з іншими технологіями Microsoft, такими як .NET і C#, які є основними технологіями, що використовуються для розробки програмного забезпечення для працівників призовної комісії. Ця інтеграція гарантує, що процес розробки буде ефективним і забезпечує плавне з’єднання між базою даних і прикладним рівнем.

Таким чином, Microsoft SQL Server забезпечує необхідні функції та продуктивність для підтримки системи призовної комісії. Його безпека,

масштабованість і надійність роблять його ідеальним вибором для керування конфіденційними даними, пов’язаними з процесами призову. Ретельно спроєктувавши базу даних і оптимізувавши її структуру, система зможе ефективно зберігати та обробляти великі обсяги даних призовників, забезпечуючи цілісність і безпеку даних.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення для системи призовної комісії розроблена як надійна, масштабована та безпечна, забезпечуючи при цьому ефективне управління особистою інформацією призовників, медичною документацією та результатами обстеження. Архітектура дотримується багаторівневого підходу, який поділяє систему на окремі компоненти, кожен з яких відповідає за певне завдання. Такий розподіл завдань гарантує, що програму легше підтримувати, вона безпечніша та працює ефективніше [17].

Система складається з трьох основних рівнів. Рівень презентації, або інтерфейс користувача (UI), відповідає за обробку взаємодії користувача. Саме тут військові комісари та призовники можуть взаємодіяти з програмним забезпеченням. Використовуючи WinForms, інтерфейс користувача надає зрозумілий і доступний інтерфейс для керування записами про призовників, медичними оцінками та іншими пов’язаними завданнями. Рівень інтерфейсу користувача взаємодіє з рівнем бізнес-логіки, щоб надсилати введені користувачем дані та відображати оброблені результати.

Рівень бізнес-логіки є ядром системи, який обробляє дані, отримані від рівня інтерфейсу користувача, і виконує необхідні операції. Цей рівень містить правила та процедури, необхідні для роботи програми, як-от керування профілями призовників, медичне обстеження та перевірки придатності. Він також виконує такі складні завдання, як створення звітів і перевірка даних.

Логічний рівень реалізовано за допомогою C#, що полегшує керування та масштабування відповідно до потреб системи.

Рівень доступу до даних (DAL) відповідає за взаємодію з базою даних MS SQL Server. Він керує всіма операціями, пов’язаними з отриманням, вставкою, оновленням або видаленням даних. Абстрагуючи деталі доступу до бази даних, DAL гарантує, що рівень бізнес-логіки може зосередитися на обробці даних, а не на управлінні підключеннями до бази даних і запитами. Він використовує ADO.NET або Entity Framework для взаємодії з базою даних і забезпечення безперебійного виконання всіх транзакцій.

Зв'язок між цими шарами чітко визначений і структурований. Рівень інтерфейсу користувача надсилає запити на рівень бізнес-логіки. Після обробки вхідних даних рівень бізнес-логіки взаємодіє з рівнем доступу до даних для виконання необхідних операцій, таких як збереження записів про призовників або оновлення медичних оглядів. Після отримання або оновлення даних рівень бізнес-логіки повертає оброблені результати на рівень інтерфейсу користувача, який потім відображає відповідну інформацію для користувача.

Безпека є ключовим аспектом системи, оскільки вона має справу з конфіденційною інформацією, такою як особисті дані та медичні записи. Архітектура включає численні заходи безпеки, включаючи рольовий контроль доступу, який гарантує, що лише авторизований персонал може отримати доступ до певних функцій. Дані також шифруються під час передачі та зберігання для захисту від несанкціонованого доступу. Крім того, вхідні дані перевіряються як на рівні інтерфейсу користувача, так і на рівні бізнес-логіки, гарантуючи, що в систему вводяться лише точні та дійсні дані.

Щоб система залишалася ефективною та оперативною під час обробки великих обсягів даних, рівень доступу до даних оптимізовано для продуктивності. Запити розроблені таким чином, щоб бути ефективними, використовуючи індекси для прискорення даних, до яких часто звертаються. Крім того, механізми кешування можуть бути реалізовані на рівні бізнес-

логіки, щоб зменшити надлишкові виклики бази даних і підвищити продуктивність системи.

Технологічний стек, який використовується для цієї системи, включає WinForms для зовнішнього інтерфейсу, C# для бізнес-логіки та MS SQL Server як систему керування базами даних. Ці технології вибрано через їхню надійність, продуктивність і функції безпеки. Інтеграція цих компонентів гарантує, що система може масштабуватися за потреби та безпечно та ефективно справлятися з вимогами керування даними призовників.

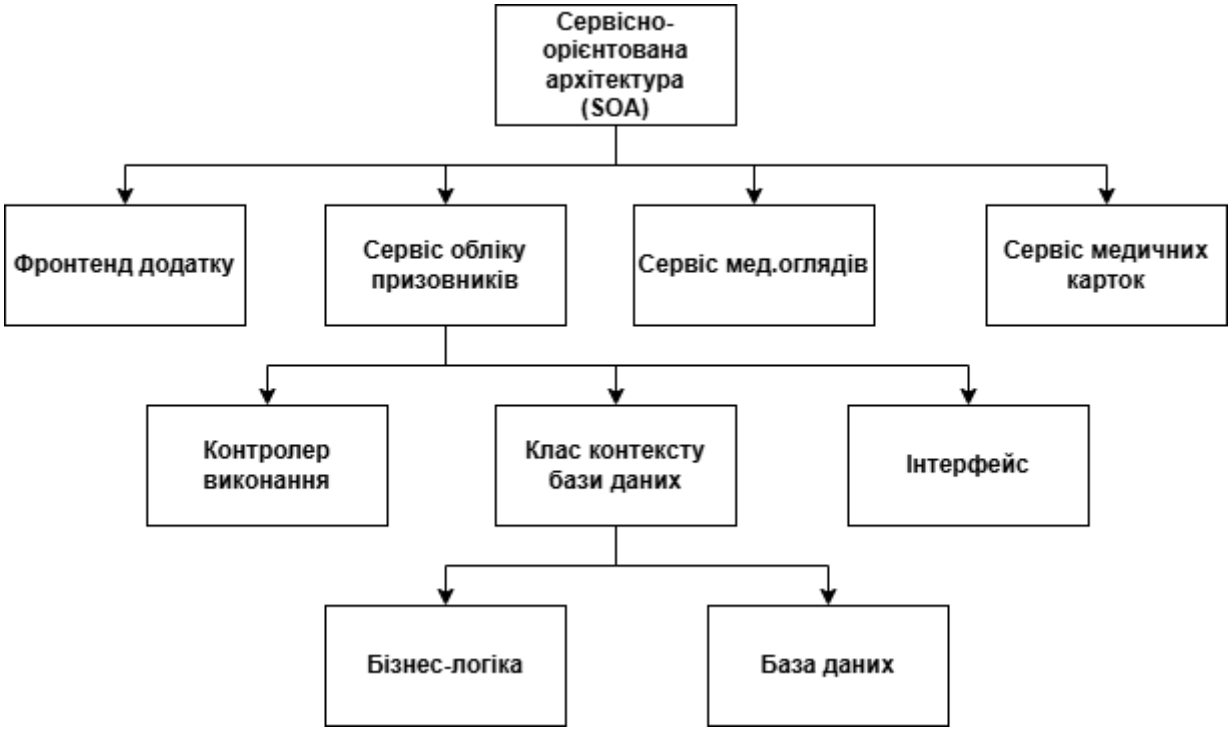


Рис. 3.3 – SOA архітектура

Підсумовуючи, архітектура програмного забезпечення для працівників призовної комісії розроблена таким чином, щоб бути безпечною та зручною для обслуговування, забезпечуючи чіткий розподіл проблем між рівнями. Використовуючи MS SQL Server, система забезпечує ефективне управління даними, зберігаючи при цьому високий рівень безпеки. Ця архітектура не тільки підтримує нагальні потреби призовної комісії, але й є достатньо гнучкою, щоб адаптуватися до майбутніх змін або розширень.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакета є важливою частиною процесу розробки програмного забезпечення, оскільки вона забезпечує високорівневе уявлення про архітектуру системи шляхом групування пов'язаних класів і компонентів у цілісні пакети. Ця діаграма допомагає представити логічну організацію системи та взаємодію різних частин, пропонуючи чіткий огляд структури системи. Це гарантує, що різні функціональні можливості системи організовані в керовані блоки, що спрощує процес підтримки та розширення програми [19].

