МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Мобільний додаток процесу інвентаризації матеріальних активів»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-91 Лебідь Дмитро Олександрович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» лютого 2021 р.

Науковий керівник _____ к.т.н., Бойко О.В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«___» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Лебідь Дмитро Олександрович
(прізвище, ім'я, по батькові)

1 Тема проекту *Мобільний додаток процесу інвентаризації матеріальних активів*

затверджена наказом по університету від «___» листопада 2020 р. № 1824-III

2 Термін здачі студентом закінченого проекту « 25 » січня 2021 р.

3 Вхідні дані до проєкту методичні вказівки, функціональні та нефункціональні вимоги, технічне завдання для розробки мобільного додатку процесу інвентаризації матеріальних активів.

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, проектування мобільного додатку процесу інвентаризації матеріальних активів, розробка мобільного додатку процесу інвентаризації матеріальних активів.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, мета кваліфікаційної роботи магістра, постановка задачі кваліфікаційної роботи магістра, аналіз існуючих мобільних додатків за схожою тематикою, структурно-функціональне моделювання мобільного додатку, моделювання варіантів використання, засоби реалізації, приклади реалізації, інструкція користувача, висновки**.**

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проєкту	Термін виконання етапів проекту	Примітка
1	Визначення мети проєкту	16.11.20 – 16.11.20	
2	Визначення аналогів	17.11.20 – 19.11.20	
3	Виявлення потреби у додатку	20.11.20 – 23.11.20	
4	Визначення вимог	24.11.20 – 26.11.20	
5	Вибір інструментарію для реалізації	27.11.20 – 30.11.20	
6	Створення WBS	01.12.20 – 02.12.20	
7	Створення OBS	01.12.20 – 02.12.20	
8	Розрахунок бюджету	07.12.20 – 09.12.20	
9	Визначення ризиків	10.12.20 – 11.12.20	
10	Розробка мобільного додатку	14.12.20 – 12.01.21	
11	Створення сховища даних	14.12.20 – 18.12.20	
12	Розробка інтерфейсу та написання kotlin скрипту	23.12.20 – 14.01.20	

13	Тестування та перевірка помилок	15.01.21 – 20.01.21	
14	Виправлення багів та зауважень	21.01.21 – 22.01.21	
15	Оформлення пояснювальної записки	17.11.20 – 15.01.21	
16	Створення настанови з використання	16.01.21 – 18.01.21	
17	Архівація	23.01.21 – 25.01.21	

Магістрант _____ Лебідь Д.О.

Керівник роботи _____ к.т.н., Бойко О.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Мобільний додаток процесу інвентаризації матеріальних активів».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 27 найменувань, додатків. Загальний обсяг роботи − 70 сторінок, у тому числі 51 сторінок основного тексту, 4 сторінки списку використаних джерел, 16 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці мобільного додатку процесу інвентаризації матеріальних активів. В роботі проаналізовано предметну область, визначені задачі, методи дослідження та потреби у розробці додатку, складений календарний план, визначені ризики та бюджет. У роботі було спроектовано мобільний додаток відповідно до теми з точки зору розробника та процес використання додатку, побудовано діаграму варіантів використання, а також структурно-функціональне моделювання. Результатом проведеної роботи є розроблений мобільний додаток процесу інвентаризації матеріальних активів. Практичне завдання роботи полягає у створенні сховища даних, написання відповідного скрипту, підключення додаткових сервісів для роботи з QR- кодом, тестування функціонування додатку згідно до поставлених вимог.

Ключові слова: мобільний додаток, інвентаризація, товарообіг, моделювання, проєктування, розробка, аналіз.

ЗМІСТ

ВСТУП

Швидкий розвиток товарообігу спричиняє складності в процесах контролю відстеження та менеджменту інвентаризації товарів. Для більшості підприємств стає важливим наявність інформаційної системи, яка дозволяє швидко відскановувати тег товару, щоб перевірити його наявність в системі матеріальних активів компанії.

Інвентаризація дозволяє встановити фактичну наявність активів шляхом перерахунку залишків та перевірки навчальних записів. І є обов'язковою процедури перед складанням річної звітності та забезпечує перевірку достовірності показників бухгалтерського балансу.

З іншого боку за останні кілька років розробка мобільних додатків стала індустрією, яка розвивається дуже стрімко. В даний час, за підрахунками, існує 2,3 мільйона розробників мобільних додатків, які прагнуть задовольнити попит різних галузей.

За даними Apple, у 2013 році в магазині програм Apple було зареєстровано 1,25 мільйона додатків, на які припадало 50 мільярдів завантажень та 5 мільярдів доларів, виплачених розробникам. Завдяки цим типам галузевих цифр незабаром стає зрозуміло, що розробка мобільних додатків є ключовим фактором успіху бізнесу.

Виходячи з вище зазначеного, ідеальним рішенням було б поєднати розвиваючу галузь - розробка мобільного додатку з проблематикою сьогодення - попит на інвентаризацію товарів.

Користувач такого програмного застосунку матиме можливість оперувати документами інвентарного опису на мобільному пристрої, а також проводити процедуру інвентаризації за допомогою зчитування QR коду.

Таким чином можна визначити мету роботи, яка полягає у створенні мобільного додатку для ведення інвентарного опису товарно-матеріальних активів підприємства.

Для досягнення поставленої мети на даному етапі необхідно виконати наступні задачі:

- провести аналіз вимог до мобільного додатку;
- дослідити аналоги додатку для ведення інвентаризації;
- визначити можливості інвентаризації товарів за допомогою QR коду;
- визначити методи дослідження для окремих задач, що виникають під час виконання кваліфікаційної роботи.
- реалізувати мобільний додаток з можливістю зчитування QR кодів;
- провести тестування мобільного додатку;
- забезпечити можливість встановлення мобільного додатку;
- визначити методи дослідження для окремих задач, що виникають під час виконання дипломної роботи;

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Роль мобільного додатку у повсякденному житті та бізнесі

Значимість смартфонів у нашому повсякденному житті та суспільної діяльності, безперечно є нескінченною. Це тому, що відбувається величезна трансформація в тому, що гаджети вже не є звичайним комунікаційним пристроєм, яким він був раніше. Розробка мобільних додатків - це техніка розробки програмного забезпечення для мобільних пристроїв, і з неї випливає основна фундаментальна концепція. Головне, що розробка різних мобільних додатків, які працюватимуть на мобільній платформі, в першу чергу називається розробкою мобільних додатків.

Переваги розробки мобільних додатків у бізнес-аспектах:

- ✓ Маркетинг на ходу: Пропозиція мобільних додатків Маркетинг на ходу полягає у тому що, клієнти можуть отримати доступ до вашого бізнесу де завгодно та в будь-який час. Регулярне використання вашого мобільного додатку сприяє покращенню бренду або бізнесу. Регулярне використання вашого додатку зміцнює бренд, коли потрібно щось придбати.
- ✓ Майбутня тенденція маркетингу: Мобільний додаток скоро стане маркетинговою тенденцією, за підрахунками, у світі використовують понад 1 мільярд смарт-гаджетів, а понад 50% пристроїв підключаються до Інтернету. Що означає, що незабаром запити пошукових систем надходять із розумних пристроїв на відміну від персонального комп’ютера.
- ✓ Збільшення продажів: бізнес-засоби для отримання продажів і доходу, додаток, що допомагає продавати продукцію, а також допомагає залучати нових потенційних клієнтів для розвитку бізнесу. Окрім продажів, мобільний додаток також сприяє підвищенню поінформованості про бренд.

✓ Соціальна платформа: бізнес стає соціальним завдяки соціальним мережам, а люди охоплені саме цими мережами. Мобільні програми дозволяють додавати інструменти соціальних медіа, які дозволяють користувачеві сподобатися, поділитися чи коментувати продукти. Тому ми повинні використовувати ці платформи як бізнес-стратегію, щоб покращити зв'язок із користувачами.

✓ Покращені продажі та обслуговування: Мобільні програми покращують продаж та послуги для бізнесу. Це дозволяє замовнику замовляти та купувати товар з будь-якого місця та в будь-який час. Найкращого способу обслуговування клієнтів вдалося досягти за допомогою мобільних додатків.

Сьогодні доступність мобільних додатків зростає настільки, що це спричиняє помітну зміну в тому, як люди відчувають і відчувають обчислювальні роботи. Кілька років тому в інших випадках для доступу до Інтернету, перевірки та читання електронної пошти доводилося користуватися комп'ютером, але сьогодні це змінилося, оскільки обчислення зараз використовуються скрізь у мобільних телефонах. Наприклад, купівля квитка на поїзд на ходу, це те, що наші предки ніколи не уявляли і не робили. Або ж наприклад, що ви не ходите в банк, але все одно переказуєте гроші родичам та друзям. Все завдяки розробникам додатків та провідним компаніям, що займаються розробкою додатків.

Очікується, що в майбутньому більшість зусиль з розробки мобільних додатків будуть зосереджені на створенні браузерних додатків, які є агностичними щодо пристроїв. Додатки на основі браузера - це просто веб-сайти, створені для мобільних браузерів. Такі сайти створені для швидкого завантаження через стільникову мережу та мають зручну навігацію.

Розробка мобільних додатків - це сукупність процесів та процедур, пов'язаних із написанням програмного забезпечення для невеликих бездротових обчислювальних пристроїв, таких як смартфони чи планшети.

1.2 Визначення загальних функцій мобільного додатка

Щоб зробити додаток привабливим для користувачів, дизайн повинен включати деякі ключові функції. Споживачі реагують на інтуїтивно зрозумілий дизайн, що робить їх досвід легшим, а не складнішим.

✓ Простота: простота - це основа будь-якого додатка. Це створює конкурентну перевагу під час перебування на борту користувачів. Використання програми має бути безперебійним досвідом, а не випробуванням для користувачів.

Якщо додаток занадто складний, користувачі звичайно ж видалять його та перейдуть до іншого, найбільш вдалого варіанту. Слід намагатися полегшити користувачеві досвід, а простота є ключовим фактором створення кращої програми.

✓ Час швидкого завантаження: Час завантаження - це ще одна ключова функція програми, яка може визначити, зберігають чи видаляють користувачі програму.

На завантаження програми не більше ніж 2 секунди, інакше існує ризик втратити користувача. Люди не хочуть чекати послуги, яку їм пропонують, особливо коли є інші програми, схожі на цю.

✓ Інтеграція соціальних медіа: Соціальні мережі стали інструментом спілкування та співпраці.

Компанії повинні скористатися цим, пропонуючи користувачам додатків можливість ділитися вмістом, який вони щойно прочитали, або предметом, який вони щойно придбали, зі своїми інтернет-спільнотами. Перетворити обмін на простий і цікавий спосіб для своїх клієнтів, а продавати товар - чудова рекламна тактика.

✓ Платежі через додаток: інтеграція рішень платіжного шлюзу є важливою для кожного додатка. Будь-який успішний мобільний додаток повинен

інтегрувати платіжні сервіси, такі як PayPal, щоб створити безпечний і швидкий процес оформлення замовлення.

Це запевняє ваших користувачів у законності додатку та створює простіший спосіб придбати товар.

✓ Відгуки користувачів: прохання про відгук є важливою функцією, яку слід не включати зі свого додатку. Якщо дозволити клієнтам висловлювати свою думку щодо наданих послуг, користувач почуватиметься цінованим та включеним до процесу оновлення.

1.3 Різновид мобільних додатків та їх властивості

Мобільні додатки розробляються дорожче, ніж веб-програми, і оскільки вони специфічні для певної платформи, запуск програми на різних платформах майже означає, що починати з нуля з точки зору дизайну та розробки. Однак вони набагато швидші і, як правило, більш досконалі з точки зору функцій та функціональних можливостей.

Як створюються рідні мобільні програми:

Власні мобільні програми - це вбудовані мови, специфічні для платформи. Рідні розробники додатків використовують Swift або Objective-C для додатків iOS, Java або C ++ для додатків Android та C # для додатків Windows Phone. Так, це означає, що якщо є потреба, щоб ваш власний додаток був доступний для завантаження на пристроях Android та iOS, потрібно буде створити кілька версій, які, швидше за все, не матимуть однакових користувацьких інтерфейсів.

Як створюються гібридні мобільні програми:

Гібридні програми поєднують в собі найкраще: як веб-програми, так і власні. Технічно свого роду мобільний додаток, гібридний додаток встановлюється як власний додаток, але при запуску, він функціонує як веб-додаток за допомогою WebView платформи. (WebView - це свого роду міні-веб-браузер, який можна налаштувати на повноекранний режим.)

Гібридні програми також створені спеціально для операційної системи, і тому вони також можуть отримувати доступ до можливостей пристрою, як власний додаток. Однак, як і веб-програми, гібридні програми пишуться у форматі HTML, CSS та JavaScript, але потім упаковуються для різних платформ.

