НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри
_____ Сергій СТІРЕНКО

“___”_____2020 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп’ютерні системи та мережі»

спеціальності 123 «Комп’ютерна інженерія»

на тему: «Додаток на Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків»

Виконав:
студент IV курсу, групи ІО-62
Кушнір Олександр Сергійович _____

Керівник:
Доцент, с.н.с., к.т.н.,
Антонюк Андрій Іванович _____

Консультант з нормоконтролю:
Професор, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Доцент, к.т.н.,
Пасько Віктор Петрович _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврьский)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___» _____2020 р.

ЗАВДАННЯ
на дипломний проект студента

Кушніра Олександра Сергійовича

1. Тема проекту «Додаток на Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків»
керівник проекту Антонюк Андрій Іванович, с.н.с., к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «07» травня 2020 року №1081-с
2. Термін здачі студентом закінченого проекту
_____
3. Вихідні дані до проекту *технічна документація*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Розробка сервісу збору даних про пацієнтів лікарень та оптимізації поставки ліків для мобільної платформи Android з використанням сучасних технологій розробки архітектури та дизайну додатку. Опис предметної області, дослідження засобів розробки програмного забезпечення, розробка алгоритму для вирішення задачі прогнозування, реалізація роботи з БД, програмна реалізація та тестування.
5. Перелік графічного матеріалу
Принципова схема, функціональна схема та структурна схема

6. Консультанти проекту, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
нормоконтроль	д.т.н., проф. Сімоненко В.П.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту	Строк виконання етапів проекту	Примітки
1.	*Затвердження теми роботи*	*1.09.2019*	
2.	*Вивчення та аналіз завдання*	*2.09.2019-15.03.2020*	
3.	*Розробка архітектури та загальної структури програми*	*15.03.2020-25.03.2020*	
4.	*Розробка структур окремих Інтерфейсів програми*	*25.03.2020-5.04.2020*	
5.	*Програмна реалізація*	*5.04.2020-15.04.2020*	
6.	*Оформлення пояснювальної записки*	*15.04.2020-20.05.2020*	
7.	*Захист програмного продукту*	*21.05.2020 – 25.05.2020*	
8.	*Передзахист*	*26.05.2020*	
9.	*Захист*		

Студент Олександр КУШНІР

Керівник Андрій АНТОНЮК

АНОТАЦІЯ

В бакалаврській дипломній роботі створено Android-додаток, необхідний приватним (чи деяким державним) лікарям для спрощення процедури збереження клієнтської бази та закупки ліків або витратних матеріалів, необхідних для роботи.

Завдяки цій програмі лікар може зберігати деякі дані про пацієнта, та в майбутньому знаходити пацієнтів в базі та переглядати ці дані. Це буде корисно для роботи лікаря, бо він завжди зможе знайти номер телефону конкретного пацієнта та зв’язатися з ним, або дізнатися попередні діагнози пацієнта, якщо він звернувся до лікаря декілька разів. Також лікар зможе отримати прогноз закупки необхідних для роботи матеріалів, виходячи з витрат попереднього місяця. Це дозволить оптимізувати поставку ліків.

Для розробки відповідної програми, використано мову Java та засобом програмування алгоритму є середа розробки Android Studio.

ANNOTATION

In the Bachelor’s Degree work created an Android application needed by private (or some public) doctors to simplify the procedure of maintaining the client base and purchasing drugs or consumables needed for the job.

With this application, the doctor can store some patient data, and in the future find patients in the database and view this data. This will be useful for the doctor, as he will always be able to find and contact the phone number of a particular patient, or find out the patient's previous diagnoses if he has consulted a doctor several times. The doctor will also be able to get a forecast of the purchase of materials needed for the work, based on the costs of the previous month. This will optimize the supply of drugs.

To develop the appropriate program, the Java language is used and the tool for programming the algorithm is the Android Studio development environment.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1.	А4		Завдання на дипломний проект	2	
2.	А4	ДП.4678.02.000 ТЗ	Технічне завдання	4	
3.	А4	ДП.4678.03.000 ПЗ	Пояснювальна записка	67	
4.	А4	ДП.4678.04.000 А1	Принципова схема алгоритму	1	
5.	А4	ДП.4678.05.000 А2	Функціональна схема	1	
6.	А4	ДП.4678.06.000 А3	Структурна схема	1	
7.	А4	ДП.4678.07.000 Б1	Лістинг програми	25	

Технічне завдання
до дипломного проекту

на тему: «Додаток на Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків»

Київ – 2020

ЗМІСТ

					ДП.4678.02.000 ТЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Кушнір О.С.			Додаток на Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків Технічне завдання	Літ.	Аркуш 1 / Аркушів 4
Перевірив		Антонюк А.І.					
Реценз.						НТУУ «КПІ», ФІОТ, ІО-62	
Н. Контр.		Сімоненко В.П.					
Затв.							

1. Загальні положення

1.1 Повне найменування програми

Повним найменуванням програми є - «Додаток на Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків»

1.2 Організація-замовник додатку

Замовником програми є кафедра Обчислювальної Техніки

1.3 Розробник додатку

Розробником додатку є студент групи ІО-62 ФІОТ Кушнір Олександр

1.4 Підстави для розробки

Підставою для розробки є технічне завдання стверджене кафедрою ОТ, факультету інформатики та обчислювальної техніки, Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

2. Область застосування

В даній роботі було розроблено додаток для операційної системи Android. Ця система є дуже розповсюдженою, тому додаток може бути використаний в необхідній сфері. Сферою застосування в першу чергу є невеликі приватні лікарні, які тільки відкриваються та не мають добре налаштованої системи поставки ліків.

3. Призначення та цілі розробки

3.1 Призначення розробки

Призначенням є створення додатку на мобільні телефони для допомоги в роботі лікарям.

3.2 Цілі розробки

На мою думку, основними цілями моєї розробки є:

- Забезпечення лікаря можливістю збору інформації про пацієнтів;
- Забезпечення можливості пошуку необхідної інформації з бази;
- Вивід кількості ліків та витратних матеріалів які є в наявності;
- Можливість додатку робити оптимізацію закупки всього необхідного шляхом прогнозування витрат на заданий період.

4. Джерела розробки

Джерелами розробки є:

- Науково-технічна література;
- Сайти необхідної тематики в Інтернеті;
- Документація Android SDK;
- Документация SQL;
- Документація Java Core.

5. Спеціальні вимоги

Проектування програми виконується мовою програмування Java в середі розробки Android Studio останньої версії. Для розробки було використано пристрій з ОС Windows 10 та процесором Intel Core-i5. На мою думку для комфортної розробки необхідно мати не менше 8гб оперативної пам’яті. При тестуванні результатів розробки використовувався вбудований в Android Studio емулятор для швидкого запуску програми на комп’ютері, який для своєї роботи використовує мінімум 2гб оперативної пам’яті.

Що стосується використання вже готової програми, то для цього немає складних вимог. Все що необхідно – це мати смартфон з версією Android не нижче за 8.0 (Oreo).

6. Етапи і стадії розробки

Етапи та стадії розробки показані нижче, в таблиці.

Етап	Зміст робіт	Результат робіт	Термін виконання
1	Створення ТЗ	Технічне Завдання	Березень 2020
2	Аналіз області застосування, аналогів, розробка схеми роботи програми	Готовий макет програми, який включає в себе схему роботи	Березень 2020
3	Розробка взаємодії БД з програмою	Готовий макет БД	Березень 2020
4	Реалізація функціоналу програми	Програма, яка вже може виконувати цілі розробки	Квітень 2020
5	Тестування програми, виправлення помилок.	Програма, яка працює без помилок	Травень 2020

Пояснювальна записка
до дипломного проекту

на тему: «Додаток на Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків»

Київ – 2020

ЗМІСТ

СПИСОК СКОРОЧЕННЬ

ОС	Операційна система
БД	База даних
VR	Virtual reality
AR	Augmented reality
ШІ	Штучний інтелект
МІС	Медичні інформаційні системи
JVM	Java Virtual Machine
MVC	Model-View-Controller
ЦП	Центральний процесор
ОП	Оперативна пам’ять

ВСТУП

Свій дипломний проект хочу почати з пояснення теми роботи та конкретної сфери застосування додатку, розробленого в цьому проекті.

Моя мама працює стоматологом. Тому все своє життя я був дуже близький до медичної сфери в нашій державі. Майже з самого дитинства я бачив недоліки нашої системи охорони здоров'я. І коли я досяг все свідомого віку мені хотілося чимось допомогти лікарні, в котрій працює моя мама та системі в цілому.

Щомісяця я міг спостерігати за тим як моя мама планує закупівлю матеріалу, необхідного для роботи. Це і знеболюючі ліки для уколів, і пломбувальний матеріал, і навіть медичні маски або рукавички. Загалом, абсолютно все, що повинна надавати держава, стоматологи мого міста закуповували самостійно і за свої кошти. І кожна така закупівля відбувалася «на око». Тобто лікар приблизно розумів скільки робочого матеріалу йому необхідно на місяць і робив замовлення.

Але на щастя ми живемо в сучасному світі і процес діджиталізації відбувається з неймовірною швидкістю, за якою нажаль не все старше покоління встигає. Тому в лікарні мого рідного міста все ще всі закупівлі відбуваються за старими принципами. Я хочу хоч трохи вплинути на систему і створити мобільний додаток, яке зможе автоматизувати процес прогнозу витрати ліків, оптимізувавши тим самим процес закупівлі.

Так само, програма, розроблена в даній роботі, буде мати функцію збору даних про пацієнтів конкретного лікаря. Це допоможе позбутися паперів і в майбутньому перенесе базу пацієнтів в смартфон. Це буде корисно, наприклад, для запису хворих на майбутнє, плануванні лікарем свого часу або надасть доступ до інформації про попередні діагнози конкретного пацієнта.

Мета роботи

Створити додаток для ОС Android, направлений на покращення умов роботи лікарів, для оптимізації процесу закупок ліків та для створення електронного обліку пацієнтів. Протестувати роботу створеної програми.

Постановка задачі

Виходячи з мети роботи, можна сформувати такі завдання:

1. Ознайомитися зі сферою застосування.
2. Провести огляд існуючих рішень, знайти їх плюси та недоліки.
3. Створити макет роботи програми ти взаємодію з БД.
4. Розробити алгоритм, необхідний для реалізації прогнозу закупок.
5. Написати код програми мовою Java.
6. Провести роботу над дизайном.
7. Протестувати програму.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ.

1.1. Аналіз предметної області

Виходячи з теми роботи відразу зрозуміло, що сфера використання мого програмного продукту - це медицина. Програма розробляється для лікарів різних спеціальностей, робота яких тісно контактує з постійною потребою в закупівлі ліків або записом пацієнтів на конкретний час. Я думаю, що в першу чергу мають потребу в цьому приватні клініки.

Тому лікар, який відкриває сучасну клініку, повинен бути зацікавлений в тому, щоб ця клініка відповідала всім сучасним стандартам медицини. Світова практика показує, що в сучасному світі IT-технології та медицина тісно пов'язані один з одним.

На наших очах формується майбутнє охорони здоров'я і це в першу чергу пов'язано з розвитком цифрових технологій. Практика показує, що сучасні IT-технології, які спочатку створювалися не для медицини, допомагають лікарям усього світу справлятися із захворюваннями. Серед таких: 3D-друк, робототехніка, VR, AR, штучний інтелект, нанотехнології.

Майбутнє охорони здоров'я полягає в тісній співпраці з цими технологіями. Медичні працівники повинні розуміти це і освоїти всі ці технології, щоб їх робота була актуальна в найближчі роки.

Ознайомимося з деякими останніми розробками щоб мати уявлення про затребуваність IT-технологій в медицині.

Штучний інтелект

Мені здається, що ШІ в майбутньому здатний повністю реорганізувати охорону здоров'я, адже його алгоритми здатні на дуже багато. ШІ може допомогти системі охорони здоров'я знаходженням більш ефективних рішень лікування важких хвороб. Наприклад, розробляти плани лікування з

огляду на статистику одужання пацієнтом, знаходити більш ефективні ліки швидше, ніж це може зробити лікар.

Наприклад, компанія DeepMind від Google створила додаток для ранньої діагностики раку. Так ось алгоритм цієї програми перевершив всі існуючі рішення і показник раннього виявлення раку молочної залози виріс на 12%. [12]

Медичні трекери і переносні пристрої

Ця технологія доступна на нашому ринку досить давно і вже набрала популярності серед всіх верств населення. Безліч фітнес-браслетів дозволяють в реальному часі стежити за своєю активністю, пульсом і навіть складають звіт про якість сну.

Цікаво те, що завдяки цьому пристрою, яке багато молодих людей купують просто як стильні годинники, збільшилось виявлення прихованих форм захворювань серцево-судинної системи. А зараз за ціною ці трекери доступні практично кожному.

3D-друк

3D-друк приносить чудеса в усі аспекти охорони здоров'я. Тепер стає доступним друк штучних кінцівок, судин, органів і багато іншого. І цей список продовжує збільшуватися постійно, ми зараз за цим спостерігаємо.

У 2019 дослідники з Політехнічного Інституту в Нью-Йорку розробили метод друку живої шкіри разом з судинами. Цей винахід зробив прорив в лікуванні людей з опіками: їм просто пересаджують нову шкіру. [13]

Віртуальна та доповнена реальність

В першу чергу, ці технології використовуються для навчання майбутніх лікарів. Наприклад, хірурги отримують хорошу практику операцій. І це працює. Хірурги, навчені сучасними технологіями, мають набагато вищу продуктивність в майбутній роботі, здійснюючи меншу кількість помилок.

Також користь цієї технології полягає в допомоги пацієнтам справлятися з болем. Пацієнтам, які терплять сильні болі, в минулому

кололи сильні знеболюючі або наркотичні засоби. Зараз, завдяки VR, таким пацієнтам потрібно вже меншу кількість шкідливих для здоров'я ліків.

Ми розглянули тільки частину сучасних технологій, що використовуються зараз в дуже розвинених країнах. Існує ще багато рішень, але про них в цій роботі я писати не буду, так як вони не відносяться безпосередньо до теми роботи. Для загального розуміння того, що медицині необхідні інформаційні технології, наведених прикладів досить. Далі перейдемо до теми медичних інформаційних систем.

1.2. Медичні інформаційні системи

Медичні інформаційні системи - це системи автоматизації обліку документообігу в медицині, що включають в себе системи підтримки ухвалення рішень лікарями, медичні картки пацієнтів, різні дані досліджень і моніторинг стану пацієнтів. МІС можуть включати в себе спілкування між співробітниками і фінансову інформацію. [1]

Від впорядкованості та злагодженості в роботі МІС залежить правильність функціонування всієї галузі медицини. Всі ці процеси розглядає медична інформатика, яка в останні роки була визнана самостійною наукою через затребуваності в сфері медицини.

Медична інформатика - це медико-технічна наука, що є результатом взаємодії інформатики та медицини, в якому інформатика є засобом виконання, а медицина - комплексом завдань. [2]

1.2.1 Класифікація медичних інформаційних систем

МІС можна класифікувати за типами та напрямку діяльності. Розглянемо обидва види класифікації на рис 1.1 та 1.2.

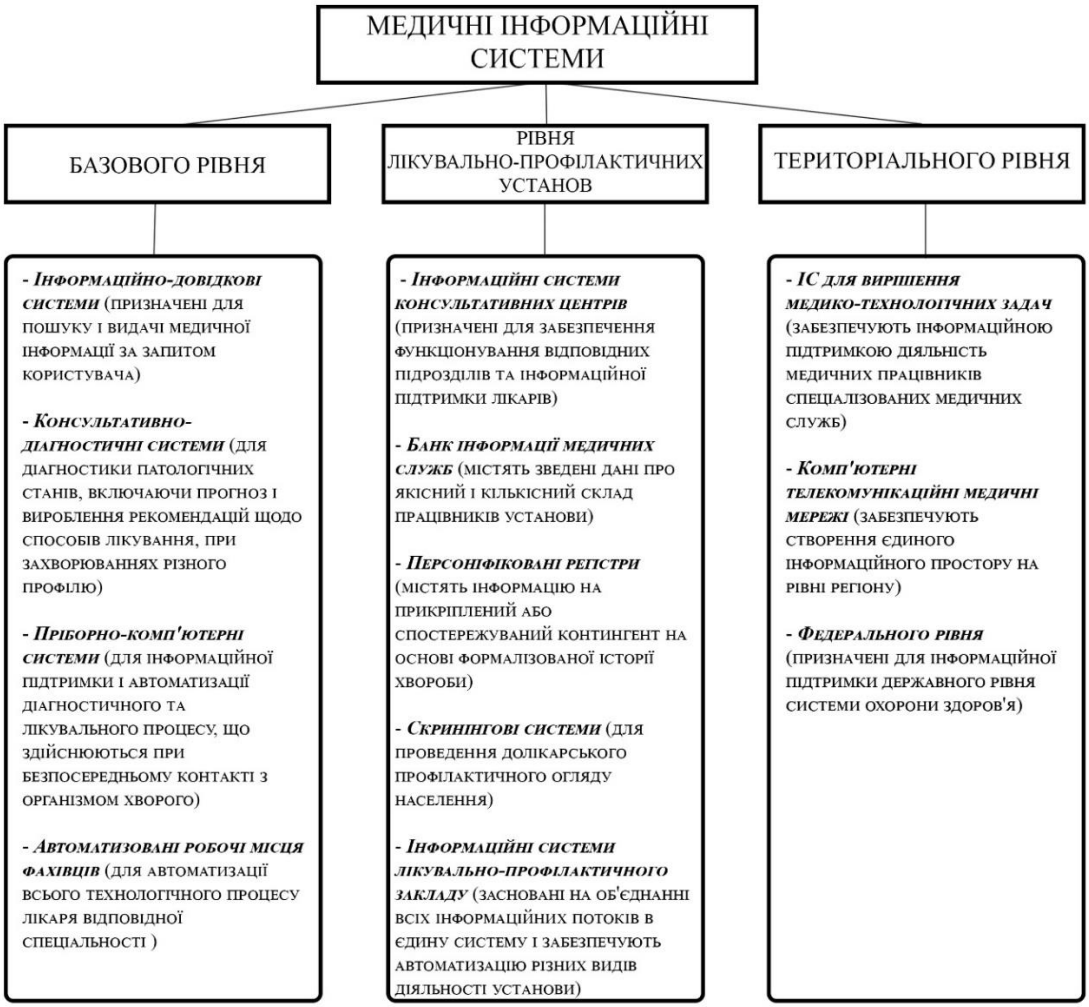


Рисунок 1.1 Класифікація МІС за типом

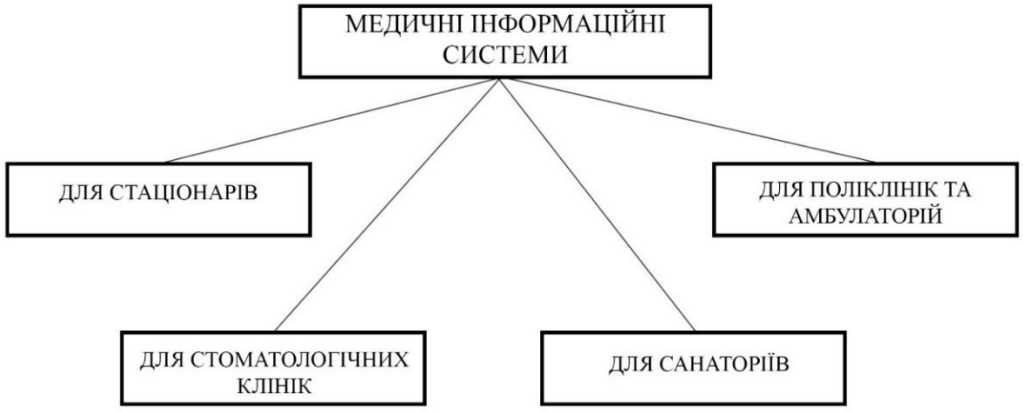


Рисунок 1.2 Класифікація МІС за напрямом діяльності

Отже, програмний продукт, виконаний в даній роботі можна віднести до медичної інформаційної системи для стоматологічних клінік інформаційно-довідникового типу.

1.2.2 Медичні інформаційні системи як продукт

Впровадження МІС приносить в сучасну медицину дуже багато плюсів, але є також і негативні фактори, що заважають швидкому впровадженню. Основна проблема пов'язана з фінансовою стороною. Медичні системи мають непрозоре ціноутворення і непрогнозовану вартість супроводу цих систем в період експлуатації.

На етапі впровадження практично неможливо врахувати всі труднощі, який спливуть в майбутньому. І часто відбувається така ситуація, в якій замовники багато чого не беруть до уваги при створенні ТЗ. Це породжує багато проблем. А коли з'ясовується, що якийсь функціонал системи не був реалізований, розробники просять додаткові гроші на виконання. Виникає конфлікт, так як система медицини не має зайвих коштів для цього, в такому випадку замовники вимагають виконати цю роботу за базовою оплатою.

Коли процес розробки переходить на етап сервісного обслуговування системи, також виникає багато конфліктів. Система зазвичай вимагає багато правок і на цьому етапі. Якщо доробка не була врахована в контракті, розробник буде вимагати оплату.

Такі конфлікти показують, що системі охорони здоров'я в нашій країні необхідні недорогі рішення. Мені здається, що системи, розроблені студентами, можуть бути дуже корисні в цьому плані.

Величезну роль грає функціонал системи, що розробляється. Важливо щоб він виконував всі необхідні для замовника функції і завдання. Так як

ми маємо справу з охороною здоров'я, тут важлива точність і відповідність завдання і результату.

В першу чергу, система повинна мати мінімальний функціонал, необхідний для роботи медичного персоналу. Але я не буду його описувати, так як це буде більш детально описано в інших розділах, коли буде описана безпосередньо розробка програми.

Зараз розглянемо конкретні деталі, які допомагають програмному продукту успішно боротися з конкурентами на ринку.

Багато уваги приділяється наявності порталу пацієнта в системі. Тобто щоб система мала інтерфейс не тільки для персоналу лікарні, а й можливість пацієнту мати доступ до обмеженої інформації. Великого успіху добиваються системи, в яких пацієнт може записатися на прийом до лікаря з програми, не виконуючи зайвих дзвінків. Лікар же, в свою чергу, більше часу витрачає на лікування хворих, ніж на телефонні дзвінки.

До речі, саме такий функціонал був реалізований в медичній реформі України. Кожен пацієнт через сайт може записатися до свого сімейного лікаря. Це дуже зручно і тому це отримує схвалення від користувачів.

Також корисна функція чату з лікарем. Якщо у пацієнта є питання щодо процесу лікування або питання після прийому, він може запитати все у лікаря через додаток.