Для програмного забезпечення, призначеного для працівника призовної комісії, діаграма пакета служить для того, щоб показати, як програма розділена на кілька ключових функціональних областей, кожна з яких відповідає за певний набір завдань. Ці області згруповані в пакети, кожен з яких обробляє різні аспекти системи. Пакети взаємодіють один з одним через чітко визначені інтерфейси, що дозволяє системі функціонувати безперебійно та гарантує, що будь-які оновлення чи зміни можуть бути реалізовані незалежно в одній частині, не викликаючи проблем в інших.

Пакет інтерфейсу користувача (UI) відповідає за керування всіма елементами, пов'язаними з візуальним інтерфейсом програми. Це включає вікна, форми та різні елементи керування, з якими взаємодіють користувачі. Пакет UI служить точкою входу для військових комісарів і призовників, дозволяючи їм переглядати дані, вводити інформацію та переміщатися системою. Зв'язок між інтерфейсом користувача та рівнем бізнес-логіки має важливе значення, оскільки інтерфейс користувача надсилає дані користувача та отримує оброблені дані для відображення результатів.

Пакет Business Logic є основою програми, керуючи такими основними операціями, як створення та оновлення записів призовників, проведення медичних оглядів і створення звітів. Він інкапсулює правила та процеси, які визначають поведінку системи. Цей пакет взаємодіє з рівнем доступу до

даних, щоб отримувати та зберігати дані з бази даних, забезпечуючи належне виконання логіки системи на основі запитів користувача.

Для керування взаємодією з базою даних використовується пакет Data Access. Цей пакет обробляє з’єднання з базою даних, виконання запитів, а також пошук або модифікацію даних. Це гарантує, що всі дані системи, включаючи відомості про призовників, медичні записи та результати оцінювання, зберігаються та ефективно управляються.

Рис. 3.4 – Діаграма пакетів

Ця діаграма пакетів відображає логічну структуру програмного забезпечення, розділяючи його на модулі та визначаючи їх взаємозв’язок. Вона демонструє, як окремі компоненти системи взаємодіють між собою та організовані для ефективного управління даними та функціоналом.

Основні пакети включають:

- користувацький інтерфейс (UI), який відповідає за взаємодію з призовниками, медичними картками та організацію медичних оглядів;
- керування даними, що містить інформацію про користувачів та механізми доступу до бази даних.

Взаємозв’язки між пакетами виражені стрілками, що показують напрямок обміну даними та виклику функціоналу. Така структура спрощує розробку та підтримку програмного забезпечення, роблячи його більш організованим і легким для масштабування.

3.5 Вибір інструментарію для створення програмного забезпечення

Розробка програмного забезпечення для працівника призовної комісії вимагає вибору відповідних інструментів, які відповідають потребам системи, забезпечують необхідну функціональність і забезпечують зручну роботу користувача. Після ретельного розгляду різних факторів, таких як вимоги до системи, масштабованість, простота використання та продуктивність, для реалізації проєкту були обрані такі інструменти: C#, WinForms, MS SQL Server. Ці інструменти забезпечують надійну основу для побудови системи, забезпечуючи оптимальну продуктивність і безпеку.

C# є основною мовою програмування, обраною для розробки цієї системи. Відомий своєю універсальністю та широким використанням у додатках корпоративного рівня, C# добре підходить як для настільних, так і для веб-додатків. Він пропонує багатий набір бібліотек і сильну об’єктно-орієнтовану парадигму, яка допомагає ефективно організувати код системи. Крім того, C# бездоганно інтегрується з іншими технологіями Microsoft, що є перевагою, враховуючи інші інструменти, вибрані для цього проєкту.

Windows Forms (WinForms) обрано як структуру інтерфейсу користувача для програмного забезпечення. WinForms — це зріла, проста у використанні структура для створення настільних програм в операційній системі Windows. Він надає багатий набір попередньо створених елементів керування, що дозволяє швидше створювати інтуїтивно зрозумілі та чутливі інтерфейси користувача. Інструмент дизайну з перетягуванням у Visual Studio також прискорює процес розробки, дозволяючи розробникам швидко

проєктувати та макетувати форми. WinForms особливо ідеальний для цього проєкту, оскільки він дозволяє ефективно керувати формами та взаємодіями між працівником призовної комісії та даними призовників у середовищі Windows.

Для керування базами даних MS SQL Server вибрано як реляційну систему керування базами даних (RDBMS). Це потужне, безпечне та добре масштабоване рішення для баз даних, яке широко використовується в корпоративних програмах. SQL Server підтримує складні запити та транзакції, необхідні для зберігання та керування великими обсягами даних призовників, медичних записів та іншої пов’язаної інформації. Його надійні функції безпеки, такі як шифрування даних, автентифікація та авторизація, гарантують надійний захист конфіденційної інформації. MS SQL Server також забезпечує високу доступність і рішення для аварійного відновлення, гарантуючи, що дані завжди доступні та безпечні.

Використовуючи C#, WinForms, MS SQL Server розробка програмного забезпечення для працівника призовної комісії будується на міцній та надійній технологічній основі. Ці інструменти забезпечують створення безпечної, ефективної та зручної системи, яка може виконувати складні вимоги щодо керування даними призовників та проведення медичних оглядів. Кожен інструмент був обраний через його здатність легко інтегруватися з іншими, забезпечуючи цілісне середовище, яке підтримує як функціональні, так і нефункціональні вимоги проєкту. Завдяки цьому пакету технологій система готова до масштабованості, простоти обслуговування та надійної безпеки, що забезпечує довгостроковий успіх і надійність.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Успішна робота програмного забезпечення, призначеного для працівників призовних комісій, залежить від виконання певних вимог до апаратного та програмного забезпечення. Ці вимоги забезпечують безперебійну роботу, точність даних і надійність під час керування записами призовників і підтримки різноманітних адміністративних завдань.

З апаратної точки зору система призначена для ефективної роботи на типовому офісному обладнанні. Сучасний багатоядерний процесор із достатньою тактовою частотою необхідний для забезпечення стабільної та швидкодіючої продуктивності, особливо коли кілька операцій виконуються одночасно. Програма також вимагає мінімум 8 ГБ оперативної пам’яті, що забезпечує ефективну обробку даних і швидке реагування під час складних процедур, таких як обробка особистих записів або створення звітів. Для зберігання великих баз даних і супровідної документації на комп’ютері має бути достатньо місця на диску — бажано на твердотільному накопичувачі для швидшого доступу. Включення стандартних периферійних пристроїв, таких як монітор із відповідною роздільною здатністю, клавіатура, миша та, можливо, принтер, є важливим для повної функціональності.