Плюси мобільних програм:

- ✓ Швидше, ніж веб-програми;

✓ Більша функціональність, оскільки вони мають доступ до системних ресурсів;

✓ Може працювати в автономному режимі;

✓ Надійність та безпека - власні програми мають бути попередньо схвалені магазином програм;

✓ Простіше створювати завдяки наявності інструментів розробника, елементів інтерфейсу та SDK;

Мінуси мобільних програм:

✓ Дорожче розроблювати, ніж веб-програми;

✓ Сумісність з різними платформами (тобто iOS та Android), як правило, означає розробку та створення програми з нуля;

✓ Дорого в обслуговуванні та оновленні;

✓ Отримати власну програму, схвалену магазином програм, може виявитися важко;

1.4 Інвентаризація товарів за допомогою QR коду у мобільних додатках

Інвентаризація товару, який також називають контролем запасів, - це процес забезпечення належної кількості поставок в організації. Завдяки відповідному внутрішньому та виробничому контролю, практика забезпечує, що компанія може задовольнити попит споживачів та забезпечити фінансову еластичність.

Для успішного контролю запасів потрібні дані про закупівлі, замовлення, доставку, складування, зберігання, отримання, задоволення споживачів, запобігання збиткам і товарообігу. Згідно зі звітністю про стан малого бізнесу за минулі роки, майже половина малих підприємств не відстежує свої запаси, навіть вручну.

Облік товарів забезпечує максимальну суму прибутку від найменшої суми інвестицій в запаси, не впливаючи на задоволеність споживачів. Якщо прискіпливо та правильно проводити облік, це дозволить компаніям оцінювати свій поточний стан щодо активів, залишків на рахунках та фінансових звітів. Контроль запасів може допомогти уникнути проблем, таких як нестачі запасів. Наприклад, Walmart підрахував, що пропустив продаж у 2014 році на суму 3 млрд доларів, оскільки його неадекватні процедури контролю запасів призвели до такого прірви.

Невід’ємною частиною контролю запасів є управління ланцюгами поставок, яке управляє потоком сировини, товарів та послуг до того моменту, коли компанія або клієнти споживають товари. Управління складами також прямо потрапляє на арену контролю запасів. Цей процес включає інтеграцію кодування товару, переупорядкування точок та звітів, усіх деталей товару, списків та підрахунків товарних запасів та методів продажу чи зберігання.

Управління запасами - це термін вищого рівня, який охоплює повний процес закупівлі, зберігання та отримання прибутку від продукції чи послуг. Хоча контроль запасів та управління запасами можуть здатися

взаємозамінними, вони не є такими. Контроль запасів регулює те, що вже є на складі. Управління запасами є ширшим і регулює все: від того, що є на складі, до того, як бізнес доставляє запаси туди та кінцеве призначення товару.

Застосування QR-коду:

QR-код - це двовимірна версія штрих-коду, яку можна читати як горизонтально, так і вертикально. Це може бути пов'язано з широкою різноманітністю інформації, яка може бути використана для посилення маркетингових кампаній, використовується для внутрішніх бізнес-процесів, а також корисна в приватному житті. QR-коди складаються з семи основних партнерів і можуть містити до 4296 символів. На основі складної матричної системи із використанням символів їх найчастіше читають за допомогою програми для сканування QR-коду, але у повсякденному житті навіть гаджети Android чи то Apple також дозволяють сканувати QR-код безпосередньо через камеру.

QR-коди були розроблені для інвентаризації:

Початковою версією QR-коду була одновимірна версія, відома як штрих-код, яку можна було прочитати лише горизонтально. Потреба в таких технологіях виникла в Японії в 1960-х роках, коли хвиля споживацтва вразила супермаркети та роздрібні магазини, яким потрібно було накопичувати більший асортимент товарів в окремих магазинах, щоб задовольнити потреби ринку. До існування штрих-кодів касири дзвонили вручну, що надзвичайно сповільнювало і здавалось дуже кропітким завданням. Коли касири почали розвивати проблеми, пов'язані зі здоров'ям, керівникам супермаркетів потрібно було знайти краще рішення. Хоча штрих-коди деякий час працювали, вони все ще не відповідали потребам, які вимагає управління масовими запасами товарів.

Винахідник Масахіро Хара та інший член команди придумали ідею QR-кодів завдяки їхній квадратній формі. Ця фігура була ключем, який дозволяв QR-кодам містити набагато більше інформації, ніж штрих-коди.

Потреби в управлінні запасами сьогодні значно перевершили те, що вимагалося раніше, тому штрих-кодів просто вже недостатньо. Завдяки складним глобальним виробничим та логістичним системам, QR-коди забезпечують можливість зберігати набагато вичерпнішу інформацію, щоб ці системи могли нормально функціонувати.

QR-код має квадратну форму, тоді як штрих-код більше схожий на прямокутник. Як зазначалося вище, форма одновимірних штрих-кодів означає, що їх можна читати лише горизонтально, обмежуючи інформацію, яку вони можуть мати. Але ще одним недоліком такої форми є те, що для їх зчитування потрібен спеціальний скануючий пристрій, що може бути значним витратами для компаній з великими запасами. Традиційні скануючі пристрої також потрібно підключити до комп’ютера або принаймні мати один поруч. Візьмемо, наприклад, Parts Brite. Управління товарно-матеріальними запасами часто відбувається на зайнятому складі завантажувальних доків і не зручно мати комп’ютер посеред усього цього. Використовувати смартфони для сканування QR-кодів набагато простіше, тому що кожен і так тримає свій телефон у кишені.

1.5 Аналіз існуючих мобільних додатків за схожою тематикою

Перш за все, щоб втілити власну бізнес ідею у реалії та перейти до розробки необхідно спочатку виконати дослідження серед конкурентів. Для пошуку схожих мобільних додатків було використано мережу Інтернет. Завдяки аналізу можна нівелювати додаткові витрати на тестування власного продукту, тож оптимальним рішенням є використання додатків у відкритому доступі для отримання майбутнього розуміння ключових моментів у власному додатку та їх інтеграції.

Мобільні додатки, які використовувалися для аналізу:

- ✓ Barcode Harvester;

Одним із головних переваг даного додатку є те, що він надає змогу сканувати не тільки QR-код, але й застарілі зразки звичайних штрих кодів, а також дуже простий формат відображення інформації товарів у вигляді Excel таблиць.

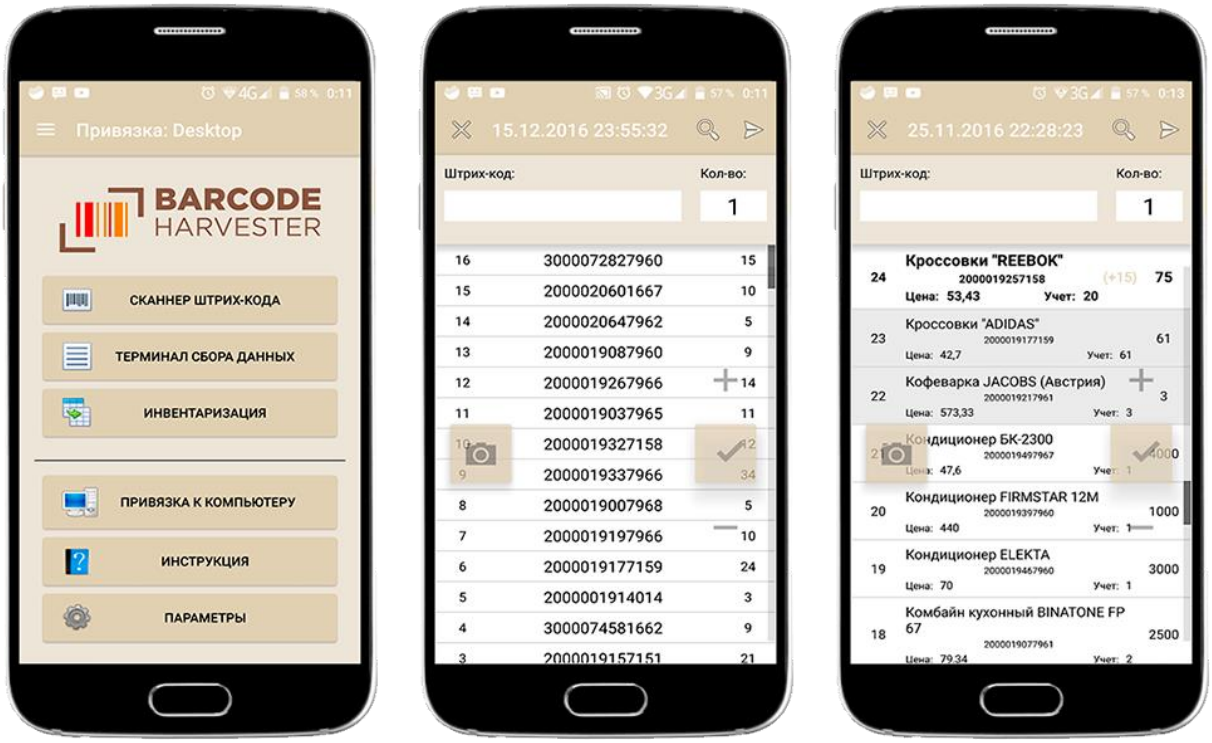

Рисунок 1.1 - «Barcode Harvester»

- ✓ Швидкий облік для Android;

Помічник при проведенні інвентаризації та управлінні запасами. Цей додаток пропонує простий і зручний функціонал, оптимізований для роботи на смартфонах і планшетах. Підтримка зовнішніх Bluetooth-сканерів забезпечує можливість його застосування також в сферах промисловості і бізнесу.

Рисунок 1.2 - «Швидкий облік для Android»

У ході проведеного аналізу також були виявлені такі переваги створення майбутнього проекту і вони полягають у наступному:

- ✓ Швидке сканування продуктів;
- ✓ Економічна ефективність;
- ✓ Простота у використанні;
- ✓ Мінімальні помилки;
- ✓ Швидка автоматизація;
- ✓ Просте усунення несправностей;
- ✓ Підвищена відмовостійкість;
- ✓ Покращені можливості зберігання даних;

Вдалий мобільний додаток повинен мати більшість з цих критеріїв для організації швидкої та необхідної роботи щоб задовольнити користувача.

Деякі системи управління запасами вимагають багато введення даних, зміни чи видалення записів про активи вручну. Навіть якщо вони впорядковані,

для отримання ефективного управління програмним забезпеченням потрібно мати багато технічних знань та навичок.

QR-коди допомагають автоматизувати операції для досягнення максимальної ефективності з мінімальними помилками. Ця миттєва передача інформації, яка зменшить кількість недоліків в інвентарних записах.

Інтеграція QR-кодів для управління запасами - це вірний спосіб оптимізувати та сприяти безперебійній транзакції системи управління запасами та забезпечує прямий доступ до відстеження, зберігання та впорядкування активів. Саме через ці переваги було обрано застосовувати QR під час розробки додатку.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі

Поєднання змоги вести зручний облік товарів та формувати відомість матеріальних активів максимально зручно та швидко на новій платформі з сучасними функціями формує практичну значущість створення мобільного додатку. З чого випливає **мета роботи**, яка полягає у створенні мобільного додатку для ведення інвентарного опису товарно-матеріальних активів підприємства.

Дана тема була обрана через ручну проблему інвентаризації товаро обліку та проблем, які можуть виникнути у ході такої інвентаризації.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- провести аналіз вимог до мобільного додатку;
- дослідити аналоги додатку для ведення інвентаризації;
- визначити можливості інвентаризації товарів за допомогою QR коду;
- визначити методи дослідження для окремих задач, що виникають під час виконання дипломної роботи.
- реалізувати мобільний додаток з можливістю читання QR кодів;
- провести тестування мобільного додатку;
- забезпечити можливість встановлення мобільного додатку.

2.2 Методи дослідження

При виконанні дипломної роботи також проведено аналіз методів дослідження, які можуть бути використані для виконання окремих робіт. Аналіз методів використовується при розробці нових продуктів або послуг та для підвищення ефективності методів, що використовуються в даний час. Якщо існує загальноприйнята процедура, то завдяки аналізу можна задокументувати її, включаючи конкретні позначення, що визначають уподобання споживача.

До основних методів, що використовуються у дипломній роботі відносяться:

- Метод аналізу для визначення аналогів, основних способів створення програмних додатків.
- Емпіричне дослідження для кількісного досліджень ринку при постановці актуальності дослідження.
- Метод моделювання для опису процесу реалізації мобільного застосунку.
- Системно-функціональне моделювання для відображення процесу роботи додатку.

3 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ПРОЦЕСУ ІНВЕНТАРИЗАЦІЇ МАТЕРІАЛЬНИХ АКТИВІВ

3.1 Структурно-функціональне моделювання

IDEF0- це група методів моделювання, які можна використовувати для опису операцій на будь-якому підприємстві чи компанії. Діаграма IDEF0, яка є частиною набору поведінкових уявлень (логічної архітектури), відображає багато контекстної інформації про взаємозв’язки декомпозиції без відображення фактичної керуючої логіки.

IDEF0 - це багато в чому простий метод. Кожне поле представляє окремий процес, як і в інших підходах, але IDEF0 відрізняється використанням та розміщенням стрілок. Окрім звичайних входів і виходів, існують ще два типи стрілок, які представляють "елементи керування" та "механізми".