Існує ще безліч корисних функцій. Наприклад, повідомлення пацієнтам через СМС або додаток. Це буде корисно для нагадування про прийом, або про необхідність вакцинації.

Крім інтерфейсу для пацієнта, можна також додати додатковий функціонал в інтерфейс лікарів. Наприклад, функцію обміну ідеями про лікування, зворотний зв'язок з розробниками для пропозиції поліпшення системи або просто реалізація можливості організації вільного часу.

Але також варто пам'ятати, що потрібно дотримуватися балансу у всьому. Не потрібно навантажувати систему непотрібними функціями, які

рідко будуть використані. МІС - це в першу чергу база даних. І краще зробити упор в її належне функціонування та в реалізацію функціоналу, пов'язаного з різними звітами і статистикою.

Вище було перераховано велику кількість сучасного функціоналу, затребуваного на ринку в цій сфері. Але в даній роботі буде розроблений мінімум, що дозволяє виконувати завдання, описану далі в роботі.

Варто також пам'ятати і про якісний графічний інтерфейс програми і його дизайн. Вдалим прикладом буде не навантажений елементами інтерфейс, а також його гнучкість під конкретного фахівця. Важливо щоб була функція виведення найбільш використовуваних сервісів на зручне місце, легкість в навігації. Це забезпечить швидке переміщення всередині програми, без зайвих витрат часу.

Уникати варто складних процесів, які вимагають складних дій і інтуїтивно незрозумілі. Безліч екранів не затребуване замовником. Кожен екран повинен виконувати конкретний функціонал.

Вище було проаналізовано сучасні тенденції МІС. З цього зрозуміло, що система, яка вводиться на рівні великої клініки, вимагає багато роботи і фахівців різного профілю. Це і front/back-end розробники, і фахівці з БД, і дизайнери. Великі IT-компанії справляються з цим, маючи необхідний ресурс працівників, а то, що буде реалізовано мною в даній роботі описано в наступному пункті.

1.3. Огляд існуючих рішень

Дана робота включає в себе дві основні задачі: облік пацієнтів та оптимізацію закупки ліків. Тому в цьому пункті розділу буде доцільно розділити огляд на дві частини, для кожної задачі окремо. Так можна буде охопити більше інформації для розгляду и отримати ширший результат.

1.3.1 Огляд існуючих МІС

В Україні вже є велика кількість інформаційних систем для лікарень. В цій роботі я зроблю огляд найпопулярніших на момент 2020 року.

Почну з найпопулярнішої та найпоширенішої системи – Helsi (рис. 1.3). Ця система була розроблена за замовленням Міністерства Охорони Здоров'я України. Сама компанія включає в себе велику команду, що складається зі спеціалістів різних сфер діяльності. Над роботою цього сервісу працюють програмісти, дизайнери, аналітики. [3]

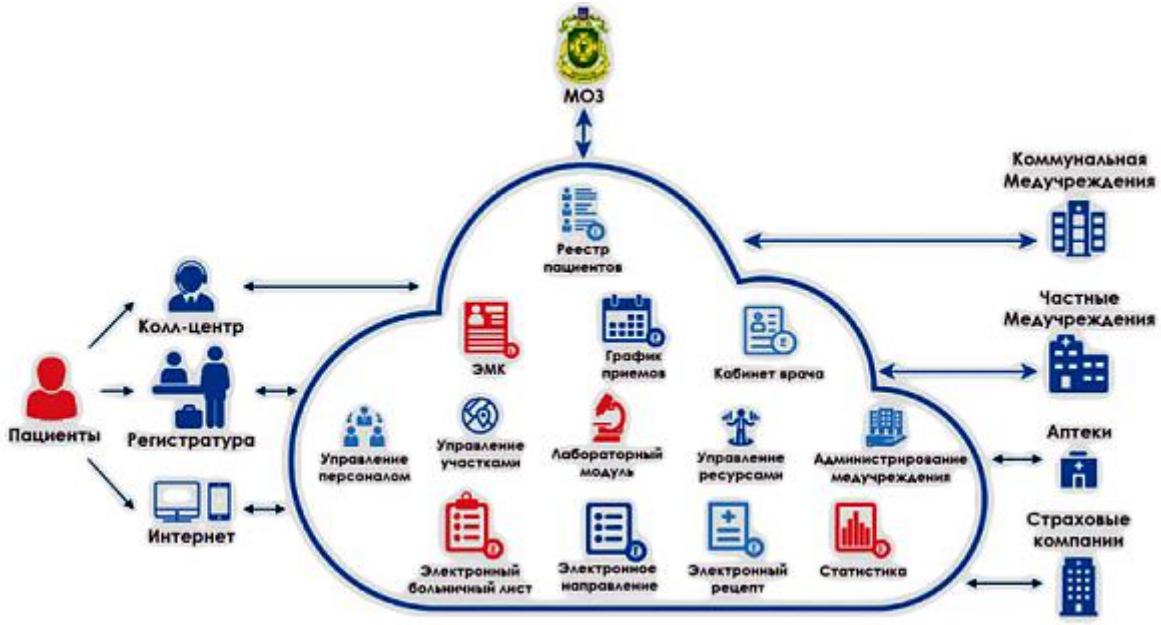


Рисунок 1.3 Схема роботи системи Helsi

Як видно зі схеми ця система має колосальні можливості та охоплює напевно все що тільки можливо. Для пацієнта є можливість обирати собі лікаря, запис на зручний час, облік медичної картки, особистий кабінет з усіма результатами досліджень, що проводилися. Також реалізовано прозору систему плану лікування, до якого пацієнт завжди має доступ.

Система надає можливість лікарям вести прийом пацієнтів, мати доступ до історії хвороби, миттєве отримання результатів дослідження пацієнтів. А медичні заклади отримують можливість налаштовувати систему під конкретний заклад та повний доступ до статистики.

Наступна система для розгляду – це ЕМСіМЕД. Ця система працює все довше, ніж попередня. Вона також має дуже широкий функціонал. Система розроблена за всіма стандартами МОЗ та рекомендована міністерством. [4] Детальніше про можливості ЕМСіМЕД можна дізнатися з рис. 1.4:

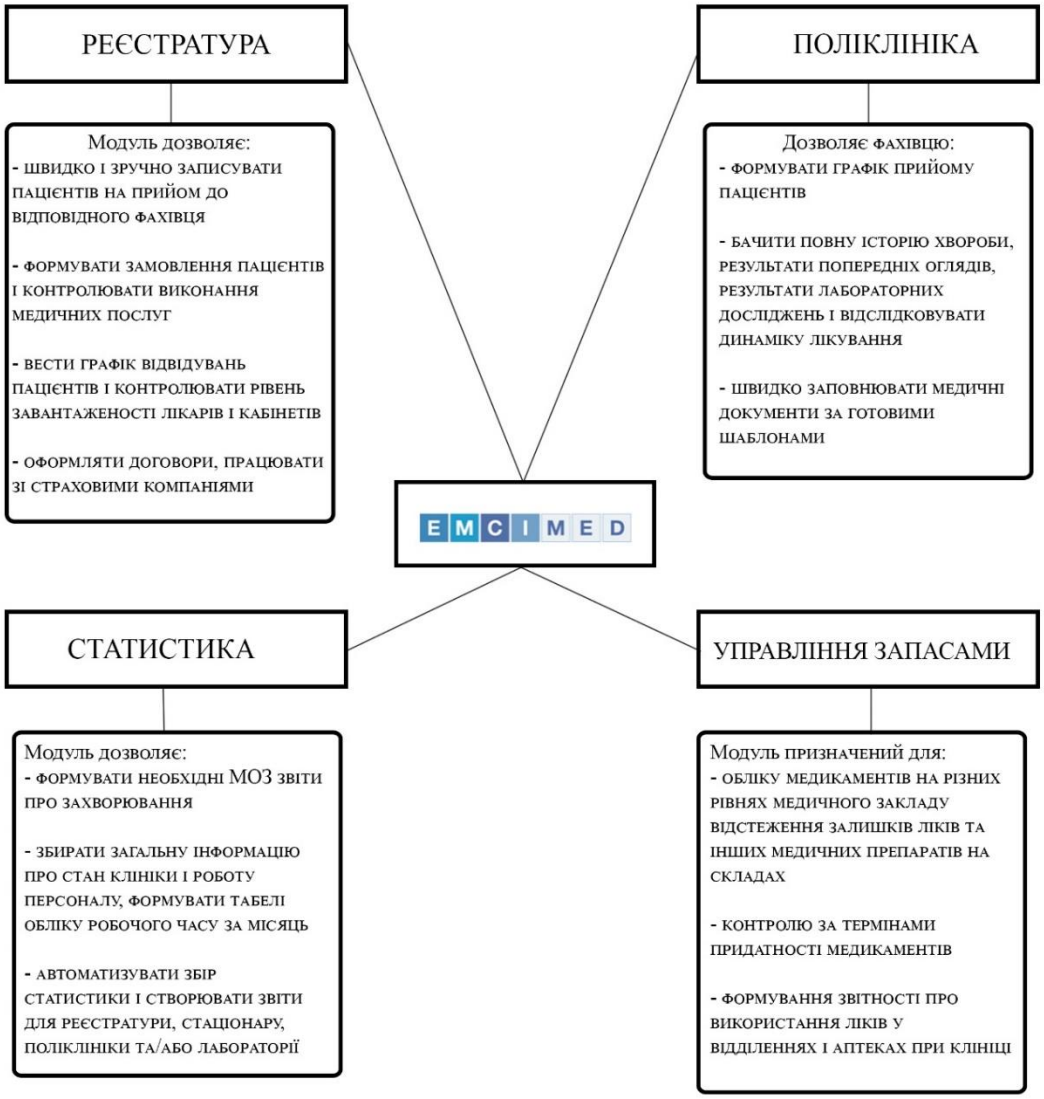


Рисунок 1.4 Схема основних модулів системи ЕМСіМЕД

На схемі зображено лише деякі з можливих функцій системи ЕМСіМЕД. Також, система має можливість управління персоналом та документами, управління всією установою в цілому. Програма має модулі

спостереження за роботою стаціонару та лабораторії. А одним з переваг цієї системи є можливість завантаження додатку на смартфон.

Однак на цьому огляд систем таких масштабів не закінчується. Найпоширенішою медичною системою України є «Доктор Елекс» (рис. 1.5). Ця система працює вже 15 років. Так як і попередні системи, вона автоматизує ключові процеси лікарні, такі як документооборот, ведення історії хвороби, статистики та різних звітів. [11]

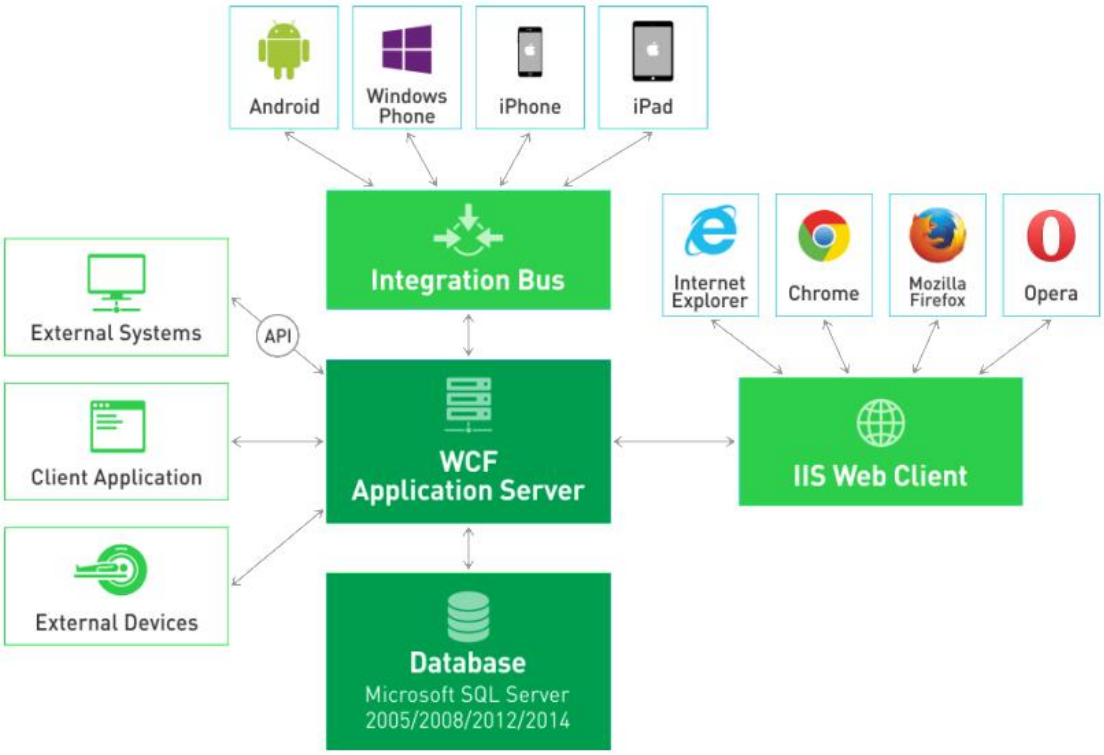


Рисунок 1.5 Архітектура системи Доктор Елекс

Я вважаю, що детальний огляд цієї системи все не потрібен, тому що функціонал також охоплює все що тільки можна уявити. Все, що мають попередні системи, реалізовано в системі Доктор Елекс.

Але на офіційному сайті системи було знайдено ціну ліцензування, програмного забезпечення, впровадження та технічного обслуговування. Проаналізувавши всі ціни можна зробити висновок, що повне впровадження цієї системи буде дуже дорого коштувати. А це означає, що даною системою не зможуть скористуватися невеликі клініки або державні лікарні.

Отже, український ринок має безліч МІС. Але можна зробити висновок, що найпопулярніші системи не зможуть охопити всю сферу, тому що мають високу вартість. Саме тому в лікарні мого рідного міста досі не впроваджено ні одної медичної системи. Я думаю що, нажаль, така ситуація в більшості лікарнях невеликих міст та селищ. А електронними медичними системами користуються в великих містах.

Тому, я вважаю, що студентські роботи можуть допомогти нашій медицині свою простотою у використанні та тим, що вони безкоштовні.

1.3.2 Огляд існуючих систем оптимізації закупок

Попит і поставки описуються цільової функцією, що має дві сторони: проблема надмірної і недостатньою закупівлі. Якщо товару закуплено занадто мало, то це породжує збиток в компанії. А так як в цій роботі я створюю додаток для медицини, проблему нестачі товару потрібно вирішувати повністю. У свою чергу, проблема надмірної закупівлі не така суттєва, тому що зазвичай у ліків термін придатності близько двох років, а за цей час, при правильній роботі зі складом, вони будуть вже використані.

Конкретною метою в цій роботі буде повністю виключити проблему нестачі в закупівлі та максимально мінімізувати надмірну закупівлю. Для цього розглянемо вже існуючі рішення і їх можливості. В даному пункті розділу я не буду акцентувати увагу на конкретні способи реалізації прогнозу, так як це буде описано в наступному розділі.

Так як темою роботи є Android-додаток, буду розглядати додатки для смартфонів.

Перший додаток, яке я розгляну, це «Trini POS» (рис 1.6). На мій погляд, програма має дуже зручний інтерфейс, мінімалістичний дизайн і широкий функціонал. Орієнтована програма під дрібний і середній бізнес.

Функціонал програми дозволяє:

- Контролювати залишки товару на складі;
- Отримувати звіт про продажі і детальну статистику;
- Отримувати звіт про популярність конкретних товарів;
- Мати зручний доступ до БД. [14]

Варто відзначити, що функціонал залишків на складі хоч і є, але має урізаний функціонал. Наголос робиться на обліку продажів і оптимізації продажів, доступом до статистики популярності конкретного товару.

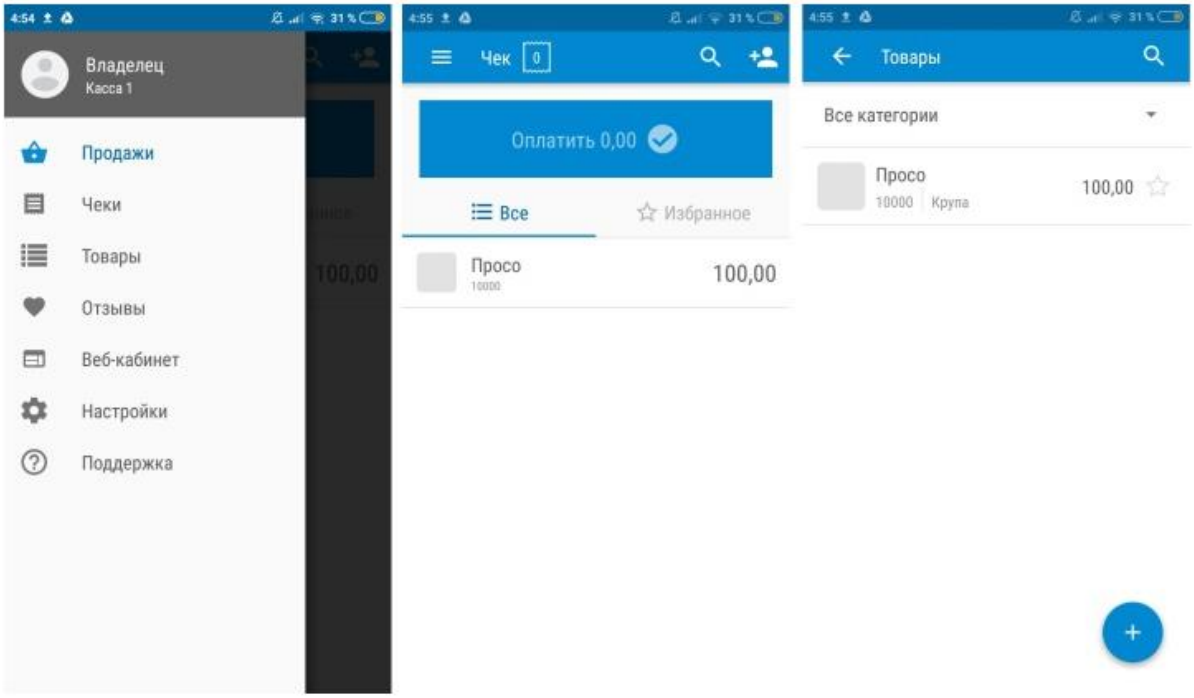


Рисунок 1.6 Інтерфейс додатку «Trini POS»

Наступною для розгляду буде програма «Облік магазину складу». Підходить вона в першу чергу для невеликих магазинів. Функціонал мінімальний, але достатній для ведення обліку бізнесу.

Програма дає можливість:

- Стежити за продажами і закупками;
- Друкувати накладні;
- Зберігати в БД не тільки назва і кількість товару, а й штрих код;
- Отримувати звіт про продажі. [14]

Я думаю, що величезним мінусом є те, що доступна програма тільки на iOS. Також, програма має досить старомодний дизайн (рис. 1.7).

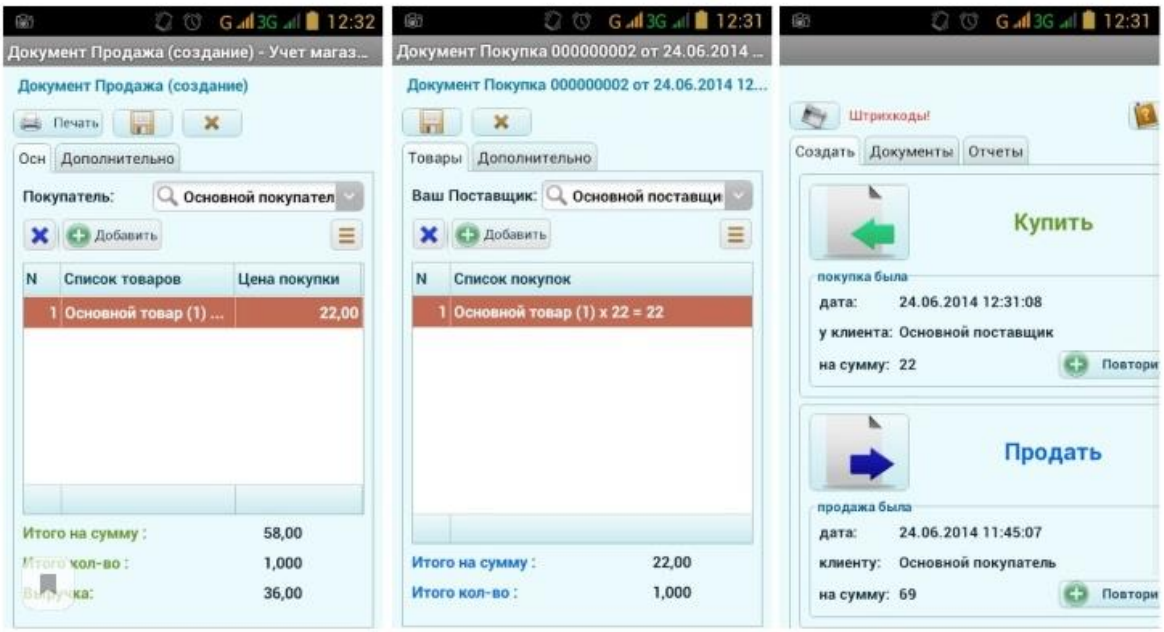


Рисунок 1.7 Інтерфейс додатку «Облік магазину складу»

Останнім розгляну додаток «Каса МойСклад». На перший погляд програма має зручний інтерфейс і хороший дизайн (рис. 1.8):

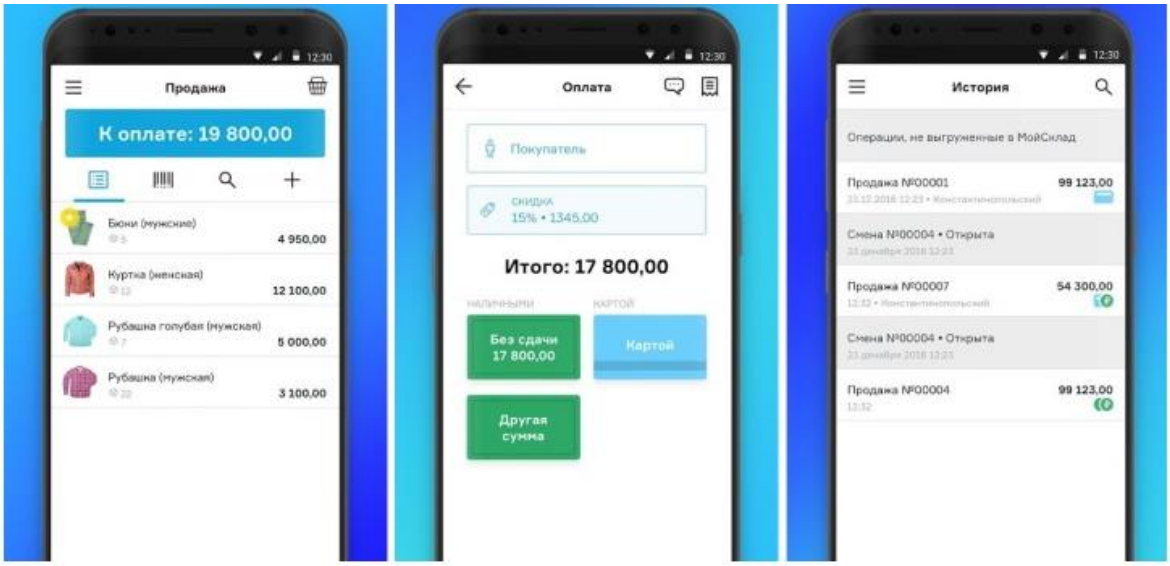


Рисунок 1.8 Інтерфейс додатку «Каса МійСклад»

Але при детальному розгляді стає зрозуміло, що безкоштовна версія програми дуже обмежена. Вона дозволяє лише реєструвати продажу,

повернення, враховувати графік роботи і формувати на його підставі зміни працівників.

