Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Мобільний додаток для обліку термінів придатності продуктів харчування
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПШПЗ.170117.01.15.ПЗ

Виконав студент IV курсу група ІПЗ-17-1 _____ С.В. Романчук
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ Н.І. Праворська
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Радельчук
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

22 червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
05 02 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Романчуку Сергію Володимировичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мобільний додаток для обліку термінів придатності продуктів харчування

Керівник проекту (роботи) Праворська Наталія Іванівна, канд. пед. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2021 р. № 11
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2021 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди)

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Радельчук Г. І., доцент кафедри ІПЗ		
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ		

7. Дата видачі завдання « 05 » лютого 2021р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних там ДП	01.12 – 30.12.2020	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2021	
3 Проектування програмного забезпечення	01.02 – 28.02.2021	
4 Програмна реалізація	01.03 – 10.04.2021	
5 Тестування програмного забезпечення	11.04 – 30.04.2021	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2021	
7 Попередній захист ДП	Травень 2021 (згідно графіка)	
8 Перевірка ДП на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки)	26.05 – 30.05.2021	
9 Підготовка до захисту та захист ДП	з 01.06.2021	

Студент _____ С. В. Романчук
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Н. І. Праворська
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: Мобільний додаток для обліку термінів придатності продуктів харчування.

Автор проекту: Романчук Сергій Володимирович.

Керівник проекту: Праворська Наталія Іванівна.

Пояснювальна записка: 101 с., 18 рис., 3 табл., 3 дод., 18 джерел.

Графічна частина: 12 слайдів.

МОБІЛЬНИЙ ДОДАТОК, ОБЛІК ПРОДУКТІВ, ТЕРМІНИ ПРИДАТНОСТІ, ANDROID, JAVA.

Метою проекту є розробка мобільного додатку для платформи Android, для обліку термінів придатності продуктів харчування.

У дипломному проекті виконано порівняння наявних мобільних додатків для ведення обліку термінів придатності продуктів харчування. Визначено функціональні задачі та розроблено ТЗ. Проведено аналіз існуючих інструментів та технологій розробки Android-додатків, спроектовано архітектуру додатка та його інтерфейс. Для реалізації програмного забезпечення використано мову програмування Java, для створення розмітки додатку було використано середовище розробки Android Studio. Після розробки було проведене модульне тестування готового застосунку.

Практична значимість отриманих результатів полягає у розробці готового до використання програмного продукту у вигляді мобільного додатку для обліку термінів придатності продуктів.

18.06.2021р.
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПШПЗ.170117.01.15.ПЗ	Пояснювальна записка	101		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	12		

Змн.	Арк.	№ докум.	Підпис	Дата	ДПШПЗ.170117.01.15.ВД		
Виконав		Романчук С.В.		18.06	Мобільний додаток для обліку термінів придатності продуктів харчування Відомість документів	Літ.	Арк. 1 / Аркушів 1
Керівник		Праворська Н.І.		22.06			
Рецензент						ХНУ, ІПЗ-17-1	
Н. контр.		Радельчук Г. І.		18.06			
Зав. каф.		Бедратюк Л.П.		17.06			

ЗМІСТ

ВСТУП

Усім відомо, що деякі сучасні холодильники можна підключати до Інтернету. Фактично в такому холодильнику передбачений вбудований комп'ютер зі зовнішнім сенсорний екраном та WiFi-модулем, за допомогою якого можна, наприклад, складати списки покупок, шукати рецепти в Мережі, слідкувати за терміном придатності продуктів, які зберігаються в холодильнику. Це, звісно, дуже зручно, але вартість таких холодильників відповідна.

Однак, якщо з'явилось бажання чи необхідність комп'ютеризувати свій холодильник, щоб завжди знати, що, в якій кількості і як довго в ньому лежить, то для цього не обов'язково наносити удар по сімейному бюджету і покупати дорогу модель холодильника лише через те, що в ньому є екран та Wi-Fi.

Справа в тому, що для таких цілей чудово підходить як спеціальні онлайн-сервіси, так і мобільні додатки, розроблені як раз для того, щоб допомогти навести порядок у холодильнику, в не залежності від його моделі.

Актуальність теми полягає в тому, що додаток для контролю термінів придатності продуктів харчування дає можливість не лише впорядкувати та контролювати ваші продукти харчування, а й зекономити гроші на холодильниках з цими ж самими вбудованими функціями, або вберегти вас від можливої непередбаченої шкоди для здоров'я, якщо ви не замітите, що в якогось із продуктів вийшов термін придатності. Також контроль вашої їжі може допомогти навколишньому середовищу, адже зараз в світі актуальна проблема викидання продуктів, в яких вийшов термін придатності. Наприклад, ви зможете завчасно побачити що в деяких продуктів підходить термін до кінця, і це дасть вам можливість приготувати його, а не просто викинути.

Метою мого проекту є розробка мобільного додатку, який дозволяє контролювати терміни придатності продуктів харчування для користувачів.

Для реалізації поставленої мети я сформував наступні завдання:

– дослідити предметну області контролю продуктів харчування та виявити потреби потенційних користувачів програмного продукту;

– проаналізувати наявне програмне забезпечення по цій темі та сформулювати вимоги на розробку нового програмного забезпечення, що відповідає потребам часу;

– спроектувати структуру мобільного додатку та розробити базу даних;

– вибрати технології для розробки додатку;

– розробити та протестувати мобільний додаток.

Результатом вирішення завдань дипломного проекту є повноцінний додаток для контролю термінів придатності продуктів харчування.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

У сучасному світі важко уявити собі мобільний пристрій, на якому не було би жодного додатку. Вони міцно увійшли в наше життя практично одночасно з планшетами та смартфонами. Тому це направлення так стрімко розвивається та захоплює ринок. Усе більше підприємців розуміють необхідність розробки мобільних додатків.

Мобільний додаток представляє з себе розроблену програму для планшетів та смартфонів, яка встановлюється на ту чи іншу платформу та має певний функціонал [2]. Простіше кажучи, виконує певні дії та вирішує задане коло питань. Серед різних методів класифікації мобільних додатків найчастіше йде мова саме про три типи: нативні, гібридні та мереживі.

Нативні додатки – це додатки, які створюються під конкретну платформу та завантажуються з магазинів Play Маркет або AppStore, використовуючи пам’ять операційної системи пристрою. Перевагою такого типу мобайлу є оптимізація під одну операційну систему, тому продукти працюють правильно та достатньо швидко. Розробка нативних застосунків під IOs чи Android проходить з використанням певних «рідних» мов програмування для цих платформ, тому органічно вписується в любі девайси. Після встановлення на гаджет, сервіс має доступ до більшості сервісів телефону: камері, мікрофону, календарю, фото- та відеофайлам, а також іншим службам. Після встановлення користувач може увімкнути push-повідомлення за рахунок доступу до системи сповіщень пристрою [3].

Серед основних недоліків нативних систем можна назвати охоплення лише однієї платформи, а це відповідно менша кількість потенційних користувачів, а також те що саме створення таких застосунків займає багато часу, особливо у

випадках коли потрібно робити по застосунку на різні платформи, а це відповідно ще й негативно впливає на вартість.

Основними перевагами є висока продуктивність та висока швидкість роботи, підтримка від конкретної операційної системи, привабливий та розширений інтерфейс, встановлення лише з офіційних маркетів.

Гібридні додатки – це додатки, які представляють з себе поєднання веб та нативних додатків. В особливості, тут мається на увазі їх доступ до функціоналу смартфону та кросплатформеність. Такі додатки можуть бути завантажені виключно офіційних маркетів від власників платформи. Серед багатьох компаній вибір частіше всього припадає саме на розробку гібридного додатку, що пояснюється тим, що гібридні додатки здатні поєднувати переваги нативних застосунків з технологічною актуальністю, яка забезпечується останніми веб-технологіями. Однак, на відміну від нативних, вартість створення гібридних на порядок нижча, а їх швидкість не сильно поступається нативним. Схожість гібридних додатків з веб-застосунками в свою чергу дає переваги у вигляді доступності у внесенні коректив [3].

Мобільні веб-додатки – це додатки, які насправді не є додатками як такими. Справа в тому, що веб-додаток, по своїй суті, це веб-сайт, який адаптований та оптимізований під любий смартфон. Для того, щоб використовувати його, достатньо мати на пристрої браузер, знати адресу сайту та мати інтернет-з’єднання. Запускаючи веб-додаток, користувач може виконувати усі ті самі дії, які він би виконував при переході на будь-який веб-сайт, а також отримує можливість встановити його на свій робочий стіл, створивши закладку сторінки веб-сайту. Недоліками таких веб-застосунків є обов’язково підключення до інтернету, низька швидкодія та низький рівень безпеки. Перевагами є кросплатформеність, швидкий процес розробки, відсутність необхідності завантаження з маркету.

Зважаючи на усі плюси та мінуси основних типів мобільних застосунків я вибрав нативний тип застосунку. Цей вибір я зробив основуючись на таких

переваг як краща швидкість роботи та продуктивнь таких типів застосунків, велика кількість інструментів для розробки, які надають власники платформ, повне функціонування без підключення до інтернету.

Оскільки нативні додатки мають свою конкретну операційну систему, під яку вони розробляються, її теж потрібно вибрати.

Операційна система — це базовий комплекс програмного забезпечення, що виконує управління апаратним забезпеченням комп'ютера або віртуальної машини, забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем [4]. Наявність операційної системи – головна особливість, яка відрізняє смартфон від простого мобільного телефону.

Зараз у світі є дві основні платформи, які займають 99.9% усіх мобільних операційних систем – це Android та iOS.

iOS – це мобільна операційна система для смартфонів, планшетів та деяких інших пристроїв, яка розроблена та підтримується компанією Apple.

В iOS використовується ядро XNU, основане на мікроядрі Mach та має в собі програмний код розроблений компанією Apple. На відміну від Android, випускається лише для пристроїв виготовлених компанією Apple. Серед плюсів можна зазначити зручне інтуїтивно зрозуміле керування та меню, комфортна, продумана до деталей синхронізація з приладами Apple, як у ручному, так і в автономному режимі, підтримка оновлень приладів з внесенням покращень в роботу гаджетів, краща систему захисту порівняно з Android. Серед недоліків можна віднести дороговизну девайсів, на яких встановлена ця операційна система, можливі ускладнення при перенесенні даних з приладів з іншими операційними системами.

Android – це операційна система для смартфонів та планшетів, яка основана на модифікованому ядрі Linux. Компанія Google придбала розробників Android Inc в 2005 році. Система підтримує понад 100 мов, має відкритих вихідний код, підходить для створення Java-додатків, в також для портів бібліотек та компонентів, які були написані на мові C. Плюсами є великий вибір телефонів,

адже усі найпопулярніші виробники телефонів, окрім Apple, випускають свої девайси на операційній системі Android, велика кількість інструментів для розробників під платформу Android. Серед мінусів можна назвати захист системи, який слабший ніж в iOS.

Проаналізувавши та порівнявши усі переваги та недоліки операційної системи Android та IOs я вирішив, що краще всього робити нативним мобільний додаток під операційну систему Android, адже вона дає більшу кількість можливостей для розробників, є більш доступною, а також найпопулярнішою мобільною операційною системою в світі.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

На сьогодні вже є значна кількість подібних додатків, які дозволяють користувачам контролювати терміни придатності продуктів харчування. Тому важливо, щоб розроблюваний додаток був зручним, функціональним, швидким та привабливим для майбутніх користувачів.

Розглянемо деякі додатки які є найпопулярнішими для платформи Android. Для цього нам потрібно перевірити додатки в магазині Play Маркет, який є офіційний додатком від Google і зазвичай попередньо встановлений на усі смартфони з операційною системою Android. Для проведення аналізу ринку скористаємось рекомендаціями по темі мого дипломного проекту, а саме термін придатності для продуктів. Після введення відповідних записів ми можемо побачити такі популярні додатки як:

- Учет Продуктов;
- Учет товаров-простой склад 2.0;
- FoodKeeper;
- Срок годности продуктов питания!;

– Expiring Things.

Далі потрібно провести детальний аналіз вищеперерахованих застосунків, звертаючи увагу на їх функціональні можливості, дизайн, переваги та недоліки, з врахуванням оцінок користувачів в Play Market та особистого досвіду використання додатків.

Першим додатком є «Учет Продуктов», на сторінці в Play Market [5] розробником представлений опис та функції свого додатку.

На рисунку 1.1 зображено додаток «Учёт Продуктов».

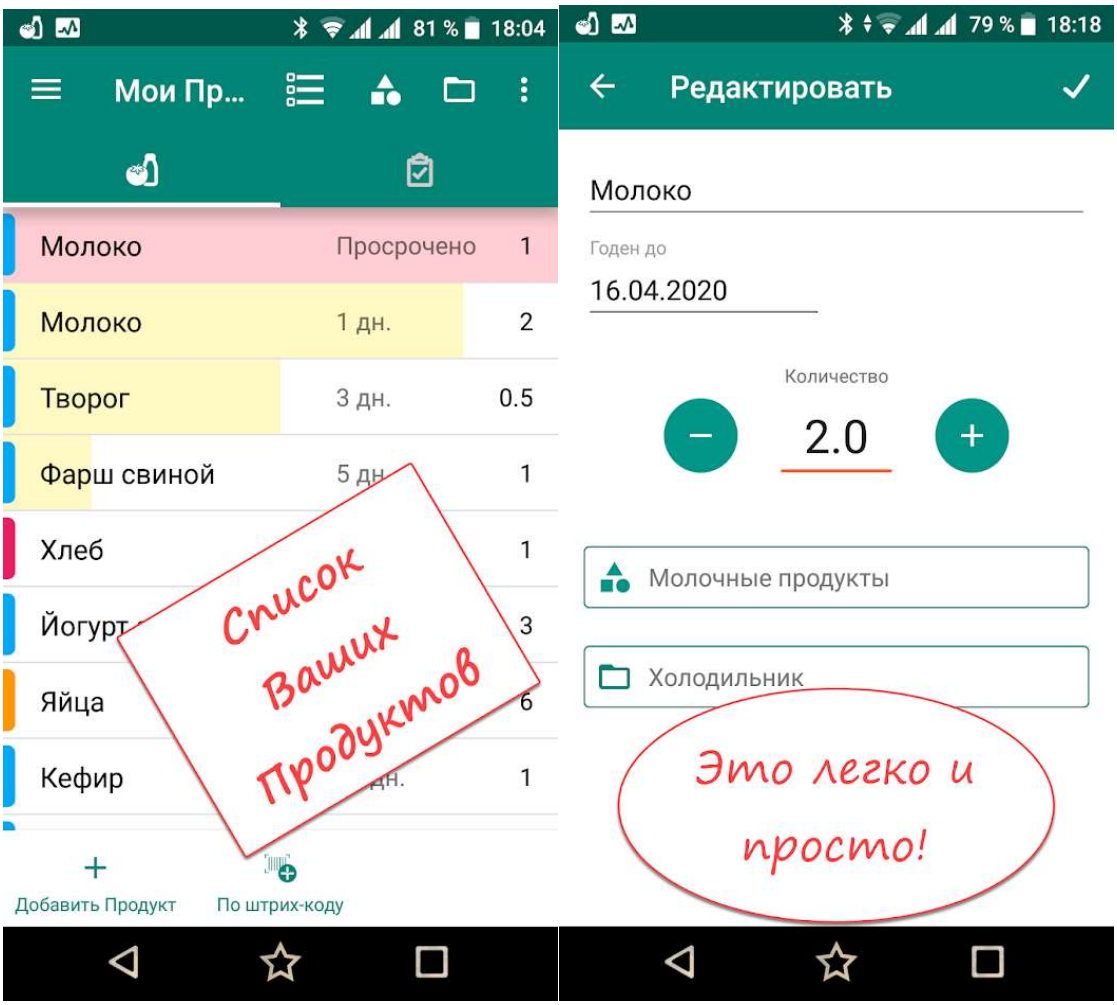


Рисунок 1.1 – Додаток «Учёт Продуктов».

Цей додаток дозволяє вести облік продуктів у вас вдома (в холодильнику, морозилці, кладовищі, де завгодно) та відслідковувати їх терміни придатності.

Також додаток дозволяє добавляти продукти в списки покупок, а після купівлі добавляти ці продукти в список продуктів які зберігаються вдома. Для зручності ви можете розсортувати продукти по групам(категоріям) та визначити місця зберігання, щоб знати, де, що і в якій кількості знаходиться. В додатку реалізована лише російська мова. Добавляти продукти можна через штрих-код товару або вручну через фотографію чи просто текст.

Наступний додаток «Учет товаров-простой склад 2.0» на його сторінці в Play Market [6] є опис додатку, його функцій та можливостей.

На рисунку 1.2 зображений додаток «Учет товаров-простой склад 2.0»

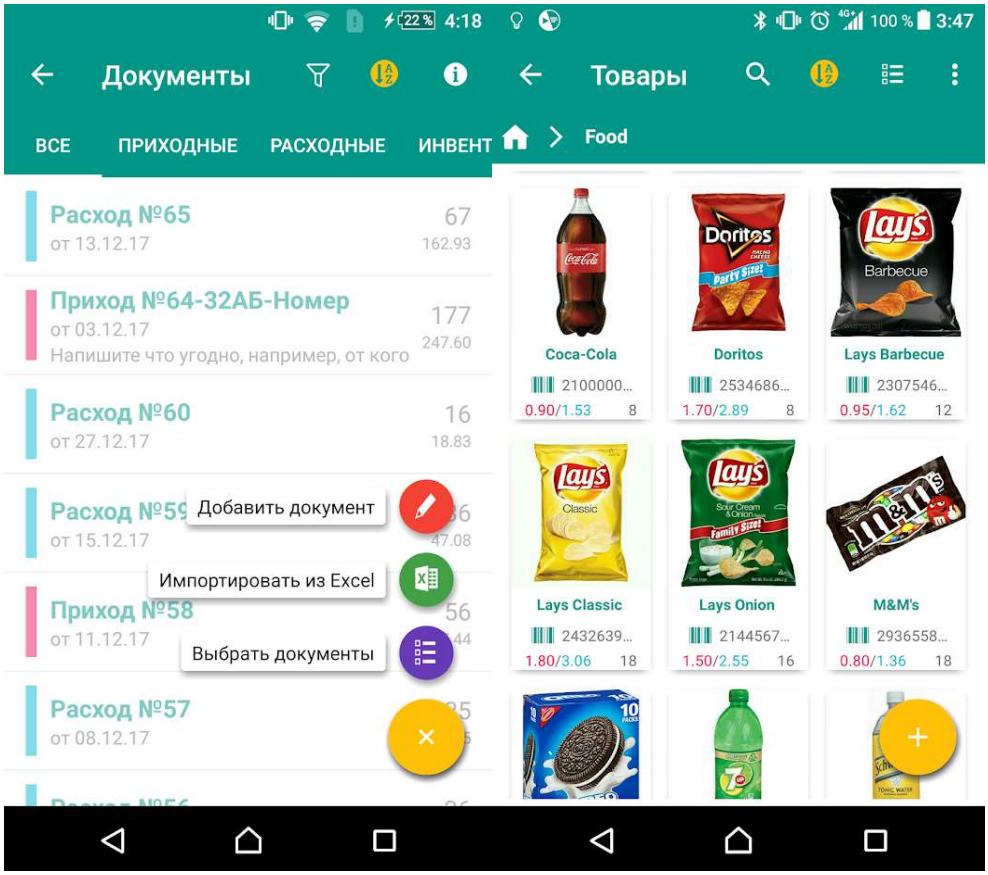


Рисунок 1.2 – Додаток «Учет товаров-простой склад 2.0»

.

Ця програма дозволяє вести простий облік товарів вдома, на складі чи в магазині за допомогою Вашого телефону або планшета. Ви можете

використовувати програму щоб вести облік своїй запасів, товарів або інших предметів вдома чи на складі.