Надійне підключення до Інтернету також має вирішальне значення, особливо якщо система працює в багатокористувацькому середовищі або спілкується із зовнішніми системами. Дротове або бездротове підключення з гідною пропускною здатністю забезпечує швидку та безпечну передачу інформації, мінімізуючи затримки та втрату даних.

Що стосується програмних вимог, операційне середовище має підтримувати Windows 10 або новішу версію. Розробка виконується за

допомогою C# у середовищі Visual Studio з використанням можливостей .NET Framework або .NET Core, залежно від остаточного сценарію розгортання. Інтерфейс створено з використанням WinForms, вибраного через його легкість у створенні інтуїтивно зрозумілих настільних програм, які потребують мінімального навчання персоналу.

Компонент керування базою даних реалізовано за допомогою Microsoft SQL Server, що забезпечує стабільну та масштабовану платформу для зберігання, отримання та захисту конфіденційних даних. Інтеграція з Entity Framework Core спрощує зв'язок між додатком і базою даних, забезпечуючи ефективне маніпулювання даними за допомогою об'єктно-орієнтованого програмування, а не необроблених запитів SQL. Для автентифікації та керування ролями система використовує ASP.NET Core Identity, що пропонує гнучке керування користувачами та безпечний контроль доступу.

Для подальшої підтримки щоденних операцій можуть знадобитися додаткові утиліти, такі як Microsoft Office і програма для читання PDF, для таких завдань, як підготовка офіційних документів або перегляд експортованих даних. Конфігурація мережі має забезпечувати безпечний обмін файлами, а механізми резервного копіювання даних — хмарні чи локальні — життєво необхідні для запобігання втраті даних і підтримки відновлення у разі технічних проблем.

Ці апаратні та програмні параметри вибрано для створення міцного та надійного робочого середовища. Вони дозволяють персоналу призовної комісії ефективно виконувати свої обов'язки, від введення та перевірки даних до звітності та безпечного доступу користувачів, зберігаючи при цьому цілісність і конфіденційність інформації, що обробляється.

Щодо встановлення програмного забезпечення – нам потрібно переконатися що:

• Зазначені параметри вищі присутні на комп'ютері, куди ми збираємося встановити програму (є Windows 10 або новіша, присутній .Net

Framework і тд.).

- Далі ми завантажуємо на комп’ютер інсталяційний пакет.
- Коли інсталяційний пакет буде завантажено, ми відкриваємо файл з розширенням .exe та обираємо параметри встановлення.
- Після встановлення системи її потрібно запустити та переконатися у її справності (перевіряємо функціонал програми)

4.2 Інсталяція на сервер

На даний момент програмне забезпечення функціонує локально, забезпечуючи коректну роботу та повний функціонал. Це дозволяє протестувати всі ключові можливості системи та налаштувати її під реальні потреби користувачів. Проте, після завершення тестування і підготовки до реального використання, програму можна буде встановити на сервер для забезпечення її доступності у багатокористувацькому середовищі.

Особливості роботи програми локально

1. Локальне середовище:

Програма працює на персональному комп’ютері, використовуючи встановлену базу даних та інші необхідні компоненти. Це забезпечує простоту налаштувань і доступ до всіх функцій.

2. Переваги локального тестування:

- Зручність у виправленні помилок та доопрацювання функцій.
- Можливість перевірки продуктивності програми в умовах реального навантаження.
- Швидке оновлення та перевірка змін.

Розгортання програми на сервері

Програма вже готова до використання в робочому середовищі, її можна розгорнути на сервері для забезпечення доступу до неї кільком користувачам одночасно. Це забезпечить кращу масштабованість, централізоване зберігання даних та підвищену безпеку. Проте це вже потрібно буде робити саме на місці

де її будуть використовувати, або на якихось захищених серверах у зв’язку зі специфікою моєї теми.

1. Підготовка серверного середовища:

Для роботи програми на сервері необхідно виконати наступні дії:

- Встановити серверну операційну систему (Windows Server 2016 або новіша).
- Налаштувати сервер для підтримки бази даних (Microsoft SQL Server)

2. Переваги серверного розгортання:

- Централізований доступ: Користувачі можуть підключатися до програми з різних комп’ютерів через локальну мережу чи Інтернет.
- Підтримка багатокористувацького середовища: Програма може обробляти запити від кількох користувачів одночасно.
- Зручність оновлень: Всі зміни та нові версії програми оновлюються одразу для всіх користувачів.

Етапи переходу до серверного використання

1. Оптимізація програми:

Потрібно переконатися, що програма готова до роботи у мережевому середовищі. Це включає:

- Налаштування підключення до віддаленої бази даних.
- Оптимізацію продуктивності для одночасної роботи кількох користувачів.

2. Розгортання:

- Перенесення програми на сервер за допомогою FTP.
- Налаштування серверних параметрів для забезпечення стабільної роботи програми.

3. Тестування:

- Після розгортання потрібно перевірити всі функції програми на сервері, щоб переконатися у її коректній роботі.

4. Запуск в експлуатацію:

- Надання доступу користувачам через локальну мережу чи Інтернет(сервер можу бути локальним або віддаленим).

Переваги переходу на сервер

Перехід на сервер дозволить програмі стати потужним інструментом у багатокористувацькому середовищі. Це забезпечить зручність, надійність та продуктивність у роботі з даними, навіть при високому навантаженні.

4.3 Тестування системи

Запустивши систему, потрапляємо на форму авторизації, тут нам треба ввести логін та пароль користувача (рис. 4.1).