Як заявляють розробники, при покупці повної версії програми стане доступним:

- Завантаження в додаток даних з уже існуючою БД;
- Меню закупок;
- Фінансові звіти. [14]

Я вважаю, що такий урізаний функціонал має мінус в тому плані, що для того щоб користувач купив повну версію програми, він повинен протестувати його роботу і повноцінно спробувати. Тому найкращим рішенням було б надати користувачу пробну версію з повним функціоналом.

Додаток доступний на Android і iOS.

Хочу відзначити, що додатки, вибрані для аналізу в цій роботі, були обрані абсолютно випадковим чином. На ринку існує безліч таких додатків і тому таке рішення, як на мене, є оптимальним в цьому випадку.

Загалом, підсумком можна вважати те, що практично всі програми реалізують функціонал, який більше підходить для магазинів і ресторанів. Функція обліку складу є, але вона не основна в цих додатках.

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі було розглянуто предметну область та впровадження інформаційних технологій в медицину. Було визначено та класифіковано медичні інформаційні системи. Було розглянуто такі системи як продукт, проведено огляд стадій розробки і впровадження, проблем, що виникають в процесі впровадження та шляхи вирішення цих проблем. Зроблено огляд затребуваного функціоналу систем у сучасній медицині. Було проведено огляд існуючих рішень в українській медицині та існуючих мобільних додатків оптимізації продажу та обліку складу.

Отже, висновком до цього розділу буде те, що в сучасній медицині вже існує багато медичних інформаційних систем. В цих системах реалізовано повний функціонал, необхідний лікарням. Але недоліком є те, що не всім лікарням доступні такі рішення через високу вартість. А існуючі мобільні додатки, серед яких є і безкоштовні, орієнтовані більше на магазини та ресторани.

Тому програма, яка розроблюється в даній роботі, зможе об’єднати ці дві сторони та бути затребуваною невеликими приватними лікарнями, або деякими державними.

РОЗДІЛ 2. ОГЛЯД ПРОГРАМНИХ ТЕХНОЛОГІЙ. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1. Теоретичні відомості

Для початку роботи над програмою необхідно спочатку оглянути технології, які будуть використані при проектуванні. В цьому пункті розділу я розгляну мову програмування, операційну систему Android та патерни програмування, які будуть використані.

2.1.1 Мова програмування Java

Java - це універсальна мова програмування, яка є паралельною, класовою і об'єктно-орієнтованою. Ця мова була спеціально розроблена, щоб мати якомога менше залежностей реалізації. Код, скомпільований на Java, може працювати на всіх платформах, що підтримують цю мову, без необхідності в перекомпіляції. [5]

Наприклад, можливо написати програму на UNIX і запустити її на комп'ютері з ОС Windows без будь-яких змін в коді. Це досягається завдяки тому, що мова Java компілюється в проміжну мову, яка називається байт-кодом. Формат такого коду є незалежним від платформи. Для виконання байт-коду на кожній платформі використовується JVM (Java Virtual Machine). [5]

Нижче показана схема роботи мови Java (рис. 2.1):

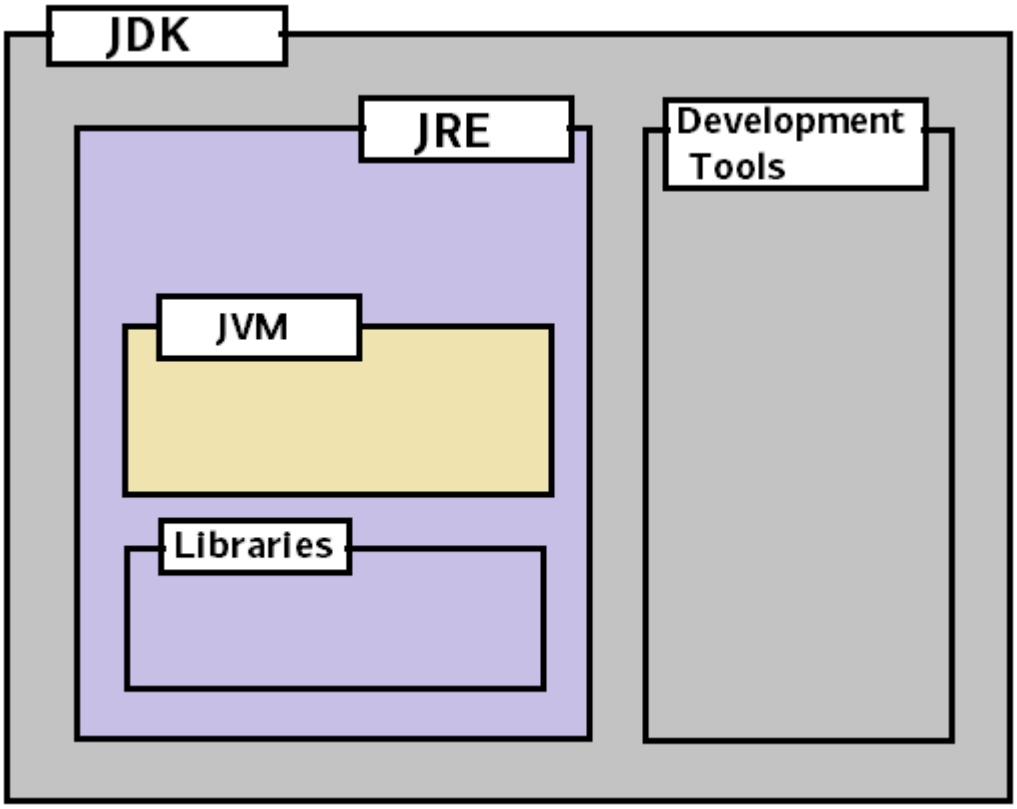


Рисунок 2.1 Схема роботи мови Java

Розшифруємо все що знаходиться на цій схемі:

JDK (Java Development Kit) - це комплект розробника програм на мові Java. [6]

JRE (Java Runtime Environment) - це реалізація віртуальної машини, необхідної для виконання програм на Java. [6]

Libraries - це все бібліотеки Java-класів.

Development Tools - це все інструменти розробки, необхідні для роботи цієї мови.

На перший погляд архітектура Java виглядає складною, але якщо розібратися з кожним пунктом, то стає зрозуміло для чого це необхідно. Тепер трохи про історію цієї мови.

Розробив мову канадський програміст Джеймс Гослінг. Випущена була мова Java в 1995 році, компанією Oracle, яка придбала цю мову. Мова є спадкоємцем мов C і C ++, але має меншу кількість об'єктів низького рівня. [7]

Мова має власний збирач сміття, який використовується для раціонального використання пам'яті. Програміст вирішує, коли об'єкти будуть створені, а вже середовище Java відповідає за їх видалення, тим самим звільняючи пам'ять, коли вони більше не використовуються. Так відбувається якщо посилання на об'єкт відсутнє.

Таке прибирання сміття може бути включено автоматично в будь-який час. У пріоритеті це відбувається, коли програма простоює. Також прибирання буде запущене автоматично якщо програмі буде недостатньо пам'яті для подальшого виконання.

2.1.2 Операційна система Android

Android - це операційна система, яка працює з більшістю мобільних пристроїв. В першу чергу це смартфони та планшети, але список пристроїв можна дуже довго продовжувати: смарт-годинники, фітнес-браслети, електронні книги і багато іншого. Система була розроблена в 2008 році компанією Android, яку в подальшому купила компанія Google. Система заснована на власній віртуальній машині Java. [8]

З моменту свого випуску ця система домоглася стрімкого успіху. Завдяки її створенню практично все людство перейшло з звичайних кнопкових телефонів на смартфони з сенсорним екраном. Цікаво те, що зараз існує дуже багато інших систем, але Android залишається лідером на ринку. На даний момент відсоток Android-пристроїв серед всіх інших становить приблизно 85%. Разом з системою iOS ці системи займають 99% всього ринку. [15]

Для того щоб почати створювати додаток, необхідно визначитися з версією операційної системи, для якої буде вестися розробка. Android влаштований так, що система, для якої створювалося додаток, підтримує цю програму, а так само всі системи, випущені після. Логічним здається в цьому

випадку розробляти програму для самої ранньої версії, щоб вона працювала на всіх пристроях. Але це не зовсім так. Ранні версії мають більш обмежений функціонал в порівнянні зі старшими версіями. Тому при розробці потрібно знайти баланс між кількістю підтримуваних пристроїв і доступним функціоналом кожної версії.

У даній роботі в пріоритеті будуть більш нові версії, але для вибору конкретної звернемося до статистики, наведеної нижче (рис. 2.2).

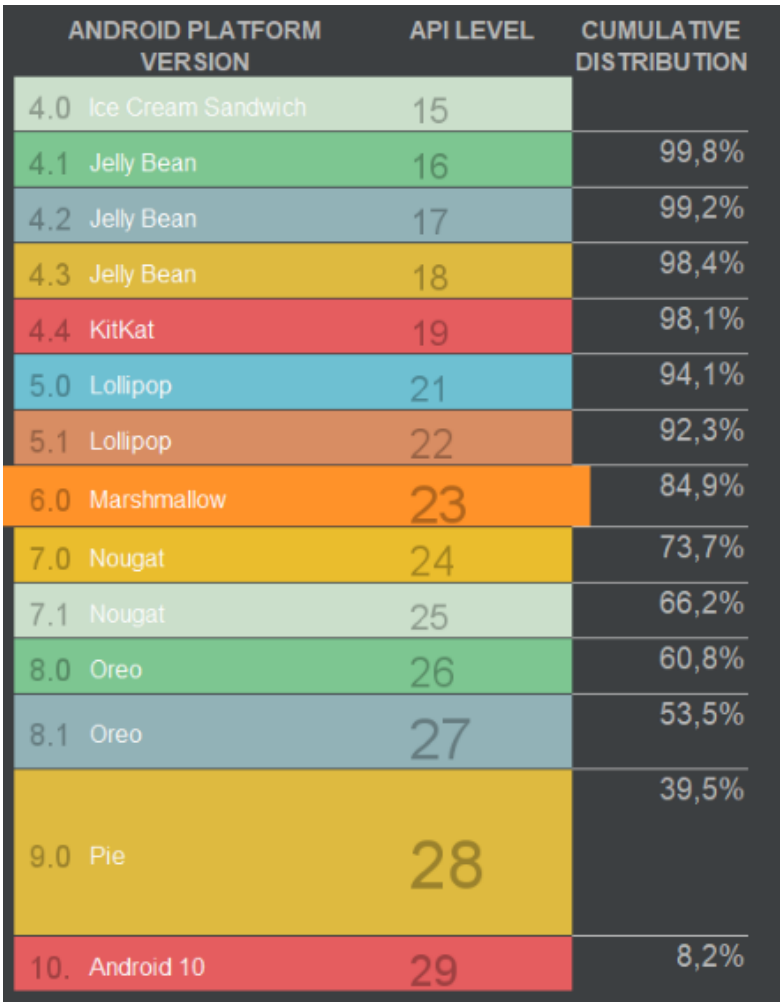


Рисунок 2.2 Статистика різних версій Android

На рисунку 2.2 зображені всі версії ОС, які використовуються на пристроях в даний момент і відсоток пристроїв, на яких додаток буде працювати при виборі цієї версії. Для того щоб повністю визначитися, подивимося на ще один рисунок 2.3.

Рисунок 2.3 Оновлення версій Android

Як бачимо на рисунку 2.3, системи нижче 8.0 вже є застарілими і втратили підтримку від Google. Тобто на них вже не будуть приходити оновлення.

Це означає, що в цій роботі я зупинюся на версії 8.0 (Oreo). Ця система актуальна в даний момент, а також додаток, розроблений під цей пристрій, буде працювати майже на 61% пристроїв. Я вважаю, що це хороший показник.

2.1.3 Шаблони проектування

При проектуванні і розробці додатку в цій роботі буде використано концепцію MVC (Model-View-Controller).

MVC («Модель-Представлення-Контролер») - це структура поділу даних додатку, що розробляється, інтерфейсу користувача і керуючої логіки на три незалежні компоненти: модель, представлення і контролер. Ці три компоненти реалізовані так, що модифікація кожного компонента може здійснюватися незалежно. [9]

Саму ідею було сформульовано в далеких 70-х роках Трюгве Реенскаугом. Завдання полягало в спрощення взаємодії користувача з комп'ютером. Необхідно було розробити модель, яка б дозволяла з одного боку реалізувати зручний інтерфейс для роботи користувача, а з іншого боку виконувати складні обчислення. Робота над проектом почалася з розробки комп'ютера для дітей, тоді вперше з'явилася потреба в реалізації

цієї моделі. Після цього робота тривала понад 10 років, а перша публікація про MVC з'явилася друком ще через 10 років. [9]

Нижче показана схема взаємодій в MVC (рис. 2.4):

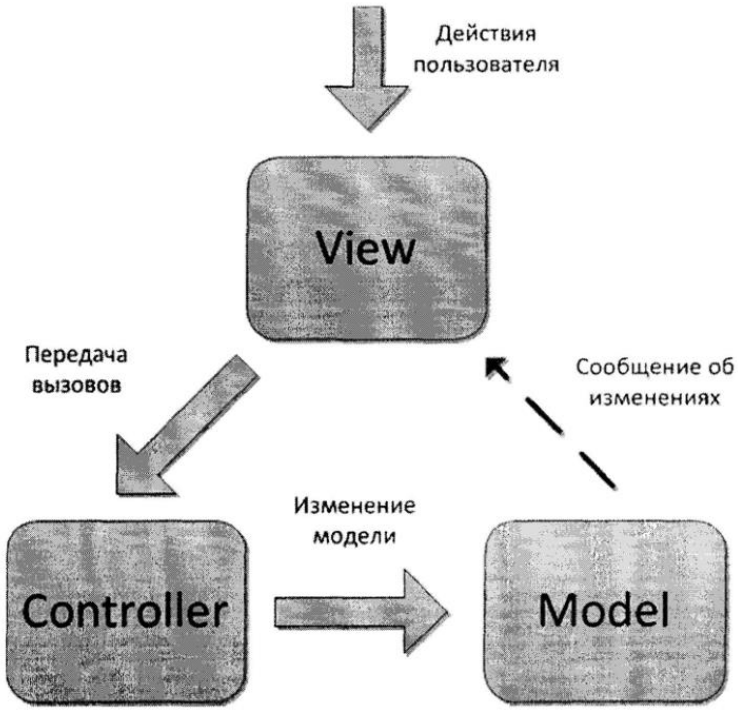


Рисунок 2.4 Схема концепції MVC

Model - це так звана модель, яка містить всю логіку програми. Це найбільш незалежна частина системи, реагує на команди контролера змінюючи свій стан. [16]

View - модуль системи що позначає вид. Відповідає за відображення всіх даних, які бачить користувач. Основне його завдання - надання інформації з моделі в зручному вигляді. Обмеження - то що він ніяк не повинен змінювати модель. [16]

Controller - частина системи, в якому зберігається код, що відповідає за обробку дій користувача. Саме через контролер користувач вносить зміни в дані, що зберігаються в моделі. [16]

Основною метою цієї концепції є відділення бізнес-логіки додатка від її візуалізації. Притримуючись принципам MVC, я спрощу написання програми і підвищу читабельність коду.

2.2. Огляд середовища для розробки

Для розробки програми було використано середовище розробки Android Studio. Це середовище є сучасним рішенням і має весь необхідний функціонал для реалізації даної роботи.

Android Studio - це інтегроване середовище розробки додатків на всі можливі пристрої, що підтримують ОС Android, написана на мовах Java, C ++ і Kotlin, яке було анонсовано в 2013 році. Розробником цієї IDE є Google. [10]

Основною перевагою Android Studio є те, що вона у вільному доступі і встановити її можна з офіційного сайту.

Основним конкурентом цього середовища розробки є Eclipse. Але мною був обраний саме перший варіант, тому що на даний момент Android Studio має великі можливості для розробки і більш зручний інтерфейс.

Переваги та можливості Android Studio:

1. Візуальний редактор макетів. Він дозволяє дуже зручно проектувати екрани додатка (Activity). Користувачеві потрібно лише вибрати необхідний елемент (View) і перенести його на екран, розташувавши в потрібному місці. Надалі цей макет можна переглядати для будь-якого розширення екрану.
2. Швидкий емулятор. Для того щоб кожен раз при тестуванні працездатності коду при розробці не встановлювати додаток на телефон, середовище розробки дає можливість запускати додаток прямо в ній, використовуючи вбудований емулятор. Це дуже зручно, але варто відзначити що для швидкої роботи емулятора потрібно мати достатню кількість оперативної пам'яті.
3. Аналізатор APK. Дає можливість порівнювати розмір програми і всіх його ресурсів.

4. Інтелектуальний редактор коду. Протягом всього процесу розробки спливають підказки з продовженням коду. Це дозволяє швидко вводити код, не відволікаючись на такі дрібниці.
5. Система збирання Gradle. Є гнучкою системою і дозволяє налаштувати декілька варіантів збірки для проекту.
6. Доступ до інформації про завантаженість системи. Дозволяє відстежувати в реальному часі завантаження ЦП, пам'яті або диска. [10]

2.3. Налаштування середовища розробки

Для початку розробки необхідно спочатку встановити мову Java на компьютер, потім завантажити Android Studio та налаштувати проект. Детальніше про це в цьому підрозділі.

Для початку необхідно перейти на офіційний сайт Java та завантажити необхідну версію. Скріншот з сайту показаний на рис. 2.5.

Рисунок 2.5 Скріншот офіційного сайту Java

Після завантаження установника клікаємо на нього, слідуємо інструкції та перезавантажуймо комп’ютер.

Перевірити правильність установки можна запустивши командну строку, та ввівши команду «java -version», як показано на рис. 2.6. Якщо все правильно встановлено, то буде написано версію встановленої Java.

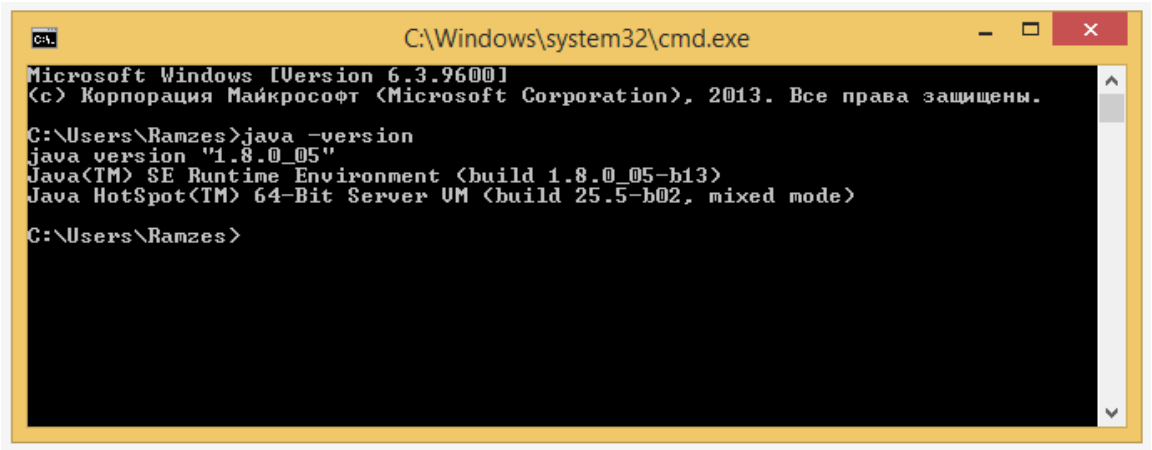


Рисунок 2.6 Скріншот командної строки з версією Java

Після цього необхідно перейти на офіційний сайт Android Studio, та завантажити останню версію середовища. На рис 2.7 можна побачити необхідну кнопку.

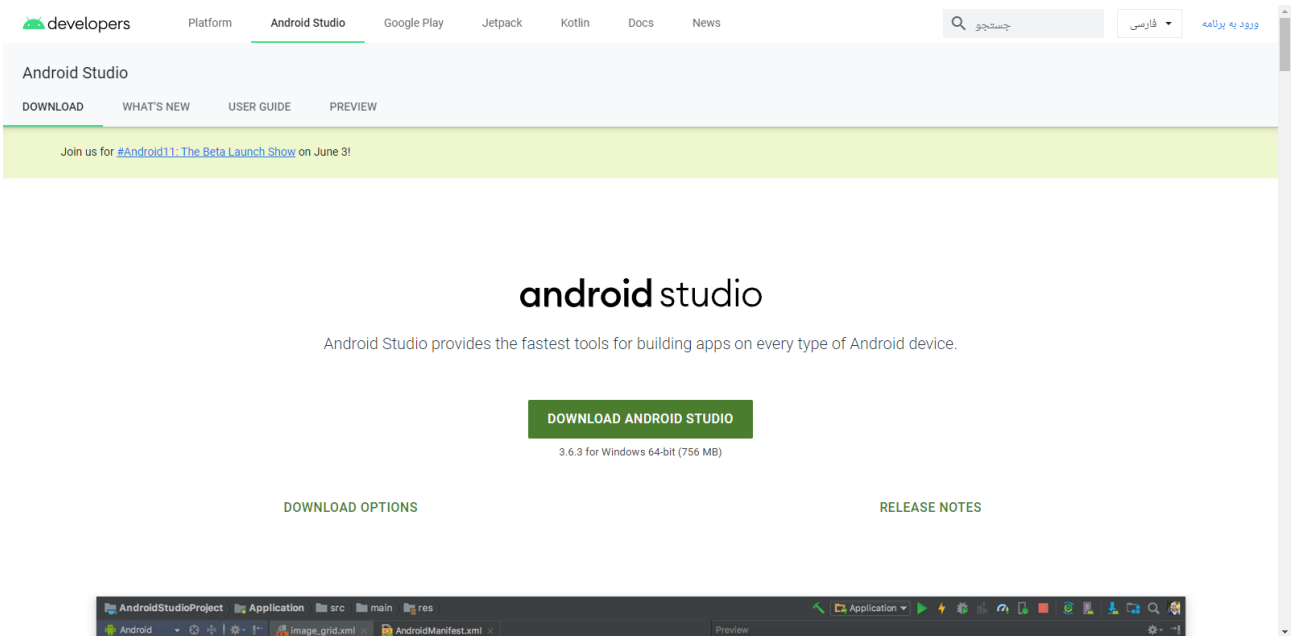


Рисунок 2.7 Скріншот офіційного сайту Android Studio

Тут необхідно зробити теж саме, слідувати інструкціям та встановити середу розробки. На робочому столі з'явиться іконка Android Studio, після кліку на яку відкриється вікно створення нового проекту.