Елементи управління є формою введення, але використовуються для керування діяльністю в процесі. Іноді виникає певна міра невизначеності щодо того, чи є елемент предметом введення чи контролю. Простий спосіб розрізнити їх полягає в тому, що входи перетворюються або змінюються якимось чином, щоб створити виходи, тоді як елементи управління дуже рідко змінюються. Стандарти, плани, шаблони та контрольні списки - це все форми контролю.

Механізми - це ресурси та інструменти, необхідні для завершення процесу. Сюди входять люди з певними навичками та інші інструменти.

Чотири типи стрілок, входи, елементи керування, виходи та механізми, спільно називаються ICOM, а IDEF - це абревіатура визначення ICOM.

Різні стрілки ICOM ідентифікуються збоку вікна активності, якого вони торкаються. Таким чином, входи знаходяться ліворуч, елементи управління - зверху, виходи - праворуч, а механізми - знизу. Це може ускладнити малювання діаграм, але спростити їх читання.

Моделювання розпочинається з контекстної діаграми, яка представляє об’єкт для усього проєкту (рис.3.1).

На вході маємо потребу та попит в розробці мобільного додатку з інвентаризації. На виході відповідно маємо функціонуючий мобільний додаток та документацію. Механізмами виступають розробник, керівник диплому, програмне та апаратне забезпечення. Управлінням у даному випадку є технічне завдання та методичні вказівки.

Наступним кроком є побудова першого рівня декомпозиції (рис. 3.2).

Аналіз предметної області включає в себе визначення ряду робіт, які необхідні для досягнення визначених цілей.

Етап створення мобільного додатку з інвентаризації декомпозується на 3 пункти(рис.3.3) та включає в себе створення та заповнення бази даних, підключення додаткових сервісів та написання kotlin скрипта.

Написання kotlin скрипта декомпозується на такі пункти як: створення алгоритму зчитування даних по QR-коду, зіставлення QR-коду з базою даних, налаштування виводу результату роботи алгоритму(рис.3.4).

Перевірка функціонування системи. Керівник диплому перевіряє розроблений мобільний додаток згідно відповідним вимогам.

Заключним етапом є створення звіту, розроблюється документація та опис робіт, яка використовувалася для досягнення поставленої цілі.

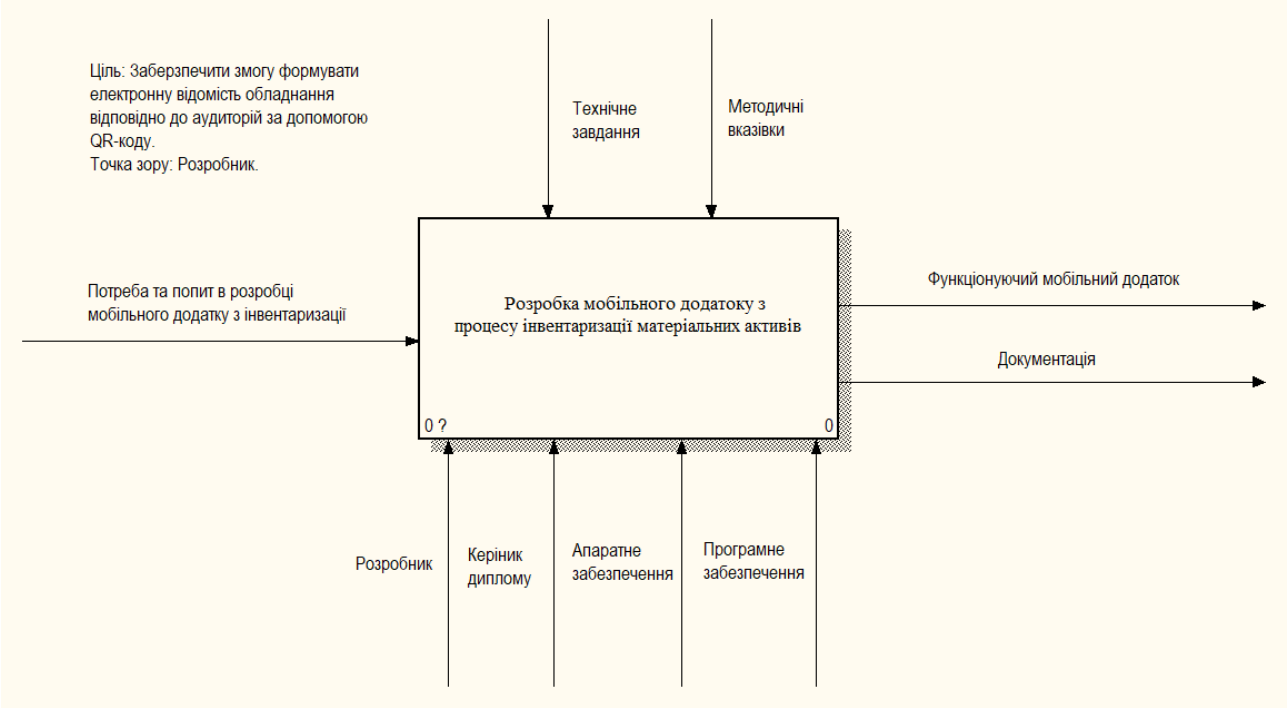

Рисунок 3.1 - Контекстна діаграма IDEF0

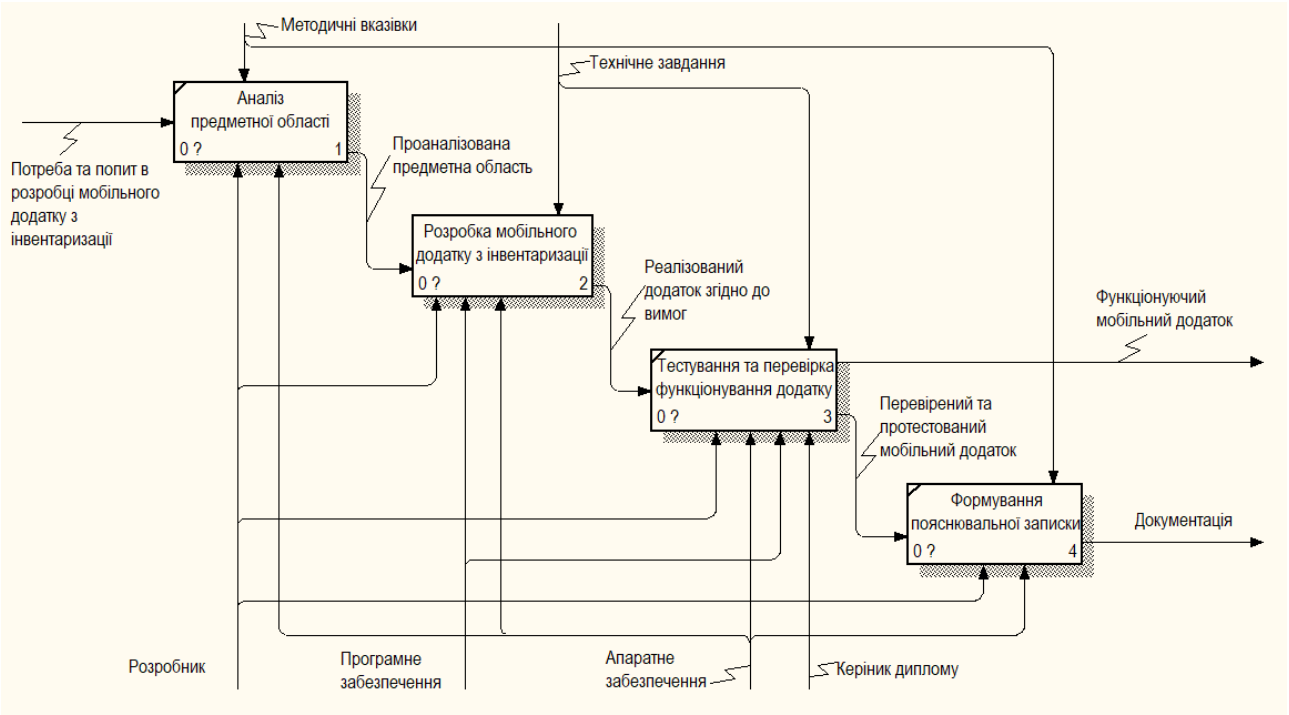

Рисунок 3.2 - Декомпозиція першого рівня

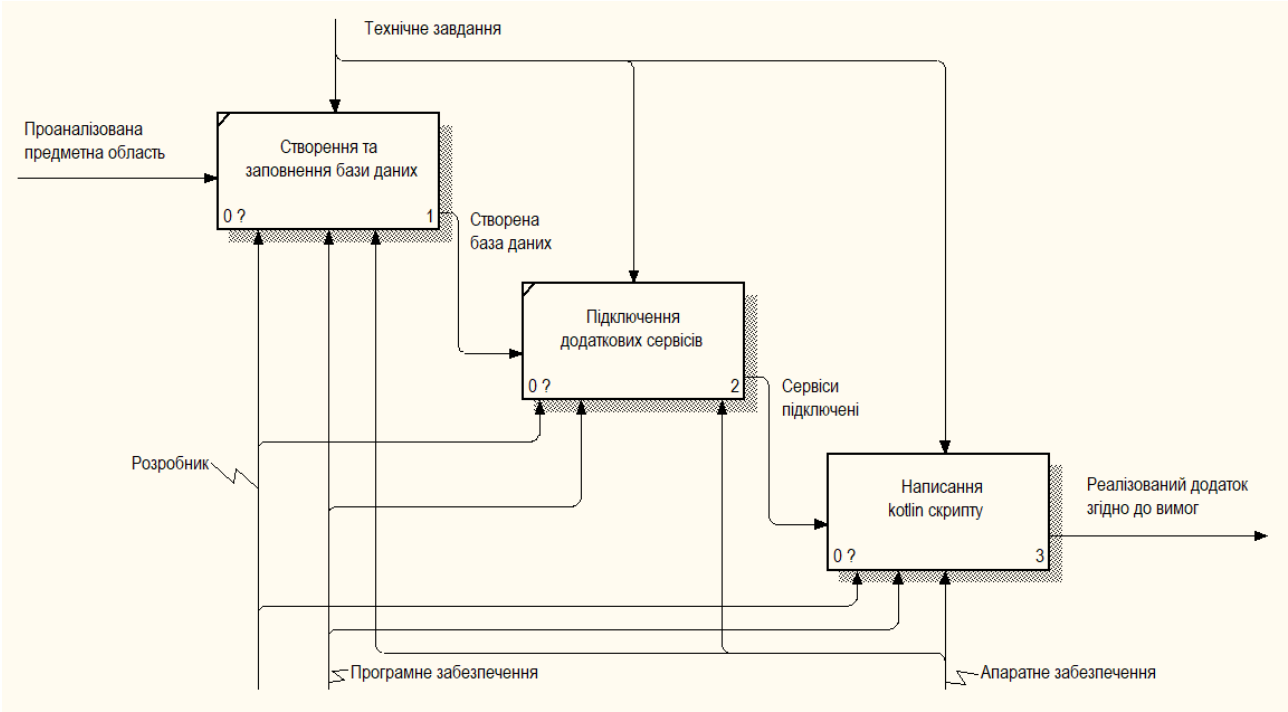

Рисунок 3.3 - Декомпозиція процесу Розробка мобільного додатку з інвентаризації

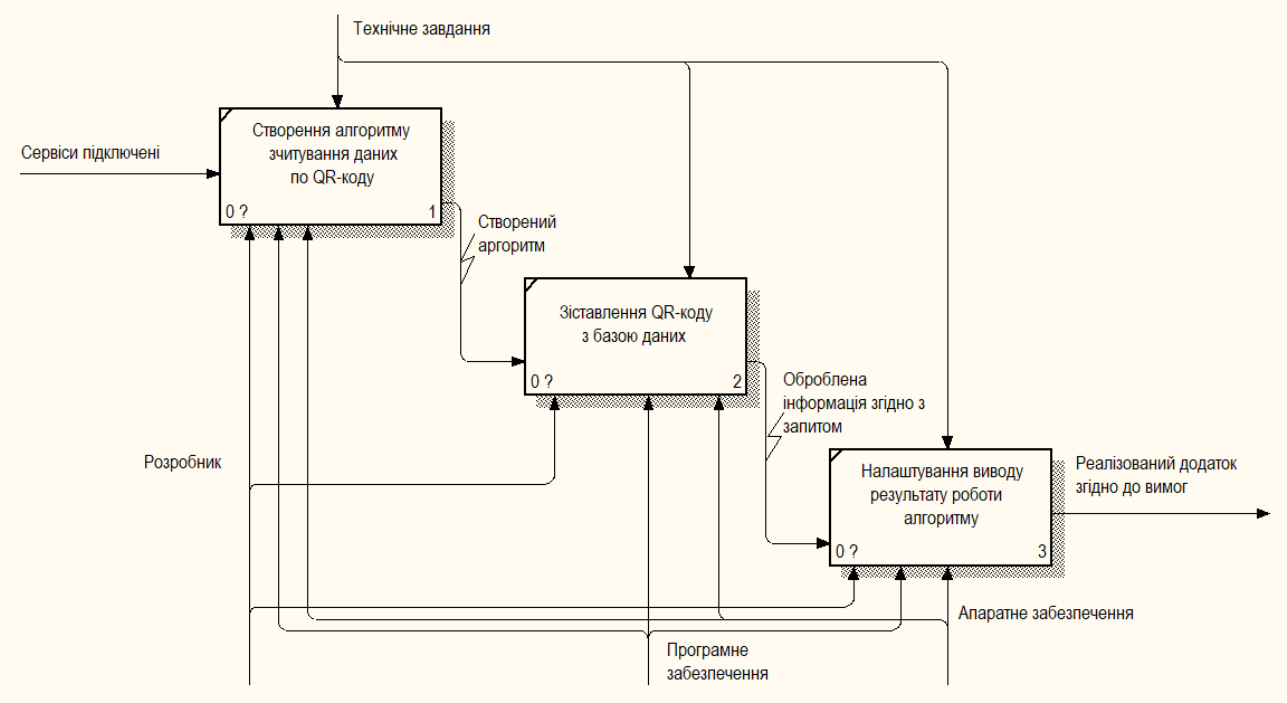

Рисунок 3.4 - Декомпозиція процесу Написання kotline скрипту

Опис блоків та стрілок наведено в таблицях 3.1-3.2.

