МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2018 р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Система ідентифікації продуктів на базі Android для людей з вадами зору

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям підготовки 6.050102 – "Комп'ютерна інженерія"

Керівник проекту: _____ (підпис) Щербакова М.Є. (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О. Критська (ініціали, прізвище)

Здобувач вищої освіти: _____ (підпис) Шехавцов П.П. (ініціали, прізвище)

Група: КІ-14ад

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень Бакалавр
Напрям підготовки 6.050102 – "Комп'ютерна інженерія"
(шифр і назва)
Спеціальність
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри
І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Шехавцова Павла Павловича
(прізвище, ім'я, по батькові)

1. Тема роботи Система ідентифікації продуктів на базі Android для людей з вадами зору
керівник проекту
(роботи) Щербакова Марина Євгенівна доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 14 05 2018 р. № 117/48
2. Термін подання студентом роботи 11.06.2018
3. Вихідні дані до
роботи Мова програмування JAVA
Програмне забезпечення для мобільного телефону, що дозволяє ідентифікувати продукцію

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз аналогічних систем, вибір засобів реалізації,проектування, розробка програмного продукту, написання проектної документаці
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	ст. викл. Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Робота з літературою та інтернет-джерелами	до 03.05.2018	
2	Аналіз отриманих результатів	до 10.05.2018	
3	Оформлення першого розділу	до 18.05.2018	
4	Підготовка змісту дипломної роботи	до 30.05.2018	
5.	Аналіз вимог до програмного продукту	до 06.06.2018	
6	Написання теоретичної частини дипломної роботи	до 10.06.2018	
7	Написання практичної частини дипломної роботи	до 15.06.2018	
8	Розділ ДР з «Охорони праці»	до 15.06.2018	
9	Захист дипломного проекту(роботи)	25.06.2018 (згідно графіку)	

Студент _____ **Шехавцов П.П.**
(підпис) (прізвище та ініціали)

Керівник _____ **Щербакова М.Є.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту бакалавра: 109 сторінок., 20 рисунків,5 таблиць, 13 бібліографічних джерел, 2 додаток.

Об’єкт розробки: Система ідентифікації продуктів на базі Android для людей з вадами зору.

Мета роботи: розробка Android додатка зчитування штрих коду для людей з вадами зору.

В проекті виконано:

1) аналіз та визначення задач мобільного додатку;

2) розробка архітектури проекту;

3) створеня макету додатку;

4) реалізація програмного продукту.

Отримано наступні результати:

В результаті виконання дипломного проекту було спроектовано та реалізовано Android-додаток зчитування штрих коду для людей з вадами зору.

Ключові слова: ANDROID, JAVA, BARCODE, АРХІТЕКТУРА ДОДАТКУ

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А, м. Сєвєродонецьк, 93400.

ЗМІСТ

ВСТУП

В наш час все частіше інформаційні технології приходять на допомогу пересічному споживачеві, для полегшення використання великого обсягу інформації, орієнтації в ньому і правильної й швидкої обробки цього потоку інформації. Це відноситься до різних сфер життєдіяльності людини і, в тому числі і до сфери охорони здоров'я. Сьогодні налічується більше 100 тисяч мобільних додатків mHealth, розроблених на двох великих платформах: iOS і Android. Обидві є абсолютними лідерами серед операційних систем для мобільних телефонів. Однак не так багато додатків, розроблених на базі Android, які допоможуть людям з вадами зору. Тому дуже зручно мати в своєму мобільному телефоні програму, яка могла допомогти ідидинфікувати продукцію.

Необхідно розробити Андроїд-додаток для сканування штрих-кодів та оголошення результатів сканування, призначений для людей з вадами зору.

1 МОВА ПРОГРАМУВАННЯ JAVA

1.1 Історія створення

Мова програмування Java зародилася в 1991 р. в лабораторіях компанії Sun Microsystems. Розробку проекту започаткував Джеймс Ґослінґ, сам проект мав назву «Green»(Зелений). Створення першої робочої версії, яка мала назву «Oak»(дуб), зайняло 18 місяців. Оскільки виявилось, що ім'я Oak уже використовувалось іншою фірмою, то в результаті тривалих суперечок навколо назви нової мови з поміж ряду запропонованих було вибрано назву Java, у 1995 р. мову було офіційно перейменовано.

Головним мотивом створення Java була потреба в мові програмування, яка б не залежала від платформи (тобто від архітектури) і яку можна було б використовувати для створення програмного забезпечення, яке вбудовується в різноманітні побутові електронні прилади, такі як мобільні засоби зв'язку, пристрої дистанційного керування тощо.

Досить скоро майже всі найпопулярніші тогочасні веб-оглядачі отримали можливість запускати «безпечні» для системи Java аплети всередині веб-сторінок. У грудні 1998 р. Sun Microsystems випустила Java 2 (спершу під назвою J2SE 1.2), де було реалізовано декілька конфігурацій для різних типів платформ. Наприклад, J2EE призначалася для створення корпоративних застосунків, а значно урізана J2ME для приладів з обмеженими ресурсами, таких як мобільні телефони. У 2006 році в маркетингових цілях, Версії J2 було перейменовано у Java EE, Java ME та Java SE, відповідно.

13 листопада 2006 року Sun випустили більшу частину Java в якості вільного та відкритого програмного забезпечення згідно з умовами GNU General Public License (GPL). 8 травня 2007 корпорація закінчила процес, в результаті

якого всі початкові коди Java були випущенні під GPL, за винятком невеликої частини коду, на який Sun не мала авторського права.

Період становлення Java збігся у часі з розквітом міжнародної інформаційної служби World Wide Web. Ця обставина відіграла вирішальну роль у майбутньому Java, оскільки Web теж вимагала платформо-незалежних програм. Як наслідок, були зміщені акценти в розробці Sun з побутової електроніки на програмування для Інтернет.

1.2 Безпечність

Один із ключових принципів розробки мови Java полягав у забезпеченні захисту від несанкціонованого доступу. Програми на Java не можуть викликати глобальні функції й одержувати доступ до довільних системних ресурсів, що забезпечує в Java рівень безпеки, недоступний для інших мов. Даний рівень безпеки виконання Java-програм забезпечує віртуальна машина Java, котра вбудована в операційну систему. Об'єктна модель у Java проста і легко розширюється, у той же час, заради підвищення продуктивності прості типи даних Java не є об'єктами.

World Wide Web висунула Java на передній план програмування, і Java, в свою чергу, сильно вплинула і навіть змінила обличчя Internet, розширивши спектр об’єктів, які можуть розповсюджуватись у кіберпросторі. Програми нової форми - аплети - завантажуються з віддаленого сервера і можуть запускатися динамічно, тобто без участі користувача. До появи Java такий підхід був неприпустимий з міркувань безпеки та переносимості. В архітектурі аплетів зроблено ряд штучних обмежень, які роблять їх цілком безпечними. Перш за все, Java є інтерпретованою мовою і простір ресурсів Java-програми обмежений так званою віртуальною Java-машиною (VJM), яка може контролювати поведінку

програми і захищати систему від побічних ефектів, які можуть виникати з вини аплета. Крім того, в мові Java є додаткові обмеження, які не дозволять аплету стати «троянським конем». Зокрема, Java-аплет не може отримати доступ до локального жорсткого диску. При такій спробі генерується виключна ситуація.

1.3 Ефективність

Оскільки аплети Java інтерпретуються, а не компілюються, то їх виконання на різних платформах значно полегшується. В цьому випадку достатньо створити для кожної платформи виконуючу Java-систему. Якщо існує така система для даної операційної системи, то будь-яка Java-програма може виконуватись в даному середовищі без додаткової компіляції на цій платформі. Проте Java не є інтерпретованою мовою в чистому розумінні. Програма на Java компілюється. Результатом роботи компілятора Java є байткод (bytecode). Байткод - це оптимізований набір команд, призначений для виконання уявним пристроєм - віртуальною Java-машиною. В такий спосіб витрати на інтерпретацію зводяться до мінімуму, оскільки байкод вже є оптимізованим, і досягається досить висока продуктивність Java-програм. Наведені вище особливості дають підставу розглядати Java не як ще одну мову програмування, а як окрему інформаційну технологію. Таким чином, інтерпретація - це найлегший шлях до перенесення програм, реалізований в Java технології. Незважаючи на те, що мова Java була розроблена в розрахунку на інтерпретацію, технічно немає нічого такого, що б перешкоджало компіляції байткоду в виконуваний код. До байткоду, який пересилається по мережі, застосовується динамічна компіляція, але це ніяк не впливає на переносимість та безпеку, оскільки роботу програми все ще контролює виконуюча система. Такий підхід застосовано в багатьох виконуючих системах Java, що забезпечує продуктивність на рівні оптимізованого коду C++.

Мова Java є однією з наймолодших в сімействі мов програмуваня і була розроблена з розрахунку на те, щоб професійний програміст міг легко її опанувати та ефективно використовувати. За основу Java взятий синтаксис C++ - безсумнівно однієї з найбільш популярних мов програмування сучасності. Проте, Java - це цілком самостійна мова програмування, і при її створенні не йшлося про будь-яку сумісність з C++. Тому деякі механізми реалізовані в Java інакше, а деякі зовсім відсутні. Ідеологічно ж Java побудована дещо інакше ніж C++. Розробники Java грунтувалися на досвіді розробки програм на C++ і прагнули позбутися можливостей, які зарекомендували себе непевними. Так, в Java відсутня перегрузка операторів а також автоматичне приведення несумісних типів - конструкції, які при неуважному використанні є джерелом важких для виявлення помилок. Взагалі, інтерфейси Java більш прості та прозорі для розуміння. Написати на Java програму з графічним інтерфейсом значно легше. Звичайно, простота інтерфейсів компенсується меншою гнучкістю, бібліотека Java не така багата, як стандартні бібліотеки C/C++. Але згадаймо, що Java задуманий для використання на різних платформах і тому реалізує в собі найбільш стандартні можливості задля легшої адаптації під конкретне середовище.

1.4 Об’єктно-орієнтована спрямованість

Від C++ Java успадкувала потужний механізм об’єктно-орієнтованого програмування. Оскільки Java розроблювався «на пустому місці», тобто не було потреби забезпечувати сумісність з попередніми версіями, розробники мали повну свободу мислення. В результаті був сформований ясний і прагматичний підхід до об’єктів. Вільно переймаючи ідеї, які реалізовувалися протягом останніх десятирічь, мові Java вдалося знайти рівновагу між парадигмою «все є об’єктом» і прагматичним підходом. Об’єктна модель Java проста і легко розширюється, в той

час як просі типи, як цілі, зберігаються як дані, що не є об’єктами, що дозволяє значно підвищити швидкість при їх обробці.

В Java вбудований набір ключових класів, що містять основні абстракції реального світу, з яким прийдеться мати справа вашим програмам. Основою популярності Java є вбудовані класи-абстракції, що зробили його мовою, дійсно незалежним від платформи.

Фактично, більшість архітектурних рішень, прийнятих при створенні Java, було продиктовано бажанням надати синтаксис, подібний із C и C++. У Java використовуються практично ідентичні вимоги для оголошення змінних, передачі параметрів, операторів і для керування потоком виконанням коду. У Java додані всі гарні риси C++, але виключені недоліки останнього.

Вказівники або адреси в пам'яті — найбільш могутня і найбільш небезпечна риса C++. Причиною більшості помилок у сьогоднішньому коді є саме неправильна робота з вказівники. Наприклад, одна з типових помилок — прорахуватися на одиницю в розмірі масиву і зіпсувати вміст комірки пам'яті, розташованої слідом за ним.

Хоча в Java дескриптори об'єктів і реалізовані у вигляді вказівників, у ній відсутні можливості працювати безпосередньо з ними. Ви не можете перетворити ціле число в вказівник, а також звернутися до довільної адреси пам'яті.

Проте на мові Java можна створювати не тільки апплети, а й консольні додатки, GUI-додатки, сервлети та JSP.

Невдовзі після появи технології сервлетів розробники стикнулися з такою проблемою: для динамічної генерації HTML-сторінок за допомогою сервлета HTML-код доводиться розміщувати в самому сервлеті. При цьому HTML-код сторінки змішується з Java-кодом (при цьому логіка роботи програми змішується із зовнішнім виглядом web-сторінки), що ускладнює роботу як програміста, так і веб-дизайнера.

Для вирішення цієї проблеми була розроблена технологія JavaServer Pages (JSP). Вона дозволяє розміщувати Java-код всередині HTML-коду web-сторінки.

При першому зверненні до jsp-сторінки її код автоматично перетворюється в сервлет і компілюється. Після цього при наступних зверненнях web-сервер викликає не jsp-сторінку, а відкомпільований сервлет. При внесенні змін в jsp-сторінку web-сервер виявляє, що сторінка змінилась, і знову оновлює відповідний сервлет.

У технологіях сервлетів і JSP введене поняття *контейнера* (container). *Servlets-контейнер* – це механізм, що відповідає за виконання сервлетів. *JSP-контейнер* – механізм, що відповідає за перетворення jsp-сторінок у сервлети і передачу цих сервлетів Servlets-контейнеру. Оскільки сервлети і jsp-сторінки викликаються через протокол HTTP, то контейнери часто супроводжує ще один компонент – web-сервер. Сукупність web-серверу і контейнерів формує *web-сервер додатків* .

Технологія Servlets і JSP були об’єднані з декількома іншими Java-технологіями, і цей комплекс був названий Java 2 Enterprise Edition (J2EE). Таким чином з’явилися сервери Java-додатків від різних компаній (IBM, BEA, IONA, Borland), у тому числі від самої компанії Sun. Кожна компанія у своєму сервері додатків реалізує технології J2EE по-своєму, але всі вони відповідають специфікації Sun.

Компанія Sun раніше пропонувала безкоштовну еталонну реалізацію Java web-сервера додатків під назвою JServ. Після виходу технології J2EE весь код був переданий компанії Apache Software Foundation, а продукт змінив свою назву на Tomcat.

На даний момент Tomcat є еталонною реалізацією Java web-сервера і входить в групу проектів Apache під назвою Jakarta.

1.5 Стійкість до помилок

Багатоплатформність середовища Web висуває надзвичайно високі вимоги до надійності програм. Як наслідок, при розробці Java приорітет був відданий

можливості створення стійких до помилок програм. Java звільняє програміста від хвилювань з приводу багатьох поширених причин, які викликають помилки програмування. Як вже згадувалося, Java є строго типізованою мовою програмування. Ще виконуюча система Java бере на себе «прибирання сміття», тобто автоматично звільняє пам’ять, яка була розподілена динамічно. Звичайно, це дещо знижує ефективність коду, але запобігає типовим помилкам, коли програміст забуває звільнити виділену пам’ять, або, навпаки, звільняє пам’ять, яка ще використовується. Java підтримує об’єктно-орієнтовану обробку виключних ситуацій подібно до C++. Але на відміну від C++ в Java обробка виключних ситуацій є обов’язкувою. Тобто неможливо скомпілювати програму, яка відкриває файл, не обробивши можливі помилки типу «файл не знайдено», які виникають при цьому. Добре написана Java-програма може сама обробляти всі помилки часу виконання.

1.6 Підтримка багатозадачності

Java розробляялася з орієнтацією на вимоги до створення інтерактивних програм, які працюють з мережею. З цією метою Java підтримує багатозадачність програмування, яке дозволяє легко розробляти програми, що викинують багато процесів одночасно. Виконання Java-програми засновано на елегантному, але в той самий час високоорганізованому рішенні багатопроцесової синхронізації, яке дозволяє вам створювати високоефективні інтерактивні системи.

У Java реалізовано кілька цікавих рішень, що дозволяють писати код, що виконує одночасно масу різних функцій і не забуває при цьому стежити за тим, що і коли повинно відбутися. У мові Java для рішення проблеми синхронізації процесів застосований найбільш елегантний із усіх коли-небудь, винайдених методів, що дозволяє конструювати прекрасні інтерактивні системи. Прості в

звертанні витончені підпроцеси Java дають можливість реалізації в програмі конкретної поведінки, не відволікаючись при цьому на побудову глобальної циклічної обробки подій.

1.7 Незалежність від архітектури

Основним питанням для розробників Java стало питання довготривалості та переносимості. Одна з головних проблем, із якою зустрілися програмісти, полягала в відсутності гарантій того, що написана сьогодні програма завтра працюватиме з тим же успіхом, причому на тій самій машині. Оновлення операційної системи, модернізація процесора та зміна об'єму оперативної пам'яті можуть призвести до збою програми. Розробники Java, прагнули змінити цю ситуацію і прийняли декілька важких рішень відносно мови Java та процесу виконання Java-програми. Їх мета полягала в тому, щоб «одного разу написане працювало всюди, в любий час і завжди». Внаслідок цього Java є системою, яка легко розширюється за рахунок створення нових стандартних класів та бібліотек.