Також ви можете добавляти продукти в список покупок, а після покупки добавляти їх в список продуктів, які зберігаються у вас вдома.

Для зручності зберігання ви можете розсортувати продукти по групам (категоріям). Можна добавляти штрих-коди товарів і сканувати їх для більш швидкої роботи.

Програма дозволяє формувати звіти:

- оборот товару;
- історія руху товару;
- прибуток та збитки;
- продажі по дням, по місяцям, по товарам та по клієнтам;
- закупки по постачальникам;
- повна історія транзакцій для вивантаження в Excel;
- імпорт та експорт даних через файли Excel.

У цього додатка є такі функції як:

- завантаження та вивантаження документів в форматі Excel;
- резервне копіювання та відновлення бази даних;
- налаштування кольорів документів;
- використання камери для сканування штрих-кодів;
- пошук, сортування та фільтри в списках;
- введення товарів в документи можна проводити за допомогою камери, імпорту з файлу Excel, вибором зі довідника товарів.
- дозамовлення товару

Більшість функцій додатка є платними.

Рейтинг цього додатка у Play Market складає 4.6/5, останнє оновлення вийшло у травні 2021 року.

Наступний додаток це «FoodKeeper». На його сторінці в Play Market [7] є короткий опис призначення та функцій.

На рисунку 1.3 зображений додаток «FoodKeeper»

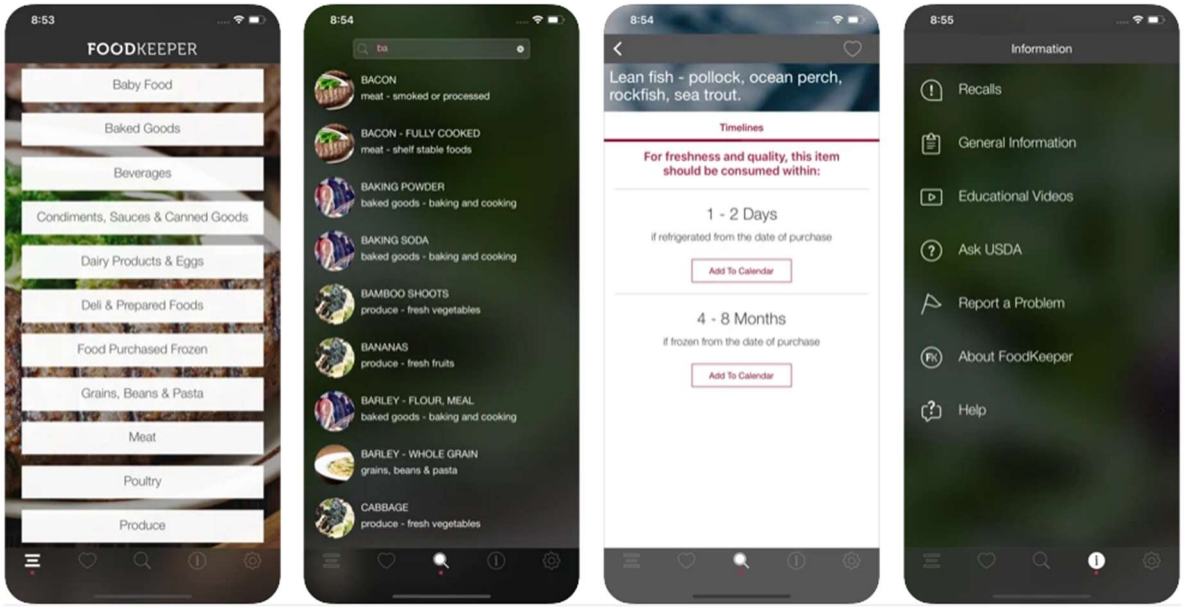


Рисунок 1.3 – Додаток «USDA FoodKeeper».

Це популярний додаток на системі iOS та Android. Він аналогічно дозволяє добавляти придбані продукти та виставляти їм терміни придатності. Його особливістю є надання рекомендацій по зберіганню про приготуванню їжі – додаток показує, при якій температурі зберігають той чи інший продукт, чи варто готувати його зараз або краще зачекати декілька днів. Сам додаток був розроблений сумісно з Міністерством сільського господарювання США, спеціалістами з Корнелльського університету та організацією Food Marketing Institute. В додатку доступна лише англійська мова.

Рейтинг на Play Market у цього додатка 3.7/5 а останнє оновлення було випущено у серпні 2020 року.

Наступний додаток це «Срок годности продуктов питания!». На його сторінці в Play Market [8] є короткий опис призначення та ключових функціональних особливостей.

На рисунку 1.4 зображений додаток «Срок годности продуктов питания!»

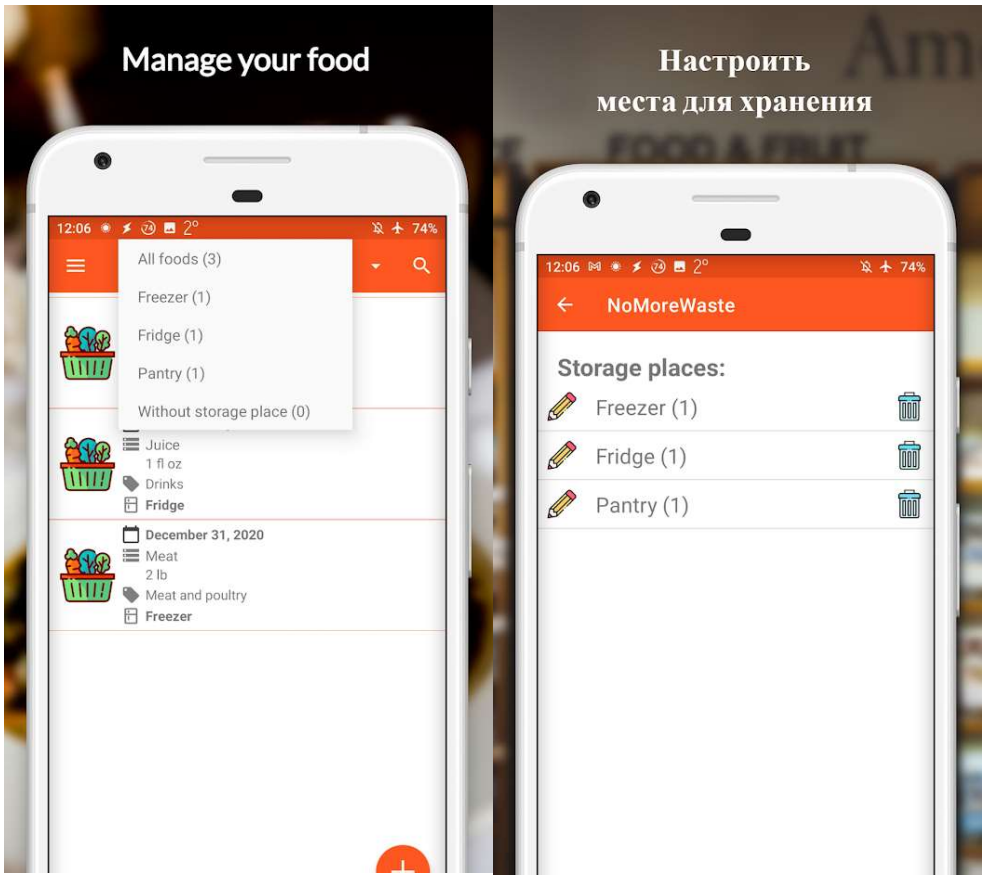


Рисунок 1.4 – Додаток «Срок годности продуктов питания!».

«Срок годности продуктов питания!» дозволяє вам відслідковувати продукти, які є у вас вдома та отримувати попередження, коли в якогось з продуктів має вийти термін придатності.

Ключовими особливостями є:

– класифікація по складу;
– сканування штрих-кодів;
– персоналізація мініатюр продуктів;
– отримання повідомлень про закінчення терміну придатності;
– налаштування імен сховищ та одиниць виміру.

В цього додатка у Play Market рейтинг 3.9, остання оновлення вийшло в березні 2021 року.

Наступний додаток це «Expiring Things». На його сторінці в Play Market [9] є короткий опис призначення та ключових функціональних особливостей.

На рисунку 1.5 зображений додаток «Expiring Things»

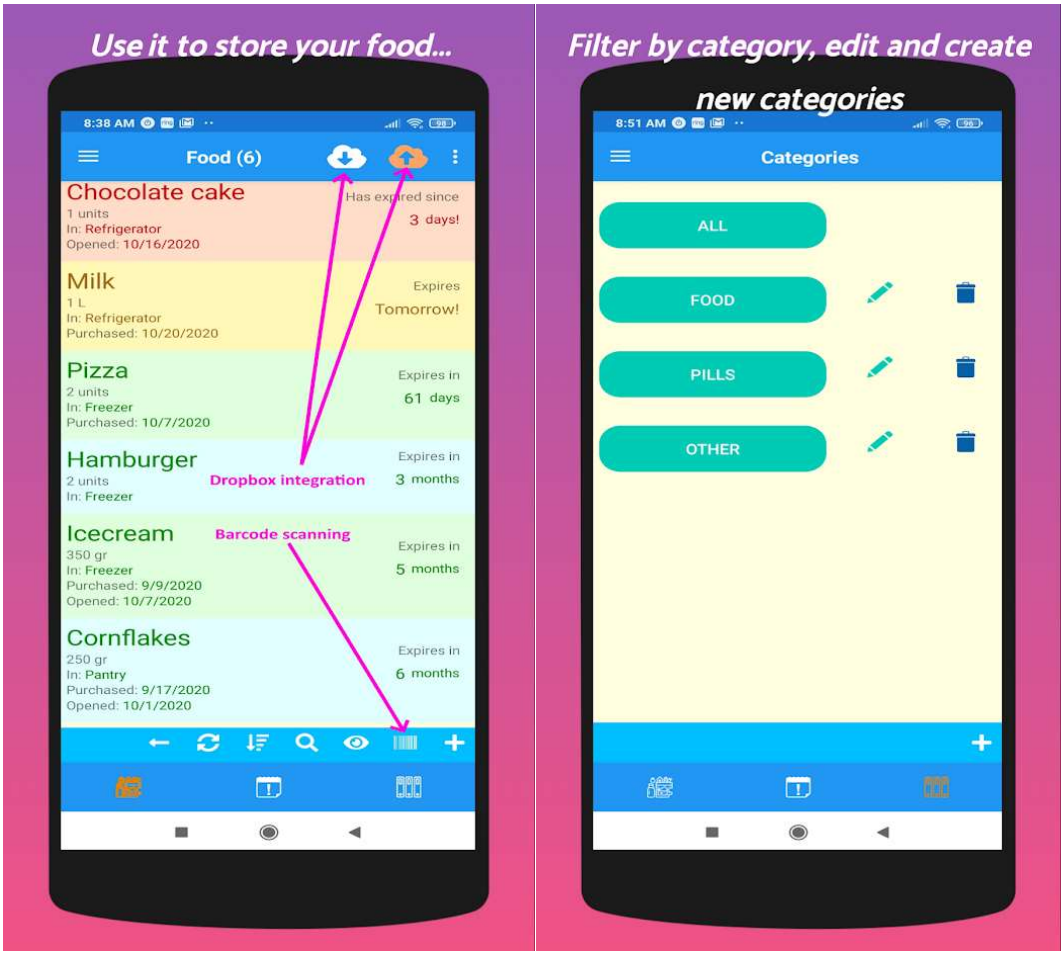


Рисунок 1.3 – Додаток «Expiring Things».

За допомогою цього додатку можна швидко та легко отримувати інформацію про Ваші продукти: кількість, термін придатності, дата купівлі та відкриття, контейнер (місце зберігання), ціна.

Основною функцією цього додатку є керування термінами придатності та кількістю продуктів.

Функціонал цього додатку:

– додавання атрибутів продуктів: назву, категорію, початкову та кінцеву кількість, одиниці вимірювання, місце

зберігання, терміни придатності, дата купівлі, дата відкриття, ціна та примітки;

– додавання продуктів вибираючи зі списку;

– додавання продуктів клонуючи уже існуючих;

– додавання продуктів за допомогою сканування їх баркоду. При першому скануванні нового продукту необхідно додати деталі вручну, оскільки додаток не використовує зовнішні сервіси;

– перегляд продуктів згрупованих по категоріям;

– перегляд продуктів згрупованих за місцем зберігання;

– сортування продуктів по назві, терміну придатності, даті купівлі, даті відкриття або кінцевій кількості;

– пошук продуктів по назві;

– отримання повідомлень про кінець терміну дії продукту;

– завантаження списку продуктів у Dropbox. Можна також ділитись списком продуктів з іншим приладом або користувачем.

– експорт та імпорт файлів даних для резервного відновлення.

Додаток є безкоштовним та не містить реклами.

Оцінка додатку на Play Market складає 4.7, а останнє оновлення вийшло в квітні 2021 року.

Після того як я провів аналіз додатків, можна провести порівняльний аналіз усіх розглянутих додатків за різними характеристиками. Основна функція усіх додатків є створення записів про терміни придатності продуктів та зберігання цих записів. Також важливою функцією є повідомлення користувача про закінчення терміну дії. У деяких додатків функціонал був орієнтований на магазини та підприємства які працюють з продуктами харчування. Для зручності користування можна виділити функцію сортування продуктів щоб, наприклад, можна було відразу бачити продукти, які вже близькі для свого кінця терміну придатності. Також надання продуктам певної категорії для того щоб швидше

знаходити потрібні продукти. Важливим критерієм є оцінка зручності інтерфейсу. Потрібно також зазначити чи є вибрані додатки платними, чи містять рекламу і чи мають вони свіжі версії, а саме чи досі оновлюються розробниками.

Таблиця 1.1 – Порівняльна характеристика наявного ПЗ

Назва додатку	Базові функції для обліку продуктів	Зручність інтерфейсу	Функція створення категорій та сортування	Функція повідомлень про закінчення терміну придатності	Безкоштовний	Підтримка розробниками	Наявність реклами
«Учёт Продуктов»	Так	3/10	Так	Так	Ні	Так	Так
«Учет товаров-простой склад 2.0»	Так	6/10	Так	Так	Ні	Так	Так
«FoodKeeper»	Так	3/10	Ні	Ні	Так	Ні	Ні
«Срок годности продуктов питания!»	Так	4/10	Відсутнє сортування	Так	Так	Так	Так
«Expiring Things»	Так	5/10	Так	Так	Так	Так	Ні

Отже, після того як я вивчив ринок мобільних додатків для обліку термінів придатності продуктів харчування, які наявні в Play Маркет, я можу зробити висновок, що на даний момент переважна кількість додатків вже має базовий функціонал, який дозволяє зберігати терміни придатності, але розширений

функціонал мають додатку які є платними. Основна проблема усіх додатків це середня або погана якість інтерфейсу, зручність якого в деяких додатках відкидає будь-яке бажання ним користуватись.

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Після проведення аналізу предметної області та наявного програмного забезпечення по темі мого диплому було прийнято рішення про розробку нового додатку.

Додаток має спрощувати контроль якості продуктів харчування, забезпечувати швидкий розрахунок терміну придатності, нагадувати про витікання терміну. Продукти можна відсортувати, а інформацію – зберегти в зручному форматі.

Має бути функція групування товарів. Додаток має дозволити користувачам групувати товари по типу, місцезнаходженню та іншим параметрам. Користувачі мають мати можливість створювати необмежену кількість груп, орієнтуючись на асортимент своєї їжі.

Також було вирішено створити функцію завантаження фотографій. При добавлені нового продукту можна додати фото. На фото піде лише декілька секунд, але завдяки йому можна буде легко орієнтуватись в загальному списку та усіх інших групах.

Одна з головних функцій це нагадування про кінець термінів придатності. Будильник термінів придатності спростить контроль свіжості продуктів і допоможе уникнути вживання потенційно небезпечного продукту. Достатньо буде лише налаштувати нагадування про терміни придатності та вибрати зручний формат повідомлень.

Також має бути реалізована система сортування продуктів. Список продуктів можна відсортувати по різним параметрам (найменуванню, терміну придатності, даті додання). Завдяки сортуванню можна швидко оцінити кількість

товарів з близьким до завершення терміном придатності і визначити ті, що вже не придатні до вживання.

Враховуючи вищеописані особливості, можна скласти перелік функціональних можливостей, які мають бути у мобільному застосунку:

- додання нових продуктів;
- можливість зробити фото продукту;
- створення груп для різних видів продуктів;
- перегляд усього списку продуктів;
- перегляд різних груп з продуктами;
- сортування продуктів в загальному списку та групах;
- отримання повідомлень про кінець терміну придатності;
- налаштування повідомлень;
- редагування та видалення продуктів;
- редагування та видалення груп;
- реалізація меню з групами та налаштуваннями.

На основі зробленого аналізу вимог до програмного забезпечення розроблено технічне завдання, яке подано у додатку А.

У цьому розділі був проведений детальний аналіз та огляд існуючих видів мобільних додатків та операційних систем для мобільних додатків. Були переглянуті та проаналізовані популярні додатки на тему контролю термінів придатності продуктів харчування. Після чого було проведено аналіз вимог до розроблюваного застосунку, на основі якого було складено технічне завдання.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКА

2.1 Архітектура та функціональна структура додатка

Архітектура – це фундаментальна організація системи, реалізована в її компонентах, їх взаємозв'язках один з одним та з навколишнім середовищем , та керуючими правилами проектування та розвитку системи. [10]

Мінімальна архітектура складається з трьох компонентів визначень (та, відповідно, трьох тематичних описів – наборів робочих продуктів/моделей):

Операційний (практичний) опис (Operational View) – система з точки зору користувача або «оператора». Сюди входять артефакти, які відносяться до етапів практичного застосування системи, сценаріями та потоками робіт:

- опис операцій в графічному або числовому вигляді;
- опис сценаріїв та варіантів використання;
- діаграма потоків задач;
- схеми організаційної структури;
- діаграми інформаційний потоків.

Логічний опис (Logical View) – системи з точки зору керівництва та замовника. Сюди входять артефакти, які визначають межу системи та її оточення, функціональні інтерфейси з зовнішніми системами, основні функції та види поведінки системи, потоки даних, внутрішні та зовнішні набори даних, внутрішні та зовнішні користувачі та внутрішні функціональні інтерфейси;

- принципіальні схеми, а інколи просто функціональна декомпозиція (data flow chart);
- діаграми IDEF0;
- схеми функціональних потоків (FFBD)

Фізичний опис (Physical View) – система з точки зору спеціалістів по проектуванню. Сюди входять артефакти, які визначають фізичну межу системи, компоненти, їх взаємозв'язок та інтерфейси між ними, внутрішні бази даних та

структури даних, інформаційно-технологічну інфраструктуру системи, стандарти, які застосовуються при її розробці:

– деталізовані фізичні блок-схеми;
– топології баз даних;
– документи по контролю сполучень;
– стандарти.

Архітектура повинна чітко показувати: як пов’язані між собою усі ці три види описів «прозорого ящика» дають на виході основні функції системи. Тим самим архітектура невідривно пов’язана з вимогами. Ті вимоги, які максимально впливають на архітектуру, називають архітектурними вимогами.

Ключовим етапом в розробці програмного забезпечення є стадія проектування, тобто моделювання теоретичної основи майбутнього продукту. Найсучасніші засоби програмування позволяють частково об’єднати етапи проектування та кодування, тобто технічної реалізації проекту, будучи заснованими на об’єктно-орієнтованому підході, але повноцінне планування потребує більш детального моделювання.

Якісний аналіз перспектив та можливостей створюваного продукту стане основою для його повноцінного функціонування та виконання усього комплексу покладених на програмне забезпечення задач. Однією з складних частин етапу проектування, наприклад, є вибір інструментів засобів та операційної системи, яких сьогодні на ринку присутня велика кількість.