Таблиця 3.1 – Опис блоків

Name (Activities)	**Definition**
Аналіз предметної області	Огляд ролі мобільних додатків у повсякденному житті, переваг розробки додатків, ідентифікація загальних функцій додатку, огляд різновиду мобільних додатків та їх властивостей, аналізування подібних додатків.
Формування пояснювальної записки	Звіт з виконання роботи/результат роботи
Розробка мобільного додатку з інвентаризації	Створення бази даних, підключення сервісів, написання kotlin скрипту.

Таблиця 3.2 – Опис стрілок

Name (Arrows)	**Definition**
Апаратне забезпечення	Необхідність використання персонального комп’ютера для розробки мобільного додатку
Програмне забезпечення	Використання програмного забезпечення та різних технологій для розробки
Документація	Написання звіту про виконану роботу
Проаналізована предметна область	Розглянуто роль мобільних додатків у повсякденному житті, переваги розробки додатків, визначення загальних функцій додатку, різновид мобільних додатків та їх властивості
Керівник диплому	Корегує процес написання роботи, перевіряє систему на відповідність вимогам
Розробник	Виконує розробку мобільного додатку, проводить аналіз предметної області та формує звіт з виконаної роботи

3.2 Створення моделі варіантів використання

Діаграма варіантів використання - це динамічна або поведінкова діаграма в UML. Діаграми варіантів використання моделюють функціональність системи за допомогою акторів та відповідних варіантів. Варіанти використання - це набір дій, послуг та функцій, які система повинна виконувати. У цьому контексті "система" - це щось, що розробляється або експлуатується, наприклад мобільний додаток. "Актори" - це люди або організації, що діють за певною роллю в системі.

Використання UML допомагає командам проектів спілкуватися, досліджувати потенційні проекти та перевіряти архітектурний дизайн програмного забезпечення. Переважно, UML використовується, як мова загального призначення для моделювання в галузі програмного забезпечення. Вони забезпечують як більш стандартизований спосіб моделювання робочих процесів, так і більш широкий спектр функцій для поліпшення читабельності та ефективності.

Діаграма варіантів використання або ж як її ще називають прецедентів-задають концептуальну модель таким чином:

- встановлюють кордони програмної системи;
- розширюють функціональну поведінку;
- дозволяє визначити функціональні вимоги.

Створимо таблицю, яка описує акторів на діаграмі.

Таблиця 3.3 - Актори діаграми

Назва актору	Опис актору
Розробник	Особа, що займається розробкою функціоналу згідно до вимог
Користувач	Особа, яка використовує мобільний додаток, вона здатна додавати нові аудиторії, робочі місця, обладнання, та користуватися вже існуючими за допомогою QR кодів
Керівник диплому	Особа, яка регулює процес написання роботи, вносить корективи та вказівки розробнику
База даних	Сховище даних , яке містить інформацію стосовно мобільного додатку, зберігає, додає чи оновлює інформацію і товари

Для відображення взаємодії між акторами з системою було побудовано діаграму варіантів використання(рис.3.5).

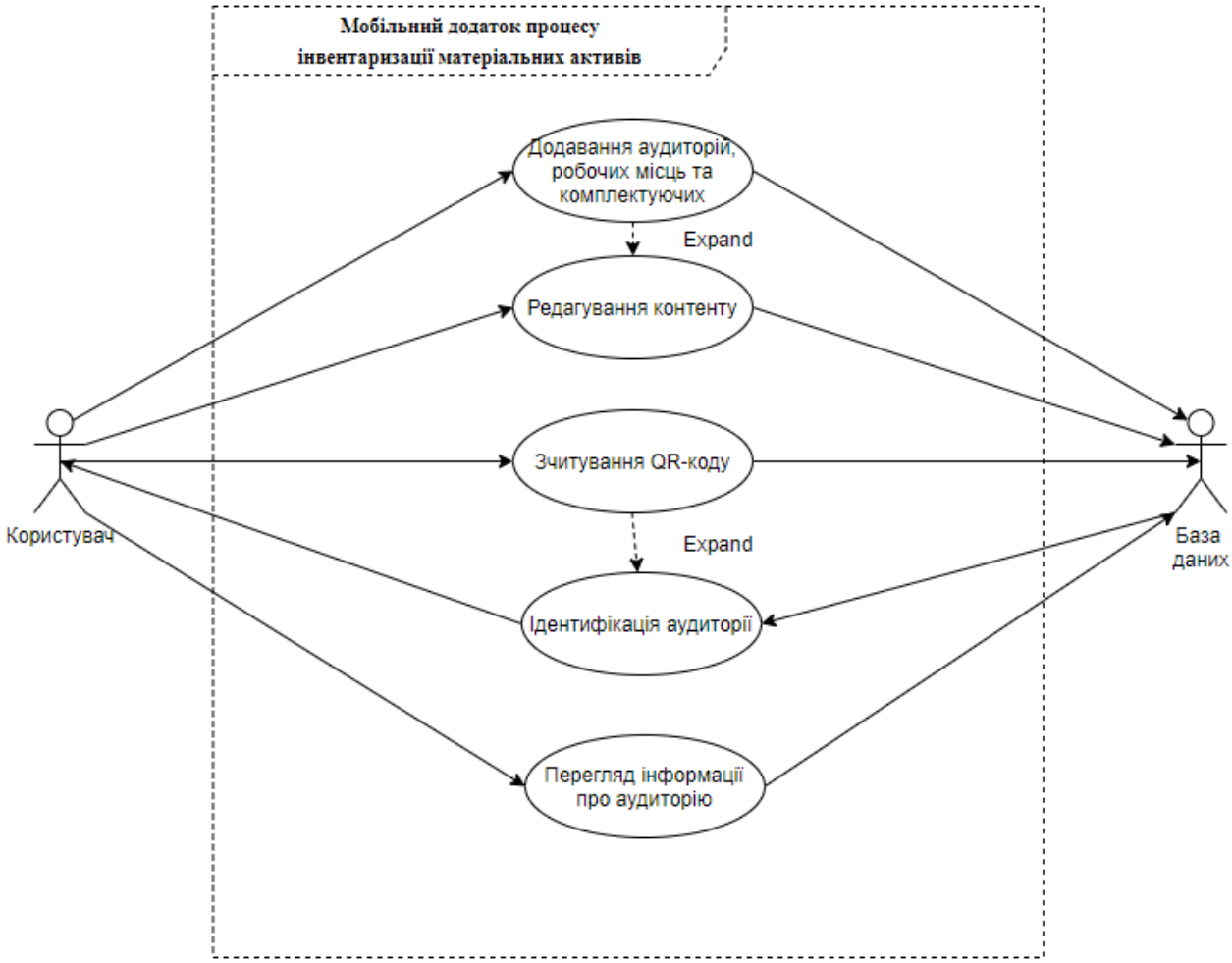

Рисунок 3.5 - Діаграма варіантів використання

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПРОЦЕСУ ІНВЕНТАРИЗАЦІЇ МАТЕРІАЛЬНИХ АКТИВІВ

4.1 Налаштування компонентів мобільного додатку

При розробці мобільного додатку було використано мову програмування kotlin. Для розпізнавання QR-коду було обрано сервіс Play Services Vision та для його генерації- ZXing.

Найбільш вагомі та основні процеси розробки наведені нижче. Вхід до мобільного додатку, отримання даних зі сховища(рис. 4.1).

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    DataHolder.obtainList( context: this)

    setListeners()


    addFragment(RoomListFragment(), addToBackStack: false)
}
```

Рисунок 4.1 - Реалізація входу у додаток

Наступним кроком є отримання доступу до камери, саме для того, щоб відкрити екран для зчитування QR-коду(рис.4.2).

```
private fun setListeners() {
    val qrButton: ImageView = findViewById(R.id.qrButton)
    qrButton.setOnClickListener { it: View!
        if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
            addFragment(BarcodeFragment())
        } else {
            requestPermissions(
                arrayOf(Manifest.permission.CAMERA),
                CAMERA_PERMISSION_REQUEST_CODE
            )
        }
    }
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    if (grantResults.isEmpty()) {
        return
    } else {
        if (grantResults.all { it == PackageManager.PERMISSION_GRANTED }) {
            if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
                addFragment(BarcodeFragment())
            }
        }
    }
}
```

Рисунок 4.2 - Доступи до камери

Далі наведений процес обробки отриманих результатів після зчитування QR- коду(рис.4.3).

```
private val onBarcodeRecognizedListener =
    object : BarcodeTrackerFactory.OnBarcodeRecognizedListener {
        override fun onBarcodeRecognized(barcode: Barcode) {
            detector?.setProcessor(null)

            val room = DataHolder.roomsList.find { it: Room
                it.id == barcode.rawValue
            }


            if (room != null) {
                processRoomFindAction(room)
            } else {
                val item = DataHolder.roomsList.flatMap { it: Room
                    it.workplaces
                }.flatMap { it: Workplace
                    it.items
                }.find { it: WorkplaceItem
                    it.id == barcode.rawValue
                }
                if (item != null) {
                    processWorkplaceItemFindAction(item)
                } else {
                    showBarcodeErrorDialog()
                }
            }
        }
    }
```

Рисунок 4.3 - Обробка результатів після зчитування QR- коду

На рисунку 4.4 відображено процес отримання та зберігання списку даних.

```
fun obtainList(context: Context){
    val sharedPreferences = context.getSharedPreferences(PREFERENCES_NAME, MODE_PRIVATE)
    val json = sharedPreferences.getString(DATA_JSON, "")
    val type = object : TypeToken<MutableList<Room>>(){}.type
    val l = Gson().fromJson<MutableList<Room>>(json, type)
    roomsList = l ?: mutableListOf()
}


fun saveList(context: Context) {
    val sharedPreferencesEditor = context.getSharedPreferences(PREFERENCES_NAME, MODE_PRIVATE).edit()
    sharedPreferencesEditor.putString(DATA_JSON, Gson().toJson(roomsList))
    sharedPreferencesEditor.apply()
}
```

Рисунок 4.4 - Зберігання списку даних

Підключення бібліотек для розпізнавання та відмальовування QR-коду(рис.4.5).

```
implementation 'com.google.android.gms:play-services-vision:20.1.3'
implementation 'com.google.zxing:core:3.2.1'
implementation 'com.journeyapps:zxing-android-embedded:3.2.0@aar'
```

Рисунок 4.5 - Підключення бібліотек

Наведений приклад структури даних, а саме аудиторій, робочих місць та їх компонентів(рис.4.6).