1.8 Переваги інтерпретованості в поєднанні з високою продуктивністю

Як вже згадувалось, Java дозвляє створювати незалежні від платформи програми шляхом компіляції в проміжне представлення, яке називається байткодом. Багато попередніх спроб знайти розв'язок проблеми незалежності від платформи були зроблені за рахунок продуктивності. Інтерпретуючі системи, подібні до BASIC, Perl, страждають на майже неподоланний дефіцит продуктивності. Це було враховано при створенні Java. Незважаючи на те, що

Java є інтерпретованою мовою, генерація байткодів була ретельно оптимізована в такий спосіб, щоб одержуваний байткод можна було легко перекладати в машинний код, який працює з дуже високою продуктивністю. Виконуючі системи такого роду не втрачають жодних переваг переносимого коду.

1.9 Розподіленість

Мова Java призначена для створення програм, які працюють в розподіленому середовищі Internet на базі протоколів TCP/IP. Насправді доступ до ресурсів за допомогою URL відрізняється від доступу до файлу. Крім того в Java наявний засіб передачі повідомлень в межах внутрішнього адресного простору. Це дозволяє забеспечити віддалене виконання процедур. Ці інтерфейси включені у пакет RMI (remote metod invocation). Цей засіб привносить високий рівень абстракції в програмування дл я середовища клієнт/сервер.

Java-програми несуть у собі значний обсяг інформації про типи часу виконання (run-time type information), яка використовується для дозволу доступу до об’єктів під час роботи програми. Це дозволяє забезпечити безпечну та оптимальну динамічну компоновку. В такий спосіб досягається захищеність середовища виконання аплетів.

1.10 Аплети

Однією з найбільших переваг мови Java – можливість створення аплетів, маленьких програм, які працюють всередині WEB-браузера. Проте, на аплети

накладені певні обмеження в зв’язку з тим, що вони виконуються на комп’ютері користувача.

Обмеження :

- Аплет не має доступу до жорсткого диску. Проте для них існує система цифрових підписів, за допомогою якої користувач може визначати чи дані аплети отримані з надійних джерел, а отже може зняти більшість обмежень
- Аплетам необхідно певний час, щоб загрузитися з Інтернету. Для зменшення цього часу всі дані, які необхідно аплету для роботи, як правило включають в jar-архів, що дозволяє швидше загрузити аплет.

Переваги:

- Для аплетів немає необхідності встановлювати їх як інші програми. Цим можна скористатися, коли необхідно постійно загружати обновлені версії програм.
- Немає необхідності хвилюватись, що загружений аплет виконає потенційно небезпечні дії. Основні дії, які можуть привести до втрати важливої інформації, пошкодження чи зміна вмісту файлів для аплетів є заборонені.
- В мові Java існує два способи створити аплет.
- Створити підклас суперкласу Applet (визначений в пакеті java.applet.Applet)
- Створити підклас суперкласу Japplet (визначений в пакеті javax.swing)

Відмінність між цими двома класи полягає лише в тому, що перший - був визначений в ранніх версіях Java, і не підтримував повну незалежність від архітектури, вигляд GUI не дуже вдалий. Другий був створений в більш пізніших версіях Java і підтримує архітектурну незалежність, має краший вигляд GUI. На сьогоднішній день в основному для створення аплетів використовують другий спосіб для створення аплетів.

На відміну від консольних додатків, які виконуються без опитування подій від користувача, аплети очікують якісь події від користувача, і змінюють всій стан у відповідності до цих дій. Він просто „висить” в пам’яті комп’ютера і перевіряє

події які відбуваються(рух мишки, натискування кнопок миші та клавіш та ін), і при поступленні перевантаженої дії змінює свій стан. [9]

Для управлінням розміщення елементів у вікні аплету використовують систему менеджерів розміщення. В Java існують багато видів менеджерів розміщення. Опишемо коротко властивості деяких з них.

- FloyLayout – розміщує компоненти послідовно зліва на право, доки вони поміщаються в одному рядку. Потім переходить на наступний рядок і т.д.
- GridLayout – представляє вікно аплета як таблицю N*M, де числа N та M вказуються при створенні менеджера. Після цього розміщує елементи аплету в клітинках так само, як попередній. Різниця полягає в тому, що для кожного елемента можна вказати його розміщення в даній таблиці.
- BolderLayout – призначений для полярного розмішення елементів. Для нього визначені значення, які відповідають сторонам світу. Тому ми можемо розміщувати елементи з певним типом вирівнювання.
- CardLayout – призначений для блокнотного розміщення компонентів. Всі елементи, які входять в контейнер визначаються як сторінки блокнота, тому в кожний момент часу видний тільки якась з сторінок.
- GridBagLayout – найбільш універсальний метод розміщення компонент. Даний менеджер розглядає контейнер як таблицю, і кожний компонент може займати більше ніж одну клітинку.
- Взаємодія аплетів між собою може бути побудована на основі двох методів:
- Створення каналу між аплетами (PipeInputStream/PipeOutputStream)
- За допомогою сокетів (якщо це окремі додатки)
- За допомогою елементів HTML-сторінки через параметри аплета.

1.11 Доступність інструментарію та ефективність розробок

Зазначена вище простота програмування на Java є причиною того, що розробки на Java коштуватимуть дешевше аналогічних на більш потужніх мовах

програмування. Цьому ж сприяє і переносимість програм на Java, оскільки ліквідуються витрати пов’язані з адаптацією програми на конкретній платформі. До того ж інтегровані програми-оболонки для розробки Java програм коштують набагато дешевше (70-100$) ніж аналогічні продукти C++, Delphi (~1000$). А набір інструментарію для пакетної компіляції Java програм JDK (Java Development Kit) є взагалі freeware. Тому платформу Java можна рекомендувати як ідеальну для створення некомерційних програмних продуктів, зокрема для галузі освіти.

1.12 Перспективи застосування

Програми на Java можуть знайти різне застосування в навчальному процесі: інтерактивні навчаючі програми (HTML в поєднанні з Java), програми-тести і особливо ділові ігри. Додаткові переваги можна отримати, якщо пистати ці програми у вигляді аплетів, які ініціалізуються з Web сервера внутрішньої мережі Intranet. В такий спосіб можна уникнути інсталяції програми на багатьох комп’ютерах - користувач просто запускає Web-браузер і завантажує потрібну сторінку. Для тестових програм, написаних на Java з використанням архітектури клієнт/сервер можна підвищити ступінь конфіденційності. База даних тестових запитань знаходиться на сервері в каталозі з обмеженим доступом. Коли користувач завантажує аплет, він автоматично підключається до програми-сервера, яка виконується на сервері і може видавати запитання з бази даних у відповідь на запит користувача. В такий спосіб унеможливлюється викрадення бази даних, за умови відсутності фізичного доступу до серверу у користувачів.

Окреме питання - навчальні ділові ігри. Під такою грою розумітимемо гру, за участю кількох користувачів, в якій мається на увазі динамічний обмін інформацією між ними. Система безпеки Java накладає обмеження, внаслідок

якого аплет може встановлювати з’єднання лише з хостом, з якого він був загружений і ні з яким більше. Але це обмеження легко обходиться: на сервері виконується програма-сервер, з якою з’єднуються усі клієнти і через яку здійснюється обмін інформацією. Таким чином така програма повинна мати архітектуру клієнт/сервер.

1.13 Завдання на розробку системи ідентифікації продуктів

Необхідно розробити Android-додаток для людей з вадами зору з такою функціональністю:

- реалізація всіх функцій додатку для роботі з камерою;
- сканування штрих-кодів продуктів;
- виведення інформації;
- оголошення результатів сканування;
- інтерфейс додатку має бути простим і зрозумілим для людей з вадами зору.

Висновки за розділом 1

Таким чином, Java-технологія є дуже перспективною для застосквання в розробках неокомерційного спрямування. Обмеженість інструментарію Java не проявляється в проектах невеликого обсягу і з лихвою компенсується простотою програмування розподілених програм, які працюють з мережею Internet/Intranet . Переносимість Java програм спрощує обмін навчальними програмами між різними навчальними закладими, відкрииває можливість сумісних розробок та створення стандартних навчальних програм, наприклад для шкіл. Крім того, дуже привабливою з точки зору ефективності, залишається ідея втілення в навчальних закладах мережевих комп’ютерів (Network Computers), які працюють на базі Java,

замість звичних персональних комп’ютерів (Personal Computers). На останок можна згадати про те що Java добре підтримує національні абетки, оскільки розроблялася для інтернаціональної мережі Internet. Ця обставина теж є важливою в умовах України.

Проте, мову програмування Java не рекомендується використовувати в системах, збій в роботі яких може призвести до смерті, травм чи значних фізичних ушкоджень (наприклад, програмне забезпечення для керування атомними електростанціями, польотами, систем життєзабезпечення чи систем озброєння) через ненадійність програм, написаних на мові програмування Java.

2 МЕТОДОЛОГІЯ РОЗРОБКИ ANDROID-ДОДАТКІВ

2.1 Вибір схеми моделі життєвого циклу

Сьогодні існує чимало стандартних моделей проектування, які дозволяють поетапно, крок за кроком, реалізувати будь-який проект від ідеї до її втілення. Їх вибір залежить тільки від розробників і тих цілей, які вони переслідують.

Прикладами таких моделями є: «Каскадна модель», модель «Спіраль» і однією з примітних таких моделей для пр. оектування додатку є модель Уолта Діснея, вона складається з трьох етапів:

– концептуальне проектування;
– логічне проектування;
– фізичне проектування.

Етапи слідують послідовно один за іншим (рис. 2.1), але в деяких випадках можливий перехід до наступної стадії без закінчення попередньої. Це може відбуватися, наприклад, коли є декілька розробників, кожен з яких працює зі своєю частиною додатку. У будь-якому випадку, після закінчення етапу фізичного проектування слід повернутися до початку і внести відповідні корективи.

Рисунок 2.1 – Етапи проектування

Концептуальне проектування. Часом буває складно оцінити ефективність додатку. Необхідно знати і розуміти критерії оцінки для того, щоб визначити хорошим чи поганим є розроблений ресурс. Є універсальний критерій, який досить точно характеризує ефективність додатку – це досягнення розробниками додатку поставлених перед ними цілей. У цьому випадку додаток перетворюється на якісний інструмент, який виконує покладені на нього функції. Концептуальне проектування служить для вказівки цілей, завдань додатку та визначення аудиторії, на яку він розрахований.

На цьому етапі проектування слід описати наступне:

- основні і другорядні цілі;
- дії, які необхідно вжити для досягнення поставлених цілей;
- аудиторію додатку;
- інтереси груп користувачів;
- розділи додатку
- критерії досягнення мети.

Після визначення поставлених цілей й інтересів користувачів, можна скласти список сервісів й розділів, які будуть розташовуватися на додатокі.

Логічне проектування. Визначені розділи додатку, на попередньому етапі, поки не впорядковані і не структуровані, тому їх потрібно привести до зручного та зрозумілого виду.

Логічне проектування включає організацію інформації в додатку, побудови її структури і навігації по розділах. На даному етапі слід задатися питанням, яким чином буде впорядкована інформація. Варіанти можуть бути самими різними і залежати від типу даних і переваг творців додатку: за часом, розділами, в алфавітному порядку, певним групам або іншими критеріями.

Одночасне використання різних способів охоплює більшу аудиторію і дозволяє швидше знайти потрібну інформацію на додатокі. На цьому етапі слід описати наступне:

– тип структури додатку (лінійна, ієрархічна, контекстна, інша);
– назви розділів;
– що буде містити в собі кожен розділ;
– організація та зв'язок розділів між собою;
– що буде розміщено на певних розділах додатку.

Кінцевим результатом логічного проектування є блок схема або структурна діаграма, що показують взаємозв'язок різних частин додатку.

Фізичне проектування. Даний етап пов'язаний з пошуком проблем, а не їх рішень, пов'язаних, здебільшого, з технічною реалізацією додатку. На цьому етапі слід описати наступне:

– технології, які будуть застосовуватися в додатку;
– програмне забезпечення, що використовується;
– можливі проблеми та способи їх усунення;
– як буде оновлюватися інформація.

Після завершення цього етапу слід повернутися до концептуального проектування і перевірити, чи не потрібно внести зміни, у зв'язку з переосмисленням проекту на інших стадіях.

При виборі схеми моделі життєвого циклу (ЖЦ) для конкретної предметної області, вирішуються питання включення важливих для створюваного продукту видів робіт або не включення несуттєвих робіт. На сьогодні основою формування нової моделі ЖЦ для конкретної прикладної системи є стандарт ISO/IEC12207, що задає повний набір процесів (понад 40), що охоплює всі можливі види робіт і завдань, пов'язаних з побудовою програмного середовища (ПС), починаючи з аналізу предметної області і закінчуючи виготовленням відповідного продукту. Цей стандарт містить основні та допоміжні процеси (рис. 2.2 та рис. 2.3).

Рисунок 2.2 – Схема основних процесів ЖЦ ПС

На рис. 2.3 представлені процеси, пов'язані безпосередньо з розробкою ПС. До категорії основних процесів відносяться також "первинні" процеси, що визначають порядок підготовки договору на розробку ПС, моніторинг діяльності постачальників ПС замовнику.

Стандарт ISO/IEC12207 надає структуру процесів ЖЦ, але не зобов'язує використовувати всі процеси в моделі ЖЦ ПЗ або в конкретній методології розробки програмного забезпечення (ПЗ).

Рисунок 2.3 – Схема допоміжних процесів ЖЦ ПЗ

При створенні додатку було використано стандарт ISO/IEC12207 [17]. На основі цього розроблено методологію додатку (рис. 2.4).

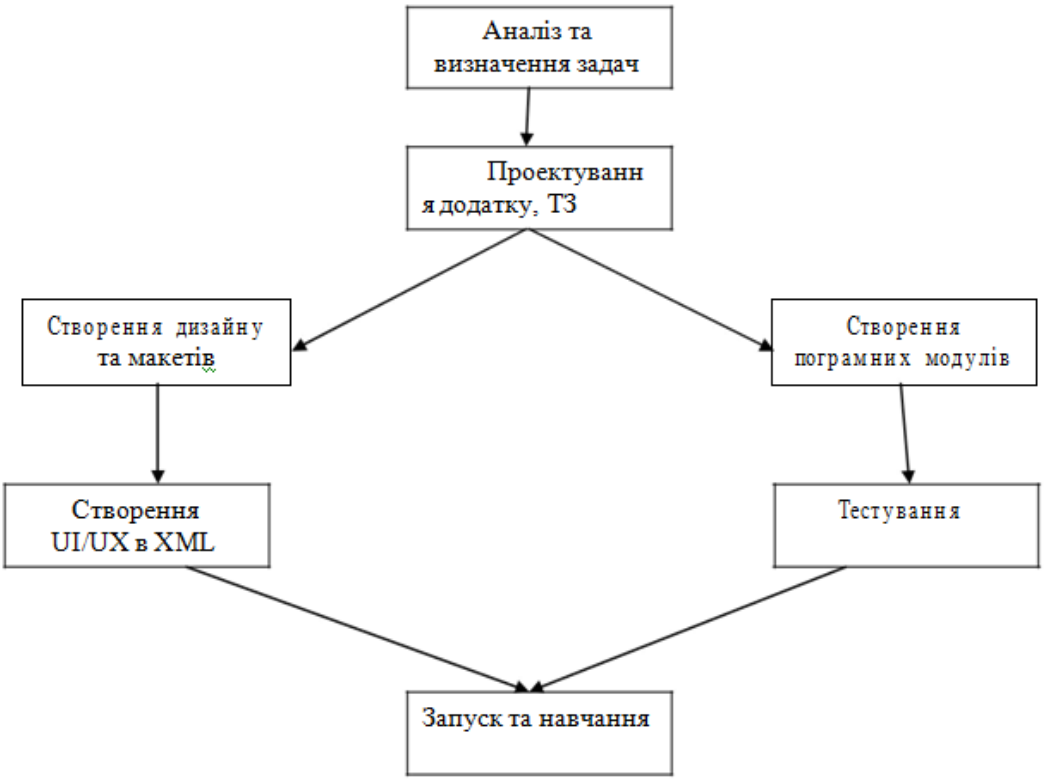

Рисунок 2.4 – Методологія створення додатку

Методологія розробки мобільного додатку включає наступні етапи:

– аналіз та визначення задач (планування) – визначення мети створення додатку, тематики і призначення майбутнього додатку, можливостей реалізації проекту;

– проектування та розробка технічного завдання – визначення структури додатку, матеріалів, вибір програмних засобів для його створення;

– веб-дизайн – створення графічних елементів макету додатку, стилів і елементів навігації;
– розробка програмного коду, модулів, бази даних і інших елементів додатку необхідних в проекті, вибір та інтегрування CMS;
– заповнення додатку матеріалами;

– тестування і розміщення додатку на хостингу в мережі інтернет;
– запуск та навчання.

2.2Аналіз та визначення задач мобільного додатку

На етапі планування, перш за все, слід визначити призначення майбутнього додатку. Тут доцільно визначити, буде додаток вузьконаправленним чи різні його розділи будуть з різнім функціоналом і яким саме. На етапі планування можна виділити наступні задачі:

– фіксація цілей і завдання додатку;
– максимально детальний опис майбутньої аудиторії;
– визначення того, що ми чекаємо від користувачів додатку;
– аналіз додатків – конкурентів;
– планування того, чим наш майбутній додаток буде відрізнятися від конкурентів;
– вибір сервісів та функцій;
– аналіз структури і функціоналу додатку;
– створення візуального плану ключових розділів;
– оцінка зручності використання додатку користувачем;
– планування способу отримання зворотного зв'язку;

– оцінка перспективи розвитку додатку .