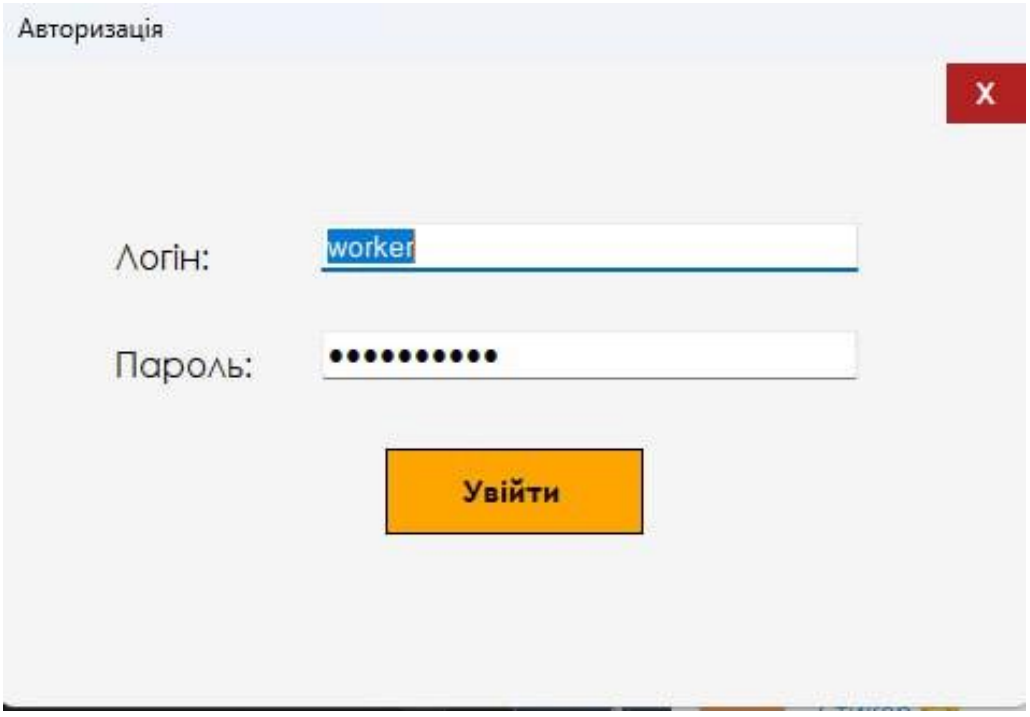
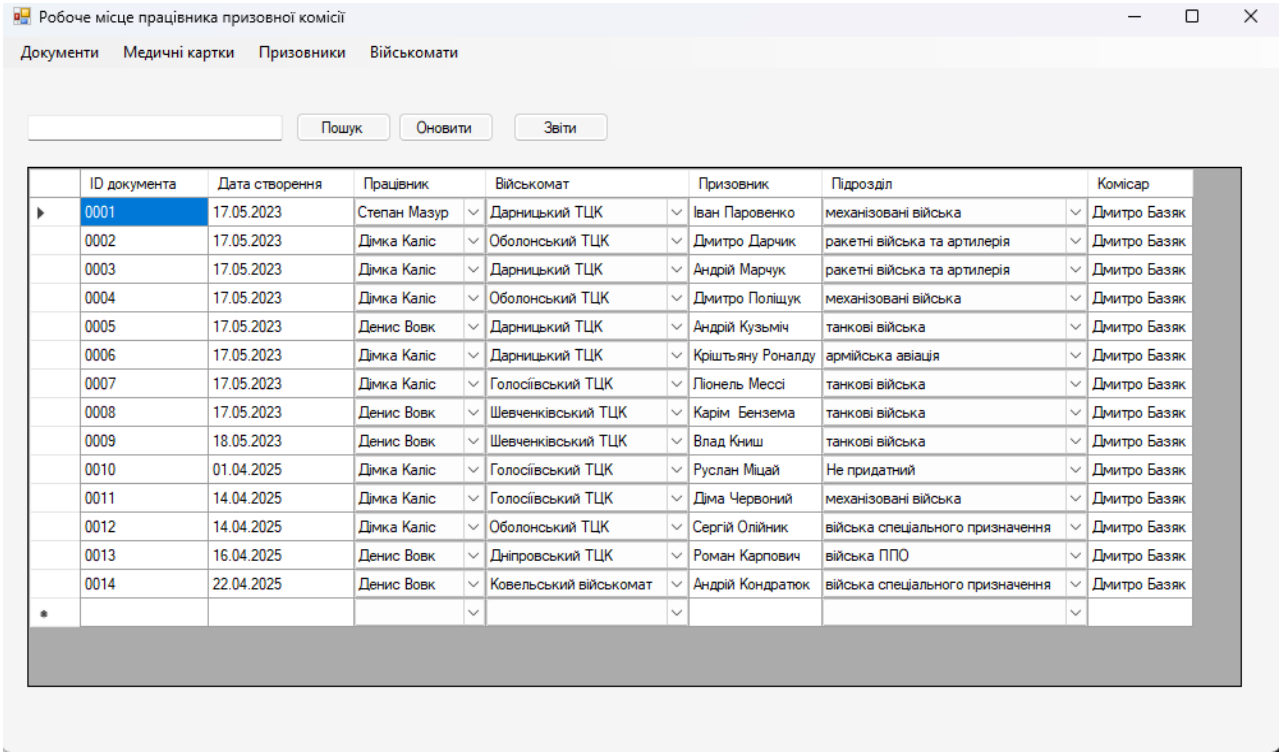


Рис. 4.1 – Форма авторизації

Після авторизації потрапляє на першу сторінку нашої програми "Документи". Тут у нас виведено весь список доданих документів у системі. Існують кнопки додавання нового документа або оновлення існуючого документа (рис. 4.2).



ID документа	Дата створення	Працівник	Військомат	Призовник	Підрозділ	Комісар
0001	17.05.2023	Степан Мазур	Дарницький ТЦК	Іван Паровенко	механізовані війська	Дмитро Базяк
0002	17.05.2023	Дімка Каліс	Оболонський ТЦК	Дмитро Дарчик	ракетні війська та артилерія	Дмитро Базяк
0003	17.05.2023	Дімка Каліс	Дарницький ТЦК	Андрій Марчук	ракетні війська та артилерія	Дмитро Базяк
0004	17.05.2023	Дімка Каліс	Оболонський ТЦК	Дмитро Поліщук	механізовані війська	Дмитро Базяк
0005	17.05.2023	Денис Вовк	Дарницький ТЦК	Андрій Кузьміч	танкові війська	Дмитро Базяк
0006	17.05.2023	Дімка Каліс	Дарницький ТЦК	Кріштьяну Роналду	армійська авіація	Дмитро Базяк
0007	17.05.2023	Дімка Каліс	Голосіївський ТЦК	Ліонель Мессі	танкові війська	Дмитро Базяк
0008	17.05.2023	Денис Вовк	Шевченківський ТЦК	Карім Бензема	танкові війська	Дмитро Базяк
0009	18.05.2023	Денис Вовк	Шевченківський ТЦК	Влад Книш	танкові війська	Дмитро Базяк
0010	01.04.2025	Дімка Каліс	Голосіївський ТЦК	Руслан Міцай	Не придатний	Дмитро Базяк
0011	14.04.2025	Дімка Каліс	Голосіївський ТЦК	Діма Червоний	механізовані війська	Дмитро Базяк
0012	14.04.2025	Дімка Каліс	Оболонський ТЦК	Сергій Олійник	війська спеціального призначення	Дмитро Базяк
0013	16.04.2025	Денис Вовк	Дніпровський ТЦК	Роман Карпович	війська ППО	Дмитро Базяк
0014	22.04.2025	Денис Вовк	Ковельський військомат	Андрій Кондратюк	війська спеціального призначення	Дмитро Базяк

Рис. 4.2 – Сторінка “Документи”

Натискає на "Додати документ" та переходимо на форму створення документа. Тут нам треба запровадити всі необхідні поля та підтвердити створення (рис. 4.4).