```
class Room {
    var id: String = ""
    var name: String = ""
    var workplaces: MutableList<Workplace> = mutableListOf()
}

class Workplace {
    var name: String = ""
    var id: String = ""
    var items: MutableList<WorkplaceItem> = mutableListOf()
    var roomName: String = ""
    var roomID: String = ""
}

class WorkplaceItem {
    var id: String = ""
    var name: String = ""
    var roomName: String = ""
    var roomID: String = ""
}
```

Рисунок 4.6 - Структура даних

На наступному скріншоті представлений кореневий каталог кваліфікаційної роботи магістра(рис.4.7).

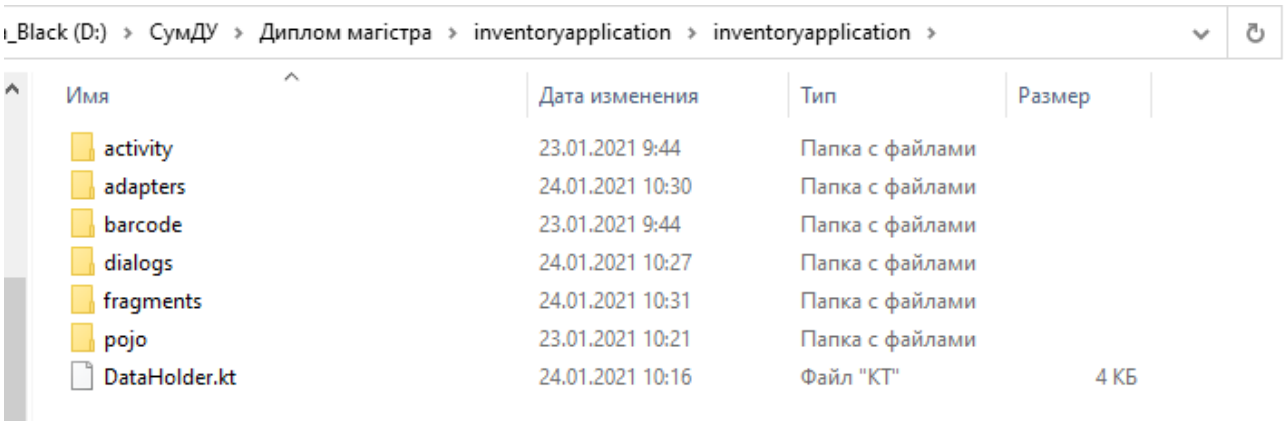

Рисунок 4.7 - Кореневий каталог

4.2 Результат реалізації мобільного додатку

На рисунку 4.8 представлено один з головних екранів Android смартфону, на якому виділено іконку мобільного додатку процесу інвентаризації матеріальних активів, який має назву “InventoryApplication”.

Рисунок 4.8 - Іконка додатку

Домашньою сторінкою мобільного додатку є екран з переліком кімнат, на якому користувач має змогу додавати нові аудиторії, редагувати та переглядати вже існуючі(рис.4.9-4.10).

Рисунок 4.9 - Екран аудиторій

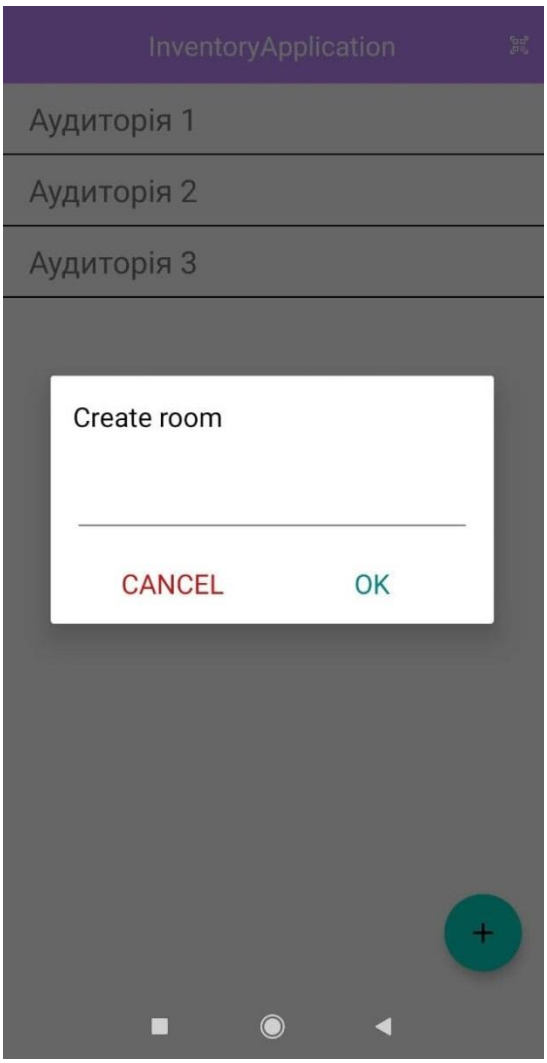

Рисунок 4.10 - Форма створення нової аудиторії

Після натискання на вже існуючу аудиторію, користувач має змогу переглянути кількість робочих місць, які закріплені за даною аудиторією(рис. 4.11). За необхідністю можна також додавати нові робочі місця(рис 4.12).

Рисунок 4.11 - Екран робочих місць

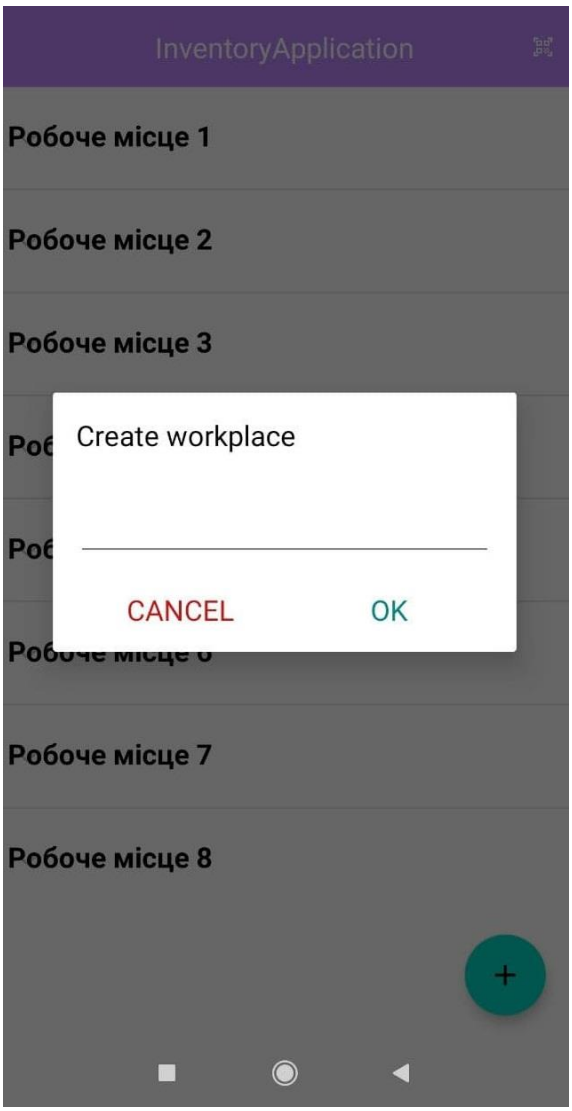

Рисунок 4.12 - Форма створення робочого місця

Для перегляду компонентів, які належать до робочого місця необхідно натиснути на вибране робоче місце, після чого з’явиться випадаючий список компонентів, які належать до вибраного користувачем робочого місця(рис. 4.13).

Існує можливість редагування цих компонентів, а також їх додавання(рис.4.14).

Рисунок 4.13 - Перегляд компонентів робочого місця

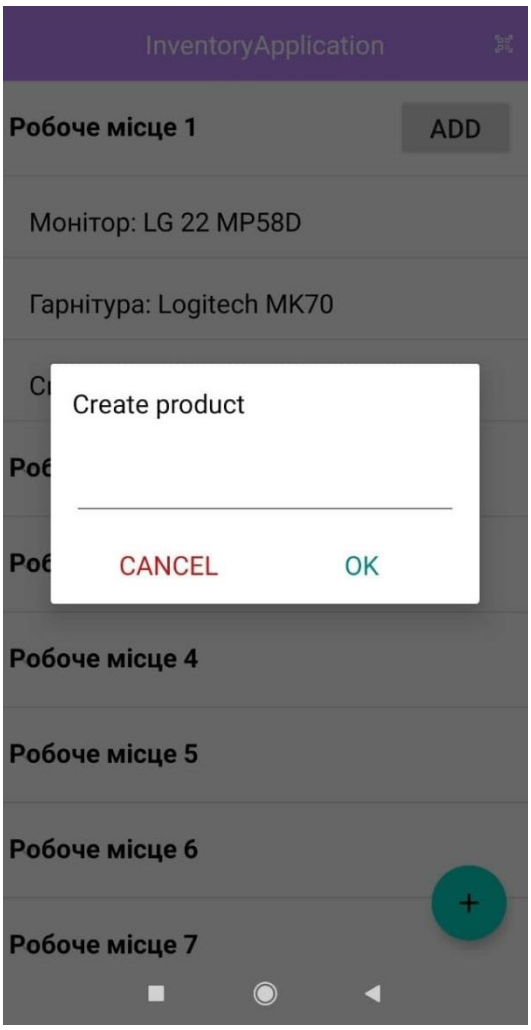

Рисунок 4.14 - Форма створення нового компоненту робочого місця

За допомогою виділеної кнопки користувач має можливість відкрити камеру та зчитати QR- код. QR- код має бути картинкою(рис 4.15).

Рисунок 4.15 - Приклад QR- коду

Вже після успішного зчитування з’явиться повідомлення, на якому буде відображено відсканований QR- код, аудиторію, яка закріплена за цим кодом(рис 4.15). Після натискання на кнопку “ОК” користувач буде перенаправлений на екран з робочими місцями, які закріплені за даною аудиторією, тобто формується звіт по аудиторії(рис. 4.16).

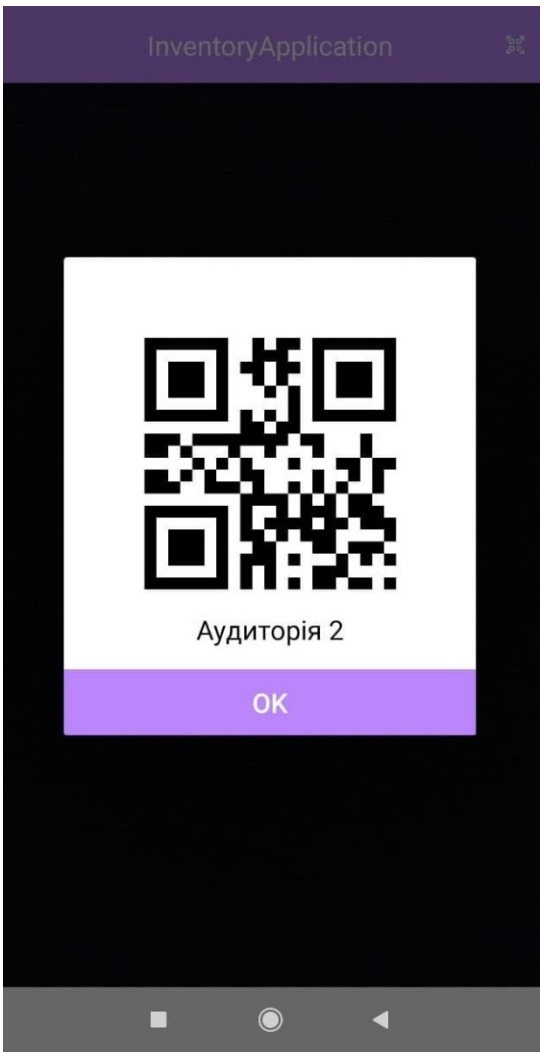

Рисунок 4.15 - Сканування QR-коду

Рисунок 4.16 - Екран робочих місць другої аудиторії після сканування QR- коду

ВИСНОВКИ

Мобільні додатки не є нічим іншим, як перетворюючим способом роботи бізнесу або інструментом для власних потреб. У 2020 році приблизно 3,5 мільярда людей у всьому світі мають смартфон. В даний час мобільні програми представляють найефективніший, прямий та настроюваний спосіб передачі інформації про товар та заохочують клієнтів зберігати вірність певним послугам.

Мобільні додатки в сучасному світі мають таке саме значення, як і веб-сайти, які мали раніше десять років тому.

Компанії постійно борються за увагу та лояльність своїх клієнтів. Конкуренція з пропозицією найкращого товару для широкої аудиторії невпинно припиняється. Отже, в такому стрімко мінливому середовищі переможець завжди виявиться найефективнішим та найзручнішим каналом спілкування зі своєю клієнтурою та їх потребами.

Аналіз предметної області дозволив сформувати мету і задачі дослідження, результати розв’язання яких представлені у пояснювальній записці.

Результати планування кваліфікаційної роботи магістра подані у вигляді WBS та OBS структур, матриці відповідальності та діаграми Ганта. Проведений аналіз ризиків і їх запобігання, виконаний розподіл бюджету.

Для реалізації розробки мобільного додатку використовувався такий інструментарій: мова програмування Kotlin, розпізнавання QR-коду за допомогою сервісу Play Services Vision та його генерація завдяки підключення ZXing.

Процес проектування представлений етапами структурно-функціонального моделювання та створенням моделі варіантів використання.

Висновок щодо функціональності розробленого додатку можна зробити з аналізу наведених прикладів використання додатку при реалізації бізнес-процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Why Mobile Applications Are Important; Especially Its Development, [Електронний ресурс], – Режим доступу до ресурсу: https://www.accelerance.com/blog/why-is-mobile-app-development-so-important-today

2. Why Mobile Apps are Important for your Business, [Електронний ресурс], – Режим доступу до ресурсу: https://mindster.com/mobile-apps-important-business/

3. Mobile Application and Its Global Impact, [Електронний ресурс], – Режим доступу до ресурсу: https://www.researchgate.net/publication/308022297_Mobile_application_and_its_global_impact

4. Impact of Mobile apps on our daily life, [Електронний ресурс], – Режим доступу до ресурсу: https://www.unicodesolutions.com/impact-mobile-apps-daily-life

5. FACTORS INFLUENCING QUALITY OF MOBILE APPS: ROLE OF MOBILE APP DEVELOPMENT LIFE CYCLE, [Електронний ресурс], – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1410/1410.4537.pdf

6. The Influence of Mobile Application Quality and Attributes on the Continuance Intention of Mobile Shopping, [Електронний ресурс], – Режим доступу до ресурсу: https://www.researchgate.net/publication/263928076_The_Influence_of_Mobile_Application_Quality_and_Attributes_on_the_Continuance_Intention_of_Mobile_Shopping

7. What Importance of Mobile Applications in Everyday Life, [Електронний ресурс], – Режим доступу до ресурсу: https://medium.com/@innversetech/what-importance-of-mobile-applications-in-everyday-life-dd2e27c6ee9e

8. Smart Reasons to Build a Mobile App for Your Business, [Електронний ресурс], – Режим доступу до ресурсу: https://www.businessnewsdaily.com/9049-small-business-app-benefits.html

9. How Mobile Apps Can Help Optimize Your Business Prospects, [Електронний ресурс], – Режим доступу до ресурсу: https://clutch.co/app-developers/resources/how-mobile-apps-help-optimize-business-prospects

10. Mobile applications in business: 14 examples, [Електронний ресурс], – Режим доступу до ресурсу: https://deviniti.com/project-management/mobile-applications-in-business/

11. Mobile Apps vs Web Apps: Which is the Better Option? , [Електронний ресурс], – Режим доступу до ресурсу: https://sagaratechnology.medium.com/mobile-apps-vs-web-apps-which-is-the-better-option-868106c88730

12. QR Codes for Inventory Management: Manage and Maximize Efficiency, [Електронний ресурс], – Режим доступу до ресурсу: https://blog.beaconstac.com/2020/07/qr-codes-for-inventory-management/#:~:text=Most%20inventory%20management%20software%20allows,before%20shipping%20to%20retail%20locations.

13. Bar-code technology for inventory and marketing management systems: A model for its development and implementation, [Електронний ресурс], – Режим доступу до ресурсу: https://www.researchgate.net/publication/222863552_Bar-code_technology_for_inventory_and_marketing_management_systems_A_model_for_its_development_and_implementation

14. How to Use QR Codes for Inventory Management, [Електронний ресурс], – Режим доступу до ресурсу: https://www.qr-code-generator.com/blog/how-to-use-qr-codes-for-inventory-management/

15. Wasp Barcode Technologies: The Barcode Solution People, [Електронний ресурс], – Режим доступу до ресурсу: http://www.waspbarcode.com/buzz/5-ways-qr-codes