Створимо проект, в якому будемо працювати над додатком. Для цього натискаємо «New Project» та у вікні, показаному на рис. 2.8, обираємо «Empty Activity» та натискаємо «Next». В цьому вікні можна обрати інші варіанти, якщо вони необхідні розробнику. Але мені зручніше робити все самому з самого початку, тому обираю саме це.

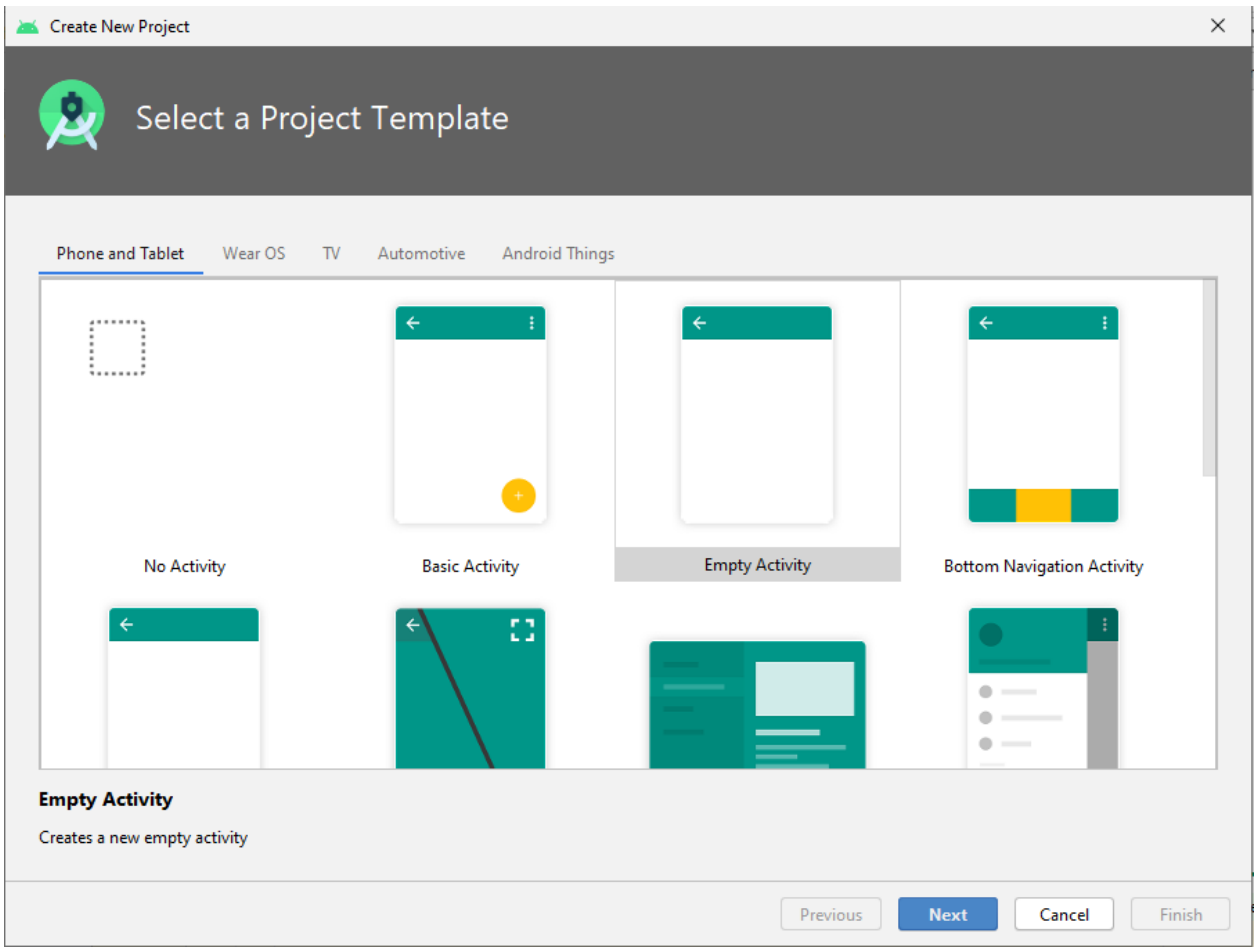


Рисунок 2.8 Вікно вибору Activity

Далі, у вікні, зображеному на рис 2.9, необхідно вказати ім’я проекту, ім’я пакету, папку для проекту та мову. Нижче обираємо версію Android та натискаємо «Finish».

Після цього деякий час відбудеться зборка проекту, коли вона завершиться, середа програмування та проект будуть готові до розробки. Якщо все зроблено правильно, то вікно проекту має виглядати так, як на рис. 2.10.

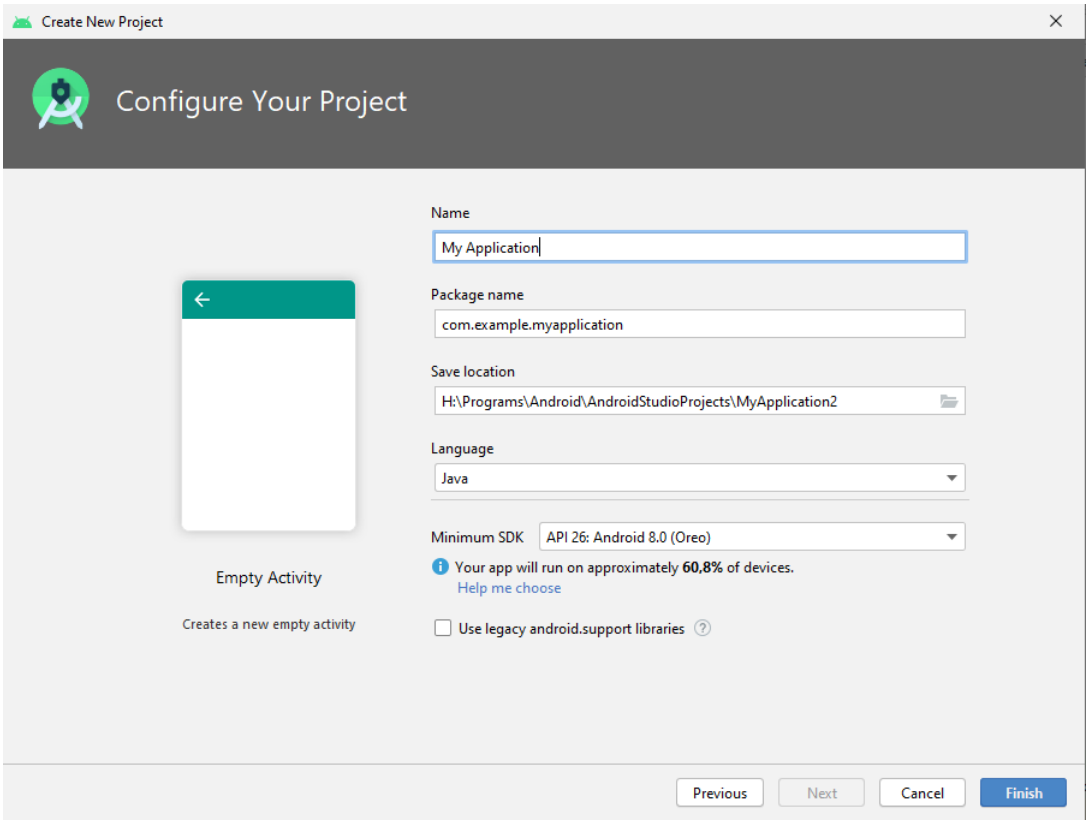


Рисунок 2.9 Вікно параметрів проекту

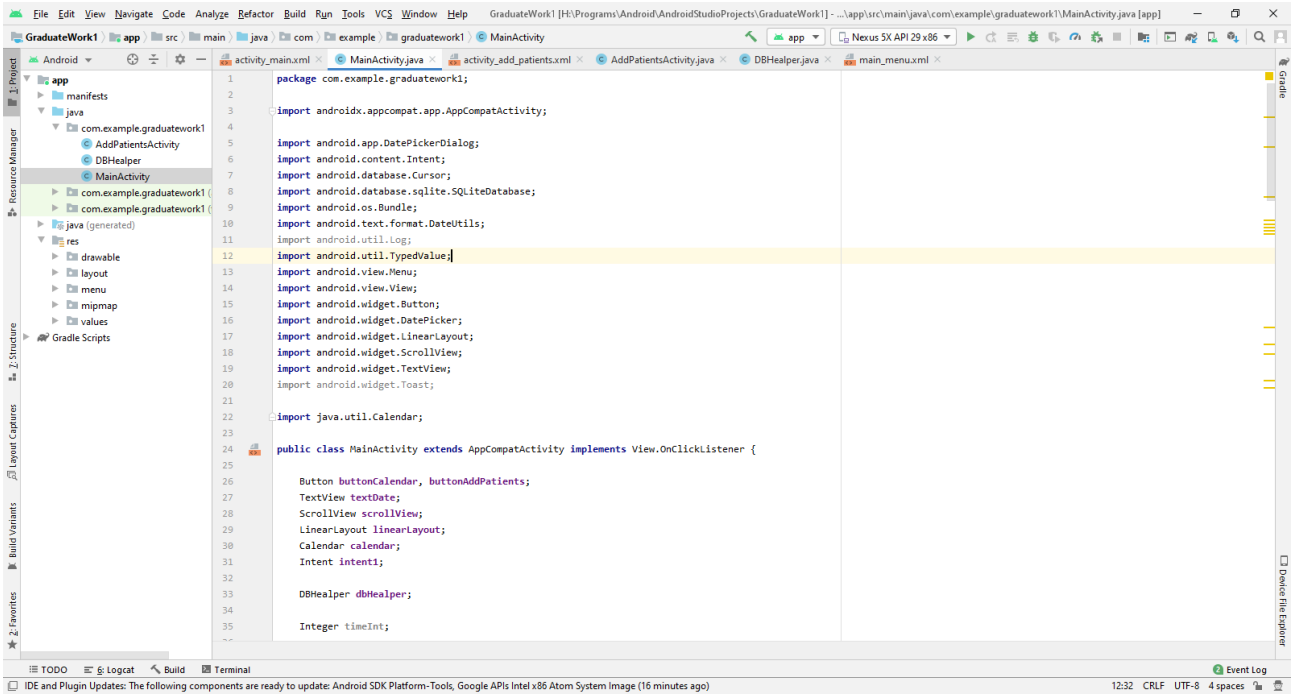


Рисунок 2.10 Головне вікно проекту

Отже, налаштування проекту можна вважати завершеним. Можна починати розробку архітектури програми та написання коду.

2.4 Постановка задачі

Проаналізувавши МІС як продукт, стає зрозуміло, що для реалізації повноцінної системи, яка буде використана в великих приватних клініках, буде потрібно дуже багато часу, сил і ресурсу. Тому в цій роботі я буду намагатися дотримуватися сучасних тенденцій та реалізовувати той функціонал, який буде необхідний сучасній медицині.

Але зрозуміло, що одному буде неможливо розробити таку систему повністю, тому буду робити упор на мінімалізмі і правильній роботі тих функцій, які були задумані. Перейду до конкретики.

В першу чергу програма буде виконувати зручний інтерфейс для введення інформації про пацієнтів і його виведення, а також пошук по БД. Лікар (він же користувач додатка) в головному вікні може вибирати час і дату. За замовчуванням встановлюється час на пристрої в момент використання, але, якщо лікарю потрібно переглянути дані про пацієнтів з минулого, він вибере відповідну дату. Якщо ж користувач хоче додати нового пацієнта в базу, він переходить на екран введення даних. Після кнопки підтвердження дані будуть внесені в БД і виведені на головному екрані.

Користувач може відкрити екран пошуку по базі і знайти необхідну інформацію, ввівши ПІБ або номер телефону пацієнта.

Іншою функцією цієї програми є ведення обліку складу витратних матеріалів із занесенням даних в БД. Користувач вводить необхідні пункти і їх кількість, система вже підвантажує системний час і записує в БД час, кількість і найменування елемента.

Мною передбачається що досить буде раз на тиждень оновлювати інформацію про кількість робочого матеріалу і ліків. Тобто раз на тиждень лікар робить перерахунок залишків ліків і всього необхідного та через інтерфейс користувача вносить цю інформацію в базу даних.

Коли користувачеві буде необхідно отримати від програми прогноз закупівлі, він переходить в відповідне вікно програми і вказує там кількість днів, на яке йому необхідно закупити товар. Система на підставі витрат попередніх тижнів робить прогноз і видає інформацію про те, яку кількість кожного елементу потрібно закупити.

Це той мінімум необхідного функціоналу, який я реалізую в даній роботі. Цього функціоналу вже буде досить для виконання головної ідеї: збереження даних і оптимізації закупівлі.

Над дизайном програми багато роботи виконуватися не буде. Буду дотримуватися принципів мінімалізму, які актуальні в сучасному світі. Буде розроблений дизайн фону і кнопок, для того щоб додаток виглядав гармонійно і вписувався в сферу медицини.

Надалі, можливо, робота над цією програмою буде продовжена для додавання функціоналу, який дозволить боротися з конкурентами на ринку і впроваджувати цю систему в деякі клініки. Але зараз буде реалізований функціонал, який дозволить протестувати цей додаток в реальних умовах і він буде працювати без помилок.

Для розуміння ринку і аналогів необхідно провести роботу над аналізом вже існуючих рішень. У наступному пункті цього розділу зробимо цю роботу.

2.5 Огляд методів прогнозування

В даній роботі для оптимізації процесу закупки ліків буде відбуватися прогнозування витрат на заданий користувачем період. Для правильної реалізації прогнозування необхідно зробити огляд існуючих методів прогнозування та обрати найбільш правильний для даної роботи.

Зазвичай прогнозуванням називають процес передбачення майбутнього, засноване на наявних даних з минулого. У нашому випадку

даними з минулого будуть кількість залишку кожного з ліків на складі для конкретних дат з минулого, а прогнозом буде кількість ліків, необхідне в майбутньому.

Розглянемо наступні методи прогнозування:

- Простої екстраполяції;
- Рухомих середніх;
- Експоненціального згладжування;
- Середнього абсолютного приросту;
- Середнього темпу зростання. [17]

Метод простої екстраполяції - це метод, побудований на підрахунку середнього значення ряду. Зазвичай він використовується якщо в майбутньому не планується тенденцій до зміни або ця зміна незначна. [17] Метод описується формулою:

$$y_{\text{прогн}} = \frac{\Sigma y_i}{n}$$

В якій $y_{\text{прогн}}$ – це прогнозована величина, y_i – значення і-го елемента з минулого, а n – кількість елементів з минулого.

Метод рухомих середніх схожий на попередній, проте в ньому використовуються в повному обсязі елементи минулого, а деяка кількість, яка визначається виходячи з завдання. Як і попередній, в основному він розрахований на процеси, що відбуваються без зміни значень або при мінімумі таких змін. За рахунок того, що при підрахунку не використовуються відразу всі дані минулого, метод є точніше попереднього. [17] Описується формулою:

$$\bar{y}_{n-H} = \frac{y_{n-m} + y_{n-m-H} + \dots y_n}{m}$$

Для простоти розглянемо на прикладі:

Таблиця 2.1 Метод рухомого середнього на практиці

T	y_i	Рахунок	Прогноз
1	30		
2	36		
3	29		
4	31	(30+36+29)/3	31,66
5	35	(36+29+31)/3	32
6	27	(29+31+35)/3	31,66
7	28	(31+35+27)/3	31
8	32	(35+27+28)/3	30
9	34	(27+28+32)/3	29
10	прогноз	(28+32+34)/3	31,33

Як можна побачити з таблиці, завдяки рухомому середньому прогноз виходить більш точним, бо використовуються лице останні дані. На перший погляд цей метод підходить для вирішення задачі, тому потрібно дослідити його більш детально. Побудуємо графік (рис 2.11).

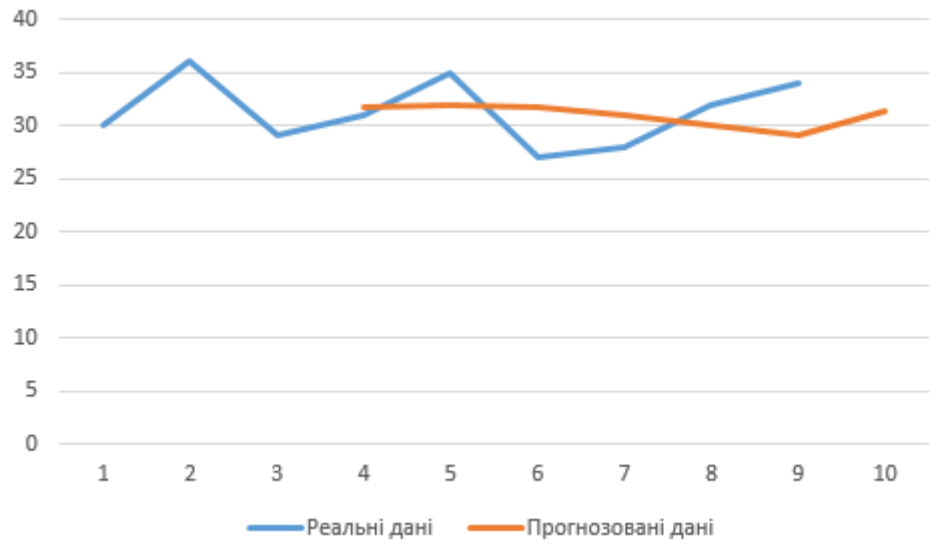


Рисунок 2.11 Порівняння реальних та прогнозованих даних методом рухомого середнього

Метод експоненціального згладжування - це схожий на попередній метод, але в ньому для попередніх рівнів задається певне вагове значення. Так як спочатку зрозуміло, що в нашому додатку не використовуватиметься вага, можна не розглядати цей метод детально, з відразу перейти до наступного. [17]

Наступним методом є метод середнього абсолютного приросту. Прогнозований рівень змінюється в залежності від середнього абсолютного приросту цієї величини в минулому. Розглянемо формулу:

$$y_{\text{прогн}} = y_0 + \bar{\Delta} * t$$

В якій y_0 – базовий рівень екстраполяції, вибирається як середнє значення деяких останніх значень ряду; $\bar{\Delta}$ - середній абсолютний приріст рівнів ряду; t – число інтервалів прогнозування.

Як становиться зрозуміло, цей метод підходить для прогнозування, коли присутня тенденція до зростання. В нашому випадку цей метод не підходить.

Останнім методом є метод середнього темпу зросту. Використовується в випадках, коли загальна тенденція характеризується експотенційною

кривою. Отже, цей метод також підходить для випадків, в яких можлива тенденція зростання, тому також не підходить. [17]

Проаналізувавши всі методи прогнозування було обрано метод рухомого середнього, адже він є найбільш точним серед тих, що підходять для вирішення завдання даної роботи.

2.6 Складання та опис алгоритму

Для початку написання коду програми необхідно скласти алгоритм, за яким буде працювати ці програма. Для цього в цьому підрозділі буде створено декілька схем роботи алгоритму, а також макет візуального інтерфейсу майбутньої програми.

Тому на рис. 2.12 зобразимо блок-схему алгоритму взаємодії всередині програми ти зв’язок з БД.

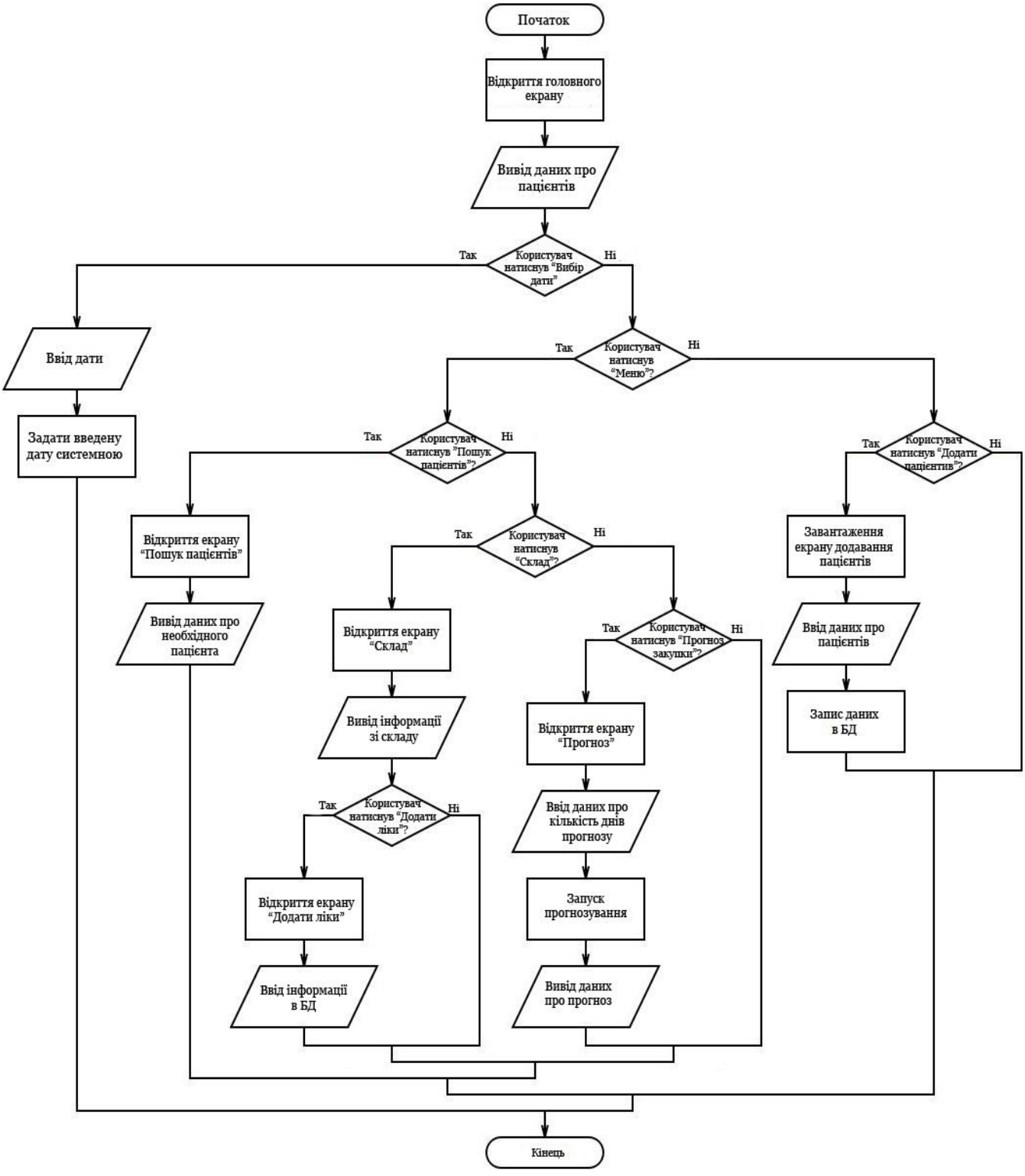


Рисунок 2.12 Блок-схема роботи програми

При запуску програми відразу встановлюється реальний системний час. Для цього часу виводяться всі пацієнти, які завантажуються з БД. Користувач може змінити дату, натиснувши «Вибір дати». Тобто, лікар може змінювати дату та переглядати пацієнтів для конкретних дат.