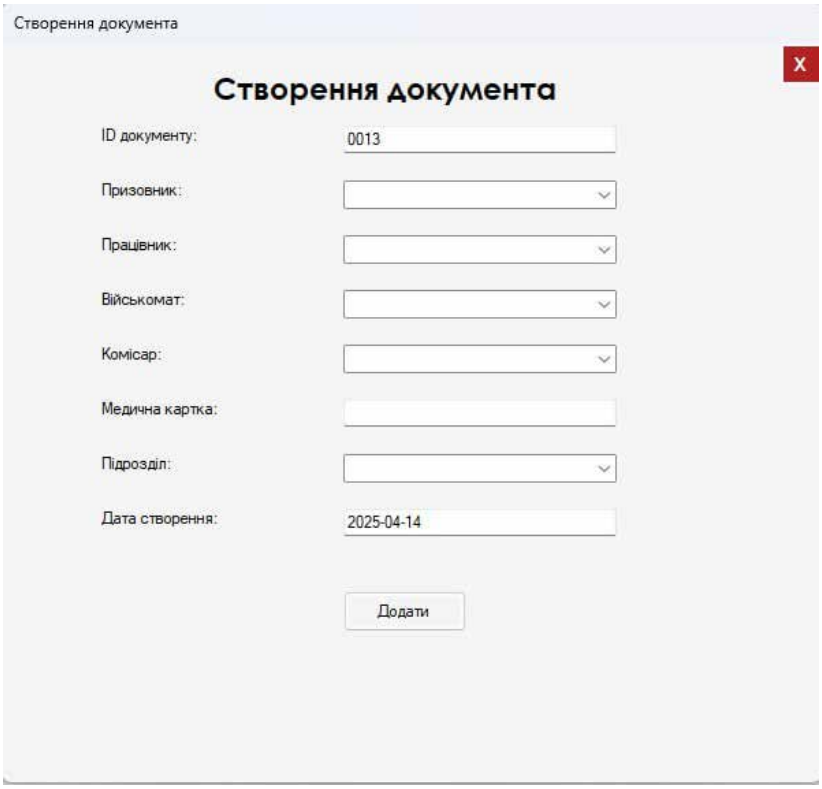


Рис. 4.4 – Форма створення документу

Відкриваємо сторінку "Медичні карти", тут у нас представлений список доданих до системи медичних карток призовників. Тут так само ми можемо оновити існуючу медичну карту або створити нову (Рис. 4.3).

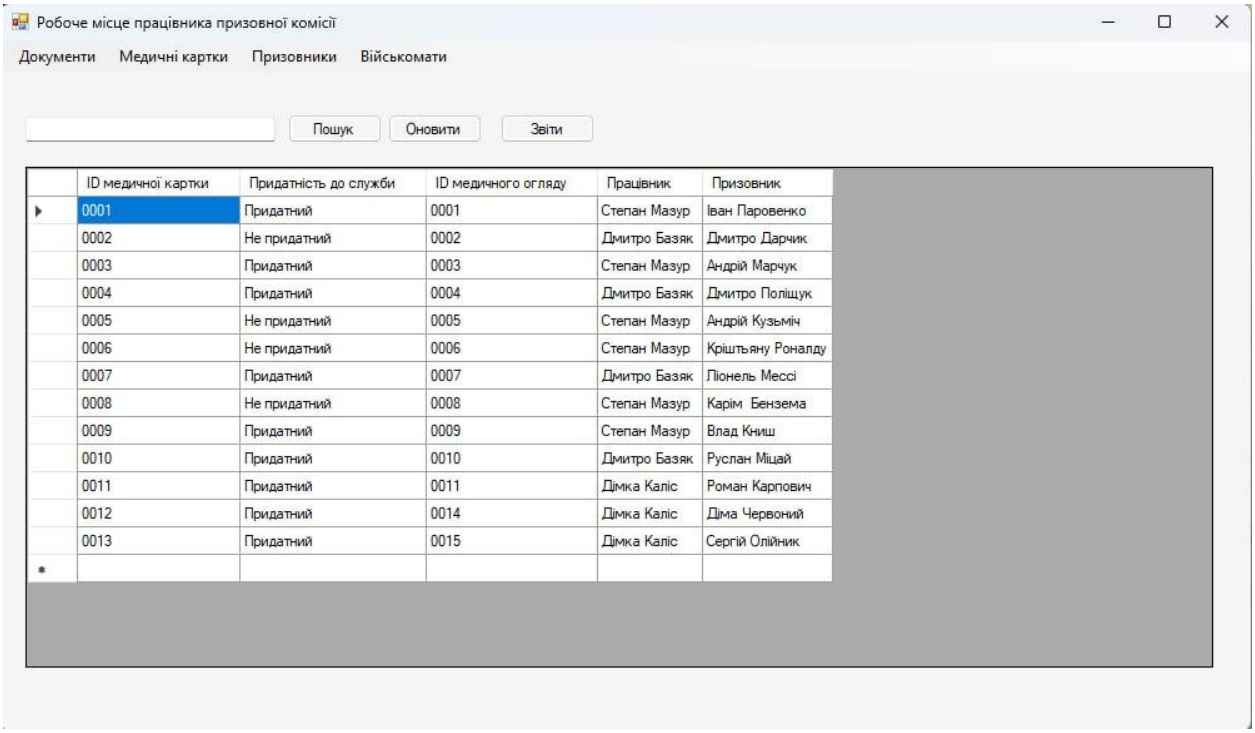


Рис. 4.3 – Сторінка “Медичні картки”

Створюючи мед.карту, нам потрібно вибрати призовника, працівника військкомату, мед.огляди та його придатність до служби (рис. 4.5).

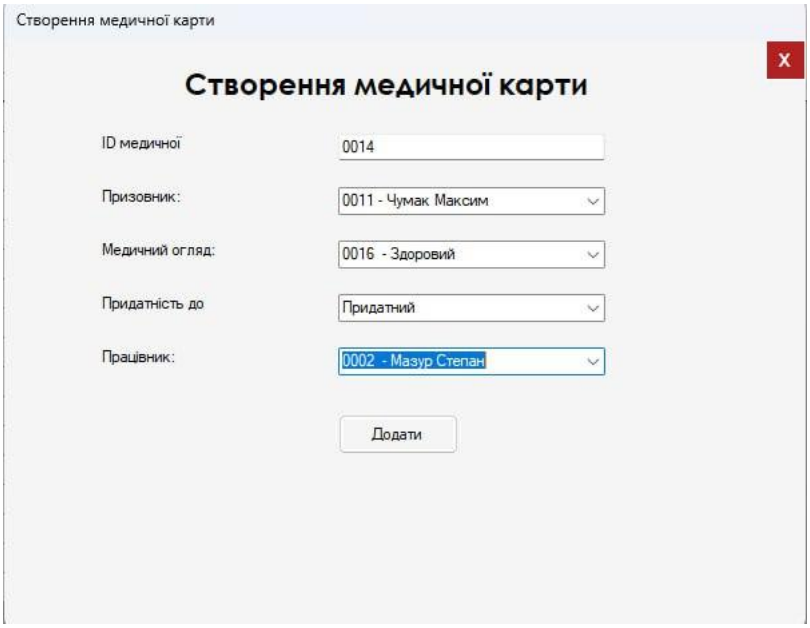


Рис. 4.5 – Форма створення мед.карти

Відкриваємо сторінку "Медичні огляди". Тут у нас представлений список проведених медоглядів за призовниками. Тут вказується результат мед.огляду, працівник військкомату та придатність до служби (Рис. 4.6).