2.3 Середовище розробки Android Studio

Android Studio - є офіційним IDE для розробки додатків Android, на основі IntelliJ IDEA. На вершині можливостей, які ви очікуєте від IntelliJ, Android Студія пропонує:

– гнучка Gradle-система збірки
– побудувати варіанти і кілька АПК покоління файлу
– шаблони коду, щоб допомогти вам побудувати загальні риси додатку
– багатий редактор макетів з підтримкою перетягування і падіння редагування теми
– інструменти, щоб фіксувати продуктивність, зручність використання,
– сумісність версії, і інші проблеми
– Proguard

Вбудована підтримка для Google Cloud Platform, що дозволяє легко інтегрувати Google Cloud повідомленнями.

Android SDK - включає в себе різноманітні бібліотеки, документацію та інструменти, які допомагають розробляти мобільні додатки для платформи Android.

API Android SDK-API бібліотеки Android, що надаються для розробки додатків.

Документація SDK-включає велику довідкову інформацію, що деталізує, що включено в кожен пакет і клас і як це використовувати при розробці додатків.

AVD (Android Virtual Device) -інтерактівний емулятор мобільного пристрою Android. Використовуючи емулятор, можна запускати і тестувати програми без використання реального Android пристрою.

Development Tools - SDK включає кілька інструментальних засобів для розробки, які дозволяють компілювати і налагоджувати створювані додатки.

Sample Code - Android SDK надає типові додатки, які демонструють деякі з можливостей Android, і прості програми, які показують, як використовувати індивідуальні особливості API у вашому коді.

Перед початком розробки додатків для Android корисно зрозуміти загальний підхід платформи до управління зміною API. Також важливозрозуміти Android API Level (Ідентифікатор рівня API) і його роль у забезпеченні сумісності вашого застосування з пристроями, на яких воно буде встановлюватися.

Рівень API - цілочисельне значення, яке однозначно визначає версію API платформи Android. Платформа забезпечує структури API, які додатки можуть використовувати для взаємодії з системою Android. Кожна наступна версія платформи Android може містити оновлення API.

Оновлення API-структури розроблені так, щоб новий API залишався сумісним з більш ранніми версіями API. Таким чином, більшість змін в API є сукупною і вводить нові функціональні можливості або виправляє попередні. Оскільки частина API постійно оновлюється, застарілі API не рекомендуються до використання, але не видаляються з міркувань сумісності з наявними додатками.

Рівень API, який використовує додаток для Android, визначається цілочисловим ідентифікатором, який вказується у файлі конфігурації кожного Android-додатки.

Перед початком розробки додатків для Android корисно зрозуміти загальний підхід платформи до управління зміною API. Також важливо
зрозуміти Android API Level (Ідентифікатор рівня API) і його роль у забезпеченні сумісності вашого застосування з пристроями, на яких воно буде встановлюватися.

Рівень API - цілочисельне значення, яке однозначно визначає версію API платформи Android. Платформа забезпечує структури API, які додатки можуть

використовувати для взаємодії з системою Android. Кожна наступна версія платформи Android може містити оновлення API.

Оновлення API-структури розроблені так, щоб новий API залишався сумісним з більш ранніми версіями API. Таким чином, більшість змін в API є сукупною і вводить нові функціональні можливості або виправляє попередні. Оскільки частина API постійно оновлюється, застарілі API не рекомендуються до використання, але не видаляються з міркувань сумісності з наявними додатками.

Рівень API, який використовує додаток для Android, визначається цілочисловим ідентифікатором, який вказується у файлі конфігурації кожного Android-додатки.

У таблиці 2.1 наведено відповідність рівня API і версії платформи Android.

Таблиця 2.1 – Відповідність версії платформи та рівня API

Версія платформи	Рівень API
Android 6.0	23
Android 5.0	21
Android 4.3	19
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

Крім емулятора, SDK також включає безліч інших інструментальних засобів для налагодження та установки створюваних додатків [15, 16] Якщо При розробці програми для Android за допомогою IDE Android Studio, багато інструментів командного рядка, що входять до складу SDK, вже використовуються при складанні і компіляції проекту. Однак крім них SDK містить ще ряд корисних інструментів для розробки і налагодження додатків:

Android - важливий інструмент розробки, що запускається з командного рядка, який дозволяє створювати, видаляти і конфігурувати віртуальні пристрої, створювати і оновлювати Android проекти (при роботі поза середовища Eclipse) і оновлювати Android SDK новими платформами,доповненнями і документацією;

Dalvik Debug Monitor Service (DDMS) - інтегрований з Dalvik Virtual Machine, стандартної віртуальні машини платформи Android, цей інструмент дозволяє управляти процесами на емуляторі, а також допомагає в налагодженні додатків. Можна використовувати цей сервіс для завершення процесів, вибору певного процесу для налагодження, генерування трасувань даних, перегляду "купи" або інформації про потоки, робити скріншоти емулятора та багато іншого;

– Hierarchy Viewer- візуальний інструмент, який дозволяє налагоджувати

– оптимізувати користувальницький інтерфейс розробляється. Він показує візуальне дерево ієрархії уявлень, аналізує швидкодію перемальовування графічних зображень на екрані і може виконувати ще багато інших функцій для аналізу графічного інтерфейсу додатків;

– Layoutopt- інструмент командного рядка, який допомагає оптимізувати схеми розмітки та ієрархії розміток в створюваному додатку. Необхідний для вирішення проблем при створенні складних графічних інтерфейсів, які можуть зачіпати продуктивність програми;
– Draw 9-patch - графічний редактор, який дозволяє легко створювати

– NinePatch графіку для графічного інтерфейсу розроблюваних додатків;

– sqlite3 - інструмент для доступу до файлів даних SQLite, створених і використовуваних додатками для Android;

– Traceview - цей інструмент видає графічний аналіз трасувань логів, які можна генерувати з додатків;

– mksdcard - інструмент для створення образу диска, який ви можете використовувати в емуляторі для симуляції наявності зовнішньої карти пам'яті (наприклад, карти SD).

– Найбільш важливий з них цих інструментів - є емулятор мобільного пристрою, однак до складу SDK входять і інші інструменти для налагодження, упаковки та інсталяції ваших додатків на емулятор.

2.4 Розробка архітектури проекту

Мобільна розробка поділяється на два етапи: верстка статичних екранів в XML та програмування бізнес логіки на Java.

Розробка здійснюється у популярній IDE від Google – Android Studio. Затверджений дизайн «нарізається» на окремі малюнки, з яких згодом складається xml-розмітка для додатку. У результаті створюється код, який можна переглядати за допомогою мобільного телефону. А типові сторінки згодом будуть використовуватися як шаблони.

Потім XML файли наповнюються потрібною інформацією динамічно за допомогою Java.

Java - об'єктно-орієнтована мова програмування. Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь якій віртуальній Java-машині незалежно від комп'ютерної архітектури (рис.2).

Java - дуже гнучка мова з відкритим вихідним кодом. Для розробки програми існують архітектурні підходи до проекту, які вирішують проблему розробки стандартного програмного забезпечення. Шаблони - це абстрактні схеми, вони не є кодом, але вони допомагають розділити логіку програми на певні модулі. При використанні шаблону проекту, цей шаблон адаптується у відповідності зі своїми певними потребами (рис.2.5).

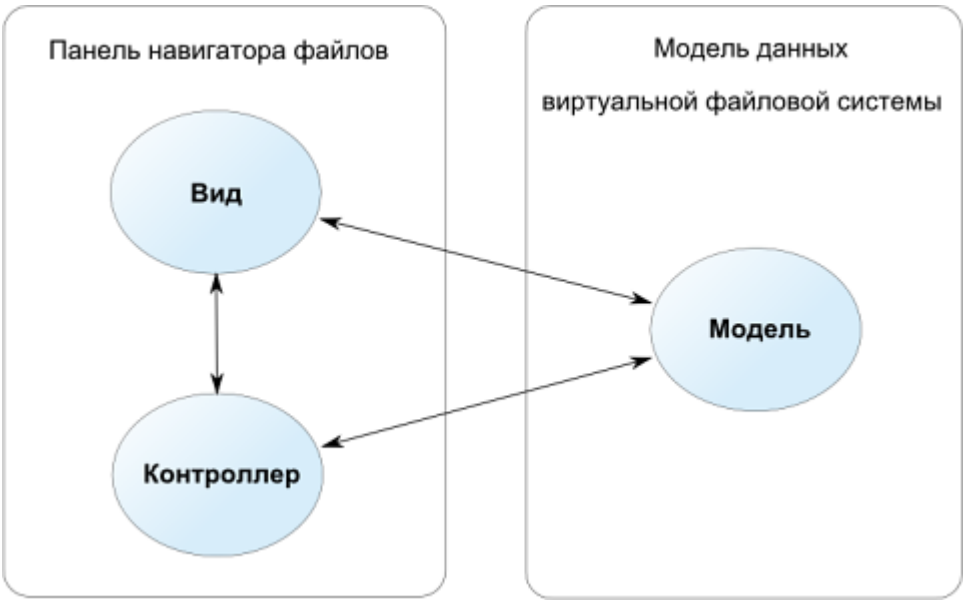

Рисунок 2.5 - Структура шаблону Модель - Вид - Контролер Модель (Model)

Модель являє собою дані, з якими оперує додаток. Це можуть бути як дані бази даних, так і будь-яка інша структура даних, що описує деякі об'єкти системи і їх стан.

Вид (View)

Вид являє собою компонент системи для відображення стану моделі в зрозумілому людині поданні. Це можуть бути діалоги, форми та інші візуальні (наприклад, синтезатор мови) засоби взаємодії людини з системою. Вид не змінює дані безпосередньо (режим тільки читання), дані змінюються за допомогою контролера.

Контролер (Controller)

Контролер є засобом, за допомогою якого користувачі взаємодіють з системою. Це може бути клавіатура, маніпулятор миша і т. Д. А також є керуючим елементом для обміну даними та повідомленнями між видом і моделлю.

Фактично, зв'язка виду і контролера є інтерфейсом користувача. Причому, якщо компоненти виду зазвичай можна повторно використовувати в інших компонентах системи, то контролер часто є специфічним для даного конкретного випадку.

Модель не залежить ні від виду, ні від контролера, що дозволяє одночасно будувати різні інтерфейси користувача для взаємодії з однією і тією ж моделлю даних. Наприклад, можна зробити як Java SWT або SWING десктопних програм, так і WEB додаток для взаємодії одними і тими ж даними.

Розрізняють як пасивний, так і активний режими роботи з моделлю даних.

При пасивному режимі дані на клієнтській стороні перечитуються при виконанні деяких дій користувача, наприклад, запиту на виведення інформації про об'єкти.

При активному режимі вводяться додаткові обробники і слухачі повідомлень, які посилають компонентам рівня подання повідомлення про те, що дані моделі були змінені і потрібне оновлення даних на стороні клієнта "View".

Практичне застосування шаблону MVC має як позитивні, так і негативні сторони. Шаблон MVC може призвести до збільшення складності по забезпеченню синхронізації даних та подання до простих додатках.

2.5 Побудова та тестування написаного додатку

Після того, як написаний весь програмний код, необхідно створити apk-файл, встановити його на телефон та розпочати тестування. Для збірки проекту в Android Studio вбудовано збірник проектів Gradle, де всі конфігурації прописуються на мові Groovy.

Тестування додатку – це перевірка додатку різними методами і способами на працездатність. Тестування необхідно як новому, так і вже працюючому додатку, для отримання гарантії його працездатності.

До основних видів тестування відносяться наступні.

Тестування usability (перевірка зручності користування додатком). У ході такого тестування визначається якість виконання і зручність інтерфейсу додатку, а так само проводяться роботи з виявлення можливих помилок в структурі. Результати дають можливість визначити, наскільки правильно використовує додаток «середньостатистичний» користувач, і як швидко він може знайти потрібні йому функції.

Тестування на стійкість до великих навантажень. Цей тест імітує одночасне користування великою кількістю користувачів (сотень або навіть тисяч) для визначення працездатності ресурсу при великих навантаженнях або ж інтенсивна, довгочасна робота додатку в умовах не великих ресурсів.

Таке тестування обов'язково для новинних додатоків, форумів і ресурсів з передбачуваною великою аудиторією. У ході тестування перевіряється не стільки

сам ресурс, скільки комплексну роботу апаратної частини сервера, модулів, програмного ядра та інших компонентів додатку.

Тестування XML коду. Перевіряється весь додаток на наявність помилок у програмному коді і відповідність стандартам.

Тестування безпеки. Перевірка безпеки включає в себе тестування як самого додатку разом з, так і веб-сервера, операційної системи і всіх
мережевих і локальних сервісів. Для збереження інформації та стабільної роботи додатку тестування безпеки необхідно проводити регулярно.

Оскільки додаток, як і будь-який інший програмний продукт не може бути без помилок, необхідно своєчасно виявляти і усувати ці помилки.

Висновки за розділом 2:

В розділ 2 було обрано оптимальну модель життєвого циклу та описано її основні етапи. Крім цього, було описано методологію розробки додатку, виділено основні задачі що підлягають подальшому аналізу та виконанню.

Також були описані середовище, мова та основні етапи розробки додатку для ОС Android.

3 СИСТЕМА ІДЕТИНФІКАЦІЇ ПРОДУКТІВ НА БАЗІ ANDROID

3.1 Аналіз та визначення задач створення додатку

Майже кожна людина має смартфон на на операційній системі Android. У світі багато людей з вадами зору тому і був створений додаток BarcodeScanner. Цей додаток взаэмодіє з камерою смартфона і допоможе їм визначати, що за продукція перед ними.

Призначенням додатку є:

- Реалізація всіх функцій додатку для роботи з камерою.
- Сканування штрих-коду.
- Виведення та оголошення результату сканування.

На основі проведеного аналізу методів та засобів розробки мобільного додатку, огляду і аналізу існуючих додатків і методології створення програмних продуктів було розроблено технічне завдання додатку.

Метою розробки є створення мобільного додатку для людей з вадами зору.

Данний додаток призначений для:

- Сканування штрих кодів.
- Виведення інформації
- Оголошення виведеної информациі.

3.2Створеня макету додатку

Для створення макету додатку було використано програму Adobe Photoshop. Опираючись на технічне завдання, було створено ключові екрани. Ними виявились – головний екран (рис. 3.1), та екран камери (рис 3.2).

Рисунок 3.1 - Головний екран додатку

Дизайн головного екрану є одночасно насичений елементами і простий. На цьому екрани зображені на лаштунки камери та кнопка яка вмикає камеру для сканування.

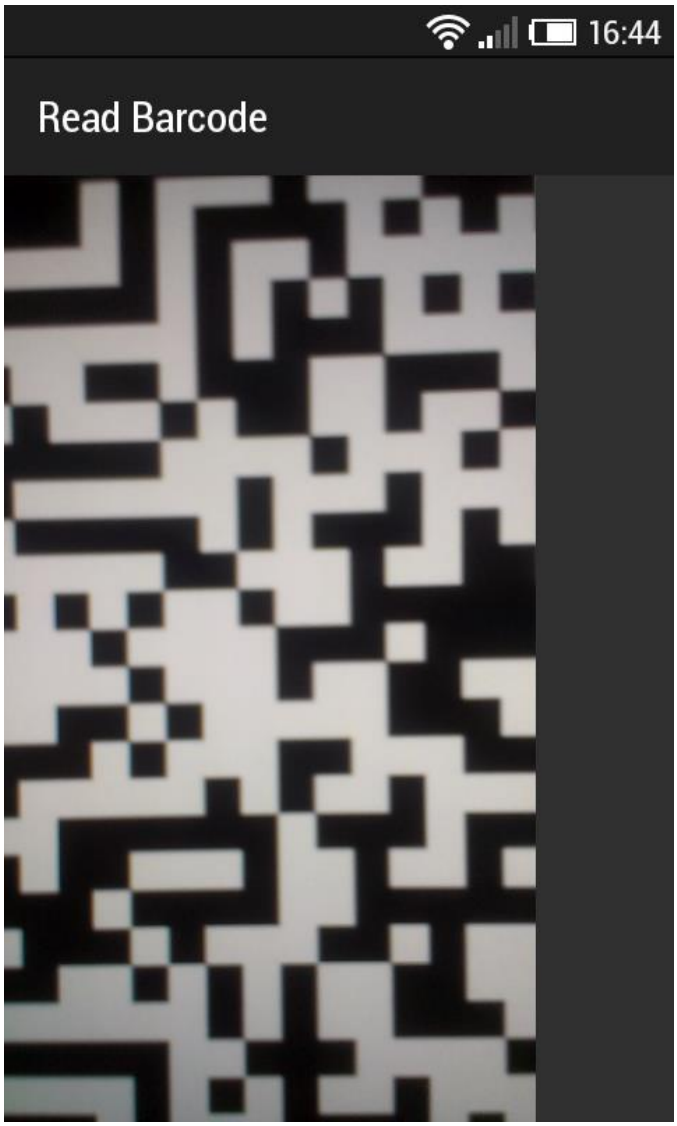

Рисунок 3.2 - Екран камери

Екран сканування штрих-коду фіксує штрих код и виділяє його рамкою.