В рамках даного етапу сторони повинні виконати:

– оцінку результатів проведеного початкового аналізу та виявлення обмежень;
– пошук критичних ділянок проекту;
– формування кінцевої архітектури створюваної системи;
– аналіз необхідності використання програмних модулів або готових рішень сторонніх розробників;

– проектування основних елементів продукту – моделі бази даних, процесів та коду;

– вибір середовища програмування та інструментів розробки, затвердження інтерфейсу програми, включаючи елементи графічного відображення даних;

– визначення основних вимог до безпеки розроблюваного програмного забезпечення.

Оскільки не існує архітектури, яка була би завжди однозначно хорошою чи поганою для поставлених задач, адже це залежить від суб'єктивної оцінки розробника, було вирішено для полегшення процесу вибрати певні шаблони проектування – патерни.

Патерн (шаблон) – це повторювана архітектурна конструкція, яка представляє з себе рішення проблеми проектування в рамках деякого часто виникаючого контексту.

Для проектування архітектури мобільних додатків краще всього підходять такі патерни проектування як:

– Model-View-Controller (Модель-Представлення-Контролер);

– Model-View-Presenter (Модель-Представлення-Представник);

– Модель-Представлення-Модель виду (MVVM);

– View–Interactor–Presenter–Entity–Router (Представлення-Взаємодія-Ведучий-Сутність-Провідник).

Головним патерном виступає Model-View-Controller (MVC). MVC – це фундаментальний патерн, який знайшов застосування в багатьох технологіях, дав розвиток новим технологіям і щодня полегшує життя розробникам. Вперше патерн MVC з'явився у мові SmallTalk. Розробники повинні були придумати архітектурне рішення, яке дозволяло би їм відокремити графічний інтерфейс від

бізнес логіки, а бізнес логіку від даних [11]. Таким чином, в класичному розумінні, MVC складається з трьох частин, які дали йому назву.

Модель (Model) – частина, яка вміщує в собі функціональну бізнес-логіку додатку. Модель повинна бути повністю незалежною від інших частин продукту. Модельний шар нічого не повинен знати про елементи дизайну, та яким чином він буде відображатись.

Модель має такі признаки:

- модель – це бізнес-логіка додатку;
- модель має знання про саму себе і не знає про контролер та представлення;
- для деяких проектів модель – це просто шар даних (DAO, бази даних, XML-файли);
- для інших проектів модель – це менеджер бази даних, набір об’єктів або просто логіка додатку.

Представлення (View) – частина, яка відображає дані, які отримує від Моделі. Представлення не може напряму впливати на модель. Можна сказати, що представлення має доступ «лише для читання» до даних.

Представлення має наступні признаки:

- в представлені реалізується відображення даних, які отримуються від моделі будь-яким методом;
- в деяких випадках, представлення може мати код, який реалізує деякі бізнес-логіку.

Контролер (Controller) – це частина, яка керую запитами користувача, які отримуються у вигляді запитів HTTP GET або POST, коли користувач натискає на елемент інтерфейсу для виконання різних дій. Його основна функція – викликати та координувати дії необхідних ресурсів та об’єктів, які задає користувач. Зазвичай контролер викликає відповідну модель для задачі та вибирає підходящий вигляд.

Контролер має наступні признаки:

– контролер визначає, які представлення повинні бути відображені в даний момент;

– події представлення можуть вплинути тільки на контролер. Контролер може вплинути на модель та визначити інше представлення;

– можливі декілька представлень лише для одного контролера.

Схема взаємодії даних в моделі MVC представлена на рисунку 2.1.

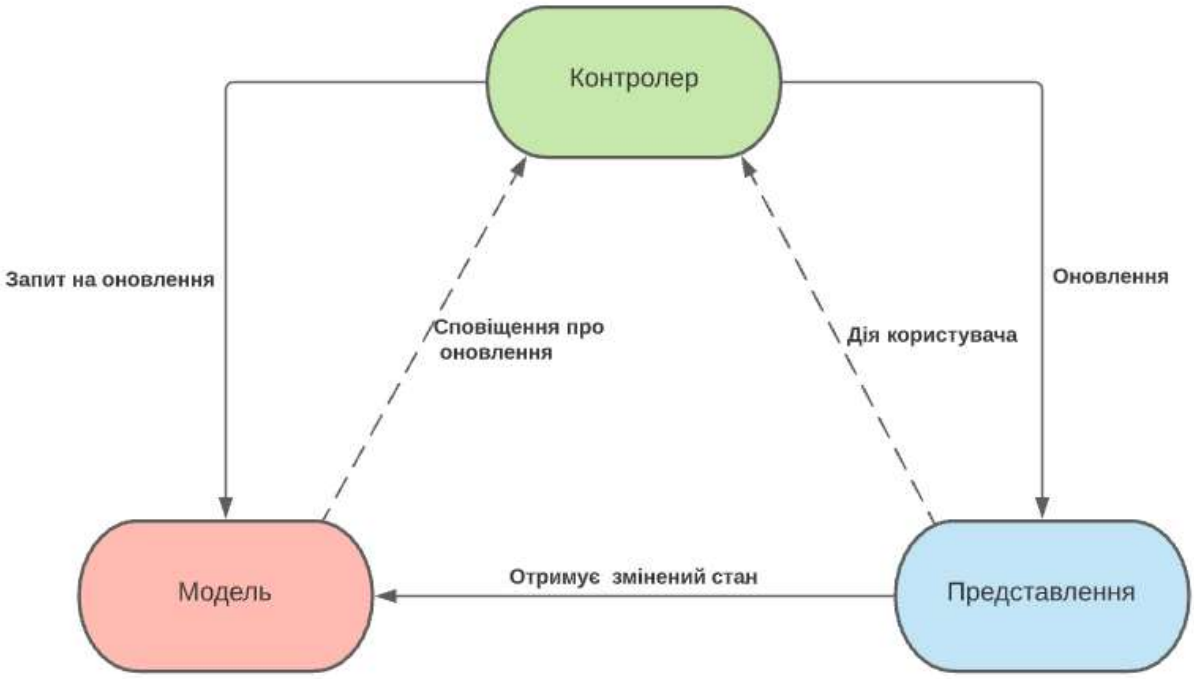


Рисунок 2.1 – Діаграма взаємодії між компонентами шаблону MVC

Серед переваг MVC можна виділити простоту реалізації, швидкість розробки, висока швидкість роботи додатку. Серед недоліків – слабке розподілення обов'язків між компонентами, що може призвести до проблем з масштабуванням проекту.

Модель-Представлення-Представник – це патерн проектування, який є похідним MVC, що відділяє візуальне зображення та поведінку обробки подій у різні класи, а саме: Представлення (View) та Представник (Presenter). Даний

підхід дозволяє створювати абстракцію представлення. Для цього необхідно виділити інтерфейс представлення з визначеними набором властивостей та методів. Представник, в свою чергу, отримує посилання на реалізацію інтерфейсу, підписується на події та по запиту змінює модель [12].

Представник має наступні признаки:

– двостороння комунікація з представником;

– представлення взаємодіє напряму з представником, шляхом вивозу власних функцій та подій екземпляра представника;

– представник взаємодіє з View шляхом використання спеціального інтерфейсу, реалізованого представленням;

– один екземпляр представлення пов’язаний з одним відображенням.

Принцип роботи MVP зображено на рисунку 2.2.

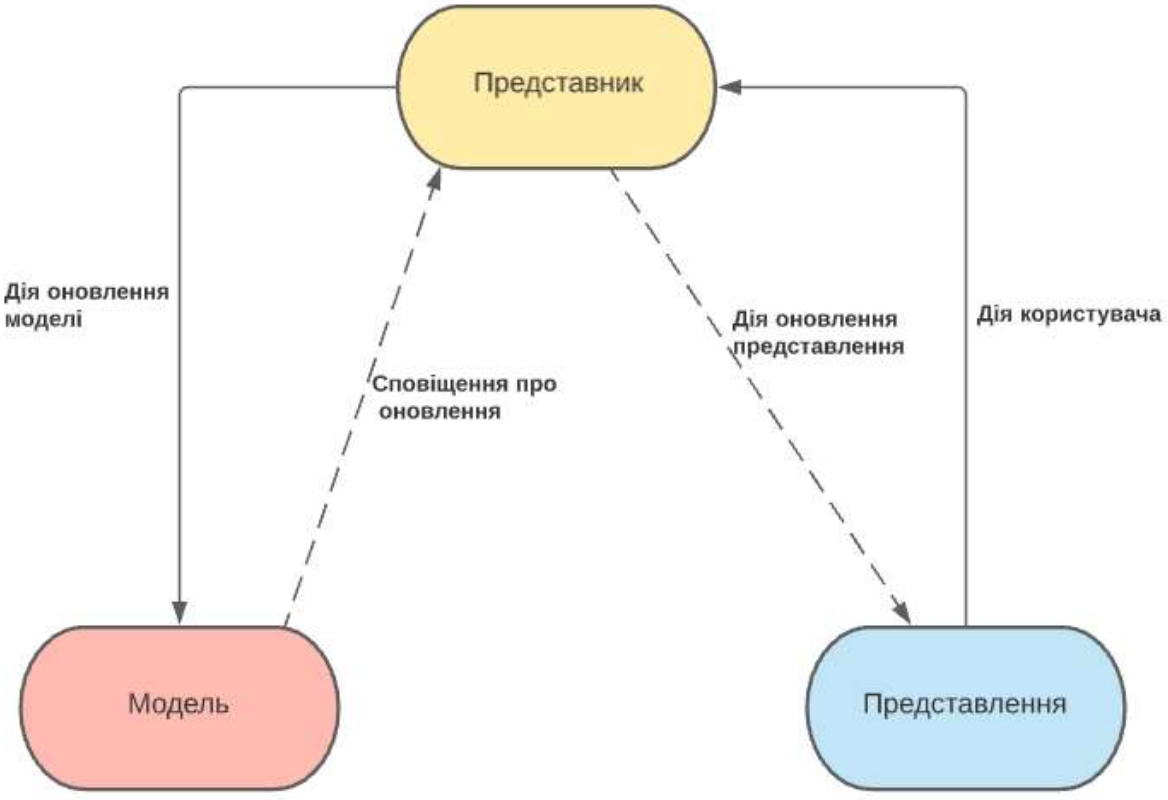


Рисунок 2.2 – Діаграма взаємодії між компонентами шаблону MVP

Серед переваг MVP є більш просте налагодження, яке дозволяє розробникам простіше робити налагодження в додатках, так як MVP використовує три різні рівні абстракції. Дозволяє повторне використання коду. При побудуванні MVP-архітектури розробники можуть багатократно використовувати уже написаний код. Це стає можливим завдяки багатьом представленням, які контролюють Views. Такий підхід більш надійний порівняно з використанням одного Presenter. Ефективне розділення функціональних можливостей додатка – під цим розуміється відділення бізнес-логіки та інших частин від класів активності та фрагментів, які в свою чергу, краще забезпечують розділення функціональних можливостей.

MVVM – це патерн проектування, який дозволяє виділити логіку додатка від візуальної частини. Такий підхід дозволяє зв'язувати елементи представлення з властивостями та подіями View-моделі. Можна сказати, що кожен шар цього патерну знає про існування іншого шару [13].

До признаків View-моделі можна віднести:

– двостороння комунікація з представленням;

– View-модель – це абстракція представлення.

– View-модель не має посилання на інтерфейс представлення.

– один екземпляр View-моделі пов'язаний з одним відображенням.

Принцип роботи MVVM зображено на рисунку 2.3.

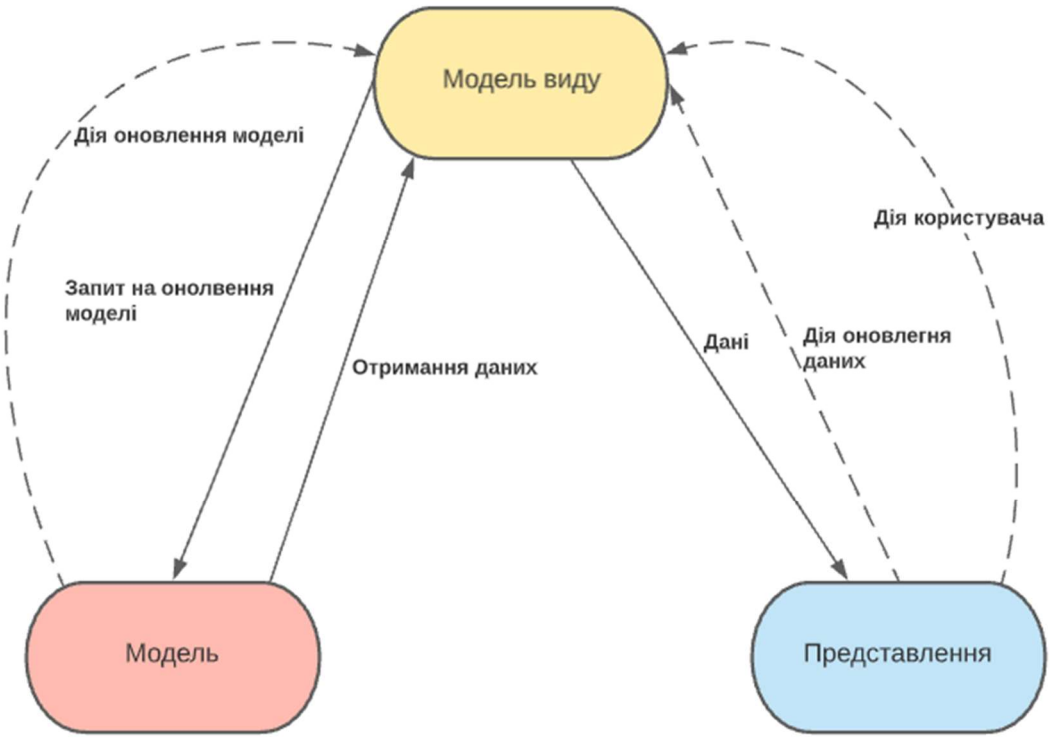


Рисунок 2.3 – Діаграма взаємодії між компонентами шаблону MVVM

Відмінність MVP від MVVM в тому що в MVVM в ролі посередника виступає View-Model (модель виду), яка і відповідає за представлення з моделі та перетворення формату даних.

Перевагою архітектури MVVM є автоматичне пов’язування представлення з моделлю, в той час як у MVP це потрібно створювати самостійно. Але з цього випливає головний недолік, оскільки для автоматичного пов’язування потрібно використовувати спеціальні фреймворки, що негативно впливає на швидкість роботи додатка. MVVM краще всього використовувати замість класичної MVC у випадках, коли в платформі, на якій проводиться розробка, є «пов’язані дані».

View–Interactor–Presenter–Entity–Router (VIPER) – це новий патерн проектування, який використовується в розробці мобільних додатків. Заснований цей патерн на принципі єдиної відповідальності (Single Responsibility) та парадигмі SOLID. Спочатку цей патерн називався VIP архітектурою, але щоб не переплутати з іншою архітектурою (“Very Important Architecture”), його перейменували в VIPER, добавивши E та R [14]

Структурно VIPER поділяється на 5 основних компонентів:

– View (Представлення) відображає те, що повідомив Presenter та передає введення даних користувачем назад в Presenter;

– Interactor (Взаємодія) вміщує в собі бізнес-логіку для керуванням об’єктами (Entity), щоб виконувати певну задачу. Задача виконується в Interactor, незалежно від любого UI;

– Presenter (Представник), який вміщує в собі логіку підготовки змісту для відображення (отриманого з Interactor) та для реакції на введення даних користувачем (запитуючи нові дані від Interactor);

– Сутність (Entity) це об’єкти, якими керує Interactor. Entity лише керує Interactor. Він ніколи не передає сутності рівень представлення (тобто Presenter);

– Провідник (Router) це компонент, який відповідає за маршрутизацію між усіма компонентами.

Принципи роботи компонентів VIPER представлені на рисунку 2.4

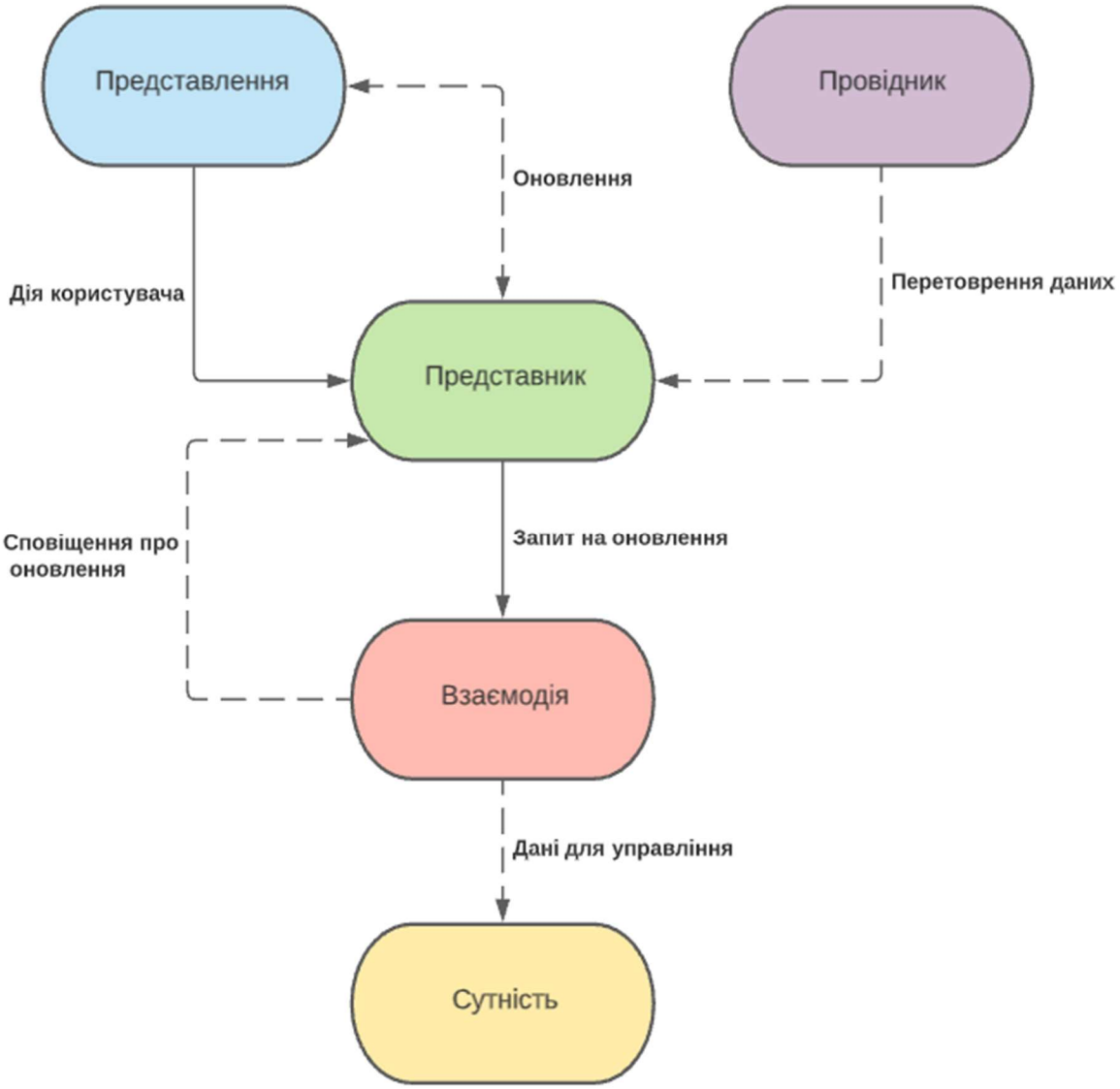


Рисунок 2.4 – Діаграма взаємодії між компонентами шаблону VIPER

Головна перевага архітектури VIPER над іншими патернами це забезпечення повної незалежності усіх компонентів, а з цього відповідно випливає легкість масштабування проекту. Недоліком цього патерну є збільшення складності проекту та його розміру.