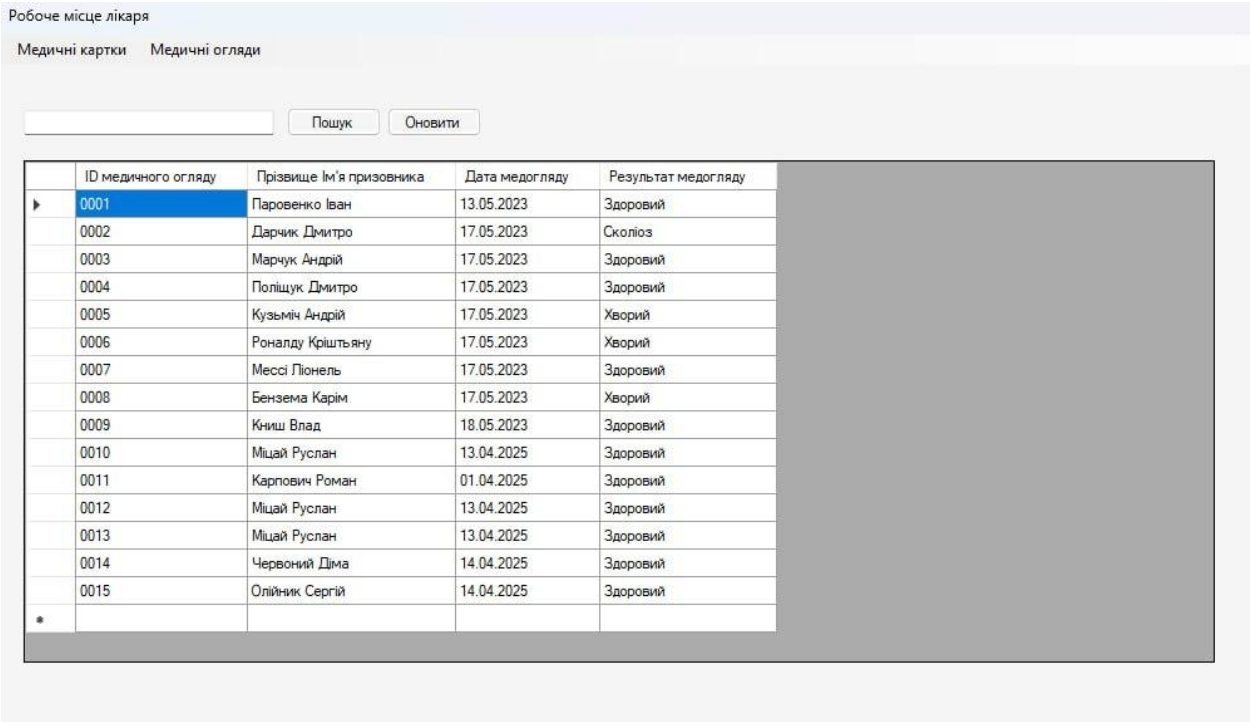


Рис. 4.6 – Сторінка “Медичні огляди”

Проводячи мед.огляд, вибираємо зі списку потрібного нам призовника, дату проведення і підходящий йому результат огляду (рис.4.7).

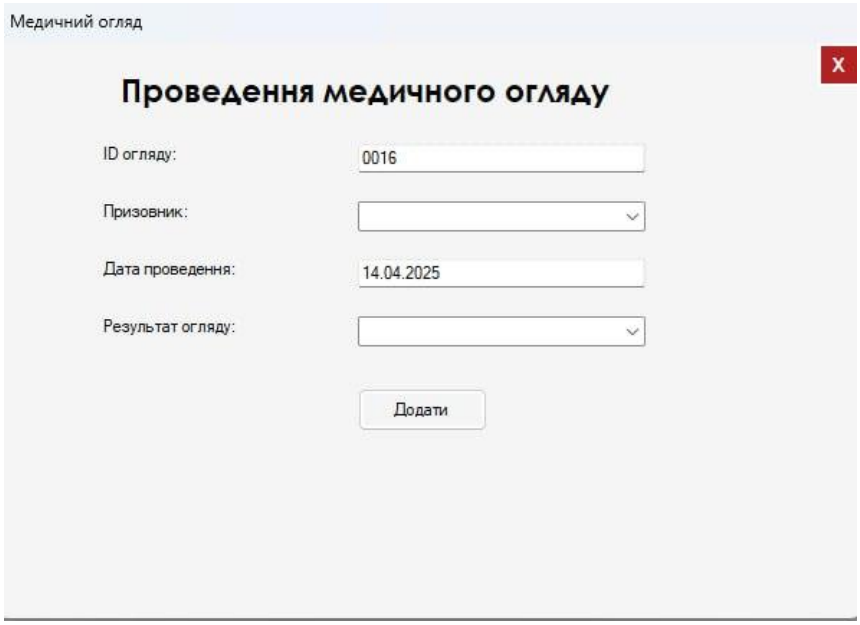


Рис. 4.7 – Форма проведення мед.огляду

Переходимо на сторінку "Звіти", де нам пропонується вибрати тип звіту, який необхідно згенерувати. На цій сторінці представлений список доступних звітів:

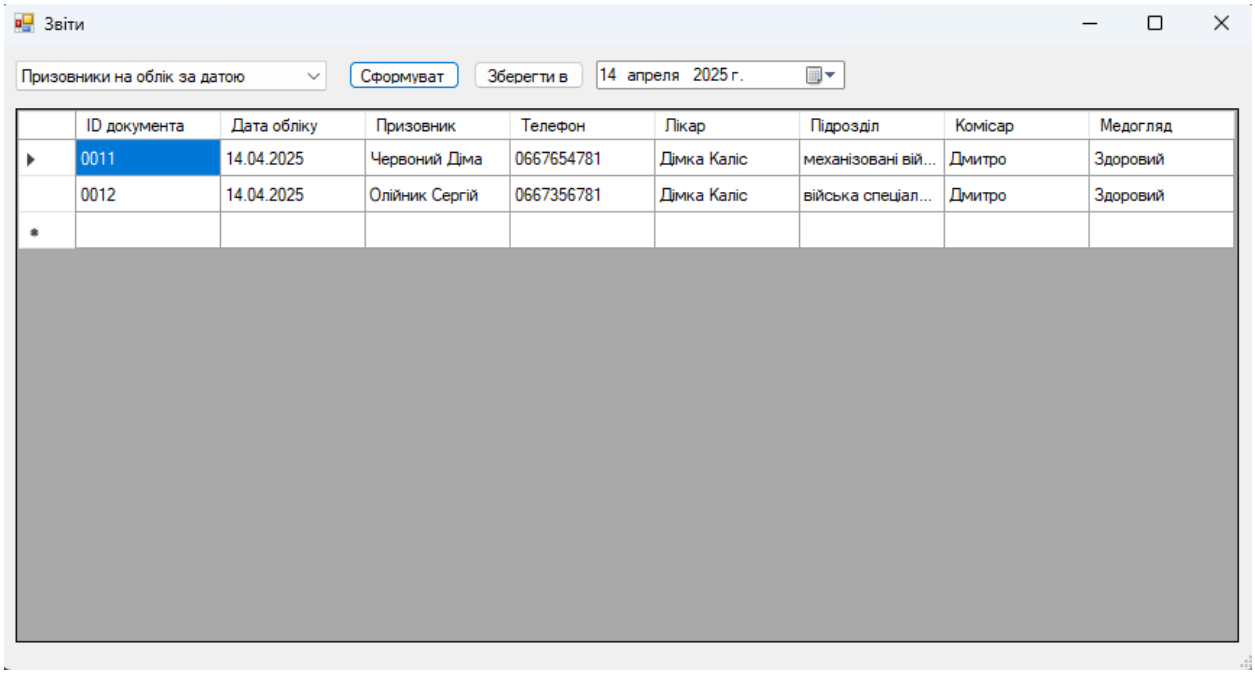


Рис. 4.8 – Форма формування звітів

Після формування звіту користувач має можливість зберегти його у форматі PDF.

Після проведення всіх етапів тестування система показала стабільну та коректну роботу відповідно до поставлених вимог. Усі функції, включаючи авторизацію, роботу з документами, медичними картами, медичними оглядами та формування звітів, успішно протестовані на реальних сценаріях використання.