3.3 Реалізація програмного продукту

Після створення дизайну програмного продукту потрібно перейти до його реалізації. Для розробки мобільних додатків під платформу Андроїд, зокрема

представленого в даній дипломній роботі, потрібне середовище розробки та відповідні плагіни до нього. Було обране середовище Android Studio, оскільки воно представлене виробника ОС Android, розвивається значно швидше від своїх конкурентів та має більші функціональні можливості, більш стабілним. Дане середовище потребує інсталяції плагінів ADT та Android SDK, але воно є більш стабільним та функціональним . Перш за все потрібно було створено архітектуру майбутнього програмного продукту, ми створили пакети, кожен із яких відповідає за певний функціонал. В пакетах були створені класи, кожен із яких мав вузький обов'язок в програмі, а також прописали відносини між об'єктами.

Класи для виявлення і аналізу штрих-кодів доступні в просторі імен com.google.android.gms.vision.barcode. Основою є клас BarcodeDetector.Він виконує обробку об'єктів Frame і повертаємасив штрих-кодів SparseArray .Тип Barcode є єдиний загальновизнанийштрих-код і його значення.У разі 1D штрих-кодів, таких як коди UPC, це буде просто номер, який закодований в штрих-коді. Його значення є в поле rawValue, в той час як тип штрих-коду (тобто його кодування) можна знайтив поле format.Для 2D штрих-кодів, які містять структурованідані, такі як QR-коди - В полі valueFormat встановлюється певний тип значення, відповідного полюданих.Так, наприклад, якщо виявлено тип URL, то поле valueFormat повернеконстанту URL, а об'єкт Barcode.UrlBookmark буде містити значенняURL-адреси.Крім URL-адрес, існує безліч різних типівданих, які QR-код може зберігати. Наприклад, поштова адреса, дату і час події календаря, захід у календарі, інформацію контакту, номер телефону, місце розташування на карті і інші дані.Використання в додатку Mobile Vision API дозволяє зчитуватиштрих-коди в будь-якому положенні.Важливо відзначити, що синтаксичний розбір всіх штрих-кодіввиконується локально, тому вам не потрібно використовувати з'єднання з сервером для читання даних з коду.Наприклад, при зчитуванні лінійного штрих-коду PDF-417,який може вмістити до 1 КБ тексту, можна відразуж отримати всю закодовану в ньому інформацію.

Отже, для розробки програми нам знадобиться:

- Середовище розробки Android Studio.
- Смартфон на Android 4.2.2 абоновішої версії - або – Емулятор.
- Android Остання версія Android SDK, включаючи компонент SDK tools. Отримуємо його за допомогою Android SDK Manager в AndroidStudio.
- I Google Play Services SDK. Отримати його також в Android SDK Manager в AndroidStudio.
-

Створюємо новий проект в Android Studio.При створенні вибираємо шаблон Empty Activity (рис 3.3).

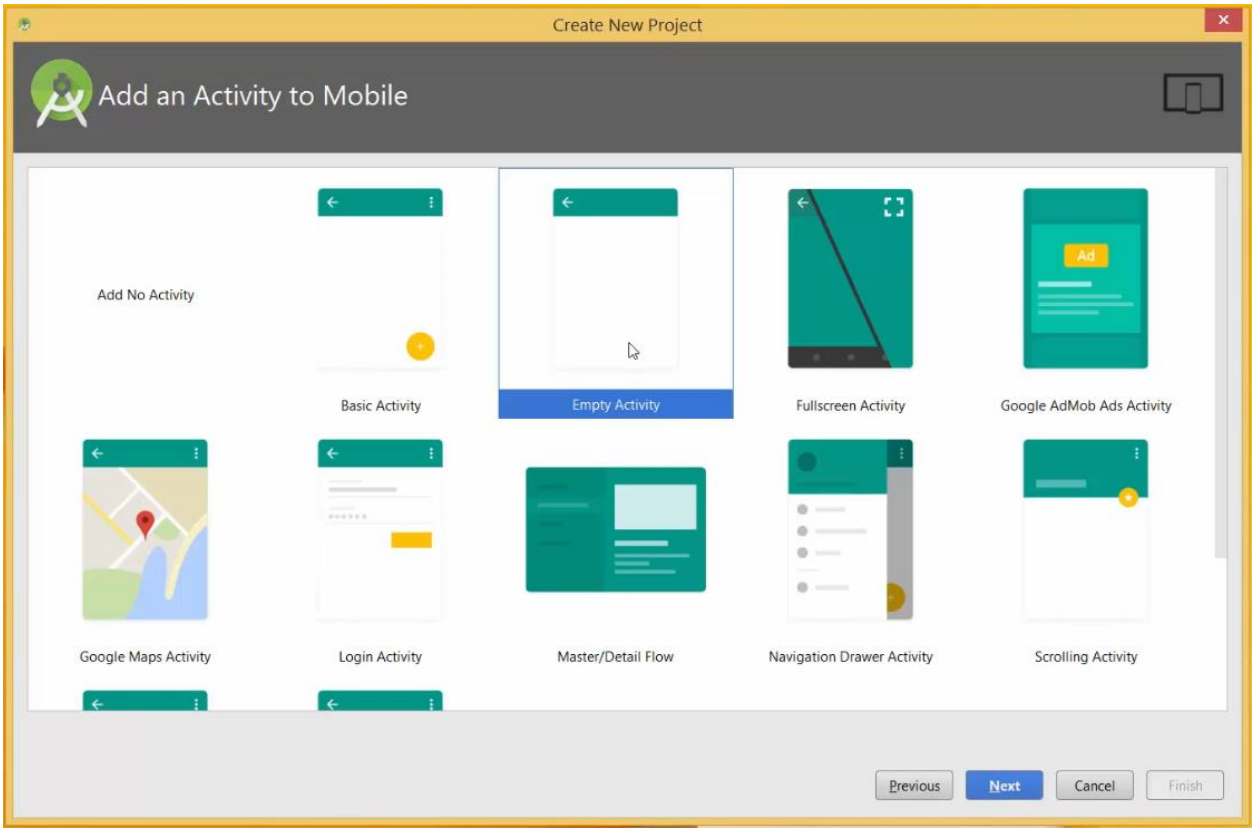

Рисунок 3.3 - Шаблон Empty Activity

Наступним кроком інсталяції переконатися, що наш додаток може використовувати служби Google Play, до складу якихвходить Mobile Vision API.Для цього потрібно оновити файл build.gradle нашого проекту.

compile **'com.android.support:support-v4:24.2.0'**
compile **'com.google.android.gms:play-services-vision:9.4.0+'**
compile **'com.android.support:design:24.2.0'**

Служби Google Play (рис. 3.4) часто оновлюються, і щоб отримати останню версію, в Android Studio вибераємо інструменти> Android> SDK Manager.Потім знайдіть рядок для сервісів Google Play .

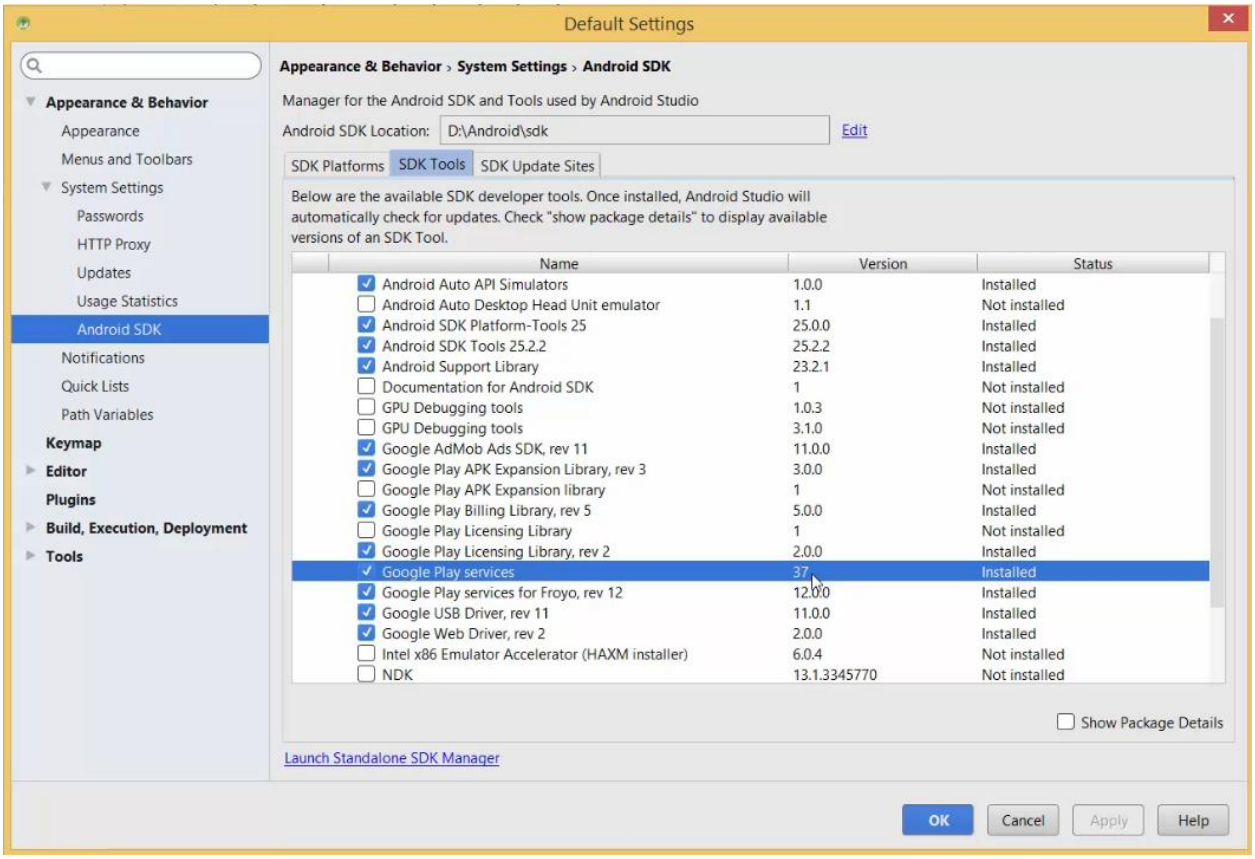

Рисунок 3.4 - Служби Google Play

Тепер створимо призначений для користувача інтерфейс. В Android Studio виберіть папку «Res» і відкрийте її вкладену папку «layout».Тут ми побачимо

«activity_main.xml».Відкриваємо його в редакторі макетів.Ми можемо бачити, що наш макет містить текстове поле <TextView> . Потрібно змінити макет таким чином.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
   android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"

tools:context="com.google.android.gms.samples.vision.barcodereader.MainActivit
y">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="@string/barcode_header"
      android:id="@+id/статус_сообщения"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_centerHorizontal="true" />

   <TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/barcode_value"
    android:layout_below="@+id/статус_сообщения"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="110dp"
    android:layout_alignRight="@+id/статус_сообщения"
    android:layout_alignEnd="@+id/статус_сообщения"                    />

<Button

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/read_barcode"
    android:id="@+id/SCANNER"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"                             />

<CheckBox

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/auto_focus"
    android:id="@+id/АВТО_ФОКУС"
    android:layout_below="@+id/barcode_value"
    android:layout_alignParentLeft="true"
```

```
    android:layout_alignParentStart="true"
    android:layout_marginTop="66dp"
    android:checked="false"                                    />

  <CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/use_flash"
    android:id="@+id/ВСПЫШКА"
    android:layout_alignTop="@+id/АВТО_ФОКУС"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:checked="false"                                    />

</RelativeLayout>
```

Тепер тут буде крім текстового поля також кнопка і зображення. Для всіх екранних компонентів прописуємо ідентифікатори,щоб потім звертатися до них в коді.При натисканні на кнопку буде відбуватися завантаження і обробка зображення штрих-коду, який буде відображатися в ImageView.Після завершення обробки штрих-коду інформація,лічена з нього, буде відображатися в TextView.Зазвичай додатки для зчитування штрих-кодів отримують зображенняз камери пристрою, або обробляють превью камери.Для реалізації цього буде потрібно досить багато коду.

У файлі MainActivity.java в методі onCreate додаємо наступний код.