Проаналізувавши усі вищезазначені шаблони проектування, я вибрав найоптимальніше архітектуру для мого мобільного застосунка, а саме MVP, оскільки вона не ускладнює код програми і підходить для невеликих проектів не впливаючи на швидкодію негативним чином.

Також однією з найголовніших частин додатку є збереження вмісту записів, тому важливо правильно спланувати що, як і куди записувати.

Збереження записів можна реалізувати двома методами:

– збереження у файл;

– збереження у базу даних.

До переваг бази даних можна віднести більші можливості, але недоліками є складніша архітектура та реалізація. Оскільки в моєму додатку потрібно зберігати лише один 1 таблицю з продуктом, було вирішено вибрати збереження у файл.

2.2 Проектування інтерфейсу користувача

Для того щоб розробити інтерфейс мобільного застосунку потрібно знати базові правила UX/UI-дизайну. По суті інтерфейс мобільного додатку це інтерфейс веб сайту, але на відміну від звичайних сайтів, які можна відкрити на комп’ютері, в мобільній версіях сайтів, а значить і в додатках, потрібно розуміти, що взаємодія з додатком відбувається не через миш та клавіатуру, а через сенсорний екран.

Користувацький інтерфейс, або UI (User Interface) – інтерфейс продукту, який може бути графічним, тактильним, голосовим або звуковим. UI-дизайн – це процес візуалізації прототипу, який розробили на основі досвіду користувача та досліджень цільової аудиторії.

UI-дизайн включає в себе роботу над графічною частиною інтерфейсу: анімацією, ілюстраціями, кнопками, меню, фотографіями та шрифтами. Сам дизайнер вибирає кольорову палітру та місцезнаходження об’єктів в інтерфейсі та чи відповідають вони таким категоріям як зручність, легкість, правильність.

UX (User Experience) – означає «досвід користувача». В загальному розумінні це поняття про увесь досвід, який отримує користувач при взаємодії з сайтом або додатком. UX-дизайн відповідає за функції, адаптивність продукту а

те, які емоції він викликає у користувача. Чим зрозуміліший інтерфейс, тим легше користувачу отримати результат. UX-дизайн – це проектування інтерфейсу на основі дослідів досвіду та поведінки користувачів [15].

Розробляючи інтерфейс мого додатка для контролю термінів придатності продуктів харчування я притримувався таких принципів:

– додаток повинен бути унікальним та виділятись серед конкурентів;

– енергозбереження, легкість в керуванні, збереження дій – усе це показує турботу про користувача;

– облік користувацького досвіду, місцезнаходження елементів на усім звичному рівні, наприклад, кнопки редагувати та видалити мають бути на достатній відстані, щоб випадково не задіти одну з них;

– пророоблення відклику, користувач повинен розуміти, що його запит виконується, відбуваються певні дії;

– легке введення тексту з мобільної клавіатури;

– упор на роботу з сенсорним екраном, усі елементи мають бути зрозумілі та легко взаємодіяти;

– мінімум спливаючих вікон.

Також потрібно дотримуватись основних вимог по розробці інтерфейсу мобільного додатка, а саме:

– використовувати звичних UI-елементів – прямокутні кнопки, місцезнаходження меню, вертикальний порядок записів про додані продукти;

– високий рівень візуалізації, який дозволяє гармонічно знаходитись в додатку;

– зниження рівня «шуму» інтерфейсу, важливі елементи повинні показуватись на самому початку та бути видимими;

– наявність виклику до дії, де воно є необхідним;

– зручне введення даних, наприклад, при виставленні дати продукту можна скористатись як графічним календарем, так і записати в ручну;

– зручне виведення даних, наприклад, зрозумілі шрифти;

– явний показ можливостей, щоб користувач відразу зрозумів, що він отримує;

– кастомізація та можливість індивідуального налаштування під потреби користувача.

Виконуючи усі ці вимоги, додаток має вийти ефективним, зрозумілим, корисним та зручним у використанні. Притримуючись цих вимог був розроблений прототип головного екрану додатку (рисунок 2.5), на який будуть виводитись усі наші добавлені продукти.

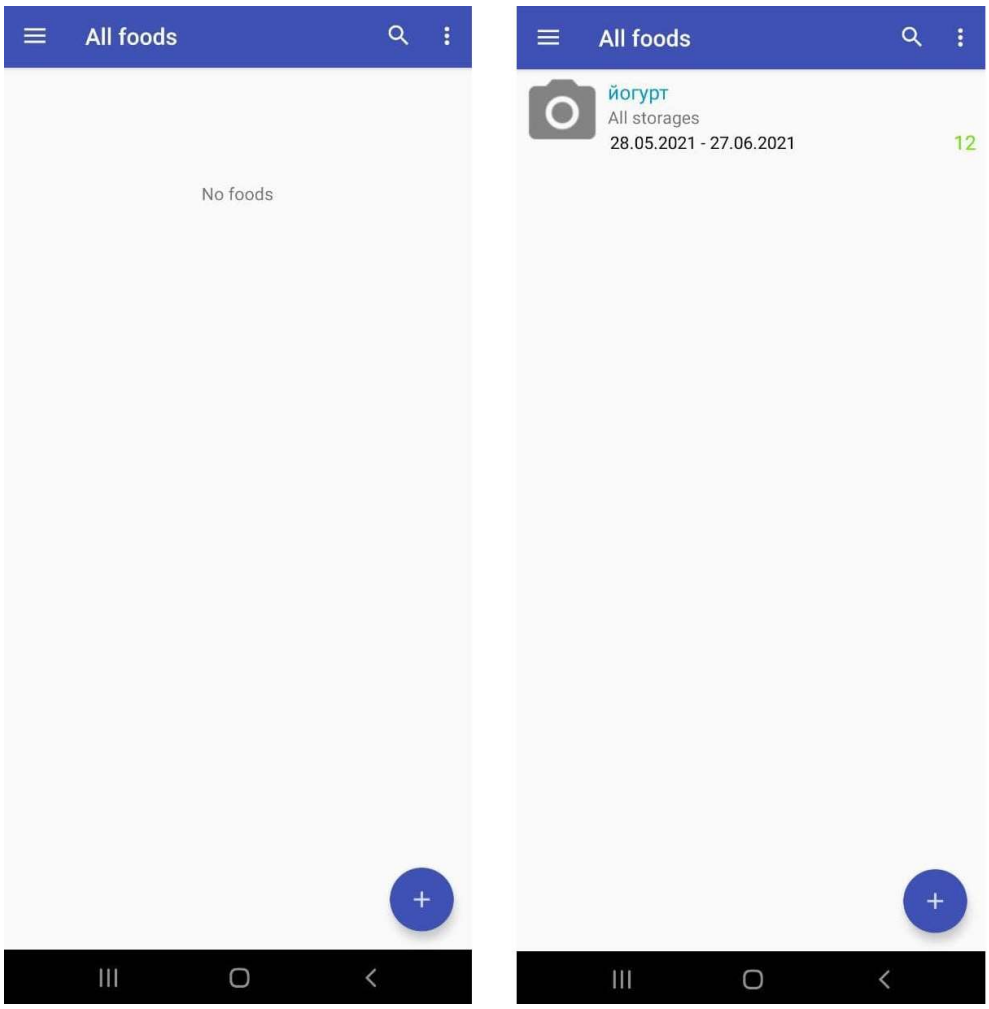


Рисунок 2.5 – Головний екран додатку

На рисунку 2.5 в правому нижньому куті сторінки розміщена кнопка за знаком плюс, при натисканні якої відкривається сторінка додавання нових записів. На рисунку 2.6 зображений екран додавання нових продуктів.

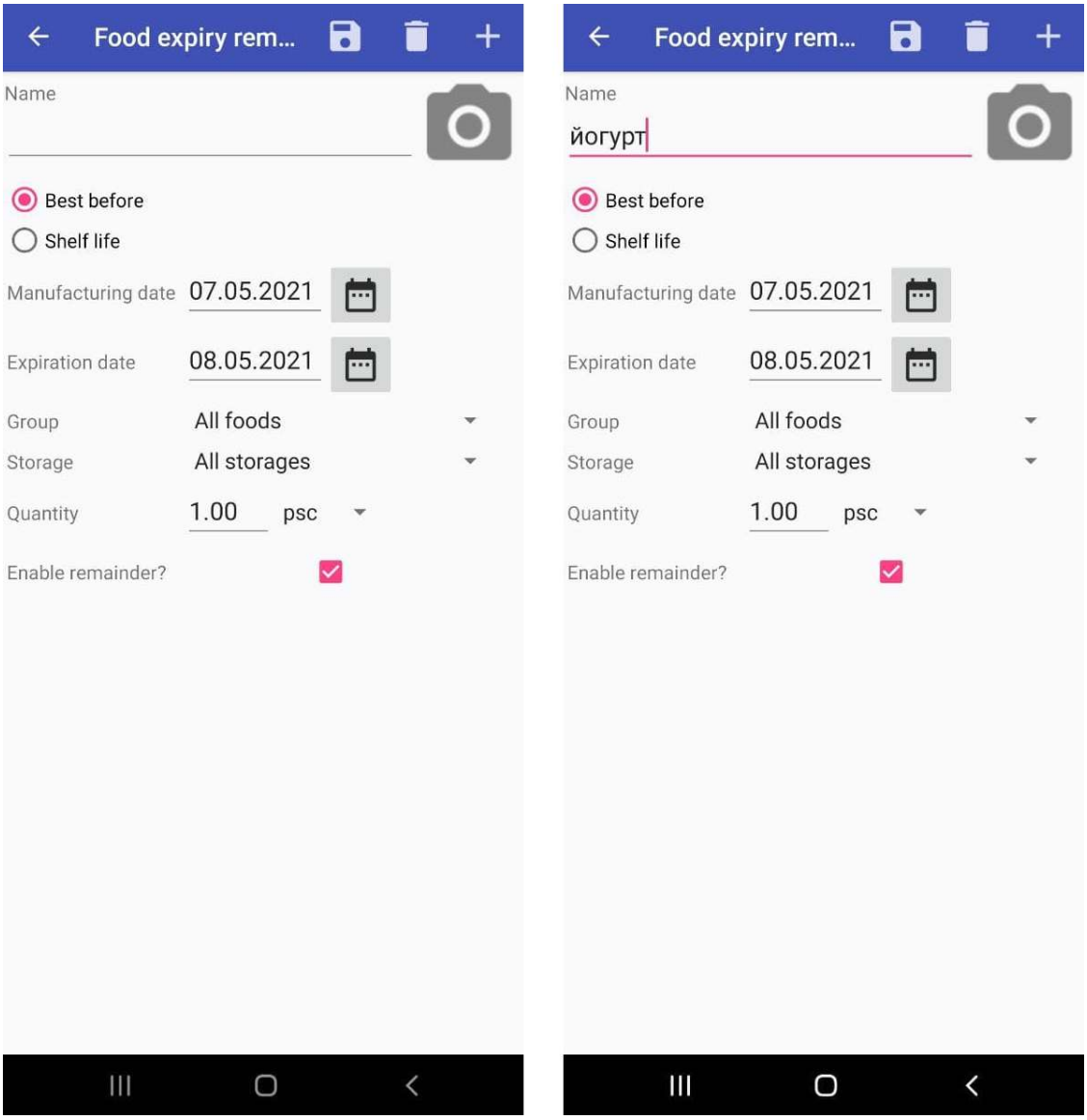


Рисунок 2.6 – Екран додавання продуктів

На цьому екрані знаходиться поле для введення назви продукту, поле для вибору його дати виготовлення та дати терміну придатності. Також тут можна вибрати групу їжі, вказати місце зберігання, кількість одиниць та тип одиниць і в кінці вибрати поставити прапорець – створювати нагадування чи ні. Зверху на

панелі інструментів знаходиться кнопка збереження, видалення та додавання наступного продукту.

На цьому етапі можна створити повну схему усіх переходів між усіма екранами додатку, де будуть зазначені усі поля для введення, пункти меню та кнопки, які є на кожному з екранів додатку. Головним (стартовим) екраном застосунку є сторінка з «мої продукти». На рисунку 2.7 зображена повна схема переходів екранів додатку.

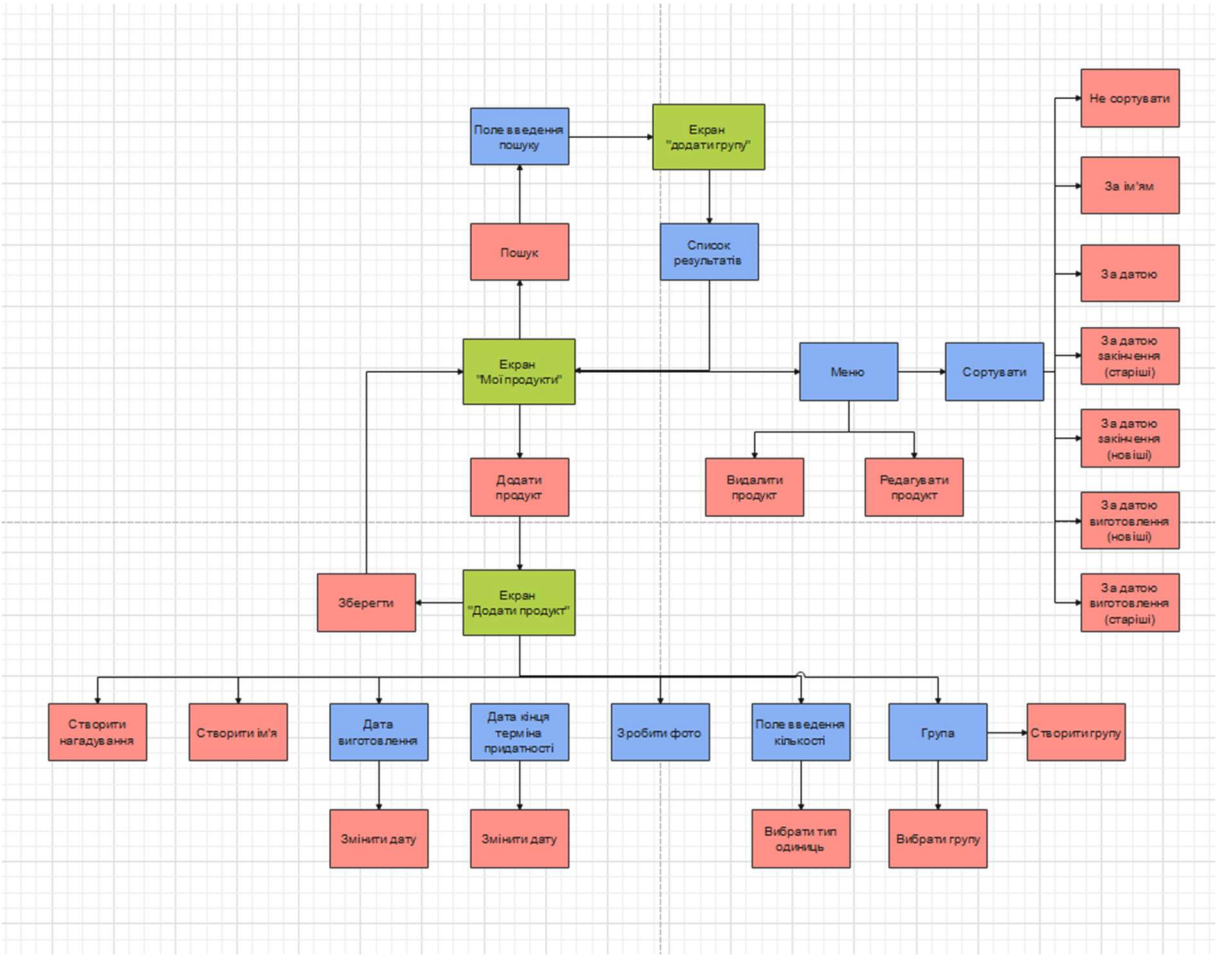


Рисунок 2.7 – Схема переходів екранів додатку

2.3 Розробка алгоритму роботи мобільного додатка

Після визначення архітектури та проектування інтерфейсу додатка можна перейти до створення алгоритму роботи програми. Для цього необхідно

визначити ключові процеси які відбуваються при взаємодії з мобільним додатком. Їх можна визначити за допомогою схеми переходів екранів додатку, адже на ній зображені основні екрани додатку, та усі переходи між ними. З цього можна виділити такі процеси:

– створення записів про продукти;
– пошук продуктів.

Процеси можна зобразити за допомогою діаграм переходів станів. Ці діаграми є графічною формою представлення кінцевого автомату – математичної абстракції, яка використовується для моделювання детермінованої поведінки технічних об’єктів або об’єктів реального світу.

Відповідно до цих процесів було створено діаграми переходів станів, що зображені на рисунках 2.8 та 2.9.