ВИСНОВКИ

Розробка програмного забезпечення для працівника призовної комісії вирішує нагальну потребу в автоматизації та оптимізації процесів, пов’язаних з організацією та веденням даних призовників. Завдяки впровадженню цієї системи спрощено такі рутинні завдання, як реєстрація призовників, ведення їх медичних карток, проведення медичних оглядів і визначення придатності до служби, що мінімізує ймовірність помилок і скорочує час, витрачений на ручну роботу з документами.

Запропоноване рішення, створене з використанням C# і WinForms framework, у поєднанні з базою даних Microsoft SQL Server, пропонує надійний і зручний інтерфейс, адаптований до потреб військових комісарів і медичного персоналу. Система розроблена з урахуванням як функціональних, так і нефункціональних вимог, що гарантує стабільну продуктивність, інтуїтивно зрозумілу роботу та безпеку даних.

Аналіз існуючих аналогів і вивчення сучасних викликів у системі призову підкреслили потребу в сучасній цифровій альтернативі. Розроблений додаток відповідає на ці виклики, забезпечуючи централізоване зберігання даних, настроювані інструменти звітності та автоматизацію ключових завдань у робочому процесі призову.

Крім безпосереднього практичного використання, розроблене програмне забезпечення сприяє досягненню ширшої мети модернізації адміністративних процедур в державних установах. Завдяки переходу від паперової документації до цифрових систем призовна комісія може забезпечити вищий рівень прозорості, підзвітності та ефективності. Це також відкриває можливості для кращої аналітики та довгострокового зберігання даних, які можуть сприяти прийняттю стратегічних рішень, таких як планування призовних кампаній або аналіз тенденцій у стані здоров’я призовників.

Заглядаючи вперед, система пропонує масштабованість і потенціал для інтеграції з іншими державними або військовими платформами. Наприклад, його можна розширити, включивши автоматизовані системи сповіщень, інтеграцію з електронними медичними записами або модулі віддаленого доступу для авторизованих користувачів. Ці вдосконалення ще більше підвищать ефективність системи та підвищать її значення для процесу призову в цілому. Таким чином, цей проєкт не тільки вирішує поточні оперативні питання, але й закладає основу для постійного технологічного прогресу в оборонному секторі.

Підсумовуючи, система, розроблена як частина цієї бакалаврської роботи, не тільки відповідає початковим цілям проєкту, але й закладає основу для подальшого вдосконалення та розширення. Вона суттєво покращує умови праці працівників призовних комісій та є вагомим кроком до цифрової трансформації державної військової служби.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Багаторівнева архітектура – [Електронний ресурс] – Режим доступу: https://simpleone.ru/glossary/mnogourovnevaya-arhitektura/
2. The Clean Architecture – [Електронний ресурс] – Режим доступу: https://medium.com/clean-code-channel/clean-architecture-the-solution-to-have-a-reusable-flexible-and-testable-code-ac7e296d1a75
3. Типи архітектури програмного забезпечення – [Електронний ресурс] – Режим доступу: https://medium.com/nuances-of-programming/4-типа-архитектуры-программного-обеспечения-917133174724
4. What is Unified Modeling Language (UML)? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
5. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/
6. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/fd_use-case-diagram.html
7. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ru/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/
8. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
9. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
10. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ru/gpg_class-diagram.html

11. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ru/uml-classes/
12. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
13. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
14. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.
15. W3Schools – [Електронний ресурс] – Режим доступу: www.w3schools.com
16. Microsoft Docs. (2021). Layered architecture pattern – [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/azure/architecture/patterns/layered
17. What is Component Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
18. Deployment Diagram in UML: Definition, Examples & Components – [Електронний ресурс] – Режим доступу: https://study.com/academy/lesson/deployment-diagram-in-uml-definition-examples-components.html
19. What is a data flow diagram? – [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/pages/data-flow-diagram
20. Соммервіль, І. (2016). Інженерія програмного забезпечення (10-е вид.). Pearson Education.
21. Прессман Р. С. та Максим Б. Р. (2015). Software Engineering: A Practitioner’s Approach (8-е видання). Освіта McGraw-Hill.

22. Пун, А., Лоу, К. (2017). «Приплив електронної комерції в індустрію гостинності: дослідження онлайн-туристичних агентств». Міжнародний журнал досліджень туризму, 19(6), 703–710.
23. Коннолі, Т., і Бегг, К. (2014). Системи баз даних: практичний підхід до проєктування, впровадження та управління (6-е видання). Pearson Education.
24. Фріман, Е., Робсон, Е. (2021). Head First Design Patterns (2-е видання). O'Reilly Media.
25. Корпорація Microsoft. (2023). Документація Microsoft SQL Server. [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/sql
26. Ріхтер, Дж. (2012). CLR через C# (4-е видання). Microsoft Press.
27. Альбахарі, Дж., і Альбахарі, Б. (2021). C# 10 in a Nutshell (8-е видання). O'Reilly Media.
28. Буч, Г., Рамбо, Дж., і Якобсон, І. (2005). Посібник користувача з уніфікованої мови моделювання (2-е видання). Аддісон-Уеслі.
29. Якобсон І., Крістерсон М., Йонссон П. та Овергаард Г. (1992). Об'єктно-орієнтоване програмне забезпечення: підхід, орієнтований на використання. Аддісон-Уеслі.
30. Васильєв, А. Н. (2020). «Автоматизація кадрового діловодства в державних установах». Журнал інформаційних систем та державного управління, 7(2), 122–130.
31. Парсонс, Д. та Оджа, Д. (2016). Нові погляди на комп’ютерні концепції 2016. Cengage Learning.
32. Беннетт С., МакРобб С. та Фармер Р. (2010). Об'єктно-орієнтований системний аналіз і проєктування з використанням UML (4-е видання). Макгроу-Хілл

ДОДАТОК А

Фрагменти програмного коду. Код підключення до бази даних під різними користувачами

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace OBD
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            bool isTruLoginAndPas = false;

            if (!string.IsNullOrWhiteSpace(textBoxLogin.Text) &&
!string.IsNullOrWhiteSpace(textBoxPassword.Text))
            {
                var dbconnection = new DBConnection();
                string login = textBoxLogin.Text;
                string pass = textBoxPassword.Text;

                var result = dbconnection.Login(login, pass);
                if (result.Success)
                {
                    dbconnection.Connect(login, pass);
                    dbconnection.openConnection();

                    string query = $"IF IS_ROLEMEMBER('Komisar', '{login}') = 1 AND
EXISTS (SELECT 1 FROM sys.sql_logins WHERE name = '{login}' AND PWDCOMPARE('{pass}',
password_hash) = 1)" +
                    "SELECT 'Користувач має роль адміністратора' AS [Результат]";
                    SqlCommand comand = new SqlCommand(query,
dbconnection.GetConnection());
                    SqlDataReader reader = comand.ExecuteReader();
                    if (reader.Read())
                    {
                        isTruLoginAndPas = true;
                        MessageBox.Show($"Ви адмін - {login}! Доступ надано!",
"Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        this.Hide();
                        FormForKomisar form2 = new FormForKomisar();
                        form2.Show();
                    }
                    reader.Close();
```