Якщо лікар хоче знайти пацієнта по імені, чи номеру телефона, він повинен натиснути на кнопку «Меню» та «Пошук пацієнтів». Відкриється екран пошуку пацієнтів в БД. Тут користувач може ввести необхідні дані, та виконати по ним пошук. В цьому екрані лікар має доступ до кожного пацієнта, вписаного в БД.

Для додавання пацієнта в базу лікар має обрати необхідну дату та натиснути кнопку «Додати пацієнта». Відкриється екран вводу даних, в якому необхідно ввести ПІБ, номер телефону, вік та діагноз. Після натискання кнопки «Зберегти», система запише ці дані в базу.

Для управління складом існує екран «Склад», який викликається з меню. В ньому лікар може додати необхідні ліки та їх кількість, або оновити вже існуючи дані. При додаванні чи оновлені даних, інформація буде записана в БД з датою додавання чи зміни. А на екрані склад завжди будуть виводитися дані для останньої дати.

Коли лікарю необхідно буде провести закупівлю ліків, він з меню переходить в екран «Прогноз», вказує там на яку кількість необхідно зробити прогноз та натискає «Створити». Система проводить необхідні обчислення та виводить результати прогнозу на екран. Після цього дані в БД складу будуть оновлені з урахуванням закупки.

На рис. 2.13 зобразимо блок-схему алгоритму прогнозування закупки.

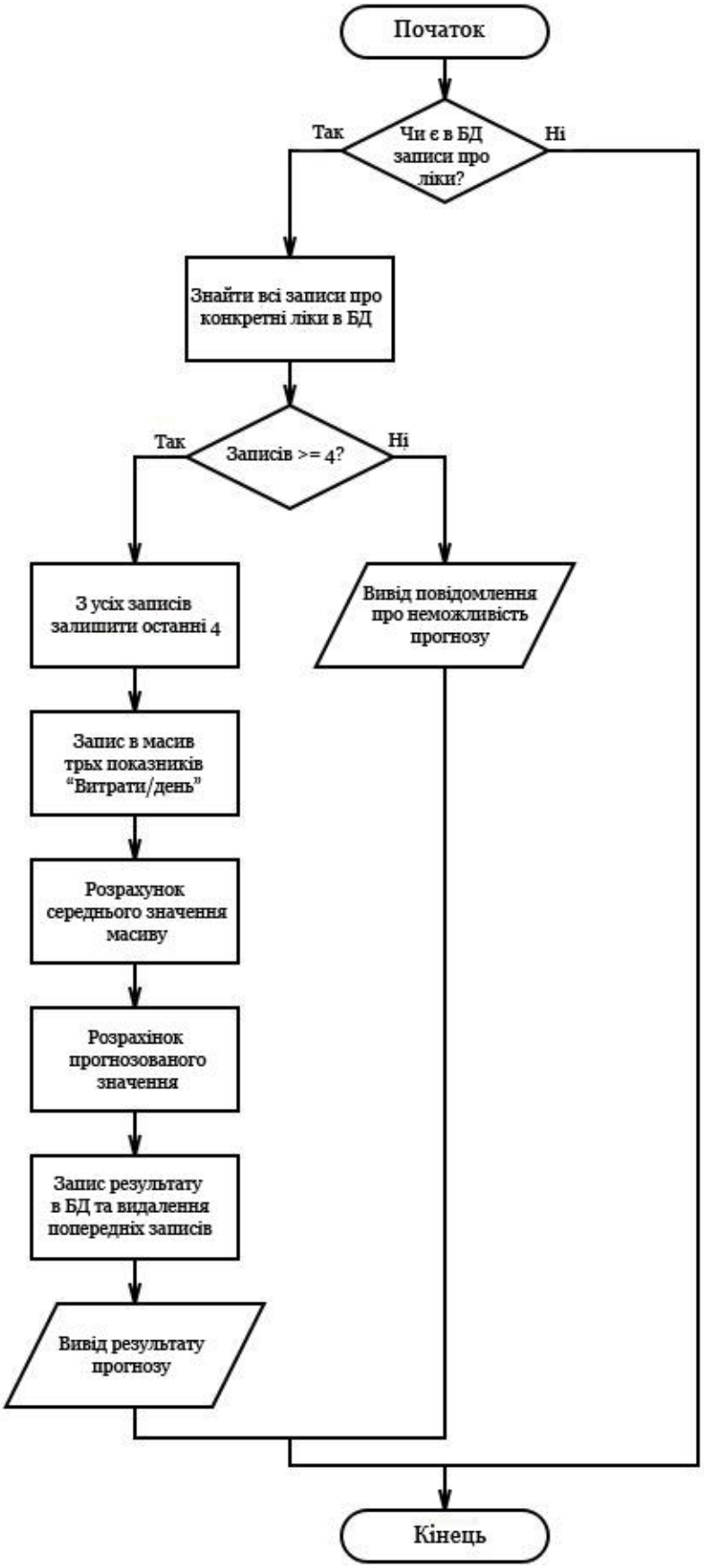


Рисунок 2.13 Блок-схема алгоритму прогнозу

2.7 Розробка моделі бази даних

Для роботи з БД в цій роботі буде використано SQLite. Це вбудована багатоплатформна база даних, яка підтримує майже повний набір команд SQL. Вибір цієї технології зумовлений тим, що робота з нею доступна не виходячи з Android Studio. Для того щоб створити базу даних, необхідно в проекті створити клас DBHealper, який буде наслідувати клас SQLiteOpenHealper.

В цьому класу необхідно імпортувати бібліотеки, які дадуть змогу працювати з SQLite:

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

В цьому класі можна створювати бази даних, таблиці для них та задавати необхідні параметри. В подальшому до цієї бази даних можна звертатися з коду додатку, зчитувати або записувати дані. В наступному розділі буде більш детально описана робота з БД, а поки що необхідно спроектувати таблиці.

Для коректної роботи та реалізації завдання буде використовуватися дві таблиці: таблиця пацієнтів та таблиця складу. Нижче наведено модель майбутніх таблиць (табл. 2.2 та 2.3).

Таблиця 2.2 Модель таблиці пацієнтів в БД

id	Прізвище	Ім’я	По-батькові	Вік	Телефон	Діагноз	Дата
INT	TEXT	TEXT	TEXT	INT	TEXT	TEXT	INT
…	…	…	…	…	…	…	…

Таблиця 2.3 Модель таблиці складу в БД

id	Назва	Кількість	Дата
INT	TEXT	REAL	INT
…	…	…	…

ВИСНОВКИ ДО РОЗДІЛУ 2

В цьому розділі було проведено огляд програмних технологій. Було опрацьовано теоретичний матеріал по мові програмування Java та операційній системі Android. Було досліджено концепцію MVC. Було розглянуто середовище для розробки Android Studio та поетапно проведено його налаштування та підготовку для початку роботи. Було зроблено огляд існуючих методів прогнозування та обрано найоптимальніший метод. Було розроблено алгоритм прогнозування, що буде використовуватися для оптимізації закупки ліків. Створено блок-схему цього алгоритму та загальну блок-схему роботи програми. Розроблено модель бази даних.

Отже, розглянувши програмні технології, можна зробити висновок того, що було обрано правильні технології для реалізації поставленої задачі. Результатом роботи над цим розділом можна вважати повністю спроектований додаток та база даних.

В наступному розділі буде описана розробка програми, на основі проекту та тестування додатку.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Структура проекту

В процесі розробки додатку було створено багато Java-класів та xml файлів, які забезпечують роботу програми. В цьому підрозділі я хочу описати загальну структуру проекту та призначення кожного елемента цієї структури. На рис. 3.1 зображено загальний вигляд проекту після розробки.

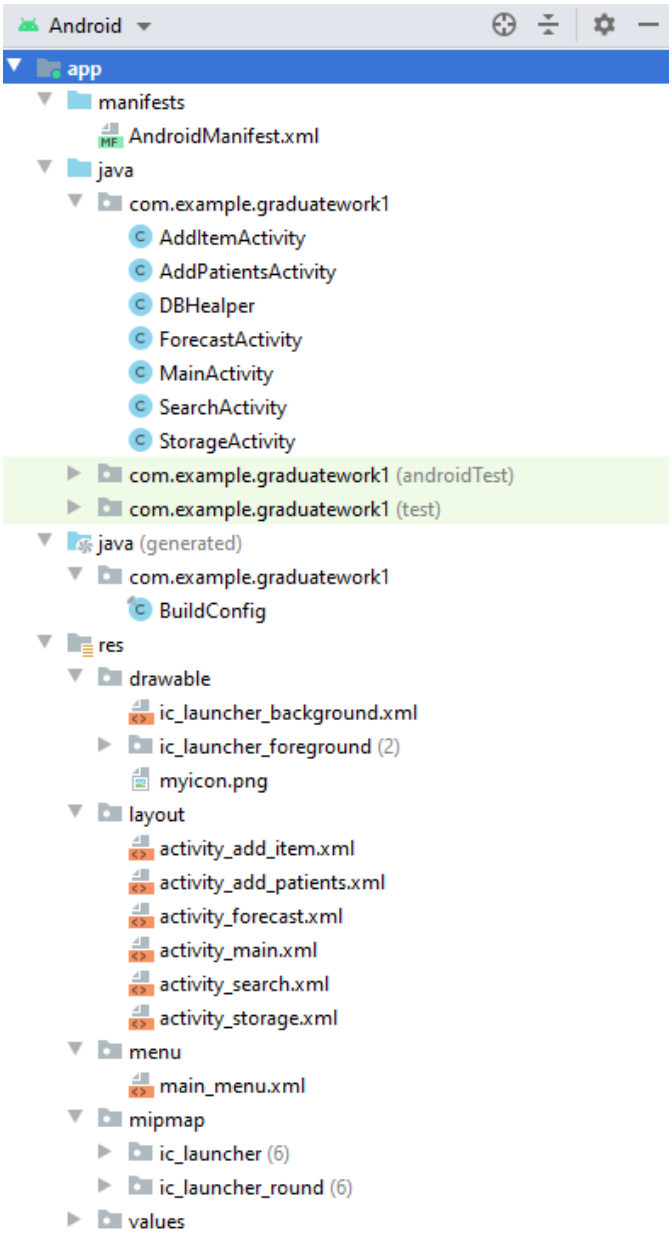


Рисунок 3.1 Загальна структура проекту

Першим є папка manifests, яка містить в собі файл AndroidManifest.xml. Цей файл є основним файлом проекту, та містить в собі конфігурацію всього додатку. Через цей файл можна задавати важливу інформацію: назву додатку, посилання на файл іконки, тему додатку, яка задає візуальний вигляд. Також, цей файл надає дозвіл застосунку на використання компонентів системи та визначає як будуть запускатися Activity, з якими фільтрами.

Наступною є папка java, яка містить вихідний код програми. Всі пакети та класи мають знаходитися саме в ній. При розробці додатку було створено сім класів, по які детальніше буде написано в наступному підрозділі.

Дуже важливою є папка res. Вона містить всі ресурси, які використовує програма: зображення, звукові і відео файли, анімації та xml-файли. В цій папці знаходяться такі основні пакети:

1. drawable – пакет, якій містить файли зображень. В даному проекті в ньому міститься лише зображення іконки додатку, тому що в програмі немає необхідності в зображеннях.
2. layout – пакет, в якому знаходяться xml-файли інтерфейсу користувача. При розробці доводилося дуже часто працювати з файлами макету кожного вікна, поміщаючи на них необхідні елементи користувацького інтерфейсу.
3. menu – містить лише xml-файл реалізації головного меню.
4. values – зберігає xml-файли для простих значень (строк, кольорів, стилів)

При розробці в основному робота відбувалася з класами Activity та layout-файлами. Детальніше про кожен клас в наступному підрозділі.

3.2 Огляд класів програми

Для забезпечення роботи додатку були розроблені класи, які виконують всі функції, необхідні для реалізації завдання. В цьому підрозділі я детально розгляну кожен клас.

MainActivity.java – основний клас програми, який реалізує одразу декілька функцій. По-перше, цей клас завантажується під час запуску програми, завантажуючи на екран інформацію з файлу activity_main.xml про елементи, які знаходяться на екрані спочатку. Відповідає він за дві основні функції програми: реалізація календаря і вивід даних про пацієнтів.

Календар реалізований таким чином, що при завантаженні програми автоматично системним часом задається реальний час. Надалі користувач може вибрати будь-яку дату. Ця дата стане системною і для неї будуть виводитися дані про пацієнтів.

Робота з календарем реалізована за допомогою бібліотеки java.util.Calendar. Вона дозволяє зберегти кількість мілісекунд з певної дати і тим самим, шляхом форматування цього числа, виводити дату на екран.

Перший виводить на екран реальний час системи, а другий відкриває діалогове вікно, в якому користувач може задати дату.

Вивід пацієнтів реалізований простим способом фільтрації пацієнтів з бази даних за датою. Система шукає всіх пацієнтів для конкретної дати з бази, створює TextView для даних про пацієнта і додає це TextView в елемент екрану ScrollView, який дозволить гортати список всіх пацієнтів.

AddPatientsActivity.java – відповідає за внесення в базу даних інформації по кожному пацієнту. Перехід на цей екран можливий з головного екрану. Цей клас відображає на екрані користувача елементи EditText, при заповненні яких користувач передасть дані програмі, яка в свою чергу звернеться до БД, прийме дані і передасть їх в БД. Все це відбувається після натискання кнопки «Додати».

SearchActivity.java – клас, який відповідає за пошук пацієнтів в БД і виведення їх на екран. Цей екран містить EditText, в який можна ввести дані, за якими і буде здійснюватися пошук. Цей клас звертається до БД, перебирає в ній всі записи, знаходячи тільки ті, які збігаються з введеним текстом. Кожен такий знайдений елемент додається в створене для нього TextView і виводиться тим самим на екран.

StorageActivity.java – відповідає за взаємодію користувача зі складом. При завантаженні цього класу виводяться на екран всі самі пізні записи в базі про кожен елемент. Фільтрація даних відбувається шляхом перевірки кожного рядка БД і виводом тієї, для якої дата найбільша. Так як дата є в БД числом типу int, що означає кількість секунд, то легко можна дізнатися найпізнішу дату шляхом порівняння цих чисел.

Так само, як і в попередніх класах, для кожної відфільтрованої записи з БД створюється TextView і заповнюється необхідною інформацією.

Для цього класу алгоритм фільтрації вже трохи складніше. Перебір елементів БД здійснюється двома курсором і порівнянням дат кожного з них.

AddItemActivity.java – клас, аналогічний класу додаванню пацієнтів, однак в цьому випадку менше елементів EditText, так як в базу необхідно внести лише назва елемента і його кількість. Дата буде завантажуватися автоматично.

ForecastActivity.java – напевно найскладніший клас програми, так як містить в собі не тільки висновок, а й обчислення прогнозу.

Алгоритм обчислення прогнозу має такий вигляд:

```
ArrayList<Float> quanForDay = new ArrayList<>();
for(int i = 0; i < 3; i++){
    float quanDiff = quantity.get((quantity.size() - 4 + i)) -
quantity.get((quantity.size() - 3 + i));
    float dateDiff = (float)dates.get(dates.size() - 3 + i) -
(float)dates.get(dates.size() - 4 + i);
    if (quanDiff >= 0 && dateDiff > 0){
        float res = quanDiff/(dateDiff/86400);
        quanForDay.add(res);
    }
```

```
}
Float average = (float) 0;
for (int i = 0; i < quanForDay.size(); i++){
    average += quanForDay.get(i);
}

Float forecastNumber = Float.parseFloat(editTextForecast.getText().toString());
Float forecastResult = forecastNumber*(average/quanForDay.size());
```

Для роботи цьому алгоритму необхідно два масиви: кількостей та дат за всі періоди. Такі масиви формуються другим курсором шляхом перебору всіх строк таблиці БД. Тобто на момент початку обрахунку вже створені масиви quantity та dates.

Далі з цих масивів беруться останні 4 записи та обраховується значення середньої витрати за день для кожного періоду. Такі значення вже записуються в новий масив.

Далі з цього масиву обраховується середнє значення та множиться на число днів, які ввів користувач. Таким чином для кожного элементу БД буде пораховано прогноз та виведено на екран.

Далі результат прогнозу буде додано до попереднього значення та записано в БД.

3.3 Огляд створеної бази даних

В цьому проекті робота з базою даних реалізована за допомогою бібліотеки android.databace.sqlite. Безпосередньо базою даних є клас *DBHealper.java*. Цей клас має два методи: onCreate() та onUpgrade().

OnCreate() буде викликатися при першому зверненні до БД, коли вона ще не буде створена. Тобто при першому доданому пацієнту, або елементу на склад. Цей метод створить дві таблички відповідно:

```
db.execSQL("create table " + TABLE_PATIENTS + "(" + KEY_ID + " integer primary
key," + KEY_SURNAME + " text," +
        KEY_NAME + " text," + KEY_PATRONYMIC + " text," + KEY_AGE + " integer," +
KEY_TELEPHONE + " text," +
        KEY_DIAGNOSIS + " text," + KEY_DATE + " integer" + ")");
```

```
db.execSQL("create table " + TABLE_STORAGE + "(" + KEY_ID + " integer primary key,"
+ KEY_DRUG_NAME + " text," +
        KEY_QUANTITY + " real," + KEY_DATE + " integet" + ")");
```

Метод onUpgrade() буде викликаний в тому випадку, якщо в коді бази даних буде змінено поле версії.

Цей метод просто видаляє вже існуючі таблички та викликає метод onCreate(), тим самим створюючи нові таблички.

Вибір SQLite зумовлений простотою в доступі до БД, без необхідності знання мови SQL. Для роботи достатньо знань принципів роботи.

Доступ до БД для читання та запису відбувається шляхом створення об'єкту класу DBHealper та визовом методу getWritableDatabace():

```
DBHealper dbHealper = new DBHealper(this);

SQLiteDatabase database = dbHealper.getWritableDatabase();
```

Після цього створюється курсор, який дозволяє переміщатися по строкам БД в будь-якому напрямі.

```
Cursor cursor = database.query(DBHealper.TABLE_PATIENTS, null, null, null, null,
null, null);
```

Далі завдяки цьому курсору можна отримати будь-які необхідні дані.

Запис в БД відбувається схожим чином. Але замість курсора створюється об'єкт класу ContentValues. Цей об'єкт заповнюється необхідними даними та додається в БД.

Таким чином, досить простими методами було реалізовано БД в цьому проекті. Цих можливостей достатньо для реалізації завдання на проект.

ВИСНОВКИ ДО РОЗДІЛУ 3

Для написання цього розділу було створено програмний код програми на основі розробленого алгоритму попереднього розділу. Безпосередньо в цьому розділі розглянуто структуру проекту та описано необхідність кожного пакету. Було описано кожен створений клас та його механізм роботи. Було створено базу даних та описано покрокове створення та її використання в інших класах.

Результатом роботи є створений програмний продукт, який виконує всі функції, які має виконувати відповідно до поставленого завдання.

РОЗДІЛ 4. ІНСТРУКЦІЯ РОБОТИ КОРИСТУВАЧА З ПРОГРАМОЮ

В цьому розділі буде проведена робота над тестуванням програми та створенню інструкцій користувача.

4.1 Реалізація збору даних пацієнтів

Для початку роботи з додатком необхідно завантажити та встановити його. Після цього натиснути відповідну іконку на робочому столі. Зовнішній вигляд іконки програми зображено на рис. 4.1.

Рисунок 4.1 Іконка програми

Після цього буде відкрито головне вікно програми, яке матиме вигляд, зображений на рис. 4.2.

Рисунок 4.2 Головне вікно програми

В цьому вікні в подальшому будуть виводитись всі пацієнти, прийняті лікарем в конкретний день. Для того, щоб змінити дату, для якої необхідно подивитися пацієнтів чи додати нових, треба натиснути «Вибір дати». На рис. 4.3 показано вікно, яке буде відкрито після натискання цієї кнопки.

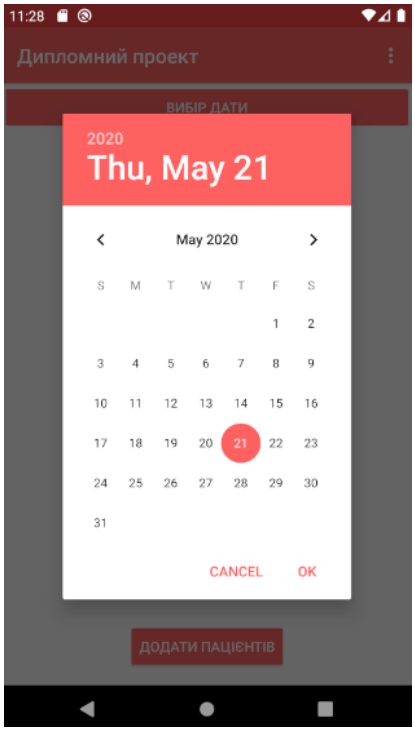


Рисунок 4.3 Вікно зміни дати

Тут треба обрати необхідну дату та натиснути «ОК».

Щоб додати нового пацієнта в вибраний день, треба натиснути кнопку «Додати пацієнтів». Відкриється нове вікно, в якому потрібно в кожне поле написати необхідні дані, як показано на рис. 4.4.