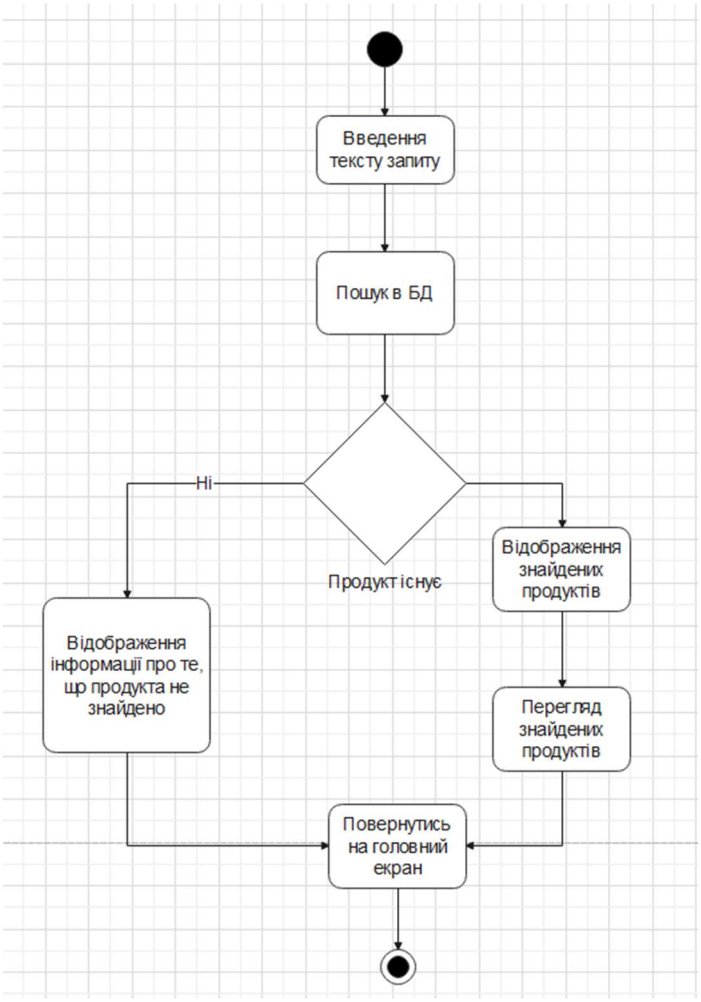


Рисунок 2.8 – Діаграма пошуку продуктів

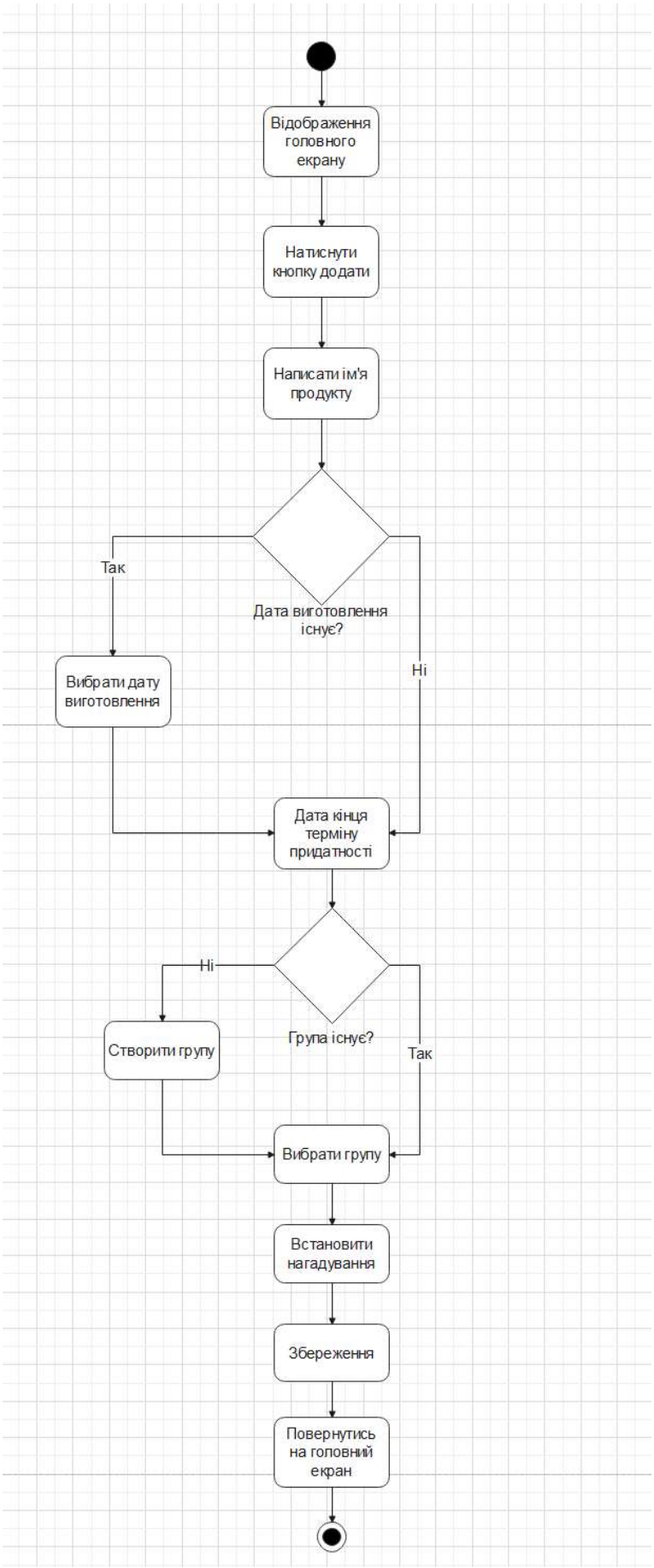


Рисунок 2.9 – діаграма створення записів про продукти

2.4 Аналіз та вибір технологій і методів реалізації мобільного додатка

Швидкий розвиток інформаційних технологій в останній час привів до появи великої кількості нових різних приладів та технологій, в тому числі планшетів, смартфонів, смарт-годинників та інших гаджетів. Усі вони більш міцно входять в наше життя і стають звичайним ділом. Зараз майже в кожної людини є смартфон і скоріше за все це смартфон з операційною системою Android, адже це найпопулярніша операційна система серед усіх мобільних девайсів. Для створення мобільних додатків під цю платформу було випущено безліч інструментів та багато мов програмування.

Оскільки при проведенні аналізу з першого розділу стосовно типів мобільних додатків було вибрано під розробку нативний мобільний додаток під систему Android, то саме під такі критерії слід вибирати мову програмування.

Переважно виділяють шести мов програмування, за допомогою яких можна створити Android-додаток:

- C#;
- Dart;
- Java;
- JavaScript;
- Kotlin;
- Python.

Потрібно проаналізувати усі ці мови програмування опираючись на плюси та мінуси цих мов, щоб вибрати ту, яка краще всього підходить під мій мобільний додаток.

Java – мультифункціональна об’єктивно-орієнтована мова програмування зі строгою типізацією. На теперішній час Java є найпопулярнішою мовою програмування для розробки мобільних додатків під систему Android. Хоча Google активно просуває Kotlin як мову, яка повинна буде замінити Java. Майже усі мови програмування під мобільне програмування будуються на Java, так як

вона має найбільш широкий функціонал в мобільному програмуванні. Операційна система Android сама по собі написана на Java (Java Virtual Machine), розробленою Googlei повністю сумісною зі звичною для Java-розробників JVM Oracle. Для розробки під Android існує велика кількість готових інструментів, більшість з яких створені Google або за участі Google. Сьогодні переважна більшість Android-розробників користується IDE Android-Studio, яка по замовчуванню «розуміє» мову Java та Kotlin. Серед плюсів мови Java є те, що вона працює на усіх платформах, є об'єктно-орієнтованою мовою (що приводить до легкого масштабування та вдосконалення додатка), величезну кількість бібліотек на усі випадки життя, чудову документацію, дуже хороші інструменти для роботи.

Kotlin представляється як сучасна, статистично типізована і одна з найбільш швидко розвиваючих мов програмування, створена та розвинута компанією JetBrains. Kotlin можна використовувати для написання найрізноманітніших видів додатків. Це і додатки для мобільних пристроїв – Android та iOS. При чому Kotlin дозволяє писати кросплатформенний код, який буде застосовуватись на усіх платформах. Це і веб-додатки, при чому як серверні, які обробляються на стороні сервера – бекенда, так і браузерні клієнтські додатки – фронтенд. Kotlin також можна використовувати для створення додатків на комп'ютер, для Data Science і тому подібне.

Таким чином, коло платформ, під які можна створювати додатки на мові Kotlin, дуже широке – Windows, Linux, Mac OS, iOS та Android.

Найпопулярнішим напрямом, де використовується Kotlin, є в перш за все розробка мобільних додатків під операційну систему Android. При чому настільки популярна, що компанія Google на конференції Google I/O 2017 найменувала Kotlin однією з офіційних мов програмування для розробки під платформу Android (на рівні з Java та C++), а інструменти для роботи з даною мовою були по замовченню включені в функціонал середовища розробки програм Android Studio.

Dart – високорівнева, інтерпретована мова програмування. Розробляється компанією Google. Є альтернативою мові JavaScript – по крайній мірі так позиціонують цю мову програмування сама компанія. Перша версія інтерпретатора стала доступною в середині осені 2011 року. Головний розробник цією мови програмування Марк Міллер пояснює необхідність створення Dart тим, що у JavaScript є «фундаментальні недоліки», які неможливо ніяк виправити, а нова мова програмування позбавлена цих проблем. На мові програмування Dart створюються мобільні додатки з описом графічного інтерфейсу та усієї логіки роботи. Результат роботи добавляється в нативний додаток, як і картинки, шрифти і тому подібне. Одночасно в нативній частині додатку створюється один єдиний екран, де довантажуються віртуальна машина Dart, яка виконує Flutter. Dart використовує SDK Flutter для мобільної розробки. Серед переваг цієї мови можна назви високу швидкодію програм, створених на цій мові, а серед недоліків те, що мова достатньо молода і під неї створена невелика кількість інструментів.

Мова C# розроблена компанією Майкрософт, одна з найпопулярніших мов програмування в усьому світі. C# використовується для розробки програм для персональних комп'ютерів, створенню важких веб-сервісів та мобільних додатків. Мова є строго статично типізованою та об'єктно орієнтованою. Для розробки мобільних додатків використовується платформа Xamarin [16]. Розробники використовують кросплатформений фреймворк Xamarin в основному для створення невеликих проектів або прототипів. Перевагами фреймворка є:

– це інструмент .NET, тому він підходить тим, хто спеціалізується саме на .NET розробці;

– дозволяє використовувати велику кількість .NET бібліотек. У .NET величезний репозиторій бібліотек;

– в розробці використовується мова C#, яка дозволяє будувати об'єктно-орієнтовану модель і ізолювати рівні;

– UI налаштовується для кожної платформи окремо нативними інструментами.

В цілому в базі Xamarin можна побудувати любе архітектурне рішення.

JavaScript – одна з найпопулярніших мов програмування у світі, чудово підходить під різні задачі. Для мови JavaScript є фреймворк React, розроблений компанією Facebook, який є одним з найпопулярніших веб-платформ JavaScript доступних на сьогоднішній день. React Native – це поріднене середовище, яке дозволяє створювати мобільні додатки на JavaScript з використанням принципів React. Інтерфейси додатків розроблені з використанням тегів макету у стилі HTML і таблиць стилів в стилі CSS. Як слідує з назви, React Native створює власні мобільні додатки. Одна з причин, чому так багато сучасних розробників люблять React Native, це тому що багато додатків на цьому фреймворці мають тенденцію бути кращими, ніж додатки написані на інших середовищах JavaScript. Серед переваг можна назвати швидку збірку проекту та просту передачу даних через мережу з використанням API.

Python – це скриптована мова, яка використовується для рішення великого об’єму даних найрізноманітніших задач. Для розробки мобільних додатків на Python є фреймворк Kivy.

Kivy – це бібліотека Python, яка має відкритий код, призначена для розробки кросплатформених GUI додатків. Вона дозволяє писати додатки з графічним інтерфейсом на чистому Python, які працюють на основних платформах (Windows, Linux, MacOS, Android, IOS). В Kivy вбудований налаштовуваний набір інструментів користувацького інтерфейсу, який надає власні кнопки, форми введення тексту і тому подібне. Це означає, що ці віджети не відображаються за допомогою елементів керування користувацького інтерфейсу власної платформи.

Проаналізувавши усі мови програмування під нативну мобільну розробку для операційної системи Android було вирішено обрати мову програмування Java,

оскільки це офіційна мова програмування, для неї є найбільша кількість інструментів і вона ідеально підходить під створення мобільних додатків.

Середовищем для написання програмного коду було обрано Android Studio. Це інтегроване середовище розробки від компанії Google, за допомогою якого розробникам доступні інструменти для створеннях додатків на платформу Android OS. Android Studio можна встановити на Windows, Mac та Linux. Обліковий запис розробника додатків в Google Play App Store коштує $25. Android Studio створювалась на базі IntelliJ IDEA [17].

У цьому розділі було розроблено архітектуру додатку, спроектовано інтерфейсу додатку, який відповідає правилам та сучасним течіям розробки мобільних додатків, розроблені алгоритми роботи додатку та вибрані інструменти для розробки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Реалізація логіки мобільного додатку

У попередньому розділі були виділені та спроектовані активності мобільного додатку. До них належить головна активність, на якій відображені додані продукти харчування, кнопка для додання нових, головне меню, меню відображення зовнішнього вигляду списку продуктів та функції редагування та видалення продуктів.

Також є активність додавання продуктів, в якій мають бути реалізовані методи по налаштуванню дати продукту, можливість зробити фотографію, визначення кількості на надання групи.

Активність по пошуку продуктів має мати метод для знаходження продуктів по назві з уже добавлених продуктів харчування.

Також реалізовані фрагменти, які виступають у ролі діалогів. Серед них є фрагмент для додання їжі, фрагмент для додання секцій, фрагменти для підтвердження редагування та видалення секції, фрагмент для редагування доданих продуктів, фрагмент який містить інформацію про сам додаток.

Для налаштувань є своя активність та свій фрагмент.

Фрагменти представляє з себе шматок візуального інтерфейсу додатку, який може використовуватись повторно та багатократно. У фрагмента може бути власний файл layout, у фрагментів є також свій власний життєвий цикл. Фрагмент існує у контексті activity і має свій життєвий цикл, поза activity сам по собі він існувати не може.

В коді класу MainActivity є методи для відкриття бокового меню, редагування секцій, перевірки нагадування, встановлення нагадування, додавання/редагування/видалення груп та продуктів.

Метод для відкриття головного бокового меню:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
```

```
// Inflate the menu;
//this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.main, menu);
// Get the SearchView and set the searchable configuration
SearchManager searchManager = (SearchManager)
getSystemService(Context.SEARCH_SERVICE);
MenuItem searchItem = menu.findItem(R.id.action_search);
SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem);
final Context context = this;
searchView.setOnQueryTextListener(
new SearchView.OnQueryTextListener() {
@Override
public boolean onQueryTextSubmit(String query){
Intent intent = new Intent(context, SearchResultActivity.class);
intent.putExtra(QUERY, query);
context.startActivity(intent);
return false;}
@Override
public boolean onQueryTextChange(String newText){
return false;}});
MenuItem filterMenuItem = menu.getItem(1);
//get filter_collection menuItem
//Menu.NONE -> Value to use for group and item identifier integers when you
don't care about them.
//get subMenu for adding itemMenus dynamically
filterSectionMenuOption = filterMenuItem.getSubMenu();
updateFilterSectionOptions();
return true;
}
```

Метод для вибору секції:

```
private String[] getSectionsArray() {
List<String> nameList = new ArrayList<>();
sectionCursor = dbUtils.getAllSections();
int nameCol =
sectionCursor.getColumnIndex(SectionContract.SectionEntry.COLUMN_SECTION_NAME);
while (sectionCursor.moveToNext()) {
nameList.add(sectionCursor.getString(nameCol));}
return nameList.toArray(new String[nameList.size()]);
}
```

Метод для редагування секції:

```
@Override
public void onEditFinished() {
updateFilterSectionOptions();
updateUi(sortOption,sectionFilterId,filterOptionActivated);
}
@Override
public void onConfirmedDeletion(int sectionId) {
new DBUtils(this, this).deleteAllFoodsInsideSection(sectionId);
new DBUtils(this, this).deleteSectionById(sectionId);
//disable filter by section at deleting any section
filterOptionActivated = false;
updateUi(sortOption, sectionFilterId, filterOptionActivated);
updateFilterSectionOptions();//update section filter options
}
```

Метод для перевірки нагадування:

```
public void checkIfSetAlarm() {
SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(this);
Boolean notificationDeliveredToday =
sharedPreferences.getBoolean(NOTIFICATION_DELIVERED_TODAY, false);//false for
default
if(notificationDeliveredToday) {
//Log.d(TAG, "Notification already delivered today");
int deliveredDay = sharedPreferences.getInt(NOTIFICATION_DELIVERED_DAY, 32);
//Log.d(TAG, "Delivered Day: " + deliveredDay);
Calendar cal = Calendar.getInstance();
cal.setTime(new Date());
int day = cal.get(Calendar.DAY_OF_MONTH);
//Log.d(TAG, "Today: " + day);
if(deliveredDay != day) {
//Log.d(TAG, "It's another day, let's schedule an alarm again");
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putBoolean(NOTIFICATION_DELIVERED_TODAY, true);
editor.commit();
setAlarm();}
} else {
```

```
//Log.d(TAG, "Notification not delivered yet today, let's schedule an alarm");
setAlarm();}}
```

Метод для створення нагадування:

```
public void setAlarm() {
//schedule an alarm at 8:00 a.m. and every day therefore
Intent serviceIntent = new Intent(this, NotificationService.class);
serviceIntent.setAction(NotificationService.ACTION_CREATE_NOTIFICATION);
//wrap intent inside the pendingIntent
PendingIntent pendingIntent = PendingIntent.getService(this, 0, serviceIntent,
0);
AlarmManager alarmManager = (AlarmManager)
this.getSystemService(Context.ALARM_SERVICE);
// Set the alarm to start at approximately 8:00 a.m.
Calendar calendar = Calendar.getInstance();
calendar.setTimeInMillis(System.currentTimeMillis());
calendar.set(Calendar.HOUR_OF_DAY, 8);
alarmManager.setInexactRepeating(
AlarmManager.RTC_WAKEUP,
calendar.getTimeInMillis(),//set alarm at 8:00 p.m.
AlarmManager.INTERVAL_DAY,//you must use AlarmManager interval constants
pendingIntent);
}
public void updateWidget() {
WidgetService.startActionUpdateWidgets(this);
}}
```

Метод для видалення продуктів:

```
@Override
public void deleteFood(Food deletedFood) {
final NotifyInterfaceUtils notifyInterfaceUtils = this;
final Context context = this;
final Food finalDeletedFood = deletedFood;
mySnackbar.setText(this.getResources().getString(R.string.snack_bar_deleted) +
finalDeletedFood.foodName);
//insert the food again
```

```
mySnackbar.setAction(this.getResources().getString(R.string.snack_bar_undo), new
View.OnClickListener() {
@Override
public void onClick(View view) {
new DBUtils(context, notifyInterfaceUtils).insertFoodIntoDB(finalDeletedFood);}
}).show();}
```

Метод для вибору опцій в меню. В цьому методі реалізований вибір з меню таких пунктів як види сортування:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId();
switch (id) {
case R.id.action_sort_remove:
sortOption = 0;
updateUi(sortOption, sectionFilterId, filterOptionActivated);
return true;
case R.id.action_sort_date:
sortOption = 1;
updateUi(sortOption, sectionFilterId, filterOptionActivated);
return true;
case R.id.action_sort_ascending:
sortOption = 2;
updateUi(sortOption, sectionFilterId, filterOptionActivated);
return true;
case R.id.action_sort_descending:
sortOption = 3;
updateUi(sortOption, sectionFilterId, filterOptionActivated);
return true;
case MENU_ITEM_ID_REMOVE:
filterOptionActivated = false;
updateUi(sortOption, sectionFilterId, filterOptionActivated);
return true;}
//get sections array from db
Section[] sections = new DBUtils(this).getAllSectionsArray();
for (int i = 0; i < sections.length; i++) {
```

```
if(id == (MENU_ITEM_ID + i)) {
filterOptionActivated = true;
sectionFilterId = sections[i].sectionId;
//update ui with filtered section array
updateUi(sortOption, sectionFilterId, filterOptionActivated);
return true;}}
return super.onOptionsItemSelected(item);}
```

Метод для відображення результатів пошуку з активності SearchResultActivity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_search_result);
getSupportActionBar().setTitle(R.string.search_activity_text);
recyclerView = (RecyclerView) findViewById(R.id.rv_search_result);
LinearLayoutManager layoutManager = new LinearLayoutManager(this);
recyclerView.setLayoutManager(layoutManager);
//to designate that the contents of the RecyclerView won't change an item's size
recyclerView.setHasFixedSize(true);
Intent intent = getIntent();
//populate ui
if(intent.hasExtra(QUERY)) {
queryString = intent.getStringExtra(QUERY);
populateUi(queryString);
}}
```

Метод для створення нових записів про продукти з фрагмента AddFoodDialogFragment:

```
public void onClick(DialogInterface dialog, int id) {
String foodName = nameEditText.getText().toString();
int foodUnit = unitSpinner.getSelectedItemPosition();
double foodQuantity = Double.parseDouble(quantityEditText.getText().toString());
int foodCategory = categorySpinner.getSelectedItemPosition();
Timestamp timeStamp = new Timestamp(System.currentTimeMillis());
String foodRegisteredTimestamp = timeStamp + "";
```