```
                    query = $"IF IS_ROLEMEMBER('Worker_', '{login}') = 1 AND EXISTS
(SELECT 1 FROM sys.sql_logins WHERE name = '{login}' AND PWDCOMPARE('{pass}',
password_hash) = 1)" +
                    "SELECT 'Користувач має роль адміністратора' AS [Результат]";
                    SqlCommand command = new SqlCommand(query,
dbconnection.GetConnection());
                    comand = new SqlCommand(query, dbconnection.GetConnection());
                    reader = comand.ExecuteReader();
                    if (reader.Read())
                    {
                        isTruLoginAndPas = true;
                        MessageBox.Show($"Ви працівник  - {login}! Доступ надано!",
"Повідомлення!", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        this.Hide();
                        FormForWorker form3 = new FormForWorker();

                        form3.Show();

                    }
                    reader.Close();

                    query = $"IF IS_ROLEMEMBER('Doktor', '{login}') = 1 AND EXISTS
(SELECT 1 FROM sys.sql_logins WHERE name = '{login}' AND PWDCOMPARE('{pass}',
password_hash) = 1)" +
                    "SELECT 'Користувач має роль адміністратора' AS [Результат]";
                    comand = new SqlCommand(query, dbconnection.GetConnection());
                    reader = comand.ExecuteReader();
                    if (reader.Read())
                    {
                        isTruLoginAndPas = true;
                        MessageBox.Show($"Ви доктор - {login}! Доступ надано!",
"Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        this.Hide();
                        FormForDoktor form4 = new FormForDoktor();
                        form4.Show();
                    }
                    reader.Close();
                }
                if(!isTruLoginAndPas)
                {
                    MessageBox.Show("Неправильний логін або пароль!", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }

            }
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            this.Close();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.FormBorderStyle = FormBorderStyle.FixedSingle;
        }

        private void textBoxLogin_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

ДОДАТОК Б

Фрагменти програмного коду. Методи заповнення DataGridView даними з бази данних з використанням Select

```
private void переглядІнформаціїПроЖурналToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Common;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace OBD
{
    public partial class FormForWorker : Form
    {
        public FormForWorker()
        {
            InitializeComponent();
        }

        private void Form3_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form form1 = Application.OpenForms.OfType<Form>().FirstOrDefault();
            this.Close();
            form1.Close();
        }

        private void переглядToolStripMenuItem_Click(object sender, EventArgs e)
        {
            dataGridView1.Show();

            DBConnection dB = new DBConnection();
            dB.Connect("", "");
            dB.openConnection();

            string query = "SELECT  idDocument as[ID документа],dateOfCreation
as[Дата створення], idWorker as[ID працівника],idConscript as[ID призовника]," +
                "idMilitarymate as[ID військомата],idMilitaryCommissar[ID
комісара],idMilitaryUnit as[ID підрозділу]," +
                "IdMedecalCard as[ID медичної карти] FROM document";
            SqlCommand command = new SqlCommand(query, dB.GetConnection());

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            adapter.Fill(table);
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = table;
```

```
        }

        private void переглядToolStripMenuItem2_Click(object sender, EventArgs e)
        {
            dataGridView1.Show();

            DBConnection dB = new DBConnection();
            dB.Connect("", "");
            dB.openConnection();

            string query = "SELECT idMedecalCard as[ID медичної
картки],suitabilityForMilitaryService as[Придатність до служби],idMedicalExamination
" +
                "as[ID медичного огляду],idWorker as[ID працівника]," +
                "idConscript as[ID призовника] FROM medicalCard";
            SqlCommand command = new SqlCommand(query, dB.GetConnection());

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            adapter.Fill(table);
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = table;
        }

        private void переглядToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            dataGridView1.Show();

            DBConnection dB = new DBConnection();
            dB.Connect("", "");
            dB.openConnection();

            string query = "SELECT idConscript as[ID призовника],nameConscript
as[Ім'я], surnameConscript as[Прізвище], gender as[Стать], " +
                "ageConscript as[Вік], hightConscript as[Ріст], weightConscript as
[Вага], phoneConscript as[Номер телефона], " +
                "idRank as[ID Рангу] FROM conscript";
            SqlCommand command = new SqlCommand(query, dB.GetConnection());

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            adapter.Fill(table);
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = table;
        }

        private void медичніОглядиToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            dataGridView1.Show();

            DBConnection dB = new DBConnection();
            dB.Connect("", "");
            dB.openConnection();

            string query = "SELECT idMedicalExamination as[ID],idConscript as[ID
призовника],dateOfMedicalExamination as[Дата огляду]," +
                "resultOfMedecalExamination as[Результат обстеження] FROM
medicalExamination";
            SqlCommand command = new SqlCommand(query, dB.GetConnection());

            DataTable table = new DataTable();
```

```
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            adapter.Fill(table);
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = table;
        }



        private void переглядToolStripMenuItem4_Click(object sender, EventArgs e)
        {
            dataGridView1.Show();

            DBConnection dB = new DBConnection();
            dB.Connect("", "");
            dB.openConnection();

            string query = "SELECT idMilitaryMate as[ID військомату]," +
                "nameMilitaryMate as[Назва військомату],adress as[Адреса] FROM
militaryMate";
            SqlCommand command = new SqlCommand(query, dB.GetConnection());

            DataTable table = new DataTable();
            SqlDataAdapter adapter = new SqlDataAdapter(command);
            adapter.Fill(table);
            dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.AllCells;
            dataGridView1.DataSource = table;

        }

        private void додаванняToolStripMenuItem2_Click(object sender, EventArgs e)
        {
            addMilitaryMate addMilitaryMate = new addMilitaryMate();
            addMilitaryMate.ShowDialog();
        }

        private void додаванняToolStripMenuItem_Click(object sender, EventArgs e)
        {
            addConscript addConscript = new addConscript();
            addConscript.ShowDialog();
        }

        private void додаванняToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            addDocument addDocument = new addDocument();
            addDocument.ShowDialog();
        }

        private void переглядToolStripMenuItem3_Click(object sender, EventArgs e)
        {

        }
    }
}


private void button1_Click(object sender, EventArgs e)
        {
            string idDoc = comboBox1.Text.ToString().Substring(0, 4);
            DateTime date = DateTime.Parse(textBox10.Text);
            string idWorker = comboBox3.Text.ToString().Substring(0, 4);
            string idMilitariMate = comboBox5.Text.ToString().Substring(0, 4);
            string idKomisar = comboBox4.Text.ToString().Substring(0, 4);
            string idUnit = comboBox6.Text.ToString().Substring(0, 4);
```

```
            DBConnection conn = new DBConnection();
            conn.updateDocument(idDoc, date, idWorker, idMilitariMate, idKomisar,
idUnit);
            MessageBox.Show($"Документ {idDoc} оновленийщш успішно.");
            string selectedComputer = comboBox1.SelectedItem.ToString();
            string computerId = selectedComputer.Split(' ')[0];

            string selectedDevice = comboBox2.SelectedItem.ToString();
            string deviceId = selectedDevice.Split(' ')[0];


            // Оновлюємо запис
            SqlCommand updateCommand = new SqlCommand("UPDATE Devices SET
ID_computer = @computerId WHERE ID_device = @deviceid",
dbconnection.GetConnection());
            updateCommand.Parameters.AddWithValue("@computerId", computerId);
            updateCommand.Parameters.AddWithValue("@deviceid", deviceId);
            updateCommand.ExecuteNonQuery();

            // Виводимо повідомлення про успішне оновлення
            MessageBox.Show("Девайс обновлено і підключений до компа");

            // Закриваємо підключення до бази даних
            dbconnection.closeConnection();
        }
```