```
public class MainActivity extends Activity implements View.OnClickListener {
  private CompoundButton autoFocus;
```

private CompoundButton **useFlash**;

private TextView **statusMessage**;

private TextView **barcodeValue**;

Це налаштовує обробник подій (onClick), що спрацьовуєколи користувач натискає кнопку.

Додаток складається з восьми класів (рис. 3.5). Три з них відповідають за інтерфейс і роботу з камерою.

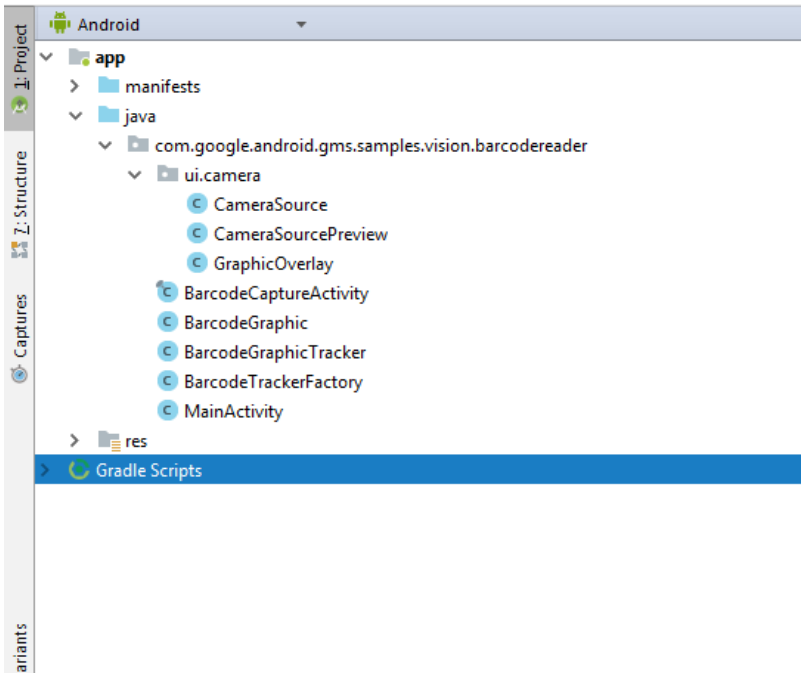

Рисунок. 3.5 - Класи додатку

– Клас Camerasource надає управління камерою для здобуття попереднього перегляду.
– Клас Camerasourcepreview відповідає за відображення превью на екрані.
– Клас Graphicoverlay відображує графічні об'єкти поверх зв'язаного попереднього перегляду камери.

- Клас Mainactivity відображує стартове вікно з налаштуваннями і кнопкою запуску сканування, і отримує дані штриха-коди для розміщення в Textview. Клас Barcodetrackerfactory реалізує патерн «Фабрика» і використовується для створення трекерів штриха-коди — поодинці для кожної штриха-коди.
- Клас Barcodegraphictracker це трекер, який використовується для виявлення штриха-код на екрані, і їх відстежування для накладення графіки, а також видалення графіки, коли штрих-код покидає зону видимості.
- Клас Barcodegraphic використовується для отрісовки екземпляра зображення, що накладається на штрих-код, з врахуванням його положення, розміру і ідентифікатора.
- Клас Barcodecaptureactivity — це актівіті, яке запускається при натисненні кнопки прочитування штриха-коди в стартовому вікні додатка. Це актівіті відображує превью камери і визначає штрих-коди на нім, виполнzет їх прочитування і накладення графічних рамок на кожен штрих-код за допомогою вищеперелічених класів.

Одразу після створення архітектури, було зверстано основні статичні екрани в XML. І фінальним етапом стала реалізація всіх методів классів. Основний функціонал программи міститься на мобільному телефоні.
Програмний код основних модулів наведений у додатках нижче.

3.4 Тестування программного продукту

Після розробки було проведено тестування, після якого були зафіксовані всі помилки для обов’язкового виправлення. Після виправлення помилок, які

були виявлені на першому етапі перевірки, додаток повторно було протестовано.

Програмний продукт був повністю протестований на usability. Також були протестовані всі функції, здійснено навантаженне тестування.

Після тестування данного програмного продукту було встановлено, що він не має критичних або ж блокуючих помилок.

Висновки за розділом 3

В даному розділі сформовано основні вимоги до програмного продукту. Розглянуто основні функції розроблюваного ПЗ та описано всі етапи розробки.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної роботи бакалавра було створення системи он-лайн замовлення їжі, і як результат було створено онлайн - систему, яка виконує всі зазначені вимоги. Так як в процесі проектування використовувався персональний комп’ютер, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп’ютера на якому буде розроблятися та використовуватися розроблена система.

4.1 Аналіз стану умов праці

4.1.1Вимоги до приміщення

Таблиця 4.1 - Розміри приміщення

Найменування	Значення
Довжина, м	4,4
Ширина, м	2,8
Висота, м	2,5
Площа, $м^2$	12,32
Об’єм, $м^3$	30,8

Згідно з ДСН 3.3.6.042-99 [18] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі. Для дотримання вимог пожежної безпеки встановлено порошковий вогнегасник та систему автоматичної пожежної сигналізації.

4.1.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за ДСанПіН 3.3.2.007-98 [19] (табл. 4.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне Значення	Нормативне Значення
Висота робочої поверхні, мм	700	680 ÷ 800
Висота простору для ніг, мм	650	не менше 600
Ширина простору для ніг, мм	540	не менше 500
Глибина простору для ніг, мм	660	не менше 650
Висота поверхні сидіння, мм	420	400 ÷ 500
Ширина сидіння, мм	410	не менше 400
Глибина сидіння, мм	420	не менше 400
Висота поверхні спинки, мм	500	не менше 300
Ширина опорної поверхні спинки, мм	400	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400

Відстань від очей до екрану дисплея, мм	750	700 ÷ 800

4.2 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.2.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00-1.28-10 [22], яке встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої. Основними робочими характеристиками персонального комп'ютера є:

– робоча напруга U=+220В +-5%;

– робочий струм I=2А;

– споживана потужність P=350 Вт.

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна Оцінка	Нормативні Документи
1	2	3	4
Фізичні:			
підвищена або знижена вологість повітря	-//-	3	[18]
підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	3	[20] [21]
Психофізіологічні:			
1	2	3	4
нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- формулювання теми; - пошук інформацію про предметну область; - пошук інформації про наявні аналоги; - проектування структур та алгоритмів; - виконання роботи; - оформлення записки.	4	[21] [22]
фізичні (статичне - сидіння)	порушення умов організації робочого часу	2	[19] [22]

	(безперервна робота)		

Робочі місця в обов’язковому порядку повинні відповідати вимогам до санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, що затверджені постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [19].

4.2.2 Пожежна безпека

Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80…100℃). При проходженні електричного струму по провідниках і деталей виділяється тепло, що в умовах їх високої щільності може привести до перегріву, і може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання між струмоведучими частинами обладнання (шини, електроди).

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), надійно захищені діелектричними щитками та/або сітками з метою недопущення потрапляння працівника під напругу.

В приміщенні наявна затверджена «План-схема евакуації з кабінету (приміщення)».

Горючими матеріалами в приміщенні, де розташовані ЕОМ, є:

1) поліамід - матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420℃;

2) полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335℃, температура самозаймання 530℃;

3) склотекстоліт ДЦ - матеріал друкарських плат, важкогорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

4) пластикат кабельний №489 - матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1;

5) деревина - будівельний і обробний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255℃, температура самозаймання 399℃.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до НАПБ Б.03.002-2007 [23] відносяться до пожежонебезпечної зони класу П-IIа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється II ступень вогнестійкості.

Причинами можливого загоряння і пожежі можуть бути:

1) несправність електроустановки;

2) конструктивні недоліки устаткування;

3) коротке замикання в електричних мережах;

4) запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол [24].

4.2.3 Електробезпека

Виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та

інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

4.3 Гігієнічні вимоги до параметрів виробничого середовища

4.3.1 Мікроклімат

Мікроклімат робочих приміщень - це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт 1а. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають ДСН 3.3.6.042-99 [18] і наведені в табл. 4.4:

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період Року	Категорія Робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря,

				м/с
Холодна	Легка-1а	22-24	40-60	0,1
Тепла	Легка-1а	23-25	40-60	0,1

4.3.2 Освітлення

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових - 1/10:

$$S_b = \left(\frac{1}{5} / \frac{1}{10}\right) * S_n \quad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2 .

$S_n = a \cdot b = 4{,}4 \cdot 2{,}8 = 12{,}32$ м^2 ,

$S = 1/10 \cdot 25 = 1{,}232$ м^2 .

Приймаємо 1 вікно площею S=1,6 м^2.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 4,4 м, ширина 2,8 м, світильниками ЛПО2П, оснащеними лампою типа ЛБ (одна - 80 Вт) з світловим потоком 5400 лм. Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n = \frac{E * S * Z * K}{F * U * M} \quad (4.2)$$

де Е - нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S - освітлювана площа, м^2 ; S = 12,32 м^2 ;

Z - поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K - коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U - коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п.

- 0,575 M - число люмінесцентних ламп в світильнику - 1;

F - світловий потік лампи - 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 * 12{,}32 * 1{,}15 * 1{,}5}{5400 * 0{,}575 * 1} \approx 2{,}0$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, оснащених лампами типа ЛБ (одна - 80 Вт) зі світловим потоком 5400 лм.

4.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при V приміщення > 40 м^3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП. Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

- експлуатацію сертифікованого обладнання;

- дотримання заходів електробезпеки;

- забезпечення оптимальних параметрів мікроклімату;

- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об'єм приміщення 20 м^3 , то потрібно подати не менш як 30 м^3 /год повітря;

б) якщо об'єм приміщення у межах від 20 до 40 м^3, то потрібно подати не менш як 20 м^3/год повітря;

в) якщо об'єм приміщення становить понад 40 м^3, допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі;

- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

- не тягнути за мережевий кабель, щоб витягти вилку з розетки;

- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

- не залишати включені електроприлади без нагляду;

4.5.1 Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [25], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з}$.:

$$R_{шт.з} = \frac{R_д * R_{пр.з}}{R_{пр.з} - R_д} \qquad (4.3)$$

де $R_{пр.з}$. - опір природних заземлювачів;

$R_д$ - допустимий опір заземлення.

Якщо природні заземлювачі відсутні, то $R_{шт.з}$.=$R_д$.

Підставивши числові значення у формулу (А.3), отримуємо:

$$R_{шт.з} = \frac{4 * 40}{40 - 4} \approx 4Ом$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом*м (табличне значення).

3) Розрахунковий питомий опір ґрунту, Ррозр, Ом·м, визначається відповідно для вертикальних заземлювачів ρрозр.в, і горизонтальних Ррозр.г, Ом·м за формулою:

$$P_{розр.} = \Psi * p \qquad (4.4)$$

де ψ - коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $Р_{розр.в}$=1,7 і горизонтальних $Р_{розр.в}$ =5,5 Ом·м

$Р_{озр.в}$ = 1, 7 * 40 = 68 Ом/м

$Р_{озр.г}$ = ·5,5 * 40 = 220 Ом/м

4) Розраховується опір розтікання струму вертикального заземлювача Rв, Ом, за (4.5).

$$R_B = \frac{P_{розр.в}}{2*\pi*1_B} * \left(\ln \frac{2*1_B}{d_{CT}} + \frac{1}{2} * \ln \frac{4*t+1_B}{4*t-1_B} \right) \qquad (4.5)$$

де $l_в$ - довжина вертикального заземлювача (для труб - 2−3 м; $l_в$=3 м);

$d_{ст}$ - діаметр стержня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t - відстань від поверхні землі до середини заземлювача, яка визначається за ф. (4.6):

$$t = h_B + \frac{1_B}{2} \qquad (4.6)$$

де $h_в$ - глибина закладання вертикальних заземлювачів (0,8 м); тоді

$$t = 0.8 + \frac{3}{2} = 2{,}3м$$

$$R_B = \frac{68}{2*\pi*3} * \left(\ln \frac{2*3}{0{,}05} + \frac{1}{2} * \ln \frac{4*2{,}3+3}{4*2{,}3-3} \right) = 18{,}5Ом$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання ηв:

$$n = \frac{2*R_B}{R_д} = \frac{2*18{,}5}{4} = 9{,}25 \qquad (4.7)$$

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_B = \frac{2*R_B}{R_д*\eta_B} = \frac{2*18{,}5}{4*0{,}57} = 16{,}2 \approx 16 \qquad (4.8)$$

7) Визначається довжина з’єднувальної стрічки горизонтального заземлювача lc, м:

$$l_c = 1{,}05 * L_B * (n_B - 1) \quad (4.9)$$

де $l_в$ - відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ - необхідна кількість вертикальних заземлювачів.

$$l_c = 1{,}05 * 3 * (16 - 1) \approx 48 м$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_Г = \frac{P_{розр.г}}{2 * \pi * l_c} * \ln \frac{2 * l_c^2}{d_{ст} * h_Г} \quad (4.10)$$

де $d_{см}$ - еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ - глибина закладання горизонтальних заземлювачів (0,5 м);

$l_с$ - довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м

$$R_Г = \frac{220}{2 * \pi * 48} * \ln \frac{2 * 48^2}{0{,}95 * 0{,}15 * 0{,}5} = 8{,}1 Ом$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_с$. відповідно до необхідної кількості вертикальних заземлювачів $n_в$. Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_B * R_Г}{R_B * \eta_c + R_Г * n_B * \eta_B} \quad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ Ом, а саме:

$$R_{заг} = \frac{18{,}5 * 8{,}1}{18{,}5 * 0{,}3 + 8{,}1 * 16 * 0{,}57} = 1{,}9 \le R_д$$

Висновки за розділом 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри

і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та значення температури, вологості й рухливості повітря, необхідна кількість ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

У дипломному проекті був розроблений андроїд-додаток, який представляє собою набір класів, за допомогою яких сканує штрих-коди та оголошує результат сканування, і який презначений для людей з вадами зору.

Данний додаток був спроектований та реалізований для людей, які погано бачать, він допоможе їм відсканувати продукцію та оголосити результат сканування.

Перед створенням додатку був проведений аналіз методів та засобів розробки мобільних додатків, огляд і аналіз існуючих додатків і методології створення програмних продуктів.

В четвертому розділі розглянуті заходи з охорони праці та безпеки в надзвичайних ситуаціях.

ПЕРЕЛІК ДЖЕРЕЛ І ПОСИЛАНЬ

1. Новая среда разработки Android Studio [Електронний ресурс]: cnews.ru Режим доступу: http://www.cnews.ru/top/2013/05/20/novaya_sreda_razrabotki_android_studio_sozdana_na_osnove_rossiyskogo_proekta529258

2. Android Studio – среда мобильной разработки на базе технологий JetBrains [Електронний ресурс]: soft.mail.ru – Режим доступу: http://soft.mail.ru/pressrl_page.php?id=51774

3. Genymotion — лучший эмулятор Андроид на ПК [Електронний ресурс]: 4idroid.com – Режим доступу: http://4idroid.com/genymotion-luchshij-emulyator-android-na-pk/

4. http://www.fandroid.info/upravlyayushhie-struktury-obshhie-svedeniya-o-tsiklah/

5. Операційна система Google Android [Електронний ресурс]: ALLS.IN.UA Режим доступу: http://alls.in.ua/13729-operacijjna-sistema-google-android.html

6. Доля Android на рынке смартфонов [Електронний ресурс]: Ferra.ru – Режим доступу: http://www.ferra.ru/ru/techlife/news/2014/01/30/strategy- analytics-2013-smartphone/#.U6DXXyjiLt0

7. Родной язык Андроида [Електронний ресурс]: toster.ru – Режим доступу: https://toster.ru/q/8860

8. Васильев А.Н. Java. Объектно-ориентированное

9. Genymotion — функциональный эмулятор Android для PC и Mac [Електронний ресурс]: androidinsider.ru – Режим доступу: http://androidinsider.ru/video/genymotion-funktsionalnyiy-emulyator-android-dlya-pc-i-mac.html

10. Роджерс Р., Ломбардо Д. Android. Разработка приложений [Текст] / Роджерс Р., Ломбардо Д. – М.: ЭКОМ Паблишерз, 2010. — 400 с.

11. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин

12. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ЕОМ
13.https://developers.google.com/android/reference/com/google/android/gms/vision/package-summary/

ДОДАТОК А

Лістинг програми

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
   android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"

tools:context="com.google.android.gms.samples.vision.barcodereader.MainActivit
y">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="@string/barcode_header"
      android:id="@+id/статус_сообщения"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_centerHorizontal="true" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:id="@+id/barcode_value"
      android:layout_below="@+id/статус_сообщения"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_marginTop="110dp"
      android:layout_alignRight="@+id/статус_сообщения"
      android:layout_alignEnd="@+id/статус_сообщения" />

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
    android:text="@string/read_barcode"
    android:id="@+id/SCANNER"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true" />

  <CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/auto_focus"
    android:id="@+id/АВТО_ФОКУС"
    android:layout_below="@+id/barcode_value"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="66dp"
    android:checked="false" />

  <CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/use_flash"
    android:id="@+id/ВСПЫШКА"
    android:layout_alignTop="@+id/АВТО_ФОКУС"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:checked="false" />

</RelativeLayout>

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.google.android.gms.samples.vision.barcodereader"
  android:installLocation="auto" >

  <uses-feature android:name="android.hardware.camera" />

  <uses-permission android:name="android.permission.CAMERA" />

  <application
    android:allowBackup="true"
    android:fullBackupContent="false"
    android:hardwareAccelerated="true"
    android:icon="@drawable/icon"
    android:label="MultiTrackerApp"
    android:supportsRtl = "true"
```

```
      android:theme="@style/Theme.AppCompat" >
      <meta-data
         android:name="com.google.android.gms.version"
         android:value="@integer/google_play_services_version" />
      <meta-data
         android:name="com.google.android.gms.vision.DEPENDENCIES"
         android:value="barcode" />

      <activity
         android:name=  ".MainActivity"
         android:label="@string/title_activity_main" >
      "@string/title_activity_main" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

      <activity android:name=".BarcodeCaptureActivity"
         android:label="Read Barcode"/>
   </application>

</manifest>
package com.google.android.gms.samples.vision.barcodereader.ui.camera;

import android.Manifest;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.os.Build;
import android.os.SystemClock;
import android.support.annotation.Nullable;
import android.support.annotation.RequiresPermission;
import android.support.annotation.StringDef;
import android.util.Log;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.WindowManager;
```

```
import com.google.android.gms.common.images.Size;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.Frame;

import java.io.IOException;
import java.lang.Thread.State;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class CameraSource {
   @SuppressLint("InlinedApi")
   public static final int CAMERA_FACING_BACK =
CameraInfo.CAMERA_FACING_BACK;
   @SuppressLint("InlinedApi")
   public static final int CAMERA_FACING_FRONT =
CameraInfo.CAMERA_FACING_FRONT;

   private static final String TAG = "OpenCameraSource";


   private static final int DUMMY_TEXTURE_NAME = 100;


   private static final float ASPECT_RATIO_TOLERANCE = 0.01f;

   @StringDef({
      Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE,
      Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO,
      Camera.Parameters.FOCUS_MODE_AUTO,
      Camera.Parameters.FOCUS_MODE_EDOF,
      Camera.Parameters.FOCUS_MODE_FIXED,
      Camera.Parameters.FOCUS_MODE_INFINITY,
      Camera.Parameters.FOCUS_MODE_MACRO
   })
   @Retention(RetentionPolicy.SOURCE)
   private @interface FocusMode {}

   @StringDef({
      Camera.Parameters.FLASH_MODE_ON,
```

```
    Camera.Parameters.FLASH_MODE_OFF,
    Camera.Parameters.FLASH_MODE_AUTO,
    Camera.Parameters.FLASH_MODE_RED_EYE,
    Camera.Parameters.FLASH_MODE_TORCH
})
@Retention(RetentionPolicy.SOURCE)
private @interface FlashMode {}

private Context mContext;

private final Object mCameraLock = new Object();


private Camera mCamera;

private int mFacing = CAMERA_FACING_BACK;



private int mRotation;

private Size mPreviewSize;


private float mRequestedFps = 30.0f;
private int mRequestedPreviewWidth = 1024;
private int mRequestedPreviewHeight = 768;


private String mFocusMode = null;
private String mFlashMode = null;


private SurfaceView mDummySurfaceView;
private SurfaceTexture mDummySurfaceTexture;

private Thread mProcessingThread;
private FrameProcessingRunnable mFrameProcessor;



private Map<byte[], ByteBuffer> mBytesToByteBuffer = new HashMap<>();
```

```
    public static class Builder {
        private final Detector<?> mDetector;
        private CameraSource mCameraSource = new CameraSource();

        public Builder(Context context, Detector<?> detector) {
            if (context == null) {
                throw new IllegalArgumentException("No context supplied.");
            }
            if (detector == null) {
                throw new IllegalArgumentException("No detector supplied.");
            }

            mDetector = detector;
            mCameraSource.mContext = context;
        }


        public Builder setRequestedFps(float fps) {
            if (fps <= 0) {
                throw new IllegalArgumentException("Invalid fps: " + fps);
            }
            mCameraSource.mRequestedFps = fps;
            return this;
        }

        public Builder setFocusMode(@FocusMode String mode) {
            mCameraSource.mFocusMode = mode;
            return this;
        }

        public Builder setFlashMode(@FlashMode String mode) {
            mCameraSource.mFlashMode = mode;
            return this;
        }