```
String myFormat = "yyyy-MM-dd";
SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.US);
String foodExpireDate = "0000-00-00";
//checks if the user input a date of expiry
if(expireDateSettled) foodExpireDate = sdf.format(myCalendar.getTime());
int sectionId = getSectionId(sectionSpinner.getSelectedItemPosition());
new DBUtils(getActivity()).insertFoodIntoDB(foodName, foodUnit, foodQuantity,
foodCategory, foodRegisteredTimestamp, foodExpireDate, sectionId);
//notify through interface to host activity for updating the ui
notifyInterfaceUtils.onAddFood();}
```

Структура проекту складається з 10 класів активностей та 8 фрагментів. На рисунку 3.1 приведена структура проекту.

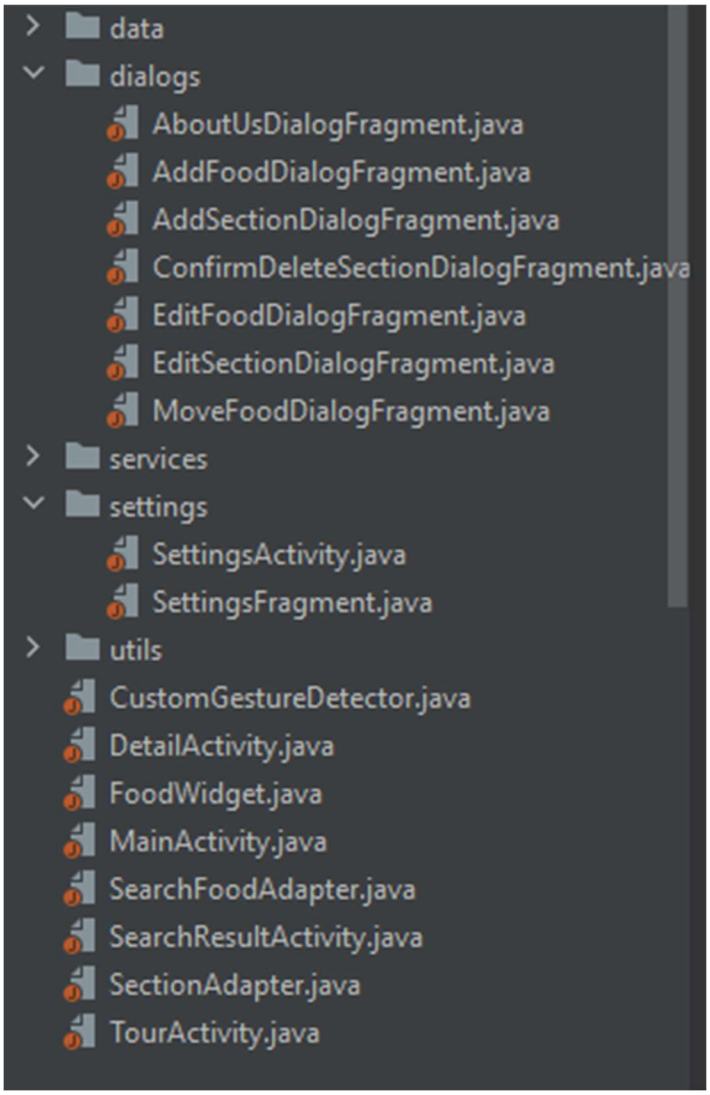


Рисунок 3.1 – Структура проекту

3.2 Реалізація розмітки мобільного додатка

Опис реалізації розмітки потрібно почати з головного екрану додатку, а саме MainActivity. Розмітка до головного екрану знаходиться у відповідному файлі розмітки activity_main. Тут знаходяться усі головні елементи з активності.

Вигляд екрану активності зображено на рисунку 3.2.

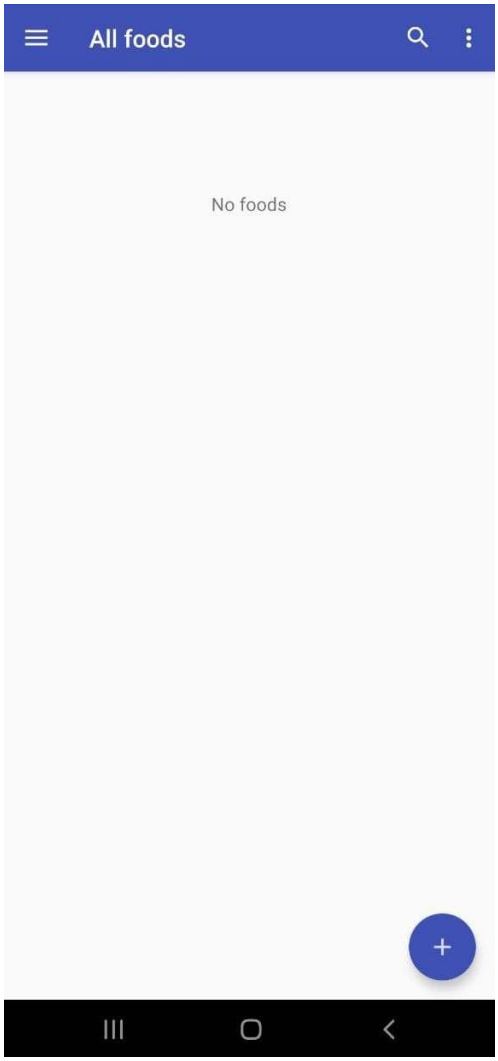


Рисунок 3.2 – Екран головної активності (MainActivity)

Код розмітки головної активності додатку:

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

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.apps.yecotec.fridgetracker.DetailActivity">
    <ImageView
        android:id="@+id/detail_food_asset"
        android:layout_width="100dp"
        android:layout_height="100dp"
        app:srcCompat="@drawable/ic_food_0"
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="32dp"
        app:layout_constraintHorizontal_bias="0.504" />
    <TextView
        android:id="@+id/detail_expiry_date_text"
    android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textStyle="bold"
        tools:text="Expiration Date: 02/10/2017"
        android:layout_marginTop="32dp"
       app:layout_constraintTop_toBottomOf="@+id/detail_food_asset"
        android:layout_marginLeft="24dp"
        app:layout_constraintLeft_toLeftOf="parent" />
    <TextView
        android:id="@+id/detail_section_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textStyle="bold"
        tools:text="Section: Section 1"
        android:layout_marginTop="8dp"
app:layout_constraintTop_toBottomOf="@+id/detail_expiry_date_text"
app:layout_constraintLeft_toLeftOf="@+id/detail_expiry_date_text" />
    <TextView
        android:id="@+id/detail_unit_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:textSize="14sp"
        android:textStyle="bold"
        tools:text="Unit: Piece"
  app:layout_constraintLeft_toLeftOf="@+id/detail_expiry_date_text"
        android:layout_marginTop="8dp"
app:layout_constraintTop_toBottomOf="@+id/detail_section_text" />
    <TextView
        android:id="@+id/detail_category_text"
        android:layout_width="wrap_content"
    android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textStyle="bold"
        tools:text="Category: Fruit"
        app:layout_constraintLeft_toLeftOf="@+id/detail_expiry_date_text"
        android:layout_marginTop="8dp"
        app:layout_constraintTop_toBottomOf="@+id/detail_unit_text" />
    <TextView
        android:id="@+id/detail_eat_all_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_corner_accent_color"
        android:text="@string/detail_eat_all_button_text"
        android:textSize="16dp"
        android:textColor="@android:color/white"
        android:textStyle="bold"
        android:paddingTop="6dp"
        android:paddingBottom="6dp"
        android:paddingStart="40dp"
        android:paddingEnd="40dp"
        android:elevation="2dp"
        android:layout_marginLeft="8dp"
    app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="16dp" />
    <TextView
        android:id="@+id/detail_quantity_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:textStyle="bold"
```

```
        tools:text="Quantity: 5"
        android:layout_marginRight="8dp"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginLeft="8dp"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginTop="8dp"
    app:layout_constraintTop_toBottomOf="@+id/detail_category_text"
        android:layout_marginBottom="8dp"
<android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />
app:layout_constraintBottom_toTopOf="@+id/detail_eat_all_button" />
    <ImageView
        android:id="@+id/detail_remove_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:background="@drawable/rounded_corner_accent_color"
        android:elevation="2dp"
        app:srcCompat="@drawable/ic_remove"
    app:layout_constraintRight_toLeftOf="@+id/detail_quantity_text"
        android:layout_marginRight="0dp"
        android:layout_marginLeft="0dp"
        app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="@+id/detail_quantity_text"
        android:layout_marginTop="0dp"
app:layout_constraintBottom_toBottomOf="@+id/detail_quantity_text"
        android:layout_marginBottom="0dp" />
    <ImageView
        android:id="@+id/detail_add_button"
        android:layout_width="40dp"
        android:layout_height="40dp"
        app:srcCompat="@drawable/ic_add"
        android:background="@drawable/rounded_corner_accent_color"
        android:elevation="2dp"
        android:layout_marginRight="0dp"
        app:layout_constraintRight_toRightOf="parent"
app:layout_constraintLeft_toRightOf="@+id/detail_quantity_text"
```

```
        android:layout_marginLeft="0dp"
app:layout_constraintBottom_toBottomOf="@+id/detail_quantity_text"
        android:layout_marginBottom="0dp"
       app:layout_constraintTop_toTopOf="@+id/detail_quantity_text"
        android:layout_marginTop="0dp" />
</android.support.constraint.ConstraintLayout>
```

Розмітка меню складається з трьох частин. Перша частина це activity_main_drawer яка тримає в собі усі переходи до інших сторінок, а саме: перехід до налаштувань, перехід на сторінку про додаток. Друга частина це menu_detail де знаходятся елементи про меню. Третя частина це menu_section де знаходяться елементи редагування, видалення та перенесення груп.

Вигляд екрану активності зображено на рисунку 3.3.

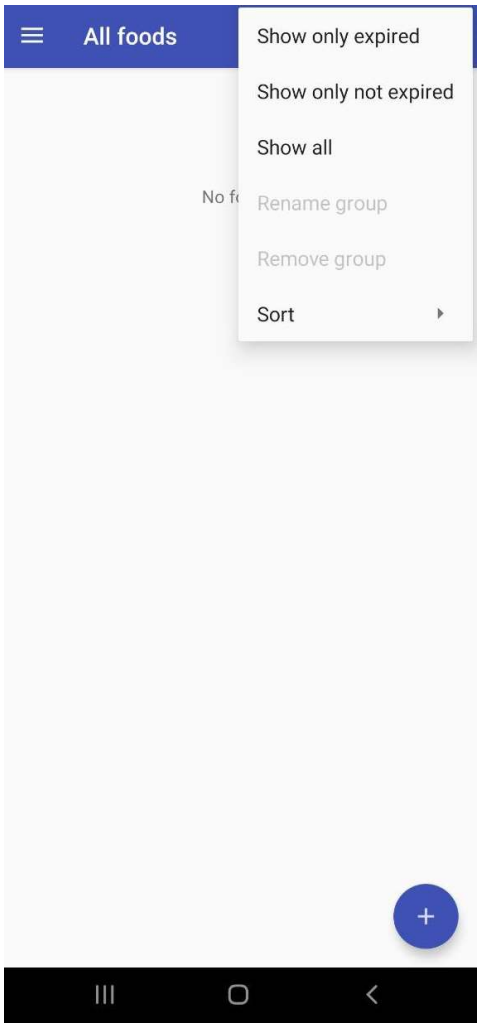


Рисунок 3.3 – Екран активності меню

Код розмітки activity_main_drawer поданий нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_create_samples"
            android:icon="@drawable/ic_samples"
            android:title="@string/navigation_drawer_create" />
        <item
            android:id="@+id/nav_go_to_settings"
            android:icon="@drawable/ic_settings"
            android:title="@string/navigation_drawer_settings" />
        <item
            android:id="@+id/nav_about"
            android:icon="@drawable/ic_about"
            android:title="@string/navigation_drawer_about" />
        <item
            android:id="@+id/menu_none"
            android:title=""
            android:visible="false"/>
    </group>
</menu>
```

Код розмітки menu_detail в якому знаходяться такі елементи як сортування та пошук продуктів:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:icon="@drawable/ic_sort"
        android:title="@string/sort_label"
        app:showAsAction="always"
        android:orderInCategory="0">
        <menu>
            <item
                android:id="@+id/action_sort_date"
            android:title="@string/sort_by_expiration_date_label"/>
```

```
            <item
                android:id="@+id/action_sort_ascending"
                android:title="@string/sort_by_ascending_label" />
            <item
                android:id="@+id/action_sort_descending"
                android:title="@string/sort_by_descending_label" />
            <item
               android:id="@+id/action_sort_remove"
               android:title="@string/sort_by_remove_sort_label" />
        </menu>
    </item>
    <item
        android:id="@+id/filter_collection"
        android:icon="@drawable/ic_filter"
        android:title="@string/filter_label"
        app:showAsAction="always"
        android:orderInCategory="0">
        <menu>
        </menu>
    </item>
    <item
        android:id="@+id/action_search"
        android:title="Action Search"
        android:icon="@drawable/ic_search"
        app:showAsAction="ifRoom|collapseActionView"
       app:actionViewClass="android.support.v7.widget.SearchView"/>
</menu>
```

Код розмітки menu_section для керування групам з функціями видалення, редагування та перенесення.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_edit_section"
        android:title="@string/menu_section_edit"
        android:orderInCategory="0"/>
    <item
        android:id="@+id/action_delete_section"
        android:title="@string/menu_section_delete"
```

```
        android:orderInCategory="1"/>
    <item
        android:id="@+id/action_move_food"
        android:title="@string/menu_section_move"
        android:orderInCategory="2"/>
</menu>
```

Вигляд екрану активності додавання продуктів зображено на рисунку 3.4

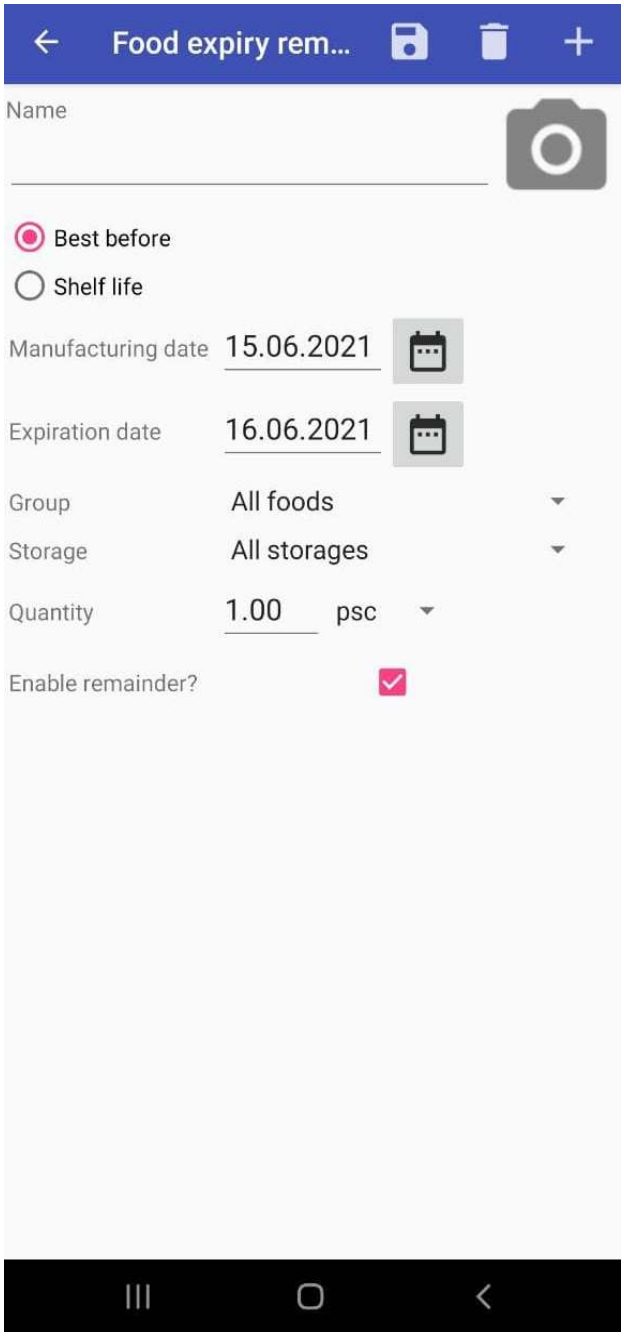


Рисунок 3.4 – Екран активності додавання продуктів

Код розмітки для додавання продуктів :

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:weightSum="1"
    android:paddingLeft="16dp"
    android:paddingRight="16dp">
    <EditText
        android:id="@+id/edit_text_food_name_dialog"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
android:hint="@string/add_food_dialog_food_name_edit_text_hint" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="8dp"
        android:weightSum="5">
        <EditText
            android:id="@+id/edit_text_food_quantity_dialog"
            android:inputType="phone"
            android:layout_width="0dp"
            android:layout_weight="3"
            android:layout_height="wrap_content"
android:hint="@string/add_food_dialog_quantity_edit_text_hint" />
        <Spinner
            android:id="@+id/spinner_unit"
            android:layout_width="0dp"
            android:layout_weight="2"
            android:layout_height="wrap_content"
            android:entries="@array/unit_array"
       android:prompt="@string/add_food_dialog_unit_prompt"/>
    </LinearLayout>
    <Spinner
```

```
        android:id="@+id/spinner_category"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:entries="@array/category_array"
        android:prompt="@string/add_food_dialog_category_prompt"
        android:layout_marginTop="16dp" />
    <Spinner
        android:id="@+id/spinner_section"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:entries="@array/category_array"
        android:prompt="@string/add_food_dialog_section_prompt"
        android:layout_marginTop="16dp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="16dp">
        <ImageView
            android:id="@+id/calendar_image_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:srcCompat="@drawable/ic_calendar"/>
        <TextView
            android:id="@+id/date_text_view"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
android:text="@string/add_food_dialog_expiration_date_label"
            android:gravity="center_vertical"
            android:layout_marginLeft="8dp"/>
    </LinearLayout>
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageView
            android:id="@+id/camera_image_view"
            android:layout_width="50dp"
            android:layout_height="50dp"
            app:srcCompat="@drawable/ic_menu_camera"
            android:scaleType="fitXY"
            android:layout_marginLeft="8dp"
            app:layout_constraintLeft_toLeftOf="parent"
```

```
            android:layout_marginRight="8dp"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            android:layout_marginBottom="16dp"
            app:layout_constraintTop_toTopOf="parent"
            android:layout_marginTop="16dp" />
    </android.support.constraint.ConstraintLayout>
    <ImageView
        android:id="@+id/captured_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal" />
</LinearLayout>
```

3.3 Керівництво користувача

Щоб почати використовувати додаток потрібно його завантажити та встановити на пристрій. Після цього потрібно запустити програму, натиснувши на її іконку, яка знаходиться на робочому столі або в меню програм. Після запуску додатку відкриється головна сторінка, на якій буде відображено усі додані продукти.

В правій нижній частині екрану знаходиться кнопка з символ плюс. Натиснувши на неї відкриється сторінка додавання продуктів. На цій сторінці знаходиться текстове поле, де потрібно ввести ім’я продукту який ви хочете додати, іконка фотографії, при натисканні на яку можна зробити фото продукту. Також на цій сторінці є поле з датою виготовлення товару та іконка календарю. Щоб встановити дату потрібно натиснути на календар та вибрати потрібну дату або в текстовому полі ввести дату вручну. Аналогічні дії потрібно проробити з наступним елементом, а саме датою закінчення терміну придатності продукту. Далі знаходиться список груп. Натиснувши на випадаючий список можна вибрати потрібну групу або створити свою. Для цього потрібно натиснути «додати нову групу» після чого відкриється вікно де потрібно ввести назву групи та натиснути клавішу «Ок». Далі знаходиться поле в якому зазначена кількість

одиниць продуктів. По замовченню виставлений 1 продукт, але кількість можна змінити вписавши в текстове поле потрібну кількість. Нижче знаходиться прапорець який відповідає за створення нагадування. Якщо прапорець встановлено, то ви отримаєте сповіщення коли термін дії продукту закінчиться. Зверху на панелі інструментів знаходяться іконки зі знаком плюс, корзина та збереження. При натисканні кнопки зі знаком збереження, ви збережете продукт та перейдете на головну сторінку, при натисканні кнопки зі знаком плюс ви збережете продукт, але залишитесь на сторінці додавання продуктів для подальшого додавання. При натисканні кнопки з іконкою корзини, ви видалите запис та повернетесь на головну сторінку.