16. Essential Guide to Inventory Control, [Електронний ресурс], – Режим доступу до ресурсу: https://www.netsuite.com/portal/resource/articles/inventory-management/what-are-inventory-management-controls.shtml

17. QR Code development story, [Електронний ресурс], – Режим доступу до ресурсу: https://www.denso-wave.com/en/technology/vol1.html

18. Инвентаризация и учет запасов для Android, [Електронний ресурс], – Режим доступу до ресурсу: https://www.tec-it.com/ru/software/mobile-data-acquisition/rapid-inventory/overview/Default.aspx

19. Штрих-код Комбайн, [Електронний ресурс], – Режим доступу до ресурсу: http://interestingsolutions.ru/BarcodeHarvester.aspx

20. Operations Management: An Integrated Approach, [Електронний ресурс], – Режим доступу до ресурсу: https://www.oreilly.com/library/view/operations-management-an/9781118122679/ch11-sec015.html#:~:text=Methods%20analysis%20is%20the%20study,for%20doing%20a%20particular%20job.

21. Your Modern Business Guide To Data Analysis Methods And Techniques, [Електронний ресурс], – Режим доступу до ресурсу: https://www.datapine.com/blog/data-analysis-methods-and-techniques/

22. Empirical Research: Definition, Methods, Types and Examples, [Електронний ресурс], – Режим доступу до ресурсу: https://www.questionpro.com/blog/empirical-research/

23. What is Empirical Research Study? , [Електронний ресурс], – Режим доступу до ресурсу: https://www.formpl.us/blog/empirical-research

24. Functional modeling and Information Flow modeling, [Електронний ресурс], – Режим доступу до ресурсу: https://www.geeksforgeeks.org/functional-modeling-and-information-flow-modeling/

25. A Responsive Web-Based QR Code for Inventory in The Laboratory of Informatics, UNESA, [Електронний ресурс], – Режим доступу до ресурсу: https://iopscience.iop.org/article/10.1088/1757-899X/288/1/012109/pdf

26. Barcodes vs Qr Codes for Inventory Management, [Електронний ресурс], – Режим доступу до ресурсу: https://www.sortly.com/blog/barcodes-vs-qr-codes-for-inventory-management/

27. Инвентаризация в цифровом формате, [Електронний ресурс], – Режим доступу до ресурсу: https://www.it-world.ru/cionews/business/143049.html

Додаток А

Планування робіт

Ідентифікація мети мобільного додатку

Мета проекту полягає у розробленні мобільного додатку з інвентаризації мобільних додатків, щоб забезпечити змогу формувати електронну відомість обладнання робочих місць відповідно до аудиторії за допомогою QR-коду. Процес впровадження діджеталізації є однією із головних цілей роботи, таким чином значно полегшиться процес контролю за обладнанням робочих місць.

Планування змісту структури робіт мобільного додатку

Це процес розділення результатів проекту та робіт проекту на більш дрібні елементи, якими легше керувати. Ієрархічна структура робіт (WBS)- це орієнтована на результат ієрархічна декомпозиція робіт, які повинна виконувати команда проекту для досягнення цілей проекту та створення потрібних результатів. WBS організує та визначає загальний зміст проекту та представляє роботи, указані в чинному узгодженому описі змісту проекту.

Продуктом у даному проекті є мобільний додаток процесу інвентаризації матеріальних активів. Виділяємо основні етапи виконання проекту:

- Ініціювання;
- Планування;
- Реалізація;
- Завершення;

Ініціювання розбиваємо на визначення мети проекту, аналогі та загальних потреб додатку. Найвагомішим етапом проекту є розробка мобільного додатку. Даний етап розбиваємо на наступні пункти: створення бази даних, підключення

додаткових сервісів, розробка інтерфейсу, написання kotlin скрипту, тестування на перевірка помилок, виправлення багів та зауважень.

Фінальний результат діаграми WBS зображено на рис.А1.

Планування структури організації, для впровадження готового проекту (OBS). Наступним плануванням структури робіт є розроблення організаційної структури проекту. Організаційна структура проекту (OBS)- є графічним відображенням учасників проекту (фізичних та юридичних осіб) та їхніх відповідальних осіб, залучених до реалізації проекту.

У проекті розроблення аналітичної підсистеми були задіяні наступні виконавці:

- викладач Бойко О.В. - керівник диплому;
- студент Лебідь Д.О. - розробник;

Діаграма OBS представлена на рис.А2.

Побудова календарного графіку з розробки мобільного додатку

Діаграма Ганта- це тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта складається із відрізків, які розміщені на горизонтальній шкалі часу. Кожен відрізок представляє собою певне завдання чи під завдання. Початок, кінець і довжина відрізку відповідає початку, завершенню та тривалості завдання.

Діаграму зображено на рисунках (рис.А3-А5).

Планування ризиків мобільного додатку

Після визначення всіх структурно-організаційні структур, необхідно передбачити всі можливі ризики котрі можуть вплинути на якість та час розробки даного проекту.

Визначаємо ризики:

- R1 - зміна ТЗ на етапі розробки;
- R2 - недотримання календарного плану;
- R3 - відсутність в мережі необхідних фреймворків;
- R4 - хвороба розробника;
- R5 - некоректне тестування;
- R6 - проблеми з розміщенням додатку в мережі Інтернет.

У таблиці А.1 показана ймовірність виникнення кожного ризика.

Таблиця А.1- Ймовірність виникнення ризиків

Ймовірність виникнення	R1	R2	R3	R4	R5	R6
Слабо ймовірний						
Малоймовірний						
Ймовірний						
Дуже ймовірний						
Майже можливий						

Наступний крок - будування таблиці витрат при виникненні ризиків.

Таблиця А.2 - Втрати при виникненні ризиків

Значимість впливу	R1	R2	R3	R4	R5	R6
Мінімальна						
Низька						
Середня						
Висока						
Максимальна						

Виходячи з цих двох, була побудована Матриця впливу (ймовірностей і наслідків) ризиків. В матриці світлим кольором позначено неважливі ризики-білий колір, темнішим- помірні, темним- критичні.

Таблиця А.3 - Матриця ймовірність-втрати

Ймовірність			
			R3
			R2
		R6	R5
	R4	R1	
	Втрати		

Виходячи з цього, було визначено два критичні ризика, такі як:

- R3 - відсутність мережі необхідних фреймворків;
- R2 - недотримання календарного плану;

Третій ризик досить реальний і якщо він станеться треба знаходити самостійні вирішення цього завдяки написанню власних фреймворків. Другий ризик теоретично може виникнути при прояві інших ризиків. Рішення- скоротити виконання окремих робіт.

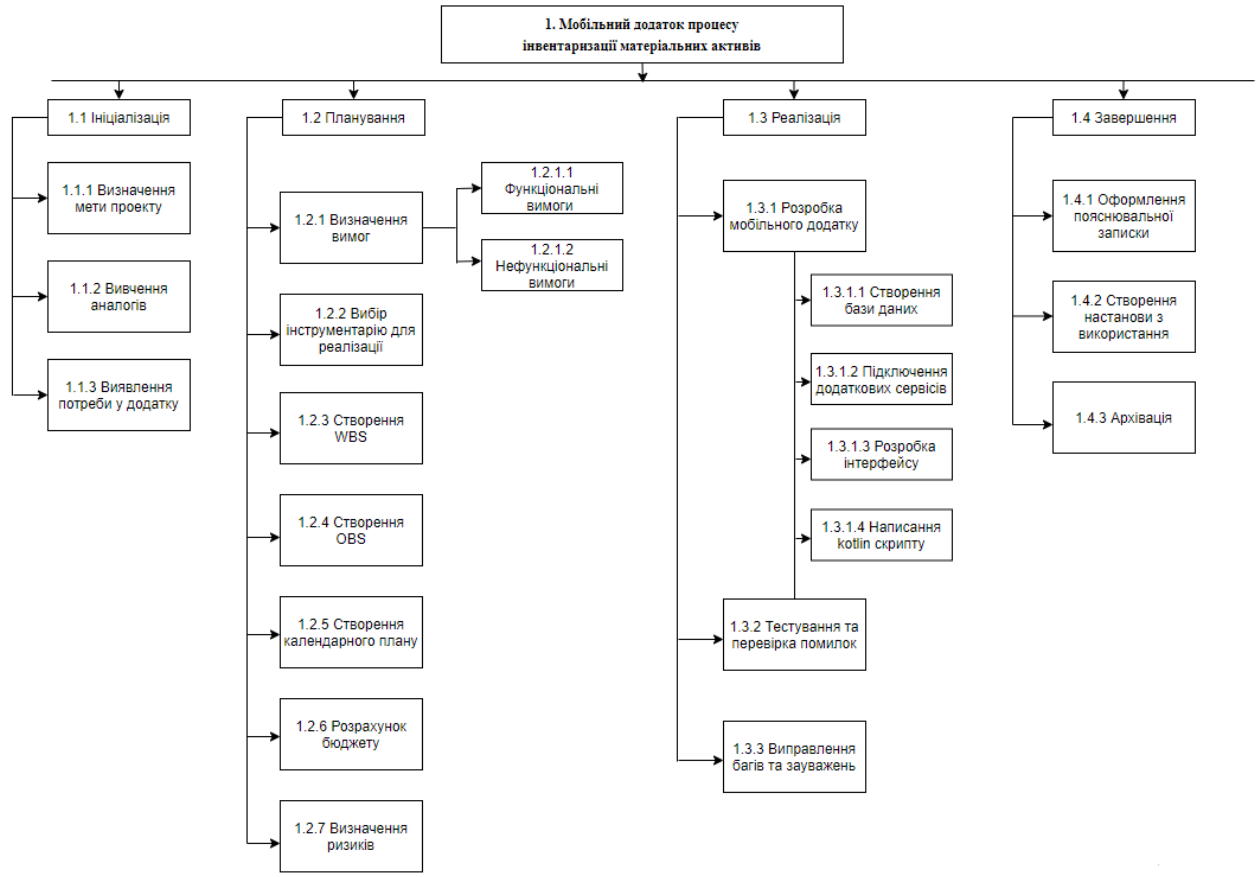

Рисунок А.1 – Діаграма WBS

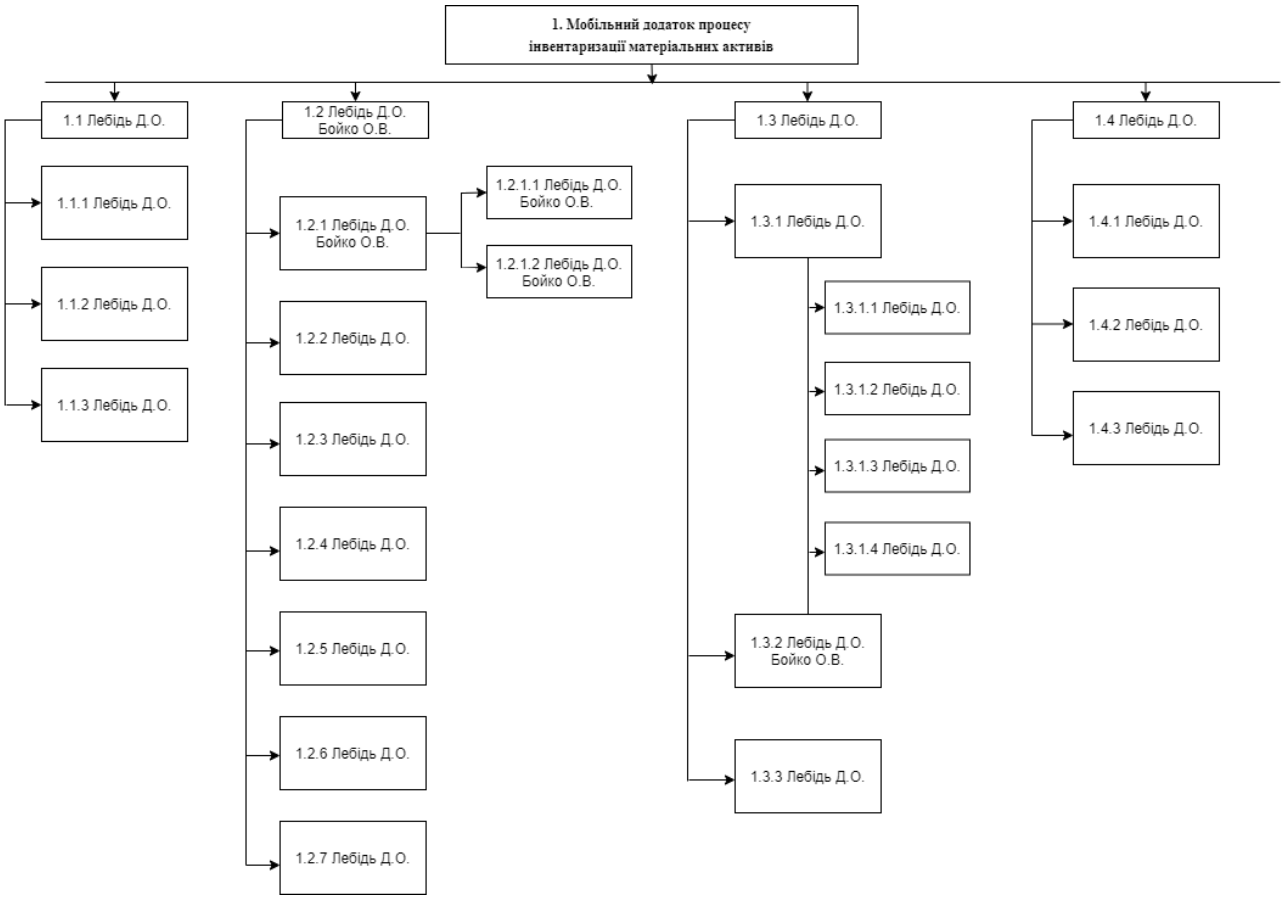

Рисунок А.2 - OBS структура

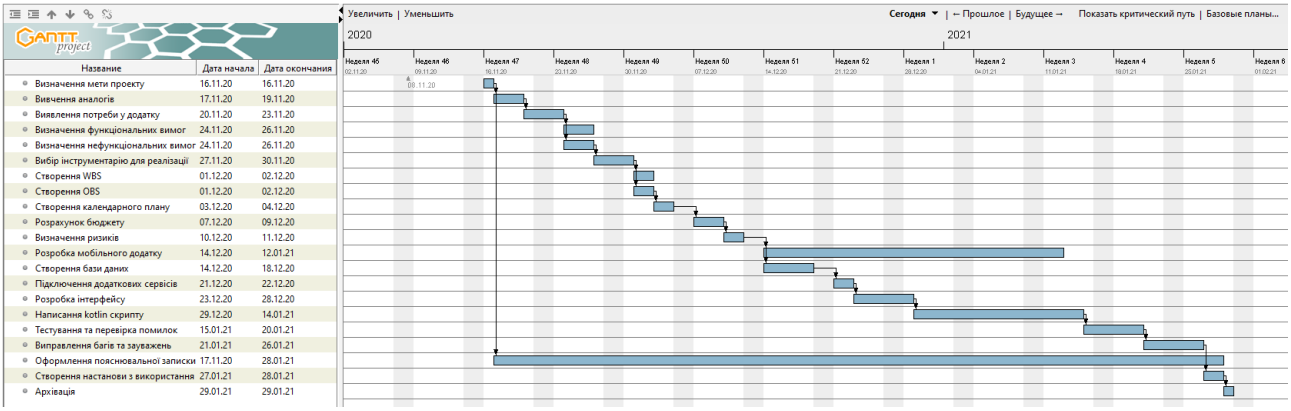

Рисунок А.3 - Діаграма Ганта

Название	Дата начала	Дата окончания
Визначення мети проекту	16.11.20	16.11.20
Вивчення аналогів	17.11.20	19.11.20
Виявлення потреби у додатку	20.11.20	23.11.20
Визначення функціональних вимог	24.11.20	26.11.20
Визначення нефункціональних вимог	24.11.20	26.11.20
Вибір інструментарію для реалізації	27.11.20	30.11.20
Створення WBS	01.12.20	02.12.20
Створення OBS	01.12.20	02.12.20
Створення календарного плану	03.12.20	04.12.20
Розрахунок бюджету	07.12.20	09.12.20
Визначення ризиків	10.12.20	11.12.20
Розробка мобільного додатку	14.12.20	12.01.21
Створення бази даних	14.12.20	18.12.20
Підключення додаткових сервісів	21.12.20	22.12.20
Розробка інтерфейсу	23.12.20	28.12.20
Написання kotlin скрипту	29.12.20	14.01.21
Тестування та перевірка помилок	15.01.21	20.01.21
Виправлення багів та зауважень	21.01.21	26.01.21
Оформлення пояснювальної записки	17.11.20	28.01.21
Створення настанови з використання	27.01.21	28.01.21
Архівація	29.01.21	29.01.21

Рисунок А.4 - Діаграма Ганта. Перша частина

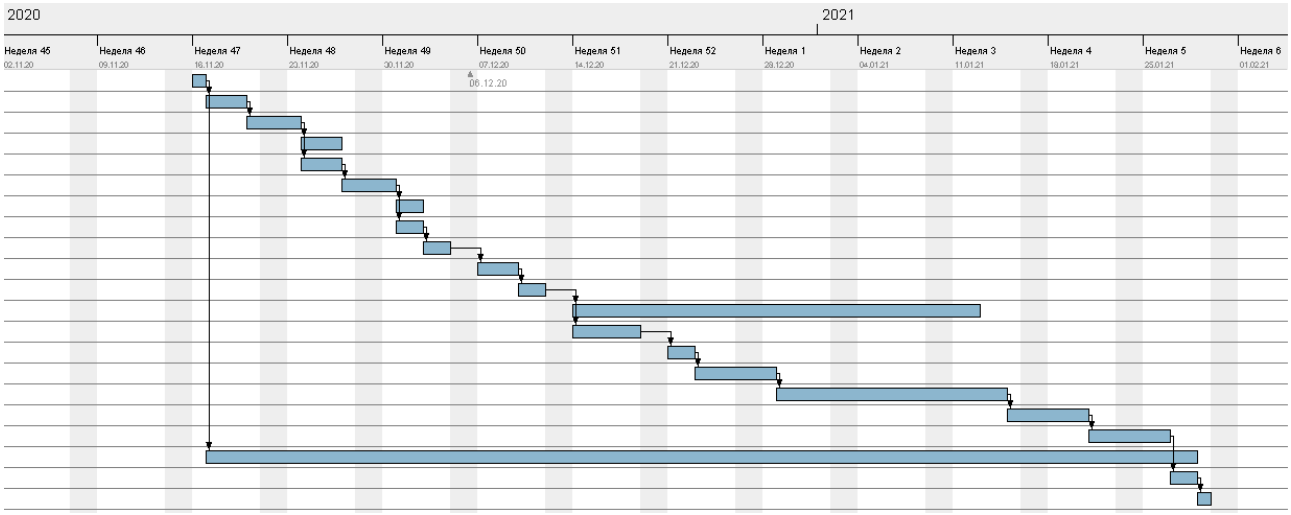

Рисунок А.5 - Діаграма Ганта. Друга частина

Додаток Б

Лістинг програмного коду

Лістинг коду файлу MainActivity:

```
package com.inventoryapplication.activity

import android.Manifest
import android.content.pm.PackageManager
import android.os.Bundle
import android.widget.ImageView
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentActivity
import com.inventoryapplication.fragments.BarcodeFragment
import com.inventoryapplication.DataHolder
import com.inventoryapplication.R
import com.inventoryapplication.fragments.RoomListFragment

class MainActivity : FragmentActivity() {

    companion object {

        private const val TAG = "MainActivity"
        private const val CAMERA_PERMISSION_REQUEST_CODE = 99999

    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        DataHolder.obtainList(this)

        setListeners()


        addFragment(RoomListFragment(), false)
    }


    private fun setListeners() {
        val qrButton: ImageView = findViewById(R.id.qrButton)
        qrButton.setOnClickListener {
            if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
                addFragment(BarcodeFragment())
            } else {
                requestPermissions(
                    arrayOf(Manifest.permission.CAMERA),
                    CAMERA_PERMISSION_REQUEST_CODE
                )
            }
        }
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
```

```
    ) {
        if (grantResults.isEmpty()) {
            return
        } else {
            if (grantResults.all { it == PackageManager.PERMISSION_GRANTED }) {
                if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
                    addFragment(BarcodeFragment())
                }
            }
        }
    }

    fun addFragment(fragment: Fragment, addToBackStack: Boolean = true) {
        val ft = supportFragmentManager.beginTransaction()
        ft.add(R.id.fragmentContainer, fragment)
        if (addToBackStack) {
            ft.addToBackStack(null)
        }
        ft.commit()
    }

}
```

Лістинг коду файлу RoomsListAdapters:

```
package com.inventoryapplication.adapters

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import android.widget.Toast
import androidx.recyclerview.widget.RecyclerView
import com.inventoryapplication.DataHolder
import com.inventoryapplication.R
import com.inventoryapplication.dialogs.UpdateItemDialog
import com.inventoryapplication.pojo.Room

class RoomsListAdapter: RecyclerView.Adapter<RoomsListAdapter.RoomHolder>() {

    private var items = DataHolder.roomsList
    private var clickListener: OnRoomClickListener? = null


    fun setOnItemCLickListener(l: OnRoomClickListener) {
        clickListener = l
    }


    private lateinit var context: Context
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RoomHolder {
        context = parent.context
        val view = LayoutInflater.from(parent.context).inflate(R.layout.room_list_item, parent, false)

        return RoomHolder(view)
    }

    override fun onBindViewHolder(holder: RoomHolder, position: Int) {
        val item  = items[position]
```

```
        with(holder) {
            name.text = item.name
            view.setOnClickListener {
                clickListener?.onRoomClicked(item)
            }
            view.setOnLongClickListener {
                val dialog = UpdateItemDialog(holder.name.context, room = item)
                dialog.create()
                dialog.setOnOkClickListener {
                    notifyDataSetChanged()
                }
                dialog.show()
                return@setOnLongClickListener true
            }
        }

    }

    override fun getItemCount() = items.size

    class RoomHolder(val view: View): RecyclerView.ViewHolder(view) {
        val name: TextView = view.findViewById(R.id.name)
    }

    interface OnRoomClickListener {
        fun onRoomClicked(room: Room)
    }

}
```

Лістинг коду файлу WorkplacesListAdapter:

```
package com.inventoryapplication.adapters

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.BaseExpandableListAdapter
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import com.inventoryapplication.R
import com.inventoryapplication.dialogs.UpdateItemDialog
import com.inventoryapplication.pojo.Workplace
import com.inventoryapplication.pojo.WorkplaceItem


class WorkplacesListAdapter(
    private val context: Context
) : BaseExpandableListAdapter (){


    private var clickListener: OnAddWorkplaceItemButtonClickListener? = null

    fun setClickListener(l: OnAddWorkplaceItemButtonClickListener) {
        clickListener = l
    }
```