        public Builder setRequestedPreviewSize(int width, int height) {

            final int MAX = 1000000;
            if ((width <= 0) || (width > MAX) || (height <= 0) || (height > MAX)) {
                throw new IllegalArgumentException("Invalid preview size: " + width +
"x" + height);
            }
            mCameraSource.mRequestedPreviewWidth = width;
            mCameraSource.mRequestedPreviewHeight = height;
```

```
            return this;
        }

        public Builder setFacing(int facing) {
            if ((facing != CAMERA_FACING_BACK) && (facing !=
CAMERA_FACING_FRONT)) {
                throw new IllegalArgumentException("Invalid camera: " + facing);
            }
            mCameraSource.mFacing = facing;
            return this;
        }


        public CameraSource build() {
            mCameraSource.mFrameProcessor = mCameraSource.new
FrameProcessingRunnable(mDetector);
            return mCameraSource;
        }
    }


    public interface ShutterCallback {

        void onShutter();
    }


    public interface PictureCallback {

        void onPictureTaken(byte[] data);
    }


    public interface AutoFocusCallback {

        void onAutoFocus(boolean success);
    }


    public interface AutoFocusMoveCallback {

        void onAutoFocusMoving(boolean start);
    }
```

```
    public void release() {
        synchronized (mCameraLock) {
            stop();
            mFrameProcessor.release();
        }
    }


    @RequiresPermission(Manifest.permission.CAMERA)
    public CameraSource start() throws IOException {
        synchronized (mCameraLock) {
            if (mCamera != null) {
                return this;
            }

            mCamera = createCamera();


            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
                mDummySurfaceTexture = new
SurfaceTexture(DUMMY_TEXTURE_NAME);
                mCamera.setPreviewTexture(mDummySurfaceTexture);
            } else {
                mDummySurfaceView = new SurfaceView(mContext);
                mCamera.setPreviewDisplay(mDummySurfaceView.getHolder());
            }
            mCamera.startPreview();

            mProcessingThread = new Thread(mFrameProcessor);
            mFrameProcessor.setActive(true);
            mProcessingThread.start();
        }
        return this;
    }


    @RequiresPermission(Manifest.permission.CAMERA)
    public CameraSource start(SurfaceHolder surfaceHolder) throws IOException {
        synchronized (mCameraLock) {
            if (mCamera != null) {
                return this;
            }
```

```
            mCamera = createCamera();
            mCamera.setPreviewDisplay(surfaceHolder);
            mCamera.startPreview();

            mProcessingThread = new Thread(mFrameProcessor);
            mFrameProcessor.setActive(true);
            mProcessingThread.start();
        }
        return this;
    }


    public void stop() {
        synchronized (mCameraLock) {
            mFrameProcessor.setActive(false);
            if (mProcessingThread != null) {
                try {

                    mProcessingThread.join();
                } catch (InterruptedException e) {
                    Log.d(TAG, "Frame processing thread interrupted on release.");
                }
                mProcessingThread = null;
            }


            mBytesToByteBuffer.clear();

            if (mCamera != null) {
                mCamera.stopPreview();
                mCamera.setPreviewCallbackWithBuffer(null);
                try {


                    if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB) {
                        mCamera.setPreviewTexture(null);

                    } else {
                        mCamera.setPreviewDisplay(null);
                    }
                } catch (Exception e) {
                    Log.e(TAG, "Failed to clear camera preview: " + e);
                }
```

```
            mCamera.release();
            mCamera = null;
        }
    }
}


public Size getPreviewSize() {
    return mPreviewSize;
}



public int getCameraFacing() {
    return mFacing;
}

public int doZoom(float scale) {
    synchronized (mCameraLock) {
        if (mCamera == null) {
            return 0;
        }
        int currentZoom = 0;
        int maxZoom;
        Camera.Parameters parameters = mCamera.getParameters();
        if (!parameters.isZoomSupported()) {
            Log.w(TAG, "Zoom is not supported on this device");
            return currentZoom;
        }
        maxZoom = parameters.getMaxZoom();

        currentZoom = parameters.getZoom() + 1;
        float newZoom;
        if (scale > 1) {
            newZoom = currentZoom + scale * (maxZoom / 10);
        } else {
            newZoom = currentZoom * scale;
        }
        currentZoom = Math.round(newZoom) - 1;
        if (currentZoom < 0) {
            currentZoom = 0;
        } else if (currentZoom > maxZoom) {
            currentZoom = maxZoom;
        }
```

```
            parameters.setZoom(currentZoom);
            mCamera.setParameters(parameters);
            return currentZoom;
        }
    }



public void takePicture(ShutterCallback shutter, PictureCallback jpeg) {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            PictureStartCallback startCallback = new PictureStartCallback();
            startCallback.mDelegate = shutter;
            PictureDoneCallback doneCallback = new PictureDoneCallback();
            doneCallback.mDelegate = jpeg;
            mCamera.takePicture(startCallback, null, null, doneCallback);
        }
    }
}

@Nullable
@FocusMode
public String getFocusMode() {
    return mFocusMode;
}


public boolean setFocusMode(@FocusMode String mode) {
    synchronized (mCameraLock) {
        if (mCamera != null && mode != null) {
            Camera.Parameters parameters = mCamera.getParameters();
            if (parameters.getSupportedFocusModes().contains(mode)) {
                parameters.setFocusMode(mode);
                mCamera.setParameters(parameters);
                mFocusMode = mode;
                return true;
            }
        }

        return false;
    }
}
```

```
@Nullable
@FlashMode
public String getFlashMode() {
   return mFlashMode;
}


public boolean setFlashMode(@FlashMode String mode) {
   synchronized (mCameraLock) {
      if (mCamera != null && mode != null) {
         Camera.Parameters parameters = mCamera.getParameters();
         if (parameters.getSupportedFlashModes().contains(mode)) {
            parameters.setFlashMode(mode);
            mCamera.setParameters(parameters);
            mFlashMode = mode;
            return true;
         }
      }

      return false;
   }
}

public void autoFocus(@Nullable AutoFocusCallback cb) {
   synchronized (mCameraLock) {
      if (mCamera != null) {
         CameraAutoFocusCallback autoFocusCallback = null;
         if (cb != null) {
            autoFocusCallback = new CameraAutoFocusCallback();
            autoFocusCallback.mDelegate = cb;
         }
         mCamera.autoFocus(autoFocusCallback);
      }
   }
}


public void cancelAutoFocus() {
   synchronized (mCameraLock) {
      if (mCamera != null) {
         mCamera.cancelAutoFocus();
      }
   }
}
```

```
    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    public boolean setAutoFocusMoveCallback(@Nullable AutoFocusMoveCallback
cb) {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
            return false;
        }

        synchronized (mCameraLock) {
            if (mCamera != null) {
                CameraAutoFocusMoveCallback autoFocusMoveCallback = null;
                if (cb != null) {
                    autoFocusMoveCallback = new CameraAutoFocusMoveCallback();
                    autoFocusMoveCallback.mDelegate = cb;
                }
                mCamera.setAutoFocusMoveCallback(autoFocusMoveCallback);
            }
        }

        return true;
    }

    private CameraSource() {
    }


    private class PictureStartCallback implements Camera.ShutterCallback {
        private ShutterCallback mDelegate;

        @Override
        public void onShutter() {
            if (mDelegate != null) {
                mDelegate.onShutter();
            }
        }
    }


    private class PictureDoneCallback implements Camera.PictureCallback {
        private PictureCallback mDelegate;

        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
```

```
            if (mDelegate != null) {
                mDelegate.onPictureTaken(data);
            }
            synchronized (mCameraLock) {
                if (mCamera != null) {
                    mCamera.startPreview();
                }
            }
        }
    }


    private class CameraAutoFocusCallback implements Camera.AutoFocusCallback {
        private AutoFocusCallback mDelegate;

        @Override
        public void onAutoFocus(boolean success, Camera camera) {
            if (mDelegate != null) {
                mDelegate.onAutoFocus(success);
            }
        }
    }


    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    private class CameraAutoFocusMoveCallback implements
Camera.AutoFocusMoveCallback {
        private AutoFocusMoveCallback mDelegate;

        @Override
        public void onAutoFocusMoving(boolean start, Camera camera) {
            if (mDelegate != null) {
                mDelegate.onAutoFocusMoving(start);
            }
        }
    }


    @SuppressLint("InlinedApi")
    private Camera createCamera() {
        int requestedCameraId = getIdForRequestedCamera(mFacing);
        if (requestedCameraId == -1) {
            throw new RuntimeException("Could not find requested camera.");
        }
```

```
        Camera camera = Camera.open(requestedCameraId);

        SizePair sizePair = selectSizePair(camera, mRequestedPreviewWidth,
mRequestedPreviewHeight);
        if (sizePair == null) {
            throw new RuntimeException("Could not find suitable preview size.");
        }
        Size pictureSize = sizePair.pictureSize();
        mPreviewSize = sizePair.previewSize();

        int[] previewFpsRange = selectPreviewFpsRange(camera, mRequestedFps);
        if (previewFpsRange == null) {
            throw new RuntimeException("Could not find suitable preview frames per
second range.");
        }

        Camera.Parameters parameters = camera.getParameters();

        if (pictureSize != null) {
            parameters.setPictureSize(pictureSize.getWidth(), pictureSize.getHeight());
        }

        parameters.setPreviewSize(mPreviewSize.getWidth(),
mPreviewSize.getHeight());
        parameters.setPreviewFpsRange(
                previewFpsRange[Camera.Parameters.PREVIEW_FPS_MIN_INDEX],
                previewFpsRange[Camera.Parameters.PREVIEW_FPS_MAX_INDEX]);
        parameters.setPreviewFormat(ImageFormat.NV21);

        setRotation(camera, parameters, requestedCameraId);

        if (mFocusMode != null) {
            if (parameters.getSupportedFocusModes().contains(
                    mFocusMode)) {
                parameters.setFocusMode(mFocusMode);
            } else {
                Log.i(TAG, "Camera focus mode: " + mFocusMode + " is not supported
on this device.");
            }
        }


        mFocusMode = parameters.getFocusMode();
```

```
        if (mFlashMode != null) {
            if (parameters.getSupportedFlashModes() != null) {
                if (parameters.getSupportedFlashModes().contains(
                        mFlashMode)) {
                    parameters.setFlashMode(mFlashMode);
                } else {
                    Log.i(TAG, "Camera flash mode: " + mFlashMode + " is not supported
on this device.");
                }
            }
        }


        mFlashMode = parameters.getFlashMode();

        camera.setParameters(parameters);


        camera.setPreviewCallbackWithBuffer(new CameraPreviewCallback());
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));

        return camera;
    }


    private static int getIdForRequestedCamera(int facing) {
        CameraInfo cameraInfo = new CameraInfo();
        for (int i = 0; i < Camera.getNumberOfCameras(); ++i) {
            Camera.getCameraInfo(i, cameraInfo);
            if (cameraInfo.facing == facing) {
                return i;
            }
        }
        return -1;
    }


    private static SizePair selectSizePair(Camera camera, int desiredWidth, int
desiredHeight) {
        List<SizePair> validPreviewSizes = generateValidPreviewSizeList(camera);
```

```
    SizePair selectedPair = null;
    int minDiff = Integer.MAX_VALUE;
    for (SizePair sizePair : validPreviewSizes) {
        Size size = sizePair.previewSize();
        int diff = Math.abs(size.getWidth() - desiredWidth) +
                Math.abs(size.getHeight() - desiredHeight);
        if (diff < minDiff) {
            selectedPair = sizePair;
            minDiff = diff;
        }
    }

    return selectedPair;
}


private static class SizePair {
    private Size mPreview;
    private Size mPicture;

    public SizePair(android.hardware.Camera.Size previewSize,
                    android.hardware.Camera.Size pictureSize) {
        mPreview = new Size(previewSize.width, previewSize.height);
        if (pictureSize != null) {
            mPicture = new Size(pictureSize.width, pictureSize.height);
        }
    }

    public Size previewSize() {
        return mPreview;
    }

    @SuppressWarnings("unused")
    public Size pictureSize() {
        return mPicture;
    }
}


private static List<SizePair> generateValidPreviewSizeList(Camera camera) {
    Camera.Parameters parameters = camera.getParameters();
    List<android.hardware.Camera.Size> supportedPreviewSizes =
            parameters.getSupportedPreviewSizes();
```

```
        List<android.hardware.Camera.Size> supportedPictureSizes =
                parameters.getSupportedPictureSizes();
        List<SizePair> validPreviewSizes = new ArrayList<>();
        for (android.hardware.Camera.Size previewSize : supportedPreviewSizes) {
            float previewAspectRatio = (float) previewSize.width / (float)
previewSize.height;

            // By looping through the picture sizes in order, we favor the higher resolutions.
            // We choose the highest resolution in order to support taking the full resolution
            // picture later.
            for (android.hardware.Camera.Size pictureSize : supportedPictureSizes) {
                float pictureAspectRatio = (float) pictureSize.width / (float)
pictureSize.height;
                if (Math.abs(previewAspectRatio - pictureAspectRatio) <
ASPECT_RATIO_TOLERANCE) {
                    validPreviewSizes.add(new SizePair(previewSize, pictureSize));
                    break;
                }
            }
        }


        if (validPreviewSizes.size() == 0) {
            Log.w(TAG, "No preview sizes have a corresponding same-aspect-ratio
picture size");
            for (android.hardware.Camera.Size previewSize : supportedPreviewSizes) {
                // The null picture size will let us know that we shouldn't set a picture size.
                validPreviewSizes.add(new SizePair(previewSize, null));
            }
        }

        return validPreviewSizes;
    }


    private int[] selectPreviewFpsRange(Camera camera, float desiredPreviewFps) {

        int desiredPreviewFpsScaled = (int) (desiredPreviewFps * 1000.0f);


        int[] selectedFpsRange = null;
        int minDiff = Integer.MAX_VALUE;
        List<int[]> previewFpsRangeList =
camera.getParameters().getSupportedPreviewFpsRange();
```

```
        for (int[] range : previewFpsRangeList) {
            int deltaMin = desiredPreviewFpsScaled -
range[Camera.Parameters.PREVIEW_FPS_MIN_INDEX];
            int deltaMax = desiredPreviewFpsScaled -
range[Camera.Parameters.PREVIEW_FPS_MAX_INDEX];
            int diff = Math.abs(deltaMin) + Math.abs(deltaMax);
            if (diff < minDiff) {
                selectedFpsRange = range;
                minDiff = diff;
            }
        }
        return selectedFpsRange;
    }


    private void setRotation(Camera camera, Camera.Parameters parameters, int
cameraId) {
        WindowManager windowManager =
                (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
        int degrees = 0;
        int rotation = windowManager.getDefaultDisplay().getRotation();
        switch (rotation) {
            case Surface.ROTATION_0:
                degrees = 0;
                break;
            case Surface.ROTATION_90:
                degrees = 90;
                break;
            case Surface.ROTATION_180:
                degrees = 180;
                break;
            case Surface.ROTATION_270:
                degrees = 270;
                break;
            default:
                Log.e(TAG, "Bad rotation value: " + rotation);
        }

        CameraInfo cameraInfo = new CameraInfo();
        Camera.getCameraInfo(cameraId, cameraInfo);

        int angle;
        int displayAngle;
```

```
        if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
            angle = (cameraInfo.orientation + degrees) % 360;
            displayAngle = (360 - angle) % 360; // compensate for it being mirrored
        } else {  // back-facing
            angle = (cameraInfo.orientation - degrees + 360) % 360;
            displayAngle = angle;
        }


        mRotation = angle / 90;

        camera.setDisplayOrientation(displayAngle);
        parameters.setRotation(angle);
    }


    private byte[] createPreviewBuffer(Size previewSize) {
        int bitsPerPixel = ImageFormat.getBitsPerPixel(ImageFormat.NV21);
        long sizeInBits = previewSize.getHeight() * previewSize.getWidth() *
bitsPerPixel;
        int bufferSize = (int) Math.ceil(sizeInBits / 8.0d) + 1;




        byte[] byteArray = new byte[bufferSize];
        ByteBuffer buffer = ByteBuffer.wrap(byteArray);
        if (!buffer.hasArray() || (buffer.array() != byteArray)) {

            throw new IllegalStateException("Failed to create valid buffer for camera
source.");
        }

        mBytesToByteBuffer.put(byteArray, buffer);
        return byteArray;
    }


    private class CameraPreviewCallback implements Camera.PreviewCallback {
        @Override
        public void onPreviewFrame(byte[] data, Camera camera) {
            mFrameProcessor.setNextFrame(data, camera);
        }
    }
```

```
private class FrameProcessingRunnable implements Runnable {
    private Detector<?> mDetector;
    private long mStartTimeMillis = SystemClock.elapsedRealtime();


    private final Object mLock = new Object();
    private boolean mActive = true;


    private long mPendingTimeMillis;
    private int mPendingFrameId = 0;
    private ByteBuffer mPendingFrameData;

    FrameProcessingRunnable(Detector<?> detector) {
        mDetector = detector;
    }


    @SuppressLint("Assert")
    void release() {
        assert (mProcessingThread.getState() == State.TERMINATED);
        mDetector.release();
        mDetector = null;
    }


    void setActive(boolean active) {
        synchronized (mLock) {
            mActive = active;
            mLock.notifyAll();
        }
    }


    void setNextFrame(byte[] data, Camera camera) {
        synchronized (mLock) {
            if (mPendingFrameData != null) {
                camera.addCallbackBuffer(mPendingFrameData.array());
                mPendingFrameData = null;
            }

            if (!mBytesToByteBuffer.containsKey(data)) {
```

```
            Log.d(TAG,
                "Skipping frame.  Could not find ByteBuffer associated with the
image " +
                "data from the camera.");
            return;
          }


          mPendingTimeMillis = SystemClock.elapsedRealtime() -
mStartTimeMillis;
          mPendingFrameId++;
          mPendingFrameData = mBytesToByteBuffer.get(data);


          mLock.notifyAll();
        }
      }

      /

      @Override
      public void run() {
        Frame outputFrame;
        ByteBuffer data;

        while (true) {
          synchronized (mLock) {
            while (mActive && (mPendingFrameData == null)) {
              try {

                mLock.wait();
              } catch (InterruptedException e) {
                Log.d(TAG, "Frame processing loop terminated.", e);
                return;
              }
            }

            if (!mActive) {

              return;
            }

            outputFrame = new Frame.Builder()
                .setImageData(mPendingFrameData, mPreviewSize.getWidth(),
```

```
                        mPreviewSize.getHeight(), ImageFormat.NV21)
                    .setId(mPendingFrameId)
                    .setTimestampMillis(mPendingTimeMillis)
                    .setRotation(mRotation)
                    .build();


            data = mPendingFrameData;
            mPendingFrameData = null;
          }



          try {
            mDetector.receiveFrame(outputFrame);
          } catch (Throwable t) {
            Log.e(TAG, "Exception thrown from receiver.", t);
          } finally {
            mCamera.addCallbackBuffer(data.array());
          }
        }
      }
    }
}
/*
 * Copyright (C) The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.google.android.gms.samples.vision.barcodereader;

import android.Manifest;
import android.annotation.SuppressLint;
```