На головній сторінці зверху знаходиться панель інструментів де є іконка лупи. При натисненні на неї, ви відкриєте поле пошуку. При введені в поле пошуку назви уже доданих товарів відкриється сторінка де будуть відображені товари які ви шукали. Також на панелі інструментів, яка знаходиться на головній сторінці є кнопка позначена трьома крапками – це меню редагування списку продуктів. Натиснувши на це меню, випаде список в якому будуть такі функції, як: сортування, показати усі продукти, показати лише прострочені продукти, показати лише не прострочені, перейменувати групу, видалити групу.

При натисненні на кнопку сортування, випаде список типів сортувань. Серед них є сортування за ім’ям, сортування за сховищем, сортування за датою виготовлення (новіші продукти), сортування за датою виготовлення (старіші продукти), сортування за датою закінчення терміну придатності (ближче до кінця), сортування за датою закінчення терміну придатності (далі від кінця). Натиснувши на один з видів сортування, усі продукти які ви додали відсортуються відповідним чином.

При натисненні на кнопку перейменувати групу відкриється вікно з полем для введення тексту, де можна ввести нову назву для групи. При натисненні на кнопку видалити групу, група буде видалена.

Щоб видалити або редагувати назву доданого продукту, потрібно натиснути на нього, утримуючи його близько секунди, після чого зверху буде кнопка для видалення та кнопка для редагування назви. Натиснувши на кнопку видалення – продукт буде видалений, натиснувши на кнопку редагування – відкриється поле введення тексту де потрібно буде ввести нову назву для продукту, яка буде збережена після введення.

3.4. Технічні характеристики мобільного додатку

В додатку використовуються прості рішення і немає складних обчислень та важких графічних навантажень, тому він не є вимогливим до технічних характеристик гаджета.

Мінімальна конфігурація:

− Android 5.1 та вище;

− любий одноядерний процесор;

− 1 ГБ ОЗП;

− 80 МБ вільної пам'яті.

Конфігурація, що рекомендується:

− Android 7.0;

− любий двоядерний процесор;

− 2 ГБ ОЗП;

− 140 МБ вільної пам'яті.

Розроблений додаток має весь необхідний функціонал, з робочими екранами, зручним інтерфейсом, який є простим, візуально привабливим та інтуїтивно зрозумілим у використанні.

У цьому розділі була виконана повна програмна реалізація додатку, був розроблений інтерфейс програми, який відповідає проектуванню з попереднього розділу, а також складено керівництво користувача та технічні вимоги додатку. Повий код програми наведений у додатку Б.

4 ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКА

4.1 Аналіз методів тестування мобільного додатка

Тестування мобільних додатків – це процес, який допомагає перевірити прикладне програмне забезпечення, яке було розроблене для мобільних пристроїв, на його зручність, функціональність та сумісність. Тестування може бути автоматизованим або мануальним.

Функціональне тестування є базовим тестом для любого додатка, яке перевіряє його відповідність до вимог. Подібно іншим додаткам, заснованим на користувацькому інтерфейсі, мобільні додати потребують переліку взаємодій людини в користувацьких сценаріях.

Тестування сумісності є найважливішим, коли діло доходить до тестування мобільних додатків. Ціль тесту на сумісність мобільного додатку, зазвичай, складається в тому, щоб головні функції додатку працювали правильним чином на конкретному приладі. Сама сумісність повинна займати всього декілька хвилин і може бути спланована завчасно. Вирішувати, які тести на сумісність мобільних приладів потрібно виконати не легка задача (оскільки тестування з усіма існуючими приладами просто неможлива). Тому потрібно підготувати тестову матрицю з кожною можливою комбінацією та розмістити пріоритети для клієнта.

Stress Testing (стрес-тестування) – це обов'язкове тестування, призначене для виявлення взаємоблокування, зависання і виключення, які можуть залишитися непоміченими під час тестування функціональності і інтерфейсу.

Ось список деяких критеріїв:

– завантажити у додаток найбільш можливу кількість даних, для того щоб перевірити його межу;

– багато разів виконати однакову операцію;

– виконати повторні операції швидко та повільно.

– залиште ваш додаток працюючим протягом тривалого періоду часу, одночасно працюючи з пристроєм і просто залишаючи його бездіяльним, або виконуючи деякі автоматичні завдання, яка займають багато часу, наприклад, слайд-шоу;

– випадково відправляти екранні натискання і натискання клавіш в вашому додатку;

– на вашому пристрої має бути запущено кілька додатків, щоб ви могли часто перемикатися між додатком і іншими додатками на пристрої.

Localization Testing (тестування локалізації). На цей момент багато додатків призначені для глобального використання, тому дуже важливо дбати про регіональні особливості, такі як мова та часові пояси. Важливо перевірити функціональність програми, коли користувач змінює мову. Необхідно враховувати, що інколи дизайн програми розроблений в східних країнах світу може не працювати з аудиторією зі західних країн або навпаки.

Performance Testing (тестування продуктивності) проводиться для визначення продуктивності клієнтських додатків, сервера і мережі. Завдяки Performance Testing можна ідентифікувати сервери, уже існуючі мережі та слабкі місця серверних додатків, враховуючи визначене навантаження і поєднання транзакцій.

Laboratory testing (лабораторне тестування), зазвичай проводяться мережевими операторами, виконуються шляхом моделювання усієї бездротової мережі. Цей тест виконується для виявлення будь-яких збоїв, наприклад коли мобільний додаток використовує передачу голосу та/або дані про виконання деяких функцій.

Security Testing допомагає виявити всі ймовірні уразливості щодо злому, управління сеансами, аутентифікації і авторизації, безпеки даних та інших стандартів безпеки. Додатки повинні шифрувати ім'я користувача та паролі при аутентифікації користувача по мережі.

Один зі способів тестування сценаріїв, пов'язаних з безпекою, полягає в маршрутизації даних вашого мобільного пристрою через проксі-сервер, такий, як OWASP Zed Attack Proxy, і пошуку уразливості.

Usability Testing (тестування зручності користування) це тест, який оцінює додаток на основі таких критеріїв як ефективність; задоволеність, повнота та точність. Дуже важливо провести юзабіліті-тестування з ранніх етапів розробки програми [18].

Для перевірки додатку доцільно використати 2 методи: модульний, конфігураційний. Саме ці типи тестування дозволять виконати повноцінну перевірку додатку на відповідність ТЗ, роботу окремих функцій та роботу додатку на різних пристроях.

4.2 Тестування додатка за допомогою емулятора

Зазвичай тестування програм починають з модульного тестування. Тому визначимо потенційні сценарії роботи з додатком та згрупуємо їх у таблиці 4.1

Таблиця 4.1 – Сценарії проведення модульних тестів

Ідентифікатор	Активність	Модуль	Вхідні дані	Очікуваний результат
1	2	3	4	5
М-М-1	MainActivity	Фрагмент	Натискання на пошук	Відкриття поля введення для пошуку
М-М-2	MainActivity	Меню	Вибір пункту меню сортування	Відкриття списку сортування
М-М-3	MainActivity	Фрагмент	Натискання на кнопку сортувати	Сортування продуктів

Кінець таблиці 4.1

1	2	3	4	5
M-M-4	MainActivity	Додання продукту	Натискання на кнопку додати	Відкриття активності додання продуктів
S-M-1	SearchActivity	Поле для введення тексту	Введення слова	Пошук слова зі списку продуктів
F-M-1	FoodActivity	Створення групи	Натиснення на кнопку створити групу	Відкриття фрагменту для створення групи
F-M-2	FoodActivity	Дата	Натиснення на кнопку календар	Відкриття вікна календаря
F-M-3	FoodActivity	Поле для введення тексту	Ввести кількість продуктів	Занесення тексту
F-M-4	FoodActivity	Фрагмент	Відкрити камеру	Відкриття фрагменту камера

По створеному сценарію були розроблені unit-тести. Нижче приведений фрагмент коду для тесту F-M-3:

```
public class ExampleUnitTest {
//testing food quantity testing
@Test
public void testCheckFoodQuantityActivity() {
 onView(withId(R.id. AmountTextView))
.perform(click())
.check(FoodActivity(isDisplayed()));
}    }
```

}

4.3 Аналіз результатів тестування мобільного додатка

Під час тестування додатка за допомогою unit-тестів та емулятора було знайдено і виправлено наступні баги:

– несправне відображення дати виготовлення продукту;

– створення дублікатів записів продуктів;

Після виправлення вищеперерахованих багів усі етапи модульного тестування були пройдені успішно та отримали наступні результати тестування, що занесені в таблицю 4.2.

Таблиця 4.2 – Результати модульного тестування

Ідентифікатор	Модуль	Вхідні значення	Отриманий результат	Результат
1	3	3	4	5
M-M-1	Фрагмент	Введення	Натискання на пошук	Правильно
M-M-2	Меню	Меню	Вибір пункту меню сортування	Правильно
M-M-3	Фрагмент	Натиснути кнопку сортувати	Список продуктів відсортовано	Правильно
M-M-4	Додання продукту	Натиск на кнопку додати	Відкриття вікна додавання продуктів	Правильно
S-M-1	Поле для введення тексту	Введення тексту «йогурт»	Виведення результатів за цим пошуком	Правильно

F-M-1	Створення групи	Натиснути створити групу	Відкриття вікна для створення групи	Правильно

Кінець таблиці 4.2

F-M-2	Дата	Встановити дату виготовлення 05.06.2021	Дата виготовлення встановлення як 05.06.2021	Правильно
F-M-3	Поле для введення тексту	Введення кількості: 3	Кількість встановлена: 3	Правильно
F-M-4	Фрагмент	Натискання на камеру	Відкриття камери	Правильно

Провівши аналізи результатів проведених тестів за допомогою вибраних методів тестування було виявлено, що функціонал додатку реалізовано відповідно до вимог складених у технічному завданні, і програма є повністю працездатною та оптимізованою до роботи на різних пристроях.

В цьому розділі описано процес тестування додатку, а саме – проведення модульного та конфігураційного тестування. Отримані результати було проаналізовано, та зроблено висновок про те, що розроблений додаток є повністю готовим, адже він може правильно виконувати усі впроваджені функції та є готовим до роботи на різних пристроях.

ВИСНОВКИ

В результаті виконаної під час дипломного проектування роботи можна зробити наступні висновки.

У першому розділі був проведений аналіз предметної області, її структурних і функціональних особливостей, також визначено специфіку та особливості розроблення програмного забезпечення для обліку термінів придатності продуктів харчування. В цьому розділі був проведений детальний аналіз та огляд існуючих видів мобільних додатків, операційних систем для мобільних додатків. Були переглянуті та проаналізовані популярні мобільні додатки на тему контролю термінів придатності продуктів харчування. Після чого було проведено аналіз вимог до розроблюваного застосунку, на основі якого було складено технічне завдання.

Як результат було вирішено розробити мобільний додаток, який задовольнить потреби користувачів і допоможе їм проводити контроль термінів придатності продуктів.

Другий розділ дипломного проекту присвячений проектуванню мобільного додатку для обліку термінів придатності. В ньому було проведено аналіз технологій та інструментів, які використовуються для створення Android застосунків. По результатам проведеного аналізу був спроектований зручний та сучасний інтерфейс, який відповідає усім функціональним особливостям та розроблена архітектура додатка.

У третьому розділі на основі архітектури та користувацького інтерфейсу була виконана програмна реалізація додатка. Також було створено керівництво користувача та визначені технічні характеристики мобільного додатку.

Після цього у четвертому розділі було проведено модульне та функціональне тестування готової програми, завдяки чому було вияснено, що усі функції, які були визначені в технічному завданні, працюють належним чином та додаток повністю готовий до використання.

При виконанні дипломного проекту було розроблено додаток для мобільних пристроїв на платформі Android, який дає змогу зручно вести облік термінів придатності продуктів харчування, завдяки чому було закріплено теоретичні та практичні знання і навички розробки життєвого циклу ПЗ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Л. П. Бедратюк. Дипломний проект : методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун, О. М. Яшина. – Хмельницький: ХНУ, 2020. – 77 с.

2. Что такое нативное мобильное приложение? Отличие натива от гибрида [Електронний ресурс] / wiserv.ru // Оксана Крючек. – Режим доступу до ресурсу: https://wiserv.ru/blog/mobile-app/about-mobile-applications

3. Что такое нативные и кроссплатформенные приложения? Плюсы и минусы [Електронний ресурс] / itvdn.com // Армен Маилян. – Режим доступу до ресурсу https://itvdn.com/ru/blog/article/native-cross-platform

4. В. А. Шеховцов. Операційні системи. – К.: Видавнича група BHV, 2005. – 576с.

5. Учет Продуктов, Сроки Годности, Список Покупок [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.chestersw.foodlist

6. Учет товаров-простой склад 2.0 [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.stockmanagment.next.app

7. FoodKeeper [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=gov.usda.fsis.foodkeeper2

8. Срок годности продуктов питания! [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.nevrgivapp.nomorewastes

9. Expiring Things [Електронний ресурс] / Play Market. – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.bfp.apps_expiring_things

10. Systems and software engineering — Architecture description: ISO/IEC/IEEE 42010

11. GUI Architectures [Електронний ресурс] / Martin Fowler. – Режим доступу до ресурсу: https://www.martinfowler.com/eaaDev/uiArchs.html

12. Potel M. MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java / Mike Potel., 1996.

13. Let's make an MVVM at Android [Електронний ресурс] / Stfalcon – Режим доступу до ресурсу: https://stfalcon.com/en/blog/post/android-mvvm

14. The Clean Architecture [Електронний ресурс] / The Clean Code Blog – Режим доступу до ресурсу: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

15. Что такое UX/UI-дизайн и как попасть в эти профессии [Електронний ресурс] / Skillbox – Режим доступу до ресурсу: https://skillbox.ru/media/design/ux_ui_dizayn_chto_eto_takoe/

16. Cross-platform mobile development in Visual Studio [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/cross-platform/cross-platform-mobile-development-in-visual-studio?view=vs-2019

17. Офіційна сторінка Android Studio [Електронний ресурс] / Developers. – Режим доступу до ресурсу: https://developer.android.com/studio

18. Тестування мобільних додатків [Електронний ресурс] / QALight – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/testuvannya-mobilnih-dodatkiv/

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки мобільного додатку для обліку термінів придатності продуктів харчування

Умовне позначення системи: мобільний додаток «FoodExpiry»

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки програмного забезпечення: Мобільний додаток для обліку термінів придатності продуктів харчування**.**

2 Призначення розробки

Додаток призначений для використання на смартфонах з операційною системою Android з метою надання зручного інструменту для ведення обліку та контролю термінів придатності продуктів.

Користувачами ПЗ є користувачі телефонів на платформі Android.

Програмне забезпечення може використовуватися на будь-якому пристрої з операційною системою Android версії вище 5.1, попередньо встановивши додаток, додаткових налаштувань для початку роботи застосунок не потребує.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Додаток повинен забезпечувати наступні функції:

- додання нових продуктів;
- можливість зробити фото продукту;
- створення груп для різних видів продуктів;
- перегляд усього списку продуктів;
- перегляд різних груп з продуктами;
- сортування продуктів в загальному списку та групах;
- отримання повідомлень про кінець терміну придатності;
- налаштування повідомлень;
- редагування та видалення продуктів;
- редагування та видалення груп;

– реалізація меню з групами та налаштуваннями.

3.2 Вимоги до надійності

Надійність системи повинна бути забезпечена за такими напрямками:

– забезпечення працездатності компонентів програмно-технічної платформи;

– збереження даних.

– виконання команд;

– виведення результату на екран.

3.3 Умови експлуатації

Для введення в експлуатацію додатка потрібно: здійснити інсталяцію програмного забезпечення, провести тестування готового продукту за різними критеріями користувачами.

При наявності зауважень або недопрацювань, складається документ «Виправлення і зауваження», де вказуються недоліки програмного забезпечення. При задовільному результаті тестування програмне забезпечення буде випущено у реліз, та опубліковано на маркеті.

3.4 Вимоги до складу параметрів технічних засобів

Додаток повинен коректно працювати на версії Android 5.1 та вище.

Мінімальна конфігурація:

− Android 5.1 та вище;

− любий одноядерний процесор;

− 1 ГБ ОЗП;

− 80 МБ вільної пам'яті.

Конфігурація, що рекомендується:

− Android 7.0;

− любий двоядерний процесор;

− 2 ГБ ОЗП;

− 140 МБ вільної пам'яті.

3.5 Вимоги до інформаційної та програмної сумісності

Програма повинна працювати під управлінням операційної системи андроїд версії вище 5.1, а також похідних від неї системах від виробників смартфонів.

4 Вимоги до програмної документації

Комплект супроводжувальної документації включає:

- код програми з коментарями та поясненнями;
- відомості про алгоритми та функціонування програми;
- технічне завдання;
- інструкція користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки проекту «мобільний додаток для обліку термінів придатності продуктів харчування» показано в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
Технічне завдання 02.01.21 – 31.01.21	Обґрунтування необхідності розробки програми	Коротка характеристика мобільного додатку; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.21 – 14.02.21	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; вибір середовища програмування; розробка і опис загальної алгоритмічної структури системи, що буде розроблюватися

Кінець таблиці А.1

Технічний проект 15.02.21 – 28.02.21	Розробка	Уточнення структури вхідних і вихідних даних; розробка інтерфейсу; розробка

	технічного проекту	докладного алгоритму; розробка структури програми; остаточне визначення конфігурацій технічних засобів
Робочий проект 01.03.21 – 10.04.21	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 11.04.21 – 20.04.21	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 21.04.21 – 30.04.21	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і випуск додатку на платформі Play Маркет

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування програмного забезпечення. Прийом мобільного додатку здійснюється після налаштування програмного забезпечення та окремих його компонентів для повноцінного функціонування.