```
private var workplacesList: List<Workplace> = emptyList()
fun swapData(l: List<Workplace>) {
    workplacesList = l
    notifyDataSetChanged()
}

override fun getGroupCount() =
    workplacesList.size


override fun getChildrenCount(position: Int) =
    workplacesList[position].items.size


override fun getGroup(position: Int) =
    workplacesList[position]

override fun getChild(groupPosition: Int, childPosition: Int) =
    workplacesList[groupPosition].items[childPosition]


override fun getGroupId(groupPosition: Int) =
    groupPosition.toLong()

override fun getChildId(groupPosition: Int, childPosition: Int) =
    childPosition.toLong()

override fun hasStableIds() =
    true

override fun getGroupView(
    groupPosition: Int,
    isExpanded: Boolean,
    convertView: View?,
    parent: ViewGroup
): View {
    val v = convertView ?: LayoutInflater.from(context).inflate(R.layout.group_view, null)
    val addButton: Button = v.findViewById(R.id.addWorkplaceItemButton)

    addButton.setOnClickListener {
        clickListener?.onAddClicked(groupPosition, workplacesList[groupPosition].id)
    }

    if (isExpanded) {
        addButton.visibility = View.VISIBLE
    } else {
        addButton.visibility = View.GONE
    }

    v.setOnLongClickListener {
        val dialog = UpdateItemDialog(parent.context, workplace = getGroup(groupPosition))
        dialog.create()
        dialog.setOnOkClickListener {
            notifyDataSetChanged()
        }
        dialog.show()
        return@setOnLongClickListener true
    }

    val textGroup: TextView = v.findViewById(R.id.textGroup)
    val text = getGroup(groupPosition).name
    textGroup.text = text
    v.setOnClickListener {
```

```
            clickListener?.onGroupClicked(groupPosition, isExpanded)
        }
        return v
    }


    override fun getChildView(
        groupPosition: Int,
        childPosition: Int,
        isLastChild: Boolean,
        convertView: View?,
        parent: ViewGroup
    ): View {
        val view = convertView ?: LayoutInflater.from(context).inflate(R.layout.child_view, null)

        val textChild: TextView = view.findViewById(R.id.textChild)
        val item = workplacesList[groupPosition].items[childPosition]
        textChild.text = item.name

        view.setOnLongClickListener {
            val dialog = UpdateItemDialog(parent.context, workplaceItem = item)
            dialog.create()
            dialog.setOnOkClickListener {
                notifyDataSetChanged()
            }
            dialog.show()
            return@setOnLongClickListener true
        }

        return view
    }

    override fun isChildSelectable(groupPosition: Int, childPosition: Int) = true


    interface OnAddWorkplaceItemButtonClickListener {

        fun onAddClicked(groupPosition: Int, workplaceID: String)


        fun onGroupClicked(groupPosition: Int, isExpanded: Boolean)

    }

}
```

Лістинг коду файлу BarcodePreviewDialog:

```
package com.inventoryapplication.barcode

import android.app.Dialog
import android.content.Context
import android.os.Bundle
import android.view.Window
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import com.google.zxing.BarcodeFormat
import com.google.zxing.MultiFormatWriter
import com.google.zxing.WriterException
```

```
import com.inventoryapplication.R
import com.inventoryapplication.pojo.Room
import com.inventoryapplication.pojo.WorkplaceItem
import com.journeyapps.barcodescanner.BarcodeEncoder

class BarcodePreviewDialog(
    context: Context,
    val room: Room? = null,
    val workplaceItem: WorkplaceItem? = null
) : Dialog(context) {


    private lateinit var preview: ImageView
    private lateinit var barcodeValue: TextView
    private lateinit var actionButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        requestWindowFeature(Window.FEATURE_NO_TITLE)
        setContentView(R.layout.barcode_preview_dialog_layout)


        preview = findViewById(R.id.preview)
        barcodeValue = findViewById(R.id.barcodeValue)
        actionButton = findViewById(R.id.actionButton)


        val name = room?.name ?: "${workplaceItem?.roomName} --> ${workplaceItem?.name}"
        barcodeValue.text = name
        generateCode()

    }


    fun setOnOkActionButtonAction(action: () -> Unit) {
        actionButton.setOnClickListener {
            action()
            dismiss()
        }
    }

    private fun generateCode() {
        val barcode = room?.id ?: (workplaceItem?.id ?: "")
        val multiFormatWriter = MultiFormatWriter()
        try {
            val bitMatrix = multiFormatWriter.encode(barcode, BarcodeFormat.QR_CODE, 700, 700)
            val barcodeEncoder = BarcodeEncoder()
            val bitmap = barcodeEncoder.createBitmap(bitMatrix)
            preview.setImageBitmap(bitmap)
        } catch (e: WriterException) {
            e.printStackTrace()
        }
    }

}
```

Лістинг коду файлу BarcodeTrackerFactory:

```
package com.inventoryapplication.barcode

import com.google.android.gms.vision.Detector
import com.google.android.gms.vision.MultiProcessor
```

```
import com.google.android.gms.vision.Tracker
import com.google.android.gms.vision.barcode.Barcode

class BarcodeTrackerFactory : MultiProcessor.Factory<Barcode> {

    private var onBarcodeRecognizedListener: OnBarcodeRecognizedListener? = null
    fun setOnBarcodeRecognizedListener(l: OnBarcodeRecognizedListener) {
        onBarcodeRecognizedListener = l
    }

    override fun create(barcode: Barcode?): Tracker<Barcode> {
        return object : Tracker<Barcode>() {
            override fun onUpdate(detectionResults: Detector.Detections<Barcode?>, barcode: Barcode?) {
                if (barcode != null) {
                    if (barcode.isRecognized) {
                        onBarcodeRecognizedListener?.onBarcodeRecognized(barcode)
                    }
                }
            }
        }
    }

    interface OnBarcodeRecognizedListener {
        fun onBarcodeRecognized(barcode: Barcode)
    }
}
```

Лістинг коду файлу CreateItemDialog:

```
package com.inventoryapplication.dialogs

import android.app.Dialog
import android.content.Context
import android.os.Bundle
import android.view.Window
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.inventoryapplication.R

class CreateItemDialog(context: Context, private val type: String): Dialog(context) {



    private lateinit var title: TextView
    private lateinit var input: EditText
    private lateinit var okButton: TextView
    private lateinit var cancelButton: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        requestWindowFeature(Window.FEATURE_NO_TITLE)
        setContentView(R.layout.create_item_layout)

        input = findViewById(R.id.input)
        cancelButton = findViewById(R.id.cancel)
        okButton = findViewById(R.id.ok)
        title = findViewById(R.id.title)

        title.text = "Create $type"
```

```
        setListeners()

    }

    fun setOnOkClickListener(action: (String) -> Unit) {
        okButton.setOnClickListener {
            val name = input.text.toString()
            if(name.isEmpty()) {
                Toast.makeText(context, "Name must not be empty", Toast.LENGTH_SHORT).show()
            } else {
                action(name)
                dismiss()
            }


        }
    }

    private fun setListeners() {
        cancelButton.setOnClickListener {
            dismiss()
        }
    }

}
```

Лістинг коду файлу UpdateItemDialog:

```
package com.inventoryapplication.dialogs

import android.app.Dialog
import android.content.Context
import android.os.Bundle
import android.view.Window
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.inventoryapplication.DataHolder
import com.inventoryapplication.R
import com.inventoryapplication.pojo.Room
import com.inventoryapplication.pojo.Workplace
import com.inventoryapplication.pojo.WorkplaceItem

class UpdateItemDialog(context: Context,
                       private val room: Room? = null,
                       private val workplace: Workplace? = null,
                       private val workplaceItem: WorkplaceItem? = null): Dialog(context) {



    private lateinit var title: TextView
    private lateinit var input: EditText
    private lateinit var okButton: TextView
    private lateinit var cancelButton: TextView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        requestWindowFeature(Window.FEATURE_NO_TITLE)
        setContentView(R.layout.create_item_layout)

        input = findViewById(R.id.input)
        cancelButton = findViewById(R.id.cancel)
```

```
        okButton = findViewById(R.id.ok)
        title = findViewById(R.id.title)

//        title.text = "Create $type"


        okButton.text = "RENAME"
        updateTitle()
        setListeners()
    }


    private fun updateTitle() {
        when {
            room != null -> {
                title.text = "ID:${room?.id}"
                input.setText(room.name)
            }
            workplace != null -> {
                title.text = "ID:${workplace?.id}"
                input.setText(workplace.name)
            }
            workplaceItem != null -> {
                title.text = "ID:${workplaceItem?.id}"
                input.setText(workplaceItem.name)
            }
        }
    }


    fun setOnOkClickListener(action: () -> Unit) {
        okButton.setOnClickListener {
            val name = input.text.toString()
            if(name.isEmpty()) {
                Toast.makeText(context, "Name must not be empty", Toast.LENGTH_SHORT).show()
            } else {
                nameUpdateAction(name)
                DataHolder.saveList(okButton.context)
                action()
                dismiss()
            }
        }
    }


    private fun nameUpdateAction(name: String) {
        when {
            room != null -> {
                DataHolder.getRoom(room.id)?.name = name
            }
            workplace != null -> {
                DataHolder.getWorkplace(workplace.id)?.name = name
            }
            workplaceItem != null -> {
                DataHolder.getWorkplaceItem(workplaceItem.id)?.name = name
            }
        }
    }


    private fun removeItemAction(context: Context) {
        when {
            room != null -> {
```

```
                DataHolder.removeRoom(context, room.id)
            }
            workplaceItem != null -> {
                DataHolder.removeWorkplaceItem(context, workplaceItem)
            }
        }
    }

    private fun setListeners() {
        cancelButton.setOnClickListener {
            dismiss()
        }
    }

}
```

Лістинг коду файлу DataHolder:

```
package com.inventoryapplication

import android.content.Context
import android.content.Context.MODE_PRIVATE
import android.util.Log
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken
import com.inventoryapplication.pojo.Room
import com.inventoryapplication.pojo.Workplace
import com.inventoryapplication.pojo.WorkplaceItem
import kotlin.random.Random

object DataHolder {

    var roomsList: MutableList<Room> = mutableListOf()
    private const val PREFERENCES_NAME = "NAME"
    private const val DATA_JSON = "DATA"



    fun getRoom(roomID: String): Room? {
        return roomsList.find {
            it.id == roomID
        }
    }

    fun removeRoom(context: Context, roomId: String) {
        if(roomsList.remove(getRoom(roomId))) {
            saveList(context)
        }
    }

    fun removeWorkplaceItem(context: Context, workplaceItem: WorkplaceItem) {
        val list = roomsList.find {
            it.id == workplaceItem.roomID
        }?.workplaces?.flatMap {
            it.items
        }?.toMutableList()


        list?.remove(workplaceItem)
        saveList(context)

    }
```

```
fun addRoom(context: Context, roomName: String, onSuccess:() -> Unit) {
    val room = Room()
    with(room) {
        name = roomName
        id = "${roomName.hashCode()}" + Random.nextInt(0, 90)
    }
    if(roomsList.add(room)) {
        saveList(context)
        onSuccess()
    }
}

fun addWorkplace(context: Context, workplaceName: String, roomId: String, onSuccess: () -> Unit) {
    val workplace = Workplace()
    val room = getRoom(roomId)
    with(workplace) {
        name = workplaceName
        id = "${workplaceName.hashCode()}" + Random.nextInt(0, 90)
        roomID = roomId
        roomName = room?.name ?: ""
    }
    if(room?.workplaces?.add(workplace) == true) {
        saveList(context)
        onSuccess()
    }
}

fun getWorkplace(workplaceId: String): Workplace? {
    return roomsList.flatMap {
        it.workplaces
    }.find {
        it.id == workplaceId
    }
}

fun getWorkplaceItem(id: String): WorkplaceItem? {
    return roomsList.flatMap {
        it.workplaces
    }.flatMap {
        it.items
    }.find {
        it.id == id
    }
}

fun addWorkplaceItem(context: Context, workplaceItemName: String, workplaceId: String, onSuccess: () -> Unit) {
    val workplaceItem = WorkplaceItem()
    val roomId = getWorkplace(workplaceId)?.roomID ?: ""
    with(workplaceItem) {
        name = workplaceItemName
        roomID = roomId
        id = "${workplaceItemName.hashCode()}" + Random.nextInt(0, 90)
        roomName = getRoom(roomId)?.name ?: ""
    }
    if(getWorkplace(workplaceId)?.items?.add(workplaceItem) == true) {
        saveList(context)
        onSuccess()
    }
}
```

```
    fun obtainList(context: Context){
        val sharedPreferences = context.getSharedPreferences(PREFERENCES_NAME, MODE_PRIVATE)
        val json = sharedPreferences.getString(DATA_JSON, "")
        val type = object : TypeToken<MutableList<Room>>(){}.type
        val l = Gson().fromJson<MutableList<Room>>(json, type)
        roomsList = l ?: mutableListOf()
    }


    fun saveList(context: Context) {
        val sharedPreferencesEditor = context.getSharedPreferences(PREFERENCES_NAME, MODE_PRIVATE).edit()
        sharedPreferencesEditor.putString(DATA_JSON, Gson().toJson(roomsList))
        sharedPreferencesEditor.apply()
    }
}
```