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.Toast;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.common.api.CommonStatusCodes;
import
com.google.android.gms.samples.vision.barcodereader.ui.camera.CameraSource;
import
com.google.android.gms.samples.vision.barcodereader.ui.camera.CameraSourcePrevie
w;

import
com.google.android.gms.samples.vision.barcodereader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.MultiProcessor;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;

import java.io.IOException;
 public final class BarcodeCaptureActivity extends AppCompatActivity implements
BarcodeGraphicTracker.BarcodeUpdateListener {
   private static final String TAG = "Barcode-reader";


   private static final int RC_HANDLE_GMS = 9001;
```

```
    private static final int RC_HANDLE_CAMERA_PERM = 2;


    public static final String AutoFocus = "AutoFocus";
    public static final String UseFlash = "UseFlash";
    public static final String BarcodeObject = "Barcode";

    private CameraSource mCameraSource;
    private CameraSourcePreview mPreview;
    private GraphicOverlay<BarcodeGraphic> mGraphicOverlay;


    private ScaleGestureDetector scaleGestureDetector;
    private GestureDetector gestureDetector;


    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.barcode_capture);

        mPreview = (CameraSourcePreview) findViewById(R.id.preview);
        mGraphicOverlay = (GraphicOverlay<BarcodeGraphic>)
findViewById(R.id.graphicOverlay);


        boolean autoFocus = getIntent().getBooleanExtra(AutoFocus, false);
        boolean useFlash = getIntent().getBooleanExtra(UseFlash, false);


        int rc = ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA);
        if (rc == PackageManager.PERMISSION_GRANTED) {
            createCameraSource(autoFocus, useFlash);
        } else {
            requestCameraPermission();
        }

        gestureDetector = new GestureDetector(this, new CaptureGestureListener());
        scaleGestureDetector = new ScaleGestureDetector(this, new ScaleListener());

        Snackbar.make(mGraphicOverlay, "Tap to capture. Pinch/Stretch to zoom",
```

```
            Snackbar.LENGTH_LONG)
            .show();
    }


    private void requestCameraPermission() {
        Log.w(TAG, "Camera permission is not granted. Requesting permission");

        final String[] permissions = new String[]{Manifest.permission.CAMERA};

        if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.CAMERA)) {
            ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
            return;
        }

        final Activity thisActivity = this;

        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ActivityCompat.requestPermissions(thisActivity, permissions,
                        RC_HANDLE_CAMERA_PERM);
            }
        };

        findViewById(R.id.topLayout).setOnClickListener(listener);
        Snackbar.make(mGraphicOverlay, R.string.permission_camera_rationale,
                Snackbar.LENGTH_INDEFINITE)
                .setAction(R.string.ok, listener)
                .show();
    }

    @Override
    public boolean onTouchEvent(MotionEvent e) {
        boolean b = scaleGestureDetector.onTouchEvent(e);

        boolean c = gestureDetector.onTouchEvent(e);

        return b || c || super.onTouchEvent(e);
    }
```

```
    @SuppressLint("InlinedApi")
    private void createCameraSource(boolean autoFocus, boolean useFlash) {
        Context context = getApplicationContext();


        BarcodeDetector barcodeDetector = new
BarcodeDetector.Builder(context).build();
        BarcodeTrackerFactory barcodeFactory = new
BarcodeTrackerFactory(mGraphicOverlay, this);
        barcodeDetector.setProcessor(
                new MultiProcessor.Builder<>(barcodeFactory).build());

        if (!barcodeDetector.isOperational()) {

            Log.w(TAG, "Detector dependencies are not yet available.");


            IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
            boolean hasLowStorage = registerReceiver(null, lowstorageFilter) != null;

            if (hasLowStorage) {
                Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
                Log.w(TAG, getString(R.string.low_storage_error));
            }
        }


        CameraSource.Builder builder = new
CameraSource.Builder(getApplicationContext(), barcodeDetector)
                .setFacing(CameraSource.CAMERA_FACING_BACK)
                .setRequestedPreviewSize(1600, 1024)
                .setRequestedFps(15.0f);


        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
            builder = builder.setFocusMode(
                    autoFocus ?
Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE : null);
        }

        mCameraSource = builder
```

```
            .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH :
null)
            .build();
    }


    @Override
    protected void onResume() {
        super.onResume();
        startCameraSource();
    }


    @Override
    protected void onPause() {
        super.onPause();
        if (mPreview != null) {
            mPreview.stop();
        }
    }


    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPreview != null) {
            mPreview.release();
        }
    }


    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode != RC_HANDLE_CAMERA_PERM) {
            Log.d(TAG, "Got unexpected permission result: " + requestCode);
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
            return;
        }

        if (grantResults.length != 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Camera permission granted - initialize the camera source");
```

```
            boolean autoFocus = getIntent().getBooleanExtra(AutoFocus,false);
            boolean useFlash = getIntent().getBooleanExtra(UseFlash, false);
            createCameraSource(autoFocus, useFlash);
            return;
        }

        Log.e(TAG, "Permission not granted: results len = " + grantResults.length +
                " Result code = " + (grantResults.length > 0 ? grantResults[0] :
"(empty)"));

        DialogInterface.OnClickListener listener = new DialogInterface.OnClickListener()
{
            public void onClick(DialogInterface dialog, int id) {
                finish();
            }
        };

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Multitracker sample")
                .setMessage(R.string.no_camera_permission)
                .setPositiveButton(R.string.ok, listener)
                .show();
    }


    private void startCameraSource() throws SecurityException {

        int code = GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
                getApplicationContext());
        if (code != ConnectionResult.SUCCESS) {
            Dialog dlg =
                    GoogleApiAvailability.getInstance().getErrorDialog(this, code,
RC_HANDLE_GMS);
            dlg.show();
        }

        if (mCameraSource != null) {
            try {
                mPreview.start(mCameraSource, mGraphicOverlay);
            } catch (IOException e) {
                Log.e(TAG, "Unable to start camera source.", e);
                mCameraSource.release();
                mCameraSource = null;
```

```
        }
    }
}


private boolean onTap(float rawX, float rawY) {

    int[] location = new int[2];
    mGraphicOverlay.getLocationOnScreen(location);
    float x = (rawX - location[0]) / mGraphicOverlay.getWidthScaleFactor();
    float y = (rawY - location[1]) / mGraphicOverlay.getHeightScaleFactor();


    Barcode best = null;
    float bestDistance = Float.MAX_VALUE;
    for (BarcodeGraphic graphic : mGraphicOverlay.getGraphics()) {
        Barcode barcode = graphic.getBarcode();
        if (barcode.getBoundingBox().contains((int) x, (int) y)) {

            best = barcode;
            break;
        }
        float dx = x - barcode.getBoundingBox().centerX();
        float dy = y - barcode.getBoundingBox().centerY();
        float distance = (dx * dx) + (dy * dy);  // actually squared distance
        if (distance < bestDistance) {
            best = barcode;
            bestDistance = distance;
        }
    }

    if (best != null) {
        Intent data = new Intent();
        data.putExtra(BarcodeObject, best);
        setResult(CommonStatusCodes.SUCCESS, data);
        finish();
        return true;
    }
    return false;
}

private class CaptureGestureListener extends
GestureDetector.SimpleOnGestureListener {
    @Override
```

```
        public boolean onSingleTapConfirmed(MotionEvent e) {
            return onTap(e.getRawX(), e.getRawY()) || super.onSingleTapConfirmed(e);
        }
    }

    private class ScaleListener implements
ScaleGestureDetector.OnScaleGestureListener {


        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            return false;
        }


        @Override
        public boolean onScaleBegin(ScaleGestureDetector detector) {
            return true;
        }


        @Override
        public void onScaleEnd(ScaleGestureDetector detector) {
            mCameraSource.doZoom(detector.getScaleFactor());
        }
    }

    @Override
    public void onBarcodeDetected(Barcode barcode) {

    }
}

package com.google.android.gms.samples.vision.barcodereader.ui.camera;

import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.View;

import com.google.android.gms.vision.CameraSource;

import java.util.HashSet;
import java.util.List;
```

```
import java.util.Set;
import java.util.Vector;


public class GraphicOverlay<T extends GraphicOverlay.Graphic> extends View {
    private final Object mLock = new Object();
    private int mPreviewWidth;
    private float mWidthScaleFactor = 1.0f;
    private int mPreviewHeight;
    private float mHeightScaleFactor = 1.0f;
    private int mFacing = CameraSource.CAMERA_FACING_BACK;
    private Set<T> mGraphics = new HashSet<>();


    public static abstract class Graphic {
        private GraphicOverlay mOverlay;

        public Graphic(GraphicOverlay overlay) {
            mOverlay = overlay;
        }


        public abstract void draw(Canvas canvas);


        public float scaleX(float horizontal) {
            return horizontal * mOverlay.mWidthScaleFactor;
        }


        public float scaleY(float vertical) {
            return vertical * mOverlay.mHeightScaleFactor;
        }


        public float translateX(float x) {
            if (mOverlay.mFacing == CameraSource.CAMERA_FACING_FRONT) {
                return mOverlay.getWidth() - scaleX(x);
            } else {
                return scaleX(x);
            }
        }
```

```
    public float translateY(float y) {
        return scaleY(y);
    }

    public void postInvalidate() {
        mOverlay.postInvalidate();
    }
}

public GraphicOverlay(Context context, AttributeSet attrs) {
    super(context, attrs);
}


public void clear() {
    synchronized (mLock) {
        mGraphics.clear();
    }
    postInvalidate();
}

public void add(T graphic) {
    synchronized (mLock) {
        mGraphics.add(graphic);
    }
    postInvalidate();
}


public void remove(T graphic) {
    synchronized (mLock) {
        mGraphics.remove(graphic);
    }
    postInvalidate();
}


public List<T> getGraphics() {
    synchronized (mLock) {
        return new Vector(mGraphics);
    }
}
```

```
    public float getWidthScaleFactor() {
        return mWidthScaleFactor;
    }


    public float getHeightScaleFactor() {
        return mHeightScaleFactor;
    }


    public void setCameraInfo(int previewWidth, int previewHeight, int facing) {
        synchronized (mLock) {
            mPreviewWidth = previewWidth;
            mPreviewHeight = previewHeight;
            mFacing = facing;
        }
        postInvalidate();
    }


    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        synchronized (mLock) {
            if ((mPreviewWidth != 0) && (mPreviewHeight != 0)) {
                mWidthScaleFactor = (float) canvas.getWidth() / (float) mPreviewWidth;

}   mHeightScaleFactor = (float) canvas.getHeight() / (float) mPreviewHeight;
            }

            for (Graphic graphic : mGraphics) {
            graphic.draw(canvas);
            }
            }
            }
package com.google.android.gms.samples.vision.barcodereader;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;

import
```

```
com.google.android.gms.samples.vision.barcodereader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.barcode.Barcode;


public class BarcodeGraphic extends GraphicOverlay.Graphic {

    private int mId;

    private static final int COLOR_CHOICES[] = {
            Color.BLUE,
            Color.CYAN,
            Color.GREEN
    };

    private static int mCurrentColorIndex = 0;

    private Paint mRectPaint;
    private Paint mTextPaint;
    private volatile Barcode mBarcode;

    BarcodeGraphic(GraphicOverlay overlay) {
        super(overlay);

        mCurrentColorIndex = (mCurrentColorIndex + 1) % COLOR_CHOICES.length;
        final int selectedColor = COLOR_CHOICES[mCurrentColorIndex];

        mRectPaint = new Paint();
        mRectPaint.setColor(selectedColor);
        mRectPaint.setStyle(Paint.Style.STROKE);
        mRectPaint.setStrokeWidth(4.0f);

        mTextPaint = new Paint();
        mTextPaint.setColor(selectedColor);
        mTextPaint.setTextSize(36.0f);
    }

    public int getId() {
        return mId;
    }

    public void setId(int id) {
        this.mId = id;
    }
```

```
    public Barcode getBarcode() {
        return mBarcode;
    }


    void updateItem(Barcode barcode) {
        mBarcode = barcode;
        postInvalidate();
    }


    @Override
    public void draw(Canvas canvas) {
        Barcode barcode = mBarcode;
        if (barcode == null) {
            return;
        }


        RectF rect = new RectF(barcode.getBoundingBox());
        rect.left = translateX(rect.left);
        rect.top = translateY(rect.top);
        rect.right = translateX(rect.right);
        rect.bottom = translateY(rect.bottom);
        canvas.drawRect(rect, mRectPaint);


        canvas.drawText(barcode.rawValue, rect.left, rect.bottom, mTextPaint);
    }
}

    @Override
    public void onNewItem(int id, Barcode item) {
        mGraphic.setId(id);
        mBarcodeUpdateListener.onBarcodeDetected(item);
    }


    @Override
    public void onUpdate(Detector.Detections<Barcode> detectionResults, Barcode
item) {
        mOverlay.add(mGraphic);
        mGraphic.updateItem(item);
    }
```

```
    @Override
    public void onMissing(Detector.Detections<Barcode> detectionResults) {
        mOverlay.remove(mGraphic);
    }


    @Override
    public void onDone() {
        mOverlay.remove(mGraphic);
    }
}
package com.google.android.gms.samples.vision.barcodereader;

import android.content.Context;

import
com.google.android.gms.samples.vision.barcodereader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.MultiProcessor;
import com.google.android.gms.vision.Tracker;
import com.google.android.gms.vision.barcode.Barcode;

package com.google.android.gms.samples.vision.barcodereader;

import android.content.Intent;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.TextView;

import com.google.android.gms.common.api.CommonStatusCodes;
import com.google.android.gms.vision.barcode.Barcode;


public class MainActivity extends Activity implements View.OnClickListener {


    private CompoundButton autoFocus;
    private CompoundButton useFlash;
    private TextView statusMessage;
    private TextView barcodeValue;
```

```
    private static final int RC_BARCODE_CAPTURE = 9001;
    private static final String TAG = "BarcodeMain";

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

        statusMessage = (TextView)findViewById(R.id.status_message);
        barcodeValue = (TextView)findViewById(R.id.barcode_value);

        autoFocus = (CompoundButton) findViewById(R.id.auto_focus);
        useFlash = (CompoundButton) findViewById(R.id.use_flash);

        findViewById(R.id.SCANNER).setOnClickListener(this);
    }


    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.SCANNER) {

            Intent intent = new Intent(this, BarcodeCaptureActivity.class);
            intent.putExtra(BarcodeCaptureActivity.AutoFocus, autoFocus.isChecked());
            intent.putExtra(BarcodeCaptureActivity.UseFlash, useFlash.isChecked());

            startActivityForResult(intent, RC_BARCODE_CAPTURE);
        }

    }



    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == RC_BARCODE_CAPTURE) {
            if (resultCode == CommonStatusCodes.SUCCESS) {
                if (data != null) {
                    Barcode barcode =
data.getParcelableExtra(BarcodeCaptureActivity.BarcodeObject);
                    statusMessage.setText(R.string.barcode_success);
                    barcodeValue.setText(barcode.displayValue);
                    Log.d(TAG, "СКАНИРОВАТЬ: " + barcode.displayValue);
```

```
            } else {
                statusMessage.setText(R.string.barcode_failure);
                Log.d(TAG, "Отсутствует штрих-код");
            }
        } else {
            statusMessage.setText(String.format(getString(R.string.barcode_error),
                    CommonStatusCodes.getStatusCodeString(resultCode)));
        }
    }
    else {
        super.onActivityResult(requestCode, resultCode, data);
    }
  }
}
class BarcodeTrackerFactory implements MultiProcessor.Factory<Barcode> {
  private GraphicOverlay<BarcodeGraphic> mGraphicOverlay;
  private Context mContext;

  public BarcodeTrackerFactory(GraphicOverlay<BarcodeGraphic>
mGraphicOverlay,
                               Context mContext) {
    this.mGraphicOverlay = mGraphicOverlay;
    this.mContext = mContext;
  }