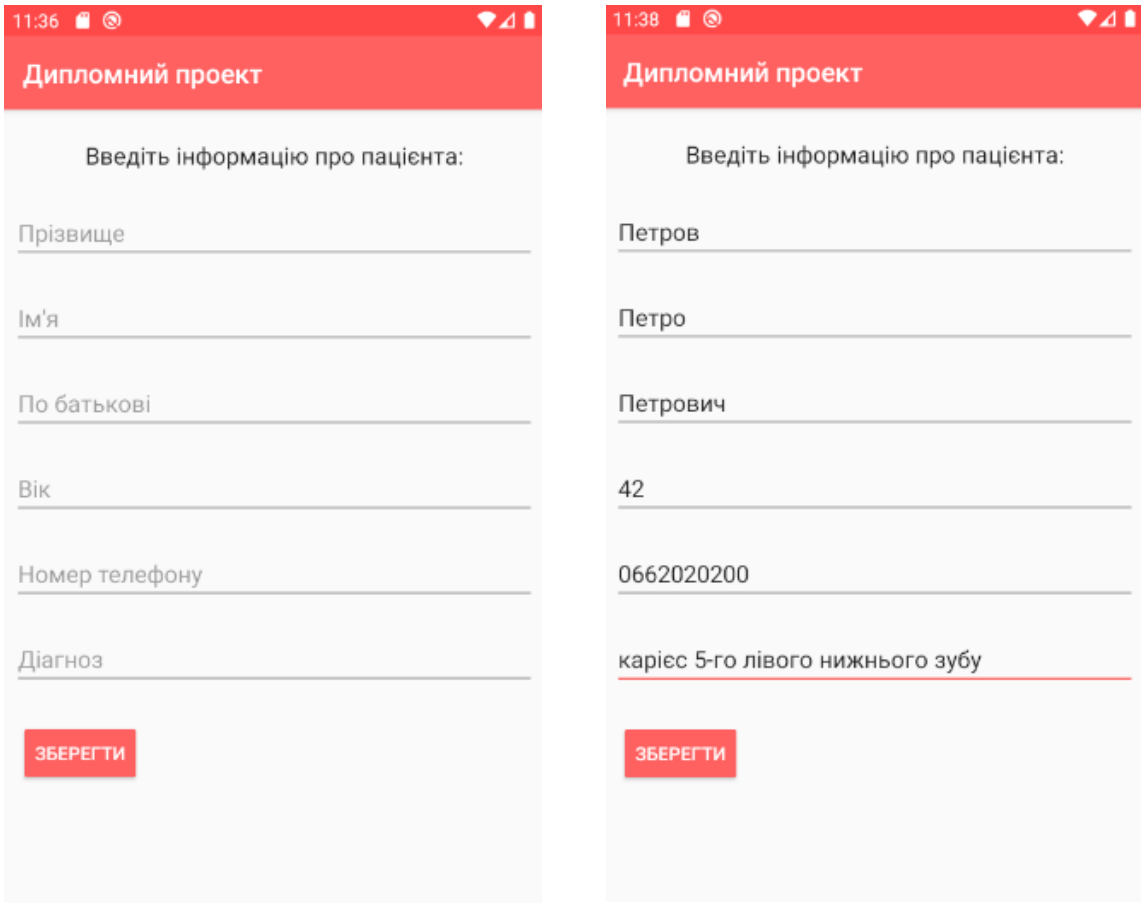


Рисунок 4.4 Ввід даних про пацієнта в вікно вводу даних

Після натискання кнопки «Зберегти», дані направляться до бази та при поверненні на головний екран, будуть виведені в відповідному полі.

Для тесту я додам дані про неіснуючих пацієнтів для різних дат. Та продемонструю вивід цих даних на головному екрані на рис. 4.5 та рис. 4.6.

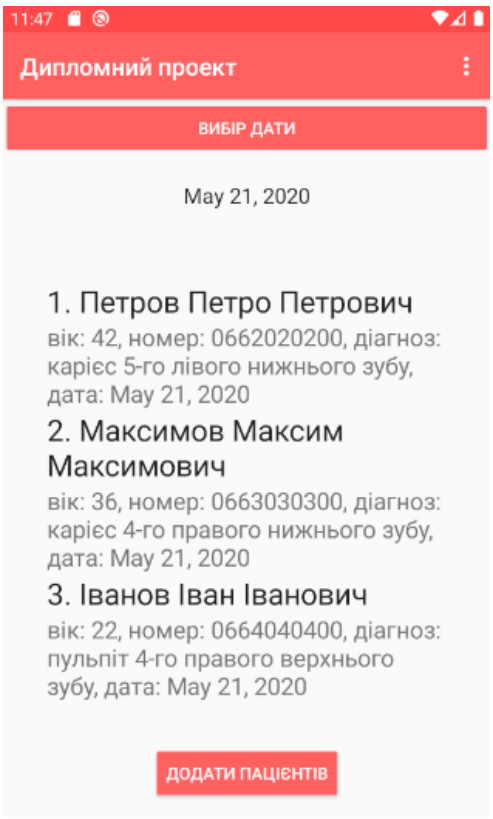


Рисунок 4.5 Вигляд головного екрану для 21 травня

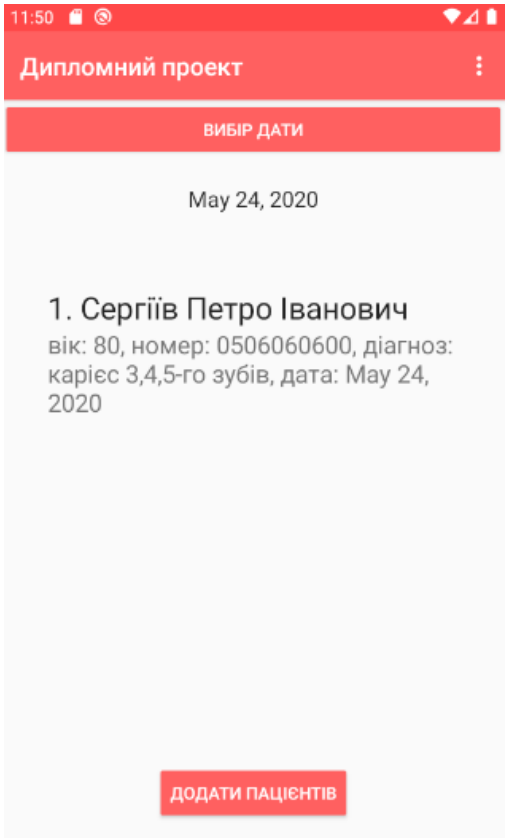


Рисунок 4.6 Вигляд головного екрану для 24 травня

Таким чином на головному екрані реалізовано зручний функціонал для перегляду інформації про пацієнтів та додавання нової.

4.2 Пошук пацієнтів в базі

Для того, щоб знайти конкретного пацієнта, лікарю не обов'язково згадувати дату, коли він його прийняв. Буде достатньо скористатися вікном «Пошук пацієнтів».

Щоб потрапити в це вікно користувачу потрібно натиснути на кнопку меню в правому верхньому кутку та обрати «Пошук пацієнтів». Він потрапить в вікно, зображене на рис. 4.7.

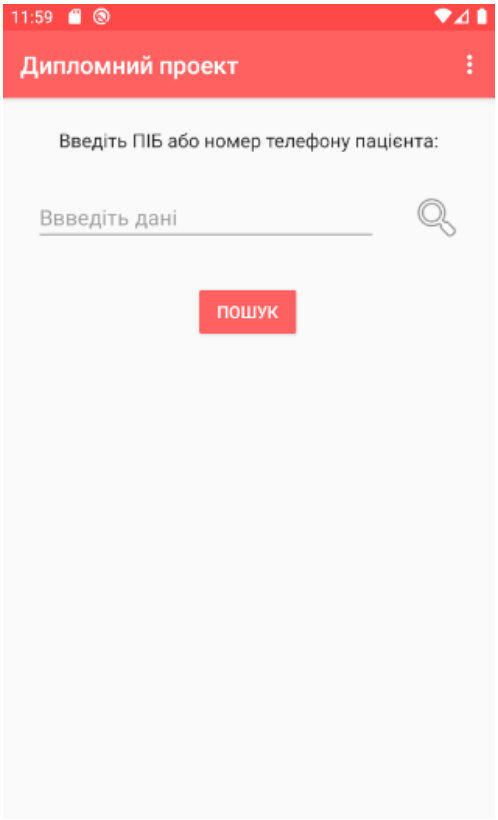


Рисунок 4.7 Вікно пошуку пацієнтів

Тут лікар може ввести будь-які дані з запропонованих та побачити всіх пацієнтів в базі з такими даними. Для тесту я введу ім'я «Петро», продемонструвавши правильну роботу програми (рис 4.8).

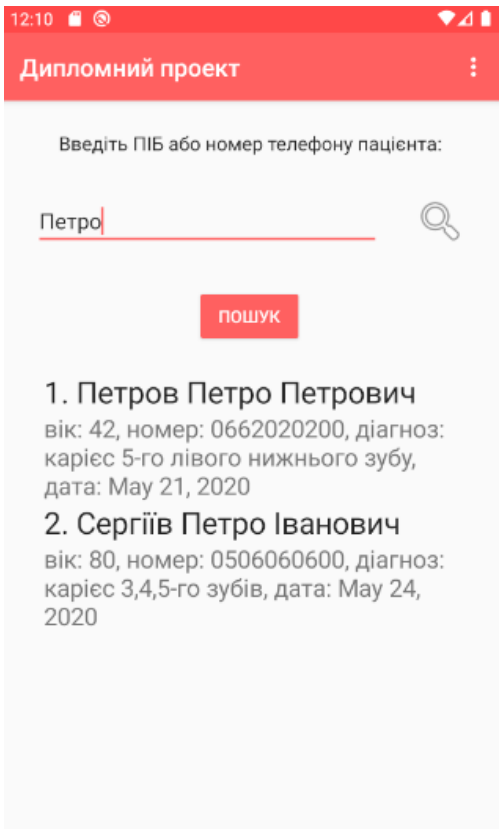


Рисунок 4.8 Результат пошуку

4.3 Робота вікна «Склад»

Для переходу в це вікно треба також скористатися меню, натиснувши в ньому «Склад». Користувач перейде в відповідне вікно, зображене на рис. 4.9.

Це вікно необхідно для додавання інформації про ліки, які є в наявності у лікаря, та перегляду цієї інформації. Користувачу важливо вчасно вносити дані про матеріали, які він використовує. Рекомендовано оновлювати інформацію хоча б раз на тиждень. Але це залежить від частоти закупок.

Інформація у вікні «Склад» буде виводитися двома кольорами: червоним та зеленим. Червоний колір означає що даних про конкретний вид ліків недостатньо для здійснення прогнозу, описаного в наступному підрозділі. Якщо ж дані зеленого кольору, то лікар може спрогнозувати закупку. Мінімум необхідно чотири записи про кожні ліки.

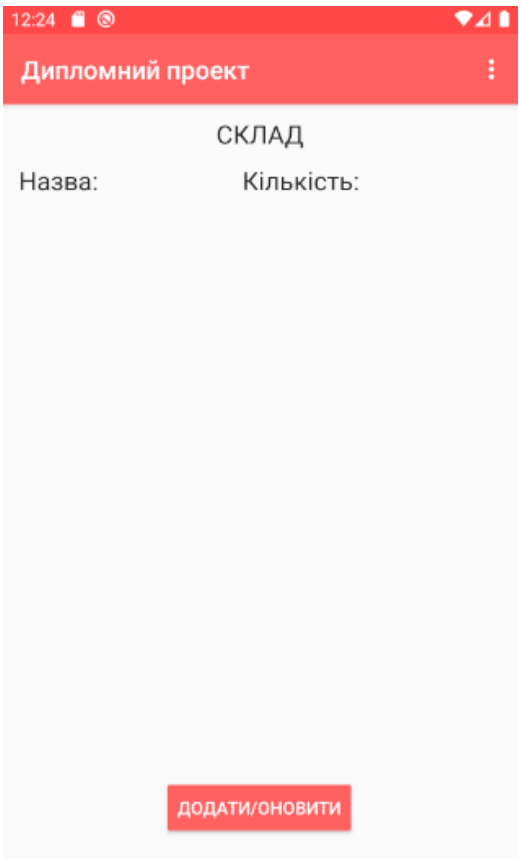


Рисунок 4.9 Вигляд вікна «Склад» при першому запуску

Для тестування цієї функції я буду вводити інформацію про матеріали кожні 5 днів. Тобто спочатку виберу сьогоднішню дату, а потім для тесту буду вибирати наступні дати і оновлювати дані про ліки. Після першого вводу екран складу буде виглядати як на рис. 4.11.

Після того як я три рази оновлю інформацію, вона стане зеленого кольору (рис. 4.12), бо системі стало достатньо інформації для прогнозу.

Така незручність буде лише при першому використанні програми, в подальшому не залежно від того коли був прогноз, можна буде робити новий. Але все ж для точності прогнозу рекомендується частіше оновлювати дані.

Щоб додати нові дані до бази, лікарю необхідно натиснути кнопку «Додати/Оновити». Система відкриє нове вікно, зображене на рис. 4.10.

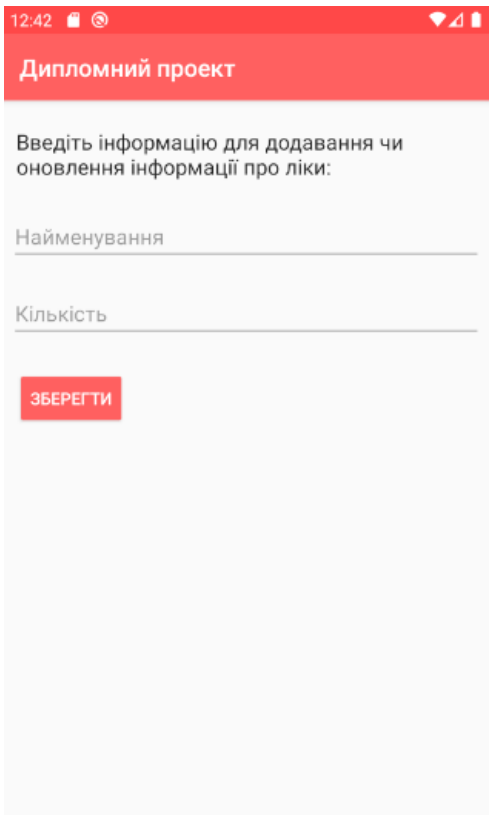


Рисунок 4.10 Вікно додавання інформації про ліки в базу

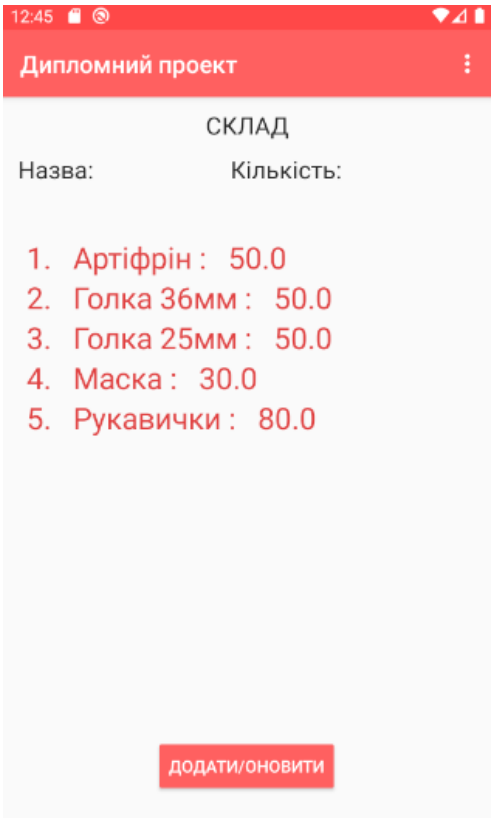


Рисунок 4.11 Вигляд вікна «Склад» після першого вводу (на 21.05)

Рисунок 4.12 Вигляд вікна «Склад» після останнього вводу (на 7.06)

Як видно, записи стали зеленого кольору, отже даних достатньо для створення прогнозу поставки цих матеріалів.

4.4 Тестування прогнозу поставки ліків

Для того щоб отримати прогноз закупки, лікар повинен відкрити меню та перейти на вікно «Прогноз закупки». Завантажиться нове вікно, яке матиме вигляд як на рис. 4.13. Тут, в спеціальне поле, користувач повинен ввести кількість днів, на яку необхідно зробити прогноз ти натиснути кнопку «Спрогнозувати».

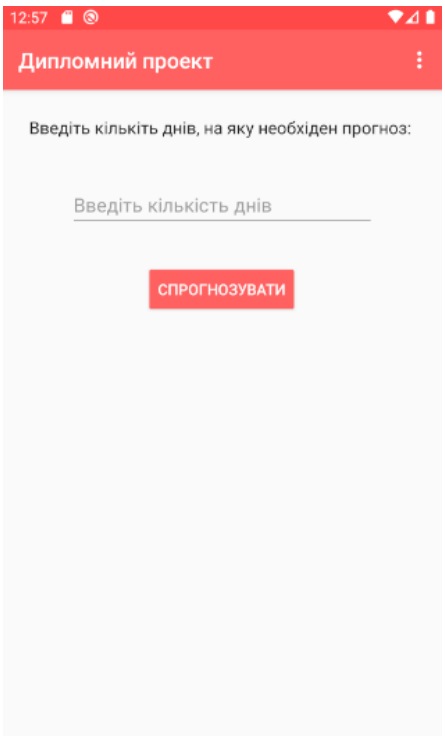


Рисунок 4.13 Вигляд вікна «Прогноз закупки»

Для тестування прогнозу та демонстрації правильності роботи програми я створюю таблицю 4.1, в яку додам всю інформацію про ліки, яка була додана до системи. Порахую приблизний прогноз в цьому випадку та порівняю з тим, що виведе програма.

Таблиця 4.1 Порівняння прогнозу програми з приблизним

№	Назва	21.05	26.05	30.05	7.06	Приблизне значення (на 14 днів)	Прогноз програми (на 14 днів)
1	Артіфрін	50	42	36	28	19	**19**
2	Голка 36мм	50	46	43	34	12	**12**
3	Голка 25мм	50	46	45	30	15	**14**
4	Маска	30	25	22	13	14	**13**
5	Рукавички	80	70	63	50	25	**25**

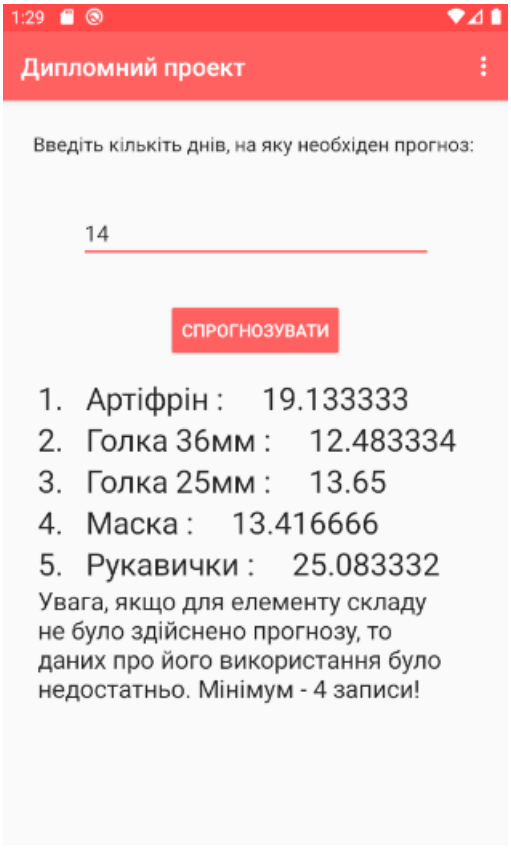


Рисунок 4.14 Результати прогнозу закупки

Отже, з результатів тестування видно що програма працює правильно, а невелике відхилення спричинене тим, що програма рахує точніше ніж я, тому є різниця.

Після оприлюднення прогнозу система автоматично запише результат в базу та оновить вікно «Склад». Тобто вважатиметься що лікар закупив ліки по прогнозу та продовжив працювати, оновлюючи дані.

ВИСНОВКИ ДО РОЗДІЛУ 4

В цьому розділі було проведено роботу над тестуванням програми та створенням інструкції користувача. Буро розглянуто кожне вікно програми та описано його функціонал. Ця інформація може бути корисною лікарю, який буде користувачем цього додатку. Було протестовано механізм оптимізації поставки ліків, запропонований в цьому проекті. Результати тестування показують що програма працює правильно, тому що результат збігається з ручним обрахунком.

Результатом роботи над цим розділом можна вважати готову інструкцію користувача та протестований програмний продукт.

ВИСНОВКИ

В даному дипломному проекті було створено додаток на операційну систему Android для збору даних про пацієнтів лікарень та оптимізації поставки ліків. Для цього спочатку було розглянуто предметну область та сучасні інформаційні технології в медицині. Розглянуто медичні інформаційні системи та їх необхідний функціонал. Оглянуто вже існуючі системи обліку пацієнтів та мобільні додатки для оптимізації процесу закупок. Було зроблено висновок що вже існує багато таких систем, але кожен має свої плюси та недоліки.

Далі було розглянуто програмні технології, які використовувалися для створення продукту. Вивчено теоретичний матеріал про мову програмування Java та операційну систему Android. Було обрано оптимальну мінімальні версію Android, для якої розроблявся додаток. Було налаштовано середовище розробки Android Studio та створено проект. Створено блок-схеми алгоритмів роботи програми.

Після цього було написано код програми, но основі створеного алгоритму. Розглянуто структуру готового проекту, описано всі класи, що забезпечують роботу системи. Розроблено мінімалістичний дизайн інтерфейсу.

Проведено тестування програмного продукту та створено інструкцію користувача для нього. В результаті тестування стало зрозуміло що програма працює правильно, та виконує всі необхідні функції, поставлені на початку роботи.

Отже, результатом роботи є готовий та протестований додаток на Android, який виконує поставлені перед ним задачі.