  @Override
  public Tracker<Barcode> create(Barcode barcode) {
    BarcodeGraphic graphic = new BarcodeGraphic(mGraphicOverlay);
    return new BarcodeGraphicTracker(mGraphicOverlay, graphic, mContext);
  }

}
```

Додаток Б

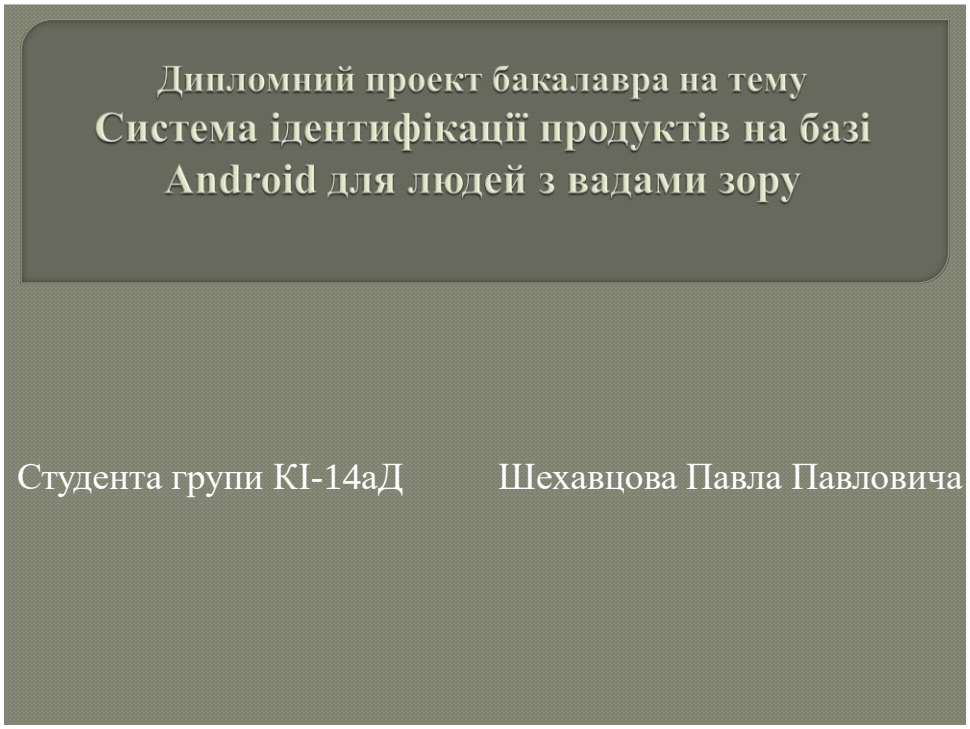

Рисунок Б.1 – Титульний лист

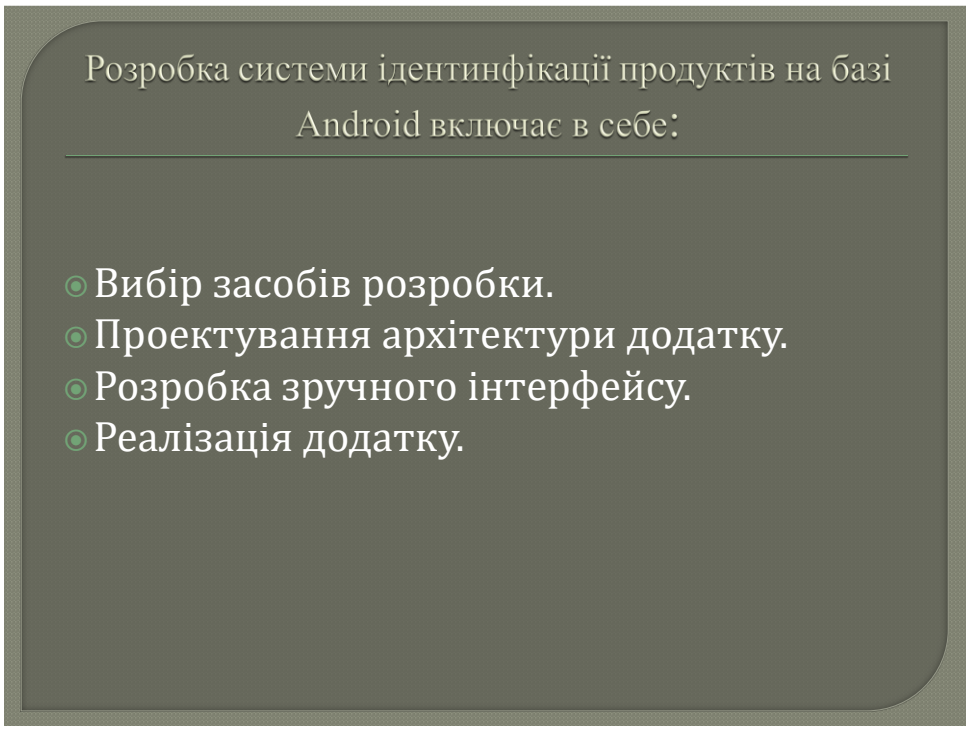

Рисунок Б.2 – Задачы розробки

Рисунок Б.3 – Вибыр засобыв розробки

Рисунок Б.4 – Методологыя створення додатку

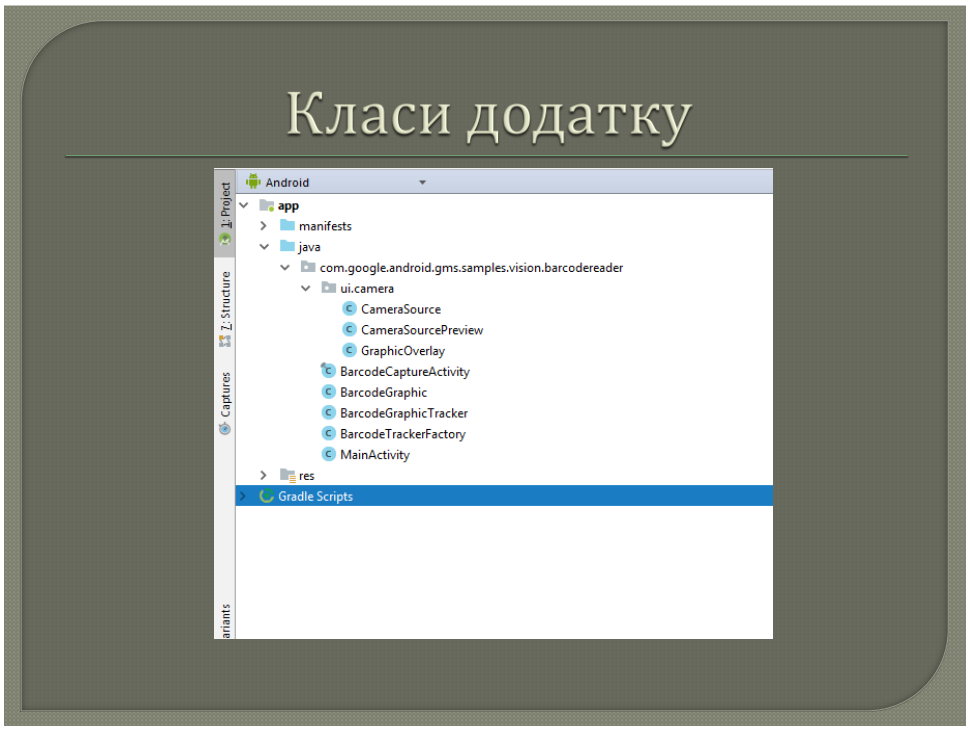

Рисунок Б.5 – Класи додатку

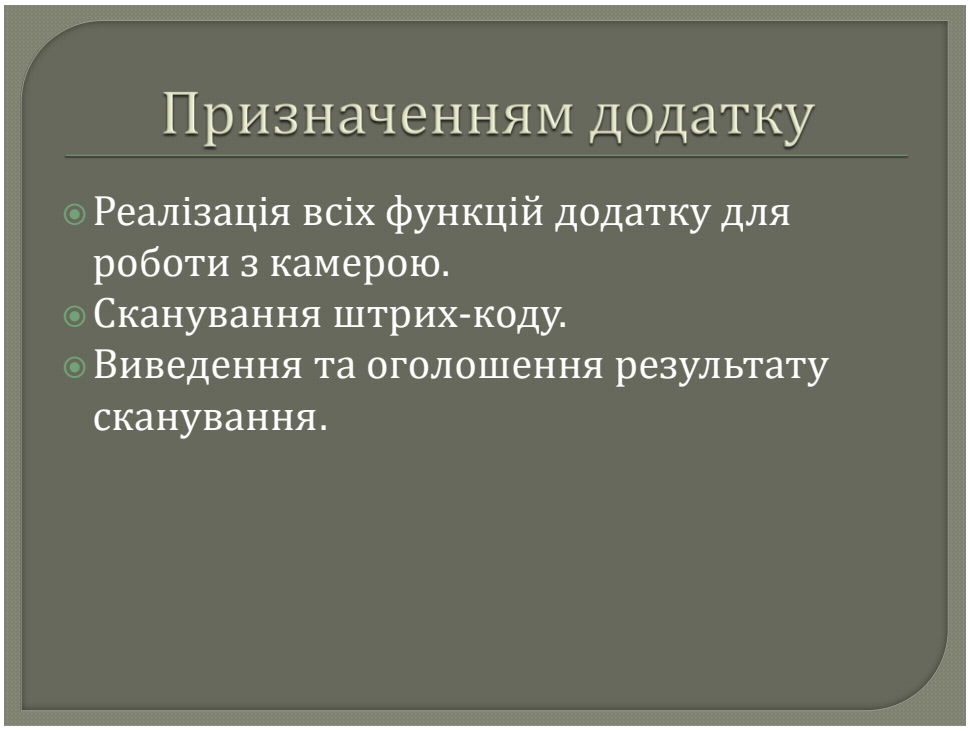

Рисунок Б.6 – Призначення додатку

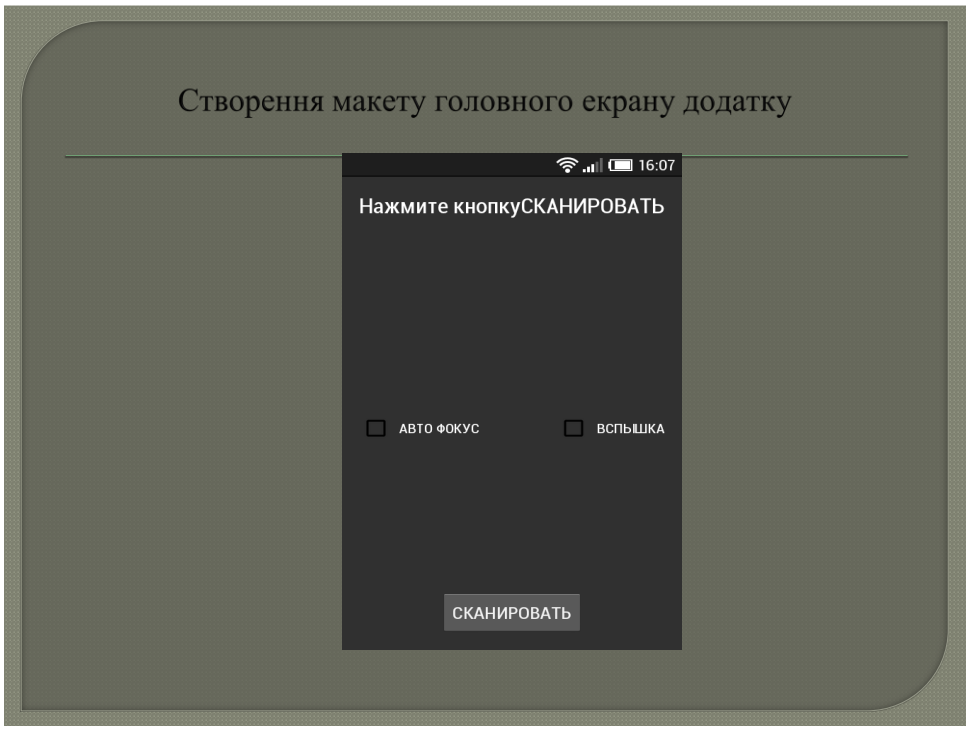

Рисунок Б. 7 – Макет головного екрану

Рисунок Б.8 – Макет екрану сканування штрих-коду

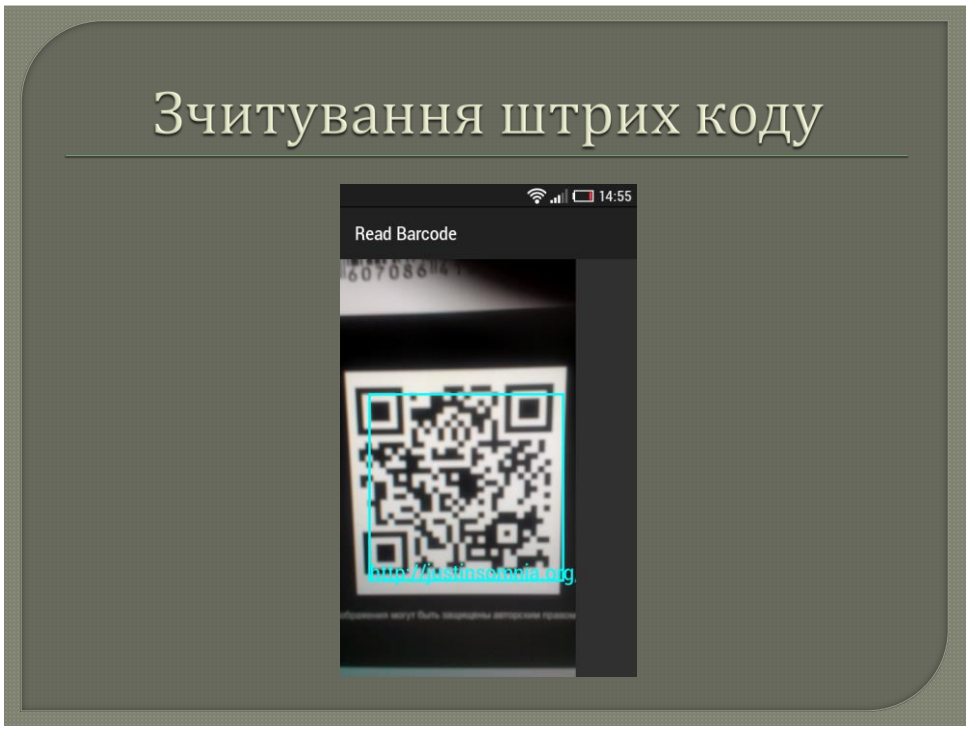

Рисунок Б.9 – Зчитування штрих коду

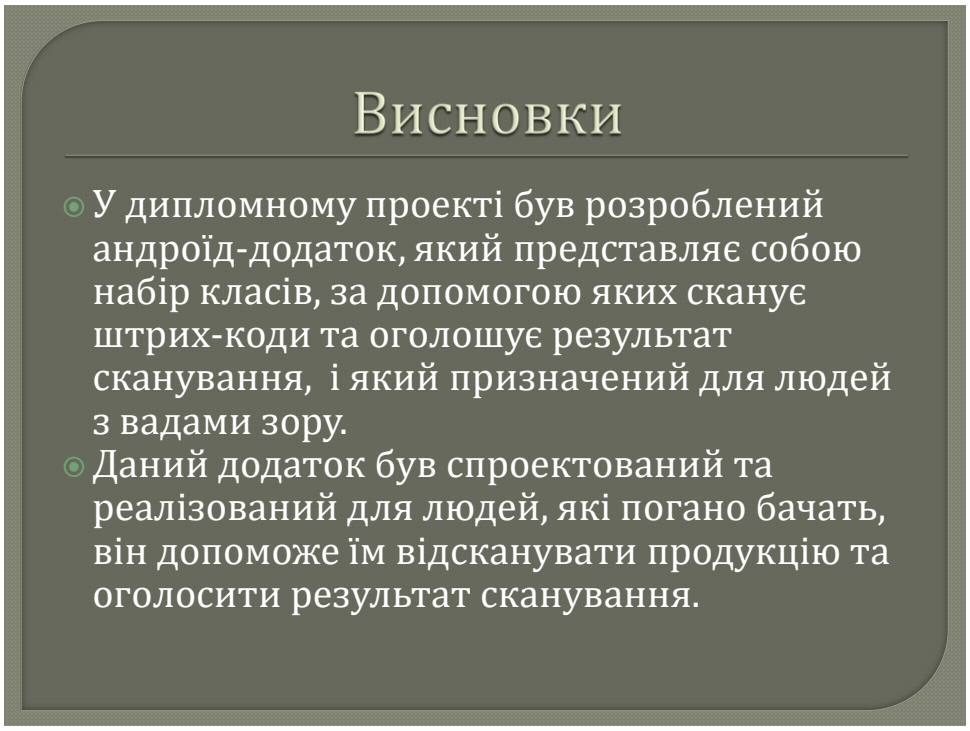

Рисунок Б. 10 - Висновки