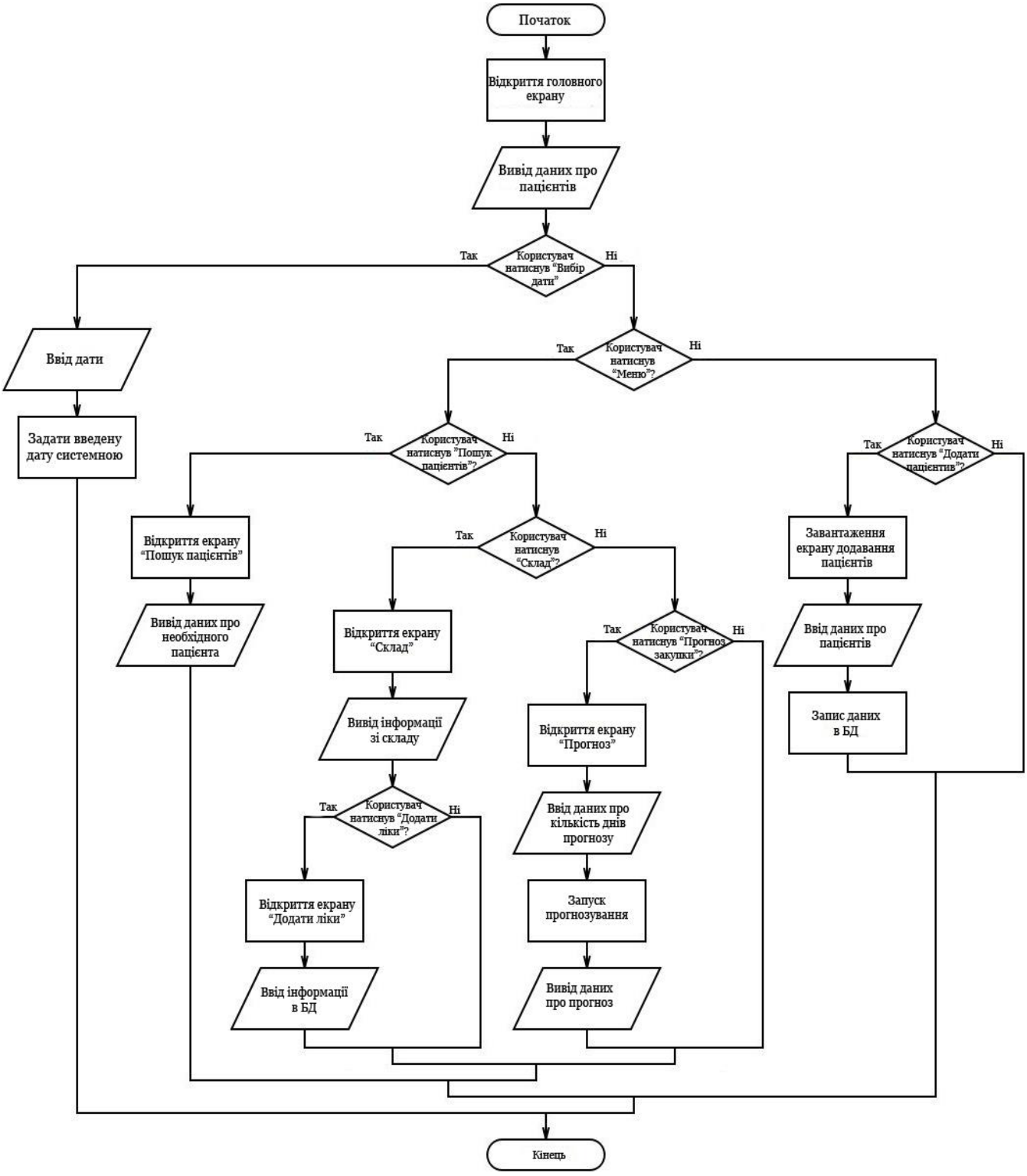
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Медицинская информационная система [сайт]. URL: https://ru.wikipedia.org/wiki/Медицинская_информационная_система.
2. Информационные технологии в медицине [сайт]. URL: https://studopedia.ru/15_20371_informatsionnie-tehnologii-v-meditsine.html.
3. Про HELSI [сайт]. URL: https://helsi.me/about.
4. Медична інформаційна система «ЕМСІМЕД» - детальна інформація [сайт]. URL: https://www.mcmed.ua/ua/about.
5. What is Java programming language? [сайт]. URL: https://howtodoinjava.com/java/basics/what-is-java-programming-language/.
6. Что такое JDK, JRE и JVM в Java? [сайт]. URL: https://javadevblog.com/chto-takoe-jdk-jre-i-jvm-v-java.html.
7. Брюс Эккель Философия Java/ Питер 2015 – 1168 с.
8. Android [сайт]. URL: https://www.android.com/.
9. Знакомство с паттерном MVC (Model-View-Controller) [сайт]. URL: https://javarush.ru/groups/posts/2536-chastjh-7-znakomstvo-s-patternom-mvc-model-view-controller.
10. Android Studio [сайт]. URL: https://developer.android.com/studio.
11. Dr. Eleks [сайт]. URL: https://doctor.eleks.com/.
12. Хабр [сайт]. URL: https://habr.com/ru/news/t/482688/
13. Футуро [сайт]. URL: https://futuro.in.ua/news/2801-vcheni-nadrukuvaly-zhyvu-shkiru-komplekti.html
14. Топ-20 мобильных приложений для учета бизнеса [сайт]. URL: https://www.ekam.ru/blogs/pos/20-mobilnyh-prilozheniy-dlya-biznesa
15. Android – википедия [сайт]. URL: https://ru.wikipedia.org/wiki/Android

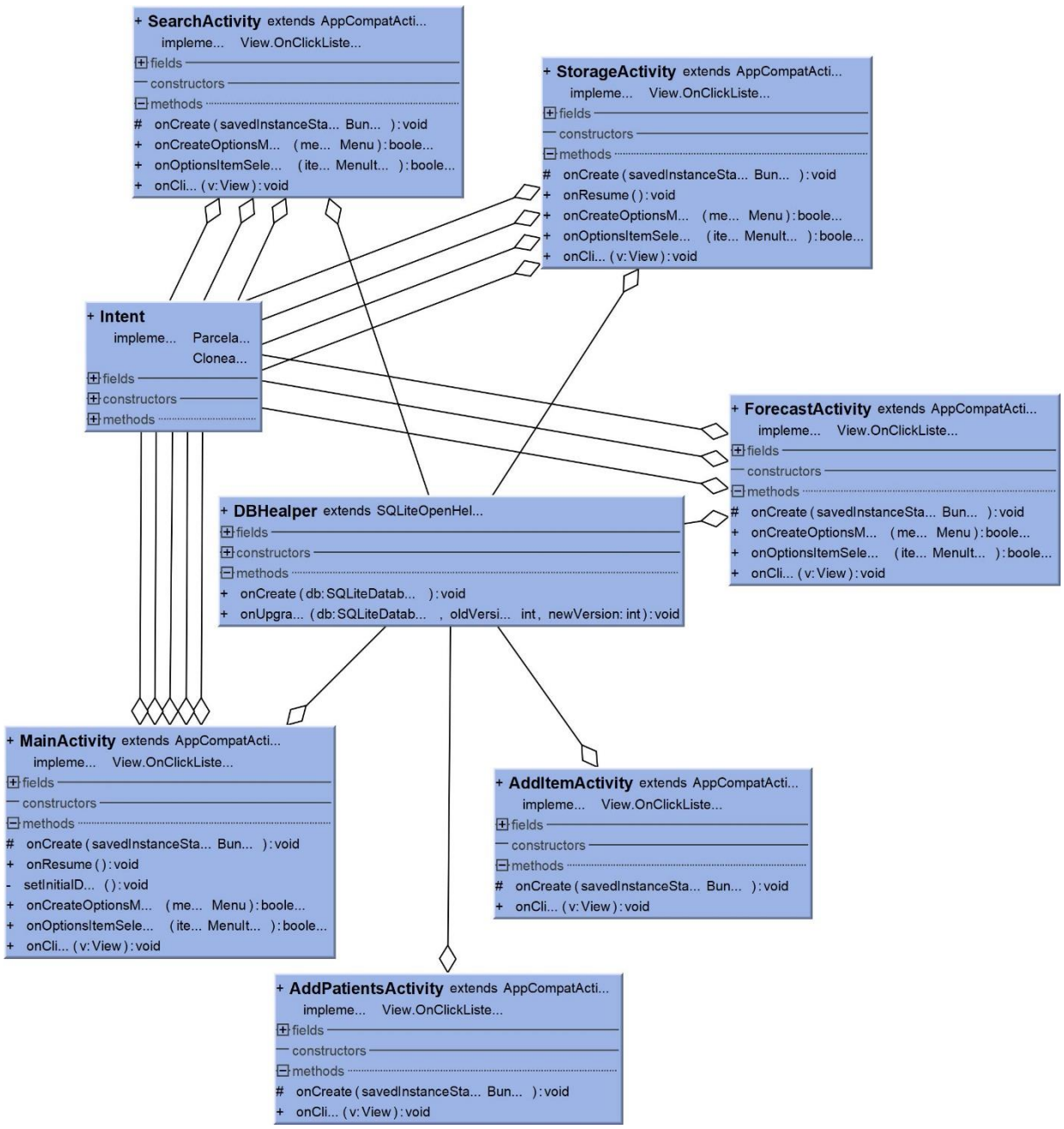
16. MVC – википедия [сайт]. URL: https://ru.wikipedia.org/wiki/Model-View-Controller
17. Математические методы прогнозирования в медицине [сайт]. URL: https://www.natural-sciences.ru/ru/article/view?id=33316

ДОДАТОК А

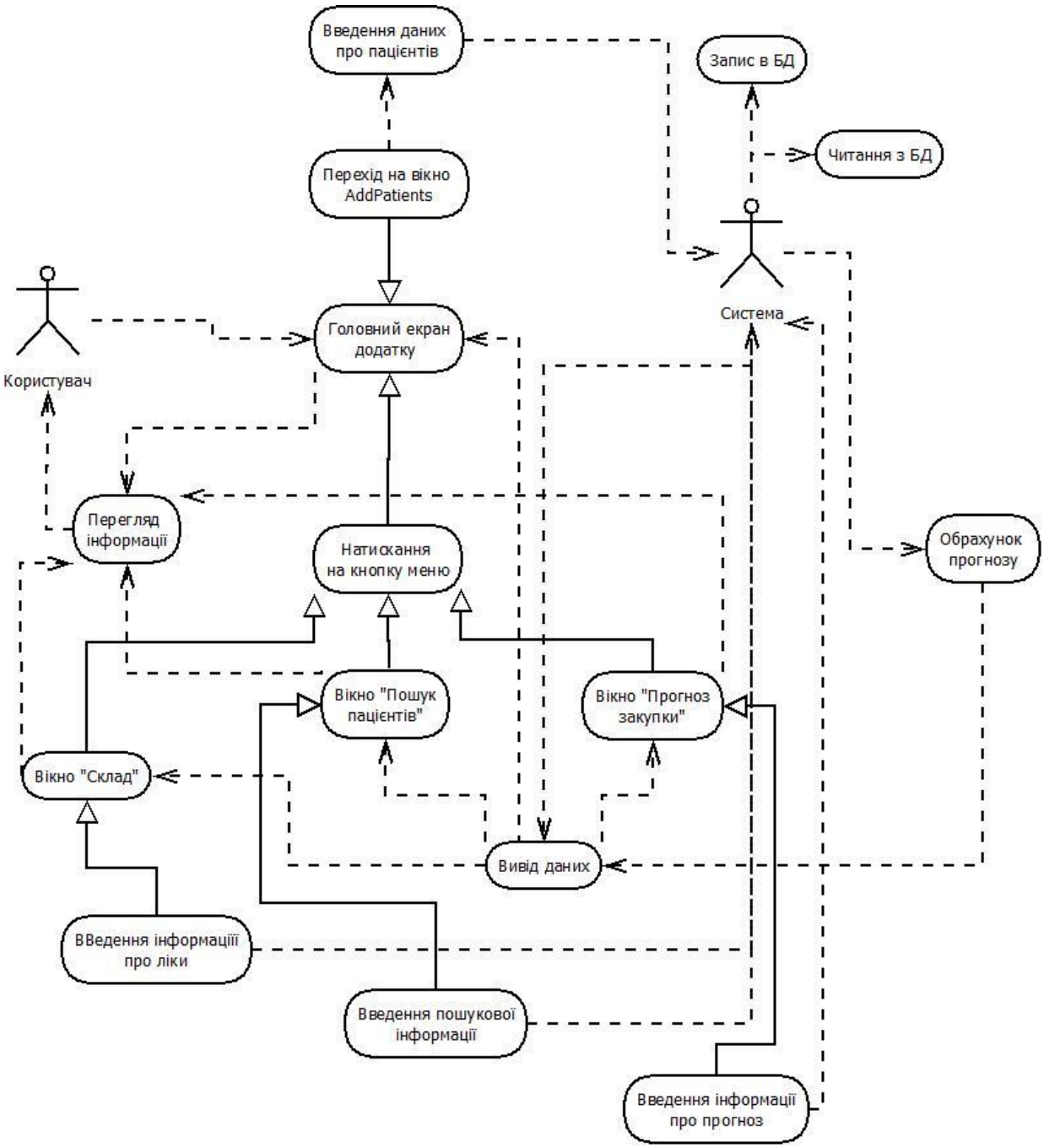
КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ



					ДП.4678.04.000 А1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		*Кушнір О.С.*			*Принципова схема алгоритму*	Літ. / Аркуш 1 / Аркушів 1
Перевірив		*Антонюк А.І.*				
Реценз.						
Н. Контр.		*Сімоненко В.П.*				*НТУУ «КПІ», ФІОТ, ІО-62*
Затвердив						



					ДП.4678.05.000 А2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		*Кушнір О.С.*			*Функціональна схема*	Літ. / Аркуш 1 / Аркушів 1
Перевірив		*Антонюк А.І.*				
Реценз.						*НТУУ «КПІ», ФІОТ, ІО-62*
Н. Контр.		*Сімоненко В.П.*				
Затвердив						



					ДП.4678.06.000 А3	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		*Кушнір О.С.*			*Структурна схема*	Літ. / Аркуш 1 / Аркушів 1
Перевірив		*Антонюк А.І.*				
Реценз.						*НТУУ «КПІ», ФІОТ, ІО-62*
Н. Контр.		*Сімоненко В.П.*				
Затвердив						

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМИ

DBHealper.java

```
package com.example.graduatework1;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHealper extends SQLiteOpenHelper {

    public static final int DATABASE_VERSION = 5;
    public static final String DATABASE_NAME = "graduateWork";

    public static final String TABLE_PATIENTS = "patients";
    public static final String TABLE_STORAGE = "storage";

    public static final String KEY_ID = "_id";
    public static final String KEY_SURNAME = "surname";
    public static final String KEY_NAME = "name";
    public static final String KEY_PATRONYMIC = "patronymic";
    public static final String KEY_AGE = "age";
    public static final String KEY_TELEPHONE = "telephone";
    public static final String KEY_DIAGNOSIS = "diagnosis";
    public static final String KEY_DATE = "date";

    public static final String KEY_DRUG_NAME = "drug_name";
    public static final String KEY_QUANTITY = "quantity";


    public DBHealper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE_PATIENTS + "(" + KEY_ID + " integer
primary key," + KEY_SURNAME + " text," +
                KEY_NAME + " text," + KEY_PATRONYMIC + " text," + KEY_AGE + "
integer," + KEY_TELEPHONE + " text," +
                KEY_DIAGNOSIS + " text," + KEY_DATE + " integer" + ")");

        db.execSQL("create table " + TABLE_STORAGE + "(" + KEY_ID + " integer
primary key," + KEY_DRUG_NAME + " text," +
                KEY_QUANTITY + " real," + KEY_DATE + " integet" + ")");

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("drop table if exists " + TABLE_PATIENTS);
        db.execSQL("drop table if exists " + TABLE_STORAGE);
        onCreate(db);
    }
}
```

MainActivity.java

```
package com.example.graduatework1;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.text.format.DateUtils;
import android.util.Log;
import android.util.TypedValue;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import java.util.Calendar;

public class MainActivity extends AppCompatActivity implements View.OnClickListener
{

    Button buttonCalendar, buttonAddPatients;
    TextView textDate;
    ScrollView scrollView;
    LinearLayout linearLayout;
    Calendar calendar;
    Intent intentAddPatients, intentStorage, intentAddItem, intentSearch,
intentForecast;

    DBHealper dbHealper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        buttonCalendar = (Button) findViewById(R.id.buttonCalendar);
        buttonAddPatients = (Button) findViewById(R.id.buttonAddPatients);
        textDate = (TextView) findViewById(R.id.textDate);
        scrollView = (ScrollView) findViewById(R.id.scrollView1);
        linearLayout = (LinearLayout) findViewById(R.id.linearLayout);

        intentAddPatients = new Intent(this, AddPatientsActivity.class);
        intentStorage = new Intent(this, StorageActivity.class);
        intentAddItem = new Intent(this, AddItemActivity.class);
        intentSearch = new Intent(this, SearchActivity.class);
        intentForecast = new Intent(this, ForecastActivity.class);

        buttonCalendar.setOnClickListener(this);
        buttonAddPatients.setOnClickListener(this);

        calendar = Calendar.getInstance();

        setInitialDate();
    }
```

```
    @Override
    public void onResume(){
        super.onResume();

        //Вывод пациентов
        linearLayout.removeAllViews();

        dbHealper = new DBHealper(this);
        SQLiteDatabase database = dbHealper.getWritableDatabase();

        Cursor cursor = database.query(DBHealper.TABLE_PATIENTS, null, null, null,
null, null, null);

        //Вывод БД в лог
        //Cursor cursor2 = database.query(DBHealper.TABLE_STORAGE, null, null,
null, null, null, null);
        //if (cursor2.moveToFirst()){
        //    do{
        //        Log.d("mLog", "ID = " +
cursor2.getInt(cursor2.getColumnIndex(DBHealper.KEY_ID))+
        //                ", name = " +
cursor2.getString(cursor2.getColumnIndex(DBHealper.KEY_DRUG_NAME)) +
        //                ", quan = " +
cursor2.getFloat(cursor2.getColumnIndex(DBHealper.KEY_QUANTITY)) +
        //                ", date = " + (DateUtils.formatDateTime(this,
(long)(cursor2.getInt(cursor2.getColumnIndex(DBHealper.KEY_DATE))) * 1000,
DateUtils.FORMAT_SHOW_DATE| DateUtils.FORMAT_SHOW_YEAR)));
        //    } while (cursor2.moveToNext());
        //} else
        //    Log.d("mLog", "0 rows");
        //cursor2.close();

        if(cursor.moveToFirst()){
            int counter = 1;
            do{
                String sur =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_SURNAME));
                String nam =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_NAME));
                String patr =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_PATRONYMIC));
                Integer ag =
cursor.getInt(cursor.getColumnIndex(DBHealper.KEY_AGE));
                String tel =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_TELEPHONE));
                String dig =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_DIAGNOSIS));
                Integer dtInt =
cursor.getInt(cursor.getColumnIndex(DBHealper.KEY_DATE));

                if (dtInt >= ((int)(calendar.getTimeInMillis()/1000))-43199 &&
dtInt <= ((int)(calendar.getTimeInMillis()/1000))+43199){

                    TextView tV2 = new TextView(MainActivity.this);
                    TextView tV3 = new TextView(MainActivity.this);
                    tV2.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 25);
                    tV3.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);
                    tV2.setTextColor(Color.parseColor("#212121"));
                    tV3.setTextColor(Color.parseColor("#757575"));
                    tV2.setText(counter + ". " + sur + " " + nam + " " + patr);
                    String dtLong = (DateUtils.formatDateTime(this, (long)dtInt *
```

```
1000, DateUtils.FORMAT_SHOW_DATE| DateUtils.FORMAT_SHOW_YEAR));
                    tV3.setText("вік: " + ag + ", номер: " + tel + ", діагноз: " +
dig + ", дата: " + dtLong);
                    linearLayout.addView(tV2);
                    linearLayout.addView(tV3);
                    counter++;
                }

            }while (cursor.moveToNext());
        }

        cursor.close();

    }
    // установка начальной даты
    private void setInitialDate() {
        textDate.setText(DateUtils.formatDateTime(this, calendar.getTimeInMillis(),
                DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_YEAR));
    }

    // установка обработчика выбора даты
    DatePickerDialog.OnDateSetListener d = new DatePickerDialog.OnDateSetListener()
{
        public void onDateSet(DatePicker view, int year, int monthOfYear, int
dayOfMonth) {
            calendar.set(Calendar.YEAR, year);
            calendar.set(Calendar.MONTH, monthOfYear);
            calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
            onResume();
            setInitialDate();
        }
    };

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {

        switch (item.getItemId()){
            case R.id.menu_home:
                break;
            case R.id.menu_storage:
                intentStorage.putExtra("date", (int)
(calendar.getTimeInMillis()/1000));
                intentStorage.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentStorage);
                break;
            case R.id.menu_forcast:
                intentForecast.putExtra("date", (int)
(calendar.getTimeInMillis()/1000));
                intentForecast.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentForecast);
                break;
            case R.id.menu_search:
                intentSearch.putExtra("date", (int)
(calendar.getTimeInMillis()/1000));
                intentSearch.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
```

```
                startActivity(intentSearch);
                break;

        }
        return super.onOptionsItemSelected(item);
    }


    @Override
    public void onClick(View v) {

        switch (v.getId()){
            case R.id.buttonCalendar:
                new DatePickerDialog(MainActivity.this, d,
calendar.get(Calendar.YEAR),
                        calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH))
                        .show();
                break;
            case R.id.buttonAddPatients:
                intentAddPatients.putExtra("date", (int)
(calendar.getTimeInMillis()/1000));
                startActivity(intentAddPatients);
                break;
        }
    }
}
```

AddPatientsActivity.java

```
package com.example.graduatework1;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


public class AddPatientsActivity extends AppCompatActivity implements
View.OnClickListener{

    EditText editTextSurname, editTextName, editTextPatronymic, editTextAge,
            editTextTelephone, editTextDiagnosis;
    Button buttonSave; //buttonClear

    DBHealper dbHealper;

    Bundle arguments;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_patients);

        editTextSurname = (EditText) findViewById(R.id.editTextSurname);
```

```
        editTextName = (EditText) findViewById(R.id.editTextName);
        editTextPatronymic = (EditText) findViewById(R.id.editTextPatronymic);
        editTextAge = (EditText) findViewById(R.id.editTextAge);
        editTextTelephone = (EditText) findViewById(R.id.editTextTelephone);
        editTextDiagnosis = (EditText) findViewById(R.id.editTextDiagnosis);

        buttonSave = (Button) findViewById(R.id.buttonSave);
        //buttonClear = (Button) findViewById(R.id.buttonClear);

        buttonSave.setOnClickListener(this);
        //buttonClear.setOnClickListener(this);

        dbHealper = new DBHealper(this);

    }


    @Override
    public void onClick(View v) {

        //запись пациентов в БД
        SQLiteDatabase database = dbHealper.getWritableDatabase();

        switch (v.getId()){
            case R.id.buttonSave:
                Toast toast = Toast.makeText(AddPatientsActivity.this, "Дані
збережено!", Toast.LENGTH_SHORT);
                toast.show();

                arguments = getIntent().getExtras();

                String surname = editTextSurname.getText().toString();
                String name = editTextName.getText().toString();
                String patronymic = editTextPatronymic.getText().toString();
                Integer age = Integer.parseInt(editTextAge.getText().toString());
                String telephone = editTextTelephone.getText().toString();
                String diagnosis = editTextDiagnosis.getText().toString();
                Integer date = arguments.getInt("date");

                ContentValues contentValues = new ContentValues();

                contentValues.put(dbHealper.KEY_SURNAME, surname);
                contentValues.put(dbHealper.KEY_NAME, name);
                contentValues.put(dbHealper.KEY_PATRONYMIC, patronymic);
                contentValues.put(dbHealper.KEY_AGE, age);
                contentValues.put(dbHealper.KEY_TELEPHONE, telephone);
                contentValues.put(dbHealper.KEY_DIAGNOSIS, diagnosis);
                contentValues.put(dbHealper.KEY_DATE, date);

                database.insert(dbHealper.TABLE_PATIENTS, null, contentValues);

                break;
                //Кнопка очистки БД
            //case R.id.buttonClear:
            //    Toast toast2 = Toast.makeText(AddPatientsActivity.this, "БД
очищена!", Toast.LENGTH_SHORT);
            //    toast2.show();
            //    database.delete(dbHealper.TABLE_PATIENTS, null, null);
            //    break;

        }
```

```
    }
}
```

SearchActivity.java

```
package com.example.graduatework1;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.text.format.DateUtils;
import android.util.TypedValue;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

public class SearchActivity extends AppCompatActivity implements
View.OnClickListener{

    Intent intentMain, intentStorage, intentForecast;
    Button buttonSearch;
    EditText editTextSearch;
    LinearLayout linearLayoutSearch;

    DBHealper dbHealper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search);

        intentMain = new Intent(this, MainActivity.class);
        intentStorage = new Intent(this, StorageActivity.class);
        intentForecast = new Intent(this, ForecastActivity.class);

        buttonSearch = (Button) findViewById(R.id.buttonSearch);
        editTextSearch = (EditText) findViewById(R.id.editTextSearch);
        linearLayoutSearch = (LinearLayout) findViewById(R.id.linearLayoutSearch);

        buttonSearch.setOnClickListener(this);
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
```

```
        switch (item.getItemId()){
            case R.id.menu_home:
                intentMain.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentMain);
                break;
            case R.id.menu_storage:
                intentStorage.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentStorage);
                break;
            case R.id.menu_forcast:
                intentForecast.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentForecast);
                break;
            case R.id.menu_search:
                break;

        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onClick(View v) {

        linearLayoutSearch.removeAllViews();

        String searchField = editTextSearch.getText().toString();

        dbHealper = new DBHealper(this);

        SQLiteDatabase database = dbHealper.getWritableDatabase();

        //Вывод лекарств на экран

        Cursor cursor = database.query(DBHealper.TABLE_PATIENTS, null, null, null,
null, null, null);

        if(cursor.moveToFirst()){
            int counter = 1;
            do{
                String sur =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_SURNAME));
                String nam =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_NAME));
                String patr =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_PATRONYMIC));
                String tel =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_TELEPHONE));
                Integer ag =
cursor.getInt(cursor.getColumnIndex(DBHealper.KEY_AGE));
                String dig =
cursor.getString(cursor.getColumnIndex(DBHealper.KEY_DIAGNOSIS));
                Integer dtInt =
cursor.getInt(cursor.getColumnIndex(DBHealper.KEY_DATE));

                if(searchField.equals(sur) || searchField.equals(nam) ||
searchField.equals(patr) || searchField.equals(tel)){
                    TextView tV2 = new TextView(SearchActivity.this);
                    TextView tV3 = new TextView(SearchActivity.this);
                    tV2.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 25);
                    tV3.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);
                    tV2.setTextColor(Color.parseColor("#212121"));
```

```
                    tV3.setTextColor(Color.parseColor("#757575"));
                    tV2.setText(counter + ". " + sur + " " + nam + " " + patr);
                    String dtLong = (DateUtils.formatDateTime(this, (long)dtInt *
1000, DateUtils.FORMAT_SHOW_DATE| DateUtils.FORMAT_SHOW_YEAR));
                    tV3.setText("вік: "+ ag + ", номер: " + tel + ", діагноз: " +
dig + ", дата: " + dtLong);
                    linearLayoutSearch.addView(tV2);
                    linearLayoutSearch.addView(tV3);
                    counter++;
                }

            }while (cursor.moveToNext());
        }

        cursor.close();

    }
}
```

StorageActivity.java

```
package com.example.graduatework1;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.util.TypedValue;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.ArrayList;

public class StorageActivity extends AppCompatActivity implements
View.OnClickListener{

    Intent intent, intentAddItem, intentSearch, intentForecast;
    Button buttonAddItem;
    LinearLayout linearLayoutOutput;
    DBHealper dbHealper;
    Bundle arguments;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_storage);

        intent = new Intent(this, MainActivity.class);
        intentAddItem = new Intent(this, AddItemActivity.class);
        intentSearch = new Intent(this, SearchActivity.class);
        intentForecast = new Intent(this, ForecastActivity.class);
```

```
        buttonAddItem = (Button) findViewById(R.id.buttonAddItem);
        linearLayoutOutput = (LinearLayout) findViewById(R.id.linearLayoutOutput);

        buttonAddItem.setOnClickListener(this);

    }

    @Override
    public void onResume(){
        super.onResume();

        linearLayoutOutput.removeAllViews();

        dbHealper = new DBHealper(this);

        SQLiteDatabase database = dbHealper.getWritableDatabase();

        //Вывод лекарств на экран

        Cursor cursor1 = database.query(DBHealper.TABLE_STORAGE, null, null, null,
null, null, null);
        Cursor cursor2 = database.query(DBHealper.TABLE_STORAGE, null, null, null,
null, null, null);


        if(cursor1.moveToFirst()){
            int counter = 1;
            ArrayList<String> names = new ArrayList<>();
            do{

if(names.contains(cursor1.getString(cursor1.getColumnIndex(DBHealper.KEY_DRUG_NAME)
))){
                }
                else {
                    int counter2 = 0;
                    Integer dateInt1 =
cursor1.getInt(cursor1.getColumnIndex(DBHealper.KEY_DATE));
                    String drugName1 =
cursor1.getString(cursor1.getColumnIndex(DBHealper.KEY_DRUG_NAME));

                    names.add(drugName1);

                    TextView tVName = new TextView(StorageActivity.this);
                    tVName.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 25);

                    if(cursor2.moveToFirst()){
                        do{
                            String drugName2 =
cursor2.getString(cursor2.getColumnIndex(DBHealper.KEY_DRUG_NAME));
                            float quan2 =
cursor2.getFloat(cursor2.getColumnIndex(DBHealper.KEY_QUANTITY));
                            Integer dateInt2 =
cursor2.getInt(cursor2.getColumnIndex(DBHealper.KEY_DATE));

                            if(drugName2.equals(drugName1) && (dateInt2 >=
dateInt1)){
                                tVName.setText(counter + ".   " + drugName2 + " :
" + quan2);
                            }
                            if(drugName2.equals(drugName1)){
                                counter2++;
```

```
                        }

                    }while (cursor2.moveToNext());
                }
                counter++;
                if(counter2 > 3){
                    tVName.setTextColor(Color.parseColor("#3b7a3e"));
                }
                else {
                    tVName.setTextColor(Color.parseColor("#e14040"));
                }
                linearLayoutOutput.addView(tVName);
            }

        }while (cursor1.moveToNext());
        names.clear();
    }

    cursor1.close();
    cursor2.close();

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {

    switch (item.getItemId()){
        case R.id.menu_home:
            intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
            startActivity(intent);
            break;
        case R.id.menu_storage:
            break;
        case R.id.menu_forcast:
            arguments = getIntent().getExtras();
            Integer date = arguments.getInt("date");
            intentForecast.putExtra("date", date);
            intentForecast.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
            startActivity(intentForecast);
            break;
        case R.id.menu_search:
            intentSearch.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
            startActivity(intentSearch);
            break;

    }

    return super.onOptionsItemSelected(item);
}

@Override
public void onClick(View v) {
    arguments = getIntent().getExtras();
    Integer date = arguments.getInt("date");
    intentAddItem.putExtra("date", date);
```

```
        startActivity(intentAddItem);
    }
}
```

AddItemActivity.java

```
package com.example.graduatework1;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddItemActivity extends AppCompatActivity implements
View.OnClickListener{

    EditText editTextNameItem, editTextQuantity;
    Button buttonSave2; //buttonClear2

    DBHealper dbHealper;

    Bundle arguments;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_item);

        editTextNameItem = (EditText) findViewById(R.id.editTextNameItem);
        editTextQuantity = (EditText) findViewById(R.id.editTextQuantity);

        buttonSave2 = (Button) findViewById(R.id.buttonSave2);
        //buttonClear2 = (Button) findViewById(R.id.buttonClear2);

        buttonSave2.setOnClickListener(this);
        //buttonClear2.setOnClickListener(this);

        dbHealper = new DBHealper(this);
    }

    @Override
    public void onClick(View v) {

        //Добавление лекарств в БД
        SQLiteDatabase database = dbHealper.getWritableDatabase();

        switch (v.getId()){
            case R.id.buttonSave2:
                Toast toast = Toast.makeText(AddItemActivity.this, "Дані
збережено!", Toast.LENGTH_SHORT);
                toast.show();
                arguments = getIntent().getExtras();
                String nameItem = editTextNameItem.getText().toString();
                Float quantity =
Float.parseFloat(editTextQuantity.getText().toString());
```

```
                Integer date = arguments.getInt("date");

                ContentValues contentValues = new ContentValues();
                contentValues.put(dbHealper.KEY_DRUG_NAME, nameItem);
                contentValues.put(dbHealper.KEY_QUANTITY, quantity);
                contentValues.put(dbHealper.KEY_DATE, date);
                database.insert(dbHealper.TABLE_STORAGE, null, contentValues);


                break;
                //Кнопка очистки БД
            //case R.id.buttonClear2:
            //    Toast toast2 = Toast.makeText(AddItemActivity.this, "БД
очищена!", Toast.LENGTH_SHORT);
            //    toast2.show();
            //    database.delete(dbHealper.TABLE_STORAGE, null, null);
            //    break;

        }

    }
}
```

ForecastActivity.java

```
package com.example.graduatework1;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.os.Bundle;
import android.util.TypedValue;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.ArrayList;

public class ForecastActivity extends AppCompatActivity implements
View.OnClickListener{

    Intent intentMain, intentStorage, intentSearch;

    Button buttonForecast;
    EditText editTextForecast;
    LinearLayout linearLayoutForecast;

    DBHealper dbHealper;
    Bundle arguments;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_forecast);

        intentMain = new Intent(this, MainActivity.class);
        intentStorage = new Intent(this, StorageActivity.class);
        intentSearch = new Intent(this, SearchActivity.class);

        buttonForecast = (Button) findViewById(R.id.buttonForecast);
        editTextForecast = (EditText) findViewById(R.id.editTextForecast);
        linearLayoutForecast = (LinearLayout)
findViewById(R.id.linearLayoutForecast);

        buttonForecast.setOnClickListener(this);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {

        switch (item.getItemId()){
            case R.id.menu_home:
                intentMain.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentMain);
                break;
            case R.id.menu_storage:
                arguments = getIntent().getExtras();
                Integer date = arguments.getInt("date");
                intentStorage.putExtra("date", date);
                intentStorage.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentStorage);
                break;
            case R.id.menu_forcast:
                break;
            case R.id.menu_search:
                intentSearch.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                startActivity(intentSearch);
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onClick(View v) {

        arguments = getIntent().getExtras();
        Integer date = arguments.getInt("date");

        linearLayoutForecast.removeAllViews();

        dbHealper = new DBHealper(this);

        SQLiteDatabase database = dbHealper.getWritableDatabase();
```

```
        //Вывод лекарств на экран

        Cursor cursor1 = database.query(DBHealper.TABLE_STORAGE, null, null, null,
null, null, null);
        Cursor cursor2 = database.query(DBHealper.TABLE_STORAGE, null, null, null,
null, null, null);


        if(cursor1.moveToFirst()){

            int counter = 1;
            ArrayList<String> names = new ArrayList<>();
            do{
                Integer id =
cursor1.getInt(cursor1.getColumnIndex(DBHealper.KEY_ID));

if(names.contains(cursor1.getString(cursor1.getColumnIndex(DBHealper.KEY_DRUG_NAME)
))){
                    //if (id.equals("")){
                    //}
                    //else {
                    //    database.delete(DBHealper.TABLE_STORAGE, DBHealper.KEY_ID
+ "=" + id, null);
                    //}
                }
                else {
                    String drugName1 =
cursor1.getString(cursor1.getColumnIndex(DBHealper.KEY_DRUG_NAME));

                    names.add(drugName1);

                    TextView tVForecast = new TextView(ForecastActivity.this);
                    tVForecast.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 25);
                    tVForecast.setTextColor(Color.parseColor("#212121"));

                    ArrayList<Float> quantity = new ArrayList<>();
                    ArrayList<Integer> dates = new ArrayList<>();

                    if(cursor2.moveToFirst()){
                        do{
                            String drugName2 =
cursor2.getString(cursor2.getColumnIndex(DBHealper.KEY_DRUG_NAME));
                            Float quan2 =
cursor2.getFloat(cursor2.getColumnIndex(DBHealper.KEY_QUANTITY));
                            Integer dateInt2 =
cursor2.getInt(cursor2.getColumnIndex(DBHealper.KEY_DATE));

                            if(drugName2.equals(drugName1)){
                                quantity.add(quan2);
                                dates.add((dateInt2));
                            }

                        }while (cursor2.moveToNext());
                    }

                    if (quantity.size() < 4 || dates.size() < 4){
                        names.remove(drugName1);
                    }
                    else {
                        ArrayList<Float> quanForDay = new ArrayList<>();
                        for(int i = 0; i < 3; i++){
```

```
                            float quanDiff = quantity.get((quantity.size() - 4 +
i)) - quantity.get((quantity.size() - 3 + i));
                            float dateDiff = (float)dates.get(dates.size() - 3 + i)
- (float)dates.get(dates.size() - 4 + i);
                            if (quanDiff >= 0 && dateDiff > 0){
                                float res = quanDiff/(dateDiff/86400);
                                quanForDay.add(res);
                            }
                        }
                        Float average = (float) 0;
                        for (int i = 0; i < quanForDay.size(); i++){
                            average += quanForDay.get(i);
                        }

                        Float forecastNumber =
Float.parseFloat(editTextForecast.getText().toString());
                        Float forecastResult =
forecastNumber*(average/quanForDay.size());
                        tVForecast.setText(counter + ".   " + drugName1 + " :     "
+ forecastResult);

                        Float newQuantity = forecastResult +
quantity.get(quantity.size() - 1);

                        //Запись в БД проноза
                        ContentValues contentValues = new ContentValues();
                        contentValues.put(dbHealper.KEY_DRUG_NAME, drugName1);
                        contentValues.put(dbHealper.KEY_QUANTITY, newQuantity);
                        contentValues.put(dbHealper.KEY_DATE, date);
                        database.insert(dbHealper.TABLE_STORAGE, null,
contentValues);
                        quanForDay.clear();

                        //if (id.equals("")){
                        //}
                        //else {
                        //    database.delete(DBHealper.TABLE_STORAGE,
DBHealper.KEY_ID + "=" + id, null);
                        //}

                    }

                    quantity.clear();
                    dates.clear();
                    counter++;
                    linearLayoutForecast.addView(tVForecast);
                }

            }while (cursor1.moveToNext());
            names.clear();

        }

        cursor1.close();
        cursor2.close();

        TextView textView = new TextView(ForecastActivity.this);
        textView.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 20);
        textView.setText("Увага, якщо для елементу складу не було здійснено
прогнозу, то даних про його використання " +
                "було недостатньо. Мінімум - 4 записи! ");
```

```
        textView.setTextColor(Color.parseColor("#212121"));
        linearLayoutForecast.addView(textView);
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="409dp"
        android:layout_height="134dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/buttonCalendar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Вибір дати"
            android:textColor="@color/icons"/>

        <TextView
            android:id="@+id/textDate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="20dp"
            android:textColor="@color/primary_text"
            android:textSize="18dp" />

    </LinearLayout>

    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="335dp"
        android:layout_height="360dp"
        app:layout_constraintBottom_toTopOf="@+id/buttonAddPatients"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout2"
        app:layout_constraintVertical_bias="0.386">

        <LinearLayout
            android:id="@+id/linearLayout"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical" />
    </ScrollView>
```

```
    <Button
        android:id="@+id/buttonAddPatients"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:text="Додати пацієнтів"
        android:textColor="@color/icons"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_add_patients.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddPatientsActivity">

    <LinearLayout
        android:layout_width="416dp"
        android:layout_height="588dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="15dp"
            android:text="Введіть інформацію про пацієнта:"
            android:textColor="@color/primary_text"
            android:textSize="18sp" />

        <EditText
            android:id="@+id/editTextSurname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="Прізвище"
            android:inputType="text" />

        <EditText
            android:id="@+id/editTextName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="Ім'я"
            android:inputType="text" />
```

```
        <EditText
            android:id="@+id/editTextPatronymic"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="По батькові"
            android:inputType="text" />

        <EditText
            android:id="@+id/editTextAge"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="Вік"
            android:inputType="number" />

        <EditText
            android:id="@+id/editTextTelephone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="Номер телефону"
            android:inputType="number" />

        <EditText
            android:id="@+id/editTextDiagnosis"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:gravity="top"
            android:hint="Діагноз"
            android:inputType="textMultiLine" />

        <Button
            android:id="@+id/buttonSave"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="15dp"
            android:gravity="center"
            android:text="Зберегти"
            android:textColor="@color/icons"/>
        <!--
            <Button
                android:id="@+id/buttonClear"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Очистить БД"
                android:textColor="@color/icons"/>
        -->

    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_search.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SearchActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Введіть ПІБ або номер телефону пацієнта:"
        android:textColor="@color/primary_text"
        android:textSize="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editTextSearch"
        android:layout_width="284dp"
        android:layout_height="44dp"
        android:layout_marginStart="28dp"
        android:layout_marginTop="32dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="Ввведіть дані"
        android:textColor="@color/primary_text"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="52dp"
        android:layout_height="43dp"
        android:layout_marginTop="32dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.508"
        app:layout_constraintStart_toEndOf="@+id/editTextSearch"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
        app:srcCompat="@android:drawable/ic_menu_search" />

    <Button
        android:id="@+id/buttonSearch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Пошук"
        android:textColor="@color/icons"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextSearch" />

    <ScrollView
        android:layout_width="340dp"
```

```
        android:layout_height="356dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonSearch"
        app:layout_constraintVertical_bias="0.467">

        <LinearLayout
            android:id="@+id/linearLayoutSearch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical" />
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_storage.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".StorageActivity">

    <TextView
        android:id="@+id/textViewStorage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="СКЛАД"
        android:textColor="@color/primary_text"
        android:textSize="20dp"
        android:layout_margin="10dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/linearLayout3"
        android:layout_width="385dp"
        android:layout_height="34dp"
        android:orientation="horizontal"
        android:layout_margin="10dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.64"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textViewStorage">

        <TextView
            android:id="@+id/textViewStorName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Назва:"
            android:textColor="@color/primary_text"
            android:textSize="20dp" />

        <TextView
```

```
            android:id="@+id/textViewStorQuan"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Кількість:"
            android:textColor="@color/primary_text"
            android:textSize="20dp" />
    </LinearLayout>

    <ScrollView
        android:id="@+id/scrollView3"
        android:layout_width="370dp"
        android:layout_height="382dp"
        app:layout_constraintBottom_toTopOf="@+id/buttonAddItem"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.515"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout3">

        <LinearLayout
            android:id="@+id/linearLayoutOutput"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" />
    </ScrollView>

    <Button
        android:id="@+id/buttonAddItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Додати/Оновити"
        android:textColor="@color/icons"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_add_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddItemActivity">

    <LinearLayout
        android:layout_width="416dp"
        android:layout_height="588dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
```

```
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="15dp"
            android:text="Введіть інформацію для додавання чи оновлення інформації
про ліки:"
            android:textColor="@color/primary_text"
            android:textSize="18sp" />

        <EditText
            android:id="@+id/editTextNameItem"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="Найменування"
            android:inputType="text" />

        <EditText
            android:id="@+id/editTextQuantity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:ems="10"
            android:hint="Кількість"
            android:inputType="numberDecimal" />


        <Button
            android:id="@+id/buttonSave2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="15dp"
            android:gravity="center"
            android:text="Зберегти"
            android:textColor="@color/icons" />
        <!--
            <Button
                android:id="@+id/buttonClear2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Очистить БД"
                android:textColor="@color/icons"/>
        -->

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_forecast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ForecastActivity">
```

```
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Введіть кількіть днів, на яку необхіден прогноз: "
        android:textColor="@color/primary_text"
        android:textSize="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editTextForecast"
        android:layout_width="284dp"
        android:layout_height="44dp"
        android:layout_marginTop="40dp"
        android:ems="10"
        android:hint="Введіть кількість днів"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <Button
        android:id="@+id/buttonForecast"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="Спрогнозувати"
        android:textColor="@color/icons"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextForecast" />

    <ScrollView
        android:layout_width="352dp"
        android:layout_height="364dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonForecast"
        app:layout_constraintVertical_bias="0.467">

        <LinearLayout
            android:id="@+id/linearLayoutForecast"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical" />
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

main_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
```

```
        android:id="@+id/menu_home"
        android:orderInCategory="1"
        android:title="@string/menu_home" />
    <item
        android:id="@+id/menu_storage"
        android:orderInCategory="2"
        android:title="Склад" />
    <item
        android:id="@+id/menu_forcast"
        android:orderInCategory="3"
        android:title="Прогноз закупки" />
    <item
        android:id="@+id/menu_search"
        android:orderInCategory="4"
        android:title="Пошук пацієнтів" />

</menu>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.graduatework1">

    <application
        android:allowBackup="true"
        android:icon="@drawable/myicon"
        android:label="Дипломний проект"
        android:roundIcon="@drawable/myicon"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".ForecastActivity"
            android:windowSoftInputMode="stateHidden|adjustPan" />
        <activity
            android:name=".SearchActivity"
            android:windowSoftInputMode="stateHidden|adjustPan"/>
        <activity android:name=".AddItemActivity" />
        <activity android:name=".StorageActivity" />
        <activity android:name=".AddPatientsActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```