ДОДАТОК Б

(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Клас MainActivity

```
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.app.SearchManager;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Bundle;
import android.os.SystemClock;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v4.app.DialogFragment;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.MenuItemCompat;
import android.support.v7.preference.PreferenceManager;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.SearchView;
import android.util.Log;
import android.view.MenuInflater;
import android.view.SubMenu;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdView;

import java.util.Calendar;
import java.util.Date;

import static android.app.SearchManager.QUERY;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener,
        View.OnClickListener,
        NotifyInterfaceUtils,
        EditSectionDialogFragment.MainActivityNotifyInterface,
        ConfirmDeleteSectionDialogFragment.MainActivityNotifyInterface,
        MoveFoodDialogFragment.MainActivityNotifyInterface {

    //for logging purpose
    String TAG = this.getClass().getSimpleName();
    Snackbar mySnackbar;

    //unique ids for creating itemMenu
    private static final int MENU_ITEM_ID = 208;
    private static final int MENU_ITEM_ID_REMOVE = 205;
    private static final int MENU_ITEM_GROUP_ID = 109;
```

```
    public static final String DELETED_FOOD = "deletedFood";

    private Boolean isMainFabOpen = false;
    private FloatingActionButton mainFab, addFoodFab,addSectionFab;
    private Animation appear,dissolve, appear_text, dissolve_text,
rotate_forward,rotate_backward;
    private TextView mainFabLabel, addFoodFabLabel, addSectionFabLabel;
    private View clickableScreen;

    private int sortOption = 0;//track the state of selected option
    private int sectionFilterId = 0;//track the state of selected sectionId
    private boolean filterOptionActivated = false;//flag that track state of filter

    private Window window;

    private NavigationView navigationView;

    private SubMenu filterSectionMenuOption;

    //member variables for recycler view
    SectionAdapter sectionAdapter;
    RecyclerView recyclerView;

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

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        window = this.getWindow();

        //find fabs and set up listeners
        mainFab = (FloatingActionButton) findViewById(R.id.main_fab);
        addFoodFab = (FloatingActionButton) findViewById(R.id.add_food_fab);
        addSectionFab = (FloatingActionButton) findViewById(R.id.add_section_fab);
        mainFab.setOnClickListener(this);
        addFoodFab.setOnClickListener(this);
        addSectionFab.setOnClickListener(this);

        //find labels and set up listeners
        mainFabLabel = (TextView) findViewById(R.id.label_main_fab);
        addFoodFabLabel = (TextView) findViewById(R.id.label_add_food_fab);
        addSectionFabLabel = (TextView) findViewById(R.id.label_add_section_fab);
        mainFabLabel.setOnClickListener(this);
        addFoodFabLabel.setOnClickListener(this);
        addSectionFab.setOnClickListener(this);

        //find clickable screen and set up listener
        clickableScreen = findViewById(R.id.clickable_screen);
        clickableScreen.setOnClickListener(this);

        //bind animation from resources
        //animation for FABs
        appear = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.appear);
        dissolve =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.dissolve);

        //animation for text labels
        appear_text = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.appear_text);
```

```
        dissolve_text =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.dissolve_text);

        //animation for main fab
        rotate_forward =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.rotate_forward);
        rotate_backward =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.rotate_backward);

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        mySnackbar = Snackbar.make(findViewById(R.id.my_coordinator_layout), "",
Snackbar.LENGTH_LONG);

        navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        //to store a reference to the RecyclerView in mNumbersList
        recyclerView = (RecyclerView) findViewById(R.id.rv_section);

        LinearLayoutManager layoutManager = new LinearLayoutManager(this);

        recyclerView.setLayoutManager(layoutManager);

        //to designate that the contents of the RecyclerView won't change an item's
size
        recyclerView.setHasFixedSize(true);

        sortOption = 0;//by default has no sort

        sectionAdapter = new SectionAdapter(new DBUtils(this).getAllSectionsArray(),
this, sortOption, this, getSupportFragmentManager());

        recyclerView.setAdapter(sectionAdapter);

        updateWidget();

        checkIfSetAlarm();

        //set up banner
        AdView adView = (AdView) findViewById(R.id.adView);

        AdRequest adRequest = new AdRequest.Builder().build();

        adView.loadAd(adRequest);

        Intent intent = getIntent();

        //set snackBar if user deleted item
        if(intent.hasExtra(DELETED_FOOD)) {
            final Food deletedFood = intent.getParcelableExtra(DELETED_FOOD);

            final NotifyInterfaceUtils notifyInterfaceUtils = this;
            final Context context = this;


mySnackbar.setText(this.getResources().getString(R.string.snack_bar_deleted)+
deletedFood.foodName);
```

```
            //insert the food again

mySnackbar.setAction(this.getResources().getString(R.string.snack_bar_undo), new
View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    new DBUtils(context,
notifyInterfaceUtils).insertFoodIntoDB(deletedFood);
                }
            }).show();
        }
    }

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

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        updateUi();

        if (requestCode == REQUEST_FOR_ACTIVITY_CODE) {
            if(resultCode == Activity.RESULT_OK){//user delete the data
                final Food deletedFood = data.getParcelableExtra(DELETED_FOOD);

                final NotifyInterfaceUtils notifyInterfaceUtils = this;
                final Context context = this;


mySnackbar.setText(this.getResources().getString(R.string.snack_bar_deleted) +
deletedFood.foodName);

                //insert the food again
mySnackbar.setAction(this.getResources().getString(R.string.snack_bar_undo), new
View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        new DBUtils(context,
notifyInterfaceUtils).insertFoodIntoDB(deletedFood);
                    }
                }).show();

            }
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
```

```
        // Get the SearchView and set the searchable configuration
        SearchManager searchManager = (SearchManager)
getSystemService(Context.SEARCH_SERVICE);
        MenuItem searchItem = menu.findItem(R.id.action_search);
        SearchView searchView = (SearchView)
MenuItemCompat.getActionView(searchItem);

        final Context context = this;

        searchView.setOnQueryTextListener(
                new SearchView.OnQueryTextListener() {

                    @Override
                    public boolean onQueryTextSubmit(String query) {
                        Intent intent = new Intent(context,
SearchResultActivity.class);
                        intent.putExtra(QUERY, query);//pass query string
                        context.startActivity(intent);
                        return false;
                    }

                    @Override
                    public boolean onQueryTextChange(String newText) {
                        return false;
                    }
                }
        );

        MenuItem filterMenuItem = menu.getItem(1);//get filter_collection menuItem

        //Menu.NONE -> Value to use for group and item identifier integers when you
don't care about them.

        //get subMenu for adding itemMenus dynamically
        filterSectionMenuOption = filterMenuItem.getSubMenu();

        updateFilterSectionOptions();

        return true;
    }

    public void updateFilterSectionOptions() {
        //clear content
        filterSectionMenuOption.clear();

        //get sections array from db
        Section[] sections = new DBUtils(this).getAllSectionsArray();

        //iterate for adding itemMenu
        for (int i = 0; i < sections.length; i++) {
            filterSectionMenuOption.add(MENU_ITEM_GROUP_ID /*group id*/, MENU_ITEM_ID
+ i/*itemId*/, Menu.NONE /*order*/, sections[i].sectionName /*text*/);
        }

        filterSectionMenuOption.add(MENU_ITEM_GROUP_ID, MENU_ITEM_ID_REMOVE,
Menu.NONE, R.string.dynamic_menu_remove_filter);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
```

```
        int id = item.getItemId();

        switch (id) {

            case R.id.action_sort_remove:
                sortOption = 0;
                updateUi(sortOption, sectionFilterId, filterOptionActivated);
                return true;

            case R.id.action_sort_date:
                sortOption = 1;
                updateUi(sortOption, sectionFilterId, filterOptionActivated);
                return true;

            case R.id.action_sort_ascending:
                sortOption = 2;
                updateUi(sortOption, sectionFilterId, filterOptionActivated);
                return true;

            case R.id.action_sort_descending:
                sortOption = 3;
                updateUi(sortOption, sectionFilterId, filterOptionActivated);
                return true;

            case MENU_ITEM_ID_REMOVE:
                filterOptionActivated = false;
                updateUi(sortOption, sectionFilterId, filterOptionActivated);
                return true;
        }

        //get sections array from db
        Section[] sections = new DBUtils(this).getAllSectionsArray();

        for (int i = 0; i < sections.length; i++) {
            if(id == (MENU_ITEM_ID + i)) {
                filterOptionActivated = true;
                sectionFilterId = sections[i].sectionId;
                //update ui with filtered section array
                updateUi(sortOption, sectionFilterId, filterOptionActivated);
                return true;
            }
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if(id == R.id.nav_create_samples) {
            //instantiate the class with notifyInterfaceUtils for updating the ui
from DBUtils class
            new DBUtils(this, this).insertSampleDataSet();
            //update menu options
            updateFilterSectionOptions();
        } else if (id == R.id.nav_go_to_settings) {
            Intent intent = new Intent(this, SettingsActivity.class);
            startActivity(intent);
        } else if(id == R.id.nav_about) {
            DialogFragment newFragment = new AboutUsDialogFragment();
            newFragment.show(getSupportFragmentManager(), "about_us");
```

```
        } else if(id == R.id.nav_visit_our_page) {
            Uri webPageUri = Uri.parse(urlAsString);

            Intent intent = new Intent(Intent.ACTION_VIEW, webPageUri);

            //Verify that this Intent can be launched and then call startActivity
            if (intent.resolveActivity(getPackageManager()) != null) {
                startActivity(intent);
            }
        } else if(id == R.id.nav_tutorial) {
            Intent intent = new Intent(this, TourActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK |Intent.FLAG_ACTIVITY_CLEAR_TOP);

            //set false for visiting tour activity
            SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(this);
            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.putBoolean(TOUR_ACTIVITY_VISITED_KEY, false);
            editor.commit();

            startActivity(intent);
        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    @Override
    public void onClick(View view) {
        int id = view.getId();
        switch (id){

            case R.id.main_fab:
                animateMainFAB();
                break;

            case R.id.add_food_fab:
                if(new DBUtils(this).thereAreRecords()) {
                    DialogFragment addFoodDialogFragment = new
AddFoodDialogFragment();
                    addFoodDialogFragment.show(getSupportFragmentManager(),
"add_food");
                } else {
                    Toast.makeText(this, R.string.toast_warning_add_food,
Toast.LENGTH_SHORT).show();
                }

                animateMainFAB();
                break;

            case R.id.add_section_fab:
                DialogFragment addSectionDialogFragment = new
AddSectionDialogFragment();
                addSectionDialogFragment.show(getSupportFragmentManager(),
"add_section");

                animateMainFAB();
                break;

            case R.id.label_main_fab:
                animateMainFAB();//cancel action with this label
                break;
```

```
            case R.id.label_add_food_fab:
                break;

            case R.id.label_add_section_fab:
                break;

            case R.id.clickable_screen:
                animateMainFAB();//cancellation action when touching the screen
                break;

        }
    }

    //here update the ui
    @Override
    public void onAddFood() {
        //update lists with selected sort and filter option
        updateUi(sortOption, sectionFilterId, filterOptionActivated);
    }

    @Override
    public void onAddSection() {
        //update menu options
        updateFilterSectionOptions();
        //update lists with selected sort and filter option
        updateUi(sortOption, sectionFilterId, filterOptionActivated);
    }

    @Override
    public void updateUi() {
        updateUi(sortOption, sectionFilterId, filterOptionActivated);
    }

    @Override
    public void deleteFood(Food deletedFood) {

        final NotifyInterfaceUtils notifyInterfaceUtils = this;
        final Context context = this;
        final Food finalDeletedFood = deletedFood;

        mySnackbar.setText(this.getResources().getString(R.string.snack_bar_deleted)
+ finalDeletedFood.foodName);

        //insert the food again
        mySnackbar.setAction(this.getResources().getString(R.string.snack_bar_undo),
new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                new DBUtils(context,
notifyInterfaceUtils).insertFoodIntoDB(finalDeletedFood);
            }
        }).show();
    }

    //dialog callbacks

    //at editing section
    @Override
    public void onEditFinished() {
        updateFilterSectionOptions();
        updateUi(sortOption, sectionFilterId, filterOptionActivated);
    }
```

```
    @Override
    public void onConfirmedDeletion(int sectionId) {
        new DBUtils(this, this).deleteAllFoodsInsideSection(sectionId);
        new DBUtils(this, this).deleteSectionById(sectionId);

        //disable filter by section at deleting any section
        filterOptionActivated = false;
        updateUi(sortOption, sectionFilterId, filterOptionActivated);

        updateFilterSectionOptions();//update section filter options
    }

    @Override
    public void onMoved() {
        updateUi();
    }

    private void updateUi(int sortOption, int sectionFilterId, boolean
filterOptionActivated) {

        Section[] sections = new DBUtils(this).getAllSectionsArray();

        sectionAdapter = new SectionAdapter(sections, this, sortOption, this,
getSupportFragmentManager());

        Section[] filteredSections = null;

        if(filterOptionActivated) {
            for (int i = 0; i < sections.length; i++) {
                if(sectionFilterId == sections[i].sectionId) {
                    filteredSections =
SectionUtils.filterSectionById(sections[i].sectionId, this);
                    i = sections.length;
                }
            }

            //update the adapter and uses this
            sectionAdapter = new SectionAdapter(filteredSections, this, sortOption,
this, getSupportFragmentManager());
        }

        recyclerView.setAdapter(sectionAdapter);
        updateWidget();
    }

    private void animateMainFAB(){

        if(isMainFabOpen){

            //fabs
            mainFab.startAnimation(rotate_backward);
            addFoodFab.startAnimation(dissolve);
            addSectionFab.startAnimation(dissolve);

            //textView
            mainFabLabel.startAnimation(dissolve_text);
            addFoodFabLabel.startAnimation(dissolve_text);
            addSectionFabLabel.startAnimation(dissolve_text);

            clickableScreen.setVisibility(View.INVISIBLE);
            window.setStatusBarColor(ContextCompat.getColor(this,
R.color.customPrimaryDark));

            addFoodFab.setClickable(false);
```

```
            addSectionFab.setClickable(false);
            mainFabLabel.setClickable(false);
            addFoodFabLabel.setClickable(false);
            addSectionFabLabel.setClickable(false);
            clickableScreen.setClickable(false);

            isMainFabOpen = false;

        } else {

            mainFab.startAnimation(rotate_forward);

            //fabs
            addFoodFab.startAnimation(appear);
            addSectionFab.startAnimation(appear);

            //textView
            mainFabLabel.startAnimation(appear_text);
            addFoodFabLabel.startAnimation(appear_text);
            addSectionFabLabel.startAnimation(appear_text);

            clickableScreen.setVisibility(View.VISIBLE);
            window.setStatusBarColor(ContextCompat.getColor(this,
R.color.mainFabOpened));

            addFoodFab.setClickable(true);
            addSectionFab.setClickable(true);
            mainFabLabel.setClickable(true);
            addFoodFabLabel.setClickable(true);
            addSectionFabLabel.setClickable(true);
            clickableScreen.setClickable(true);

            isMainFabOpen = true;

        }
    }

    public void checkIfSetAlarm() {

        SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(this);
        Boolean notificationDeliveredToday =
sharedPreferences.getBoolean(NOTIFICATION_DELIVERED_TODAY, false);//false for default

        if(notificationDeliveredToday) {
            //Log.d(TAG, "Notification already delivered today");
            int deliveredDay = sharedPreferences.getInt(NOTIFICATION_DELIVERED_DAY,
32);
            //Log.d(TAG, "Delivered Day: " + deliveredDay);

            Calendar cal = Calendar.getInstance();
            cal.setTime(new Date());
            int day = cal.get(Calendar.DAY_OF_MONTH);

            //Log.d(TAG, "Today: " + day);
            if(deliveredDay != day) {
                //Log.d(TAG, "It's another day, let's schedule an alarm again");
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putBoolean(NOTIFICATION_DELIVERED_TODAY, true);
                editor.commit();

                setAlarm();
            }
```

```
        } else {
            //Log.d(TAG, "Notification not delivered yet today, let's schedule an
alarm");
            setAlarm();
        }

    }

    public void setAlarm() {
        //schedule an alarm at 8:00 a.m. and every day therefore
        Intent serviceIntent = new Intent(this, NotificationService.class);
        serviceIntent.setAction(NotificationService.ACTION_CREATE_NOTIFICATION);

        //wrap intent inside the pendingIntent
        PendingIntent pendingIntent = PendingIntent.getService(this, 0,
serviceIntent, 0);

        AlarmManager alarmManager = (AlarmManager)
this.getSystemService(Context.ALARM_SERVICE);

        // Set the alarm to start at approximately 8:00 a.m.
        Calendar calendar = Calendar.getInstance();
        calendar.setTimeInMillis(System.currentTimeMillis());
        calendar.set(Calendar.HOUR_OF_DAY, 8);

        alarmManager.setInexactRepeating(
                AlarmManager.RTC_WAKEUP,
                calendar.getTimeInMillis(),//set alarm at 8:00 p.m.
                AlarmManager.INTERVAL_DAY,//you must use AlarmManager interval
constants
                pendingIntent);
    }

    public void updateWidget() {
        WidgetService.startActionUpdateWidgets(this);
    }
}
```

Клас SearchFoodAdapter

```
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffColorFilter;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.TextView;

public class SearchFoodAdapter extends
RecyclerView.Adapter<SearchFoodAdapter.SearchFoodAdapterViewHolder> {

    Food[] foods;
    Context context;

    public SearchFoodAdapter(Food[] foods, Context context) {
        this.foods = foods;
        this.context = context;
    }
```

```
    @Override
    public SearchFoodAdapterViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        Context context = parent.getContext();
        int layoutIdForListItem = R.layout.food_list_item;
        LayoutInflater inflater = LayoutInflater.from(context);
        boolean shouldAttachToParentImmediately = false;
        View view = inflater.inflate(layoutIdForListItem, parent,
shouldAttachToParentImmediately);

        SearchFoodAdapterViewHolder viewHolder = new
SearchFoodAdapterViewHolder(view);

        return viewHolder;
    }

    @Override
    public void onBindViewHolder(SearchFoodAdapterViewHolder holder, int position) {
        holder.bind(position);
    }

    @Override
    public int getItemCount() {
        return foods.length;
    }

    public class SearchFoodAdapterViewHolder extends RecyclerView.ViewHolder {

        View itemView;
        ImageView circleBackGround;
        ImageView foodImageView;
        TextView foodNameTextView;
        TextView foodQuantityTextView;
        ImageView calendarImageView;
        TextView expirationDateTextView;

        public SearchFoodAdapterViewHolder(View itemView) {
            super(itemView);
            this.itemView = itemView;
            circleBackGround = (ImageView)
itemView.findViewById(R.id.image_view_circle_back_ground);
            foodImageView = (ImageView) itemView.findViewById(R.id.image_view_food);
            foodNameTextView = (TextView)
itemView.findViewById(R.id.text_view_food_name);
            foodQuantityTextView = (TextView)
itemView.findViewById(R.id.text_view_food_quantity);
            calendarImageView = (ImageView)
itemView.findViewById(R.id.image_view_calendar);
            expirationDateTextView = (TextView)
itemView.findViewById(R.id.text_view_expiration_date);
        }

        void bind(int position) {
            //set up attributes
            Drawable circleDrawable =
context.getResources().getDrawable(R.drawable.circle_color_1, null);
            circleDrawable.setColorFilter(new PorterDuffColorFilter(new
DBUtils(context).getSectionById(foods[position].sectionId).sectionColor,
PorterDuff.Mode.SRC_IN));
            circleBackGround.setBackground(circleDrawable);


foodImageView.setImageResource(FoodUtils.getFoodAsset(foods[position].foodCategory));
```

```
            foodNameTextView.setText(foods[position].foodName);
            foodQuantityTextView.setText(foods[position].foodQuantity + " " + new
FoodUtils(context).getUnitString(foods[position].foodUnit));
            if(foods[position].foodUnit == 0) foodQuantityTextView.setText((int)
foods[position].foodQuantity + " " + new
FoodUtils(context).getUnitString(foods[position].foodUnit));

            Drawable calendarDrawable =
context.getResources().getDrawable(R.drawable.ic_calendar, null).mutate();
            calendarDrawable.setColorFilter(new PorterDuffColorFilter(new
DBUtils(context).getSectionById(foods[position].sectionId).sectionColor,
PorterDuff.Mode.SRC_IN));
            calendarImageView.setImageDrawable(calendarDrawable);

            expirationDateTextView.setText(foods[position].foodExpireDate);

            final int finalPosition = position;

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Intent intent = new Intent(context, DetailActivity.class);
                    intent.putExtra(FOOD_ID, foods[finalPosition].foodId);
                    ((Activity) context).startActivityForResult(intent,
REQUEST_FOR_ACTIVITY_CODE);
                }
            });
        }
    }
}
```

Клас SearchResultActivity

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.Menu;

import static android.app.SearchManager.QUERY;

public class SearchResultActivity extends AppCompatActivity {

    SearchFoodAdapter searchFoodAdapter;
    RecyclerView recyclerView;
    String queryString;

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

        getSupportActionBar().setTitle(R.string.search_activity_text);

        recyclerView = (RecyclerView) findViewById(R.id.rv_search_result);

        LinearLayoutManager layoutManager = new LinearLayoutManager(this);

        recyclerView.setLayoutManager(layoutManager);

        //to designate that the contents of the RecyclerView won't change an item's
size
```

```
        recyclerView.setHasFixedSize(true);

        Intent intent = getIntent();

        //populate ui
        if(intent.hasExtra(QUERY)) {
            queryString = intent.getStringExtra(QUERY);
            populateUi(queryString);
        }

    }

    private void populateUi(String queryString) {
        searchFoodAdapter = new SearchFoodAdapter(new
DBUtils(this).queryFoods(queryString),this);
        recyclerView.setAdapter(searchFoodAdapter);
    }

}
```

Клас DetailActivity

```
import android.app.Activity;
import android.content.Intent;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

public class DetailActivity extends AppCompatActivity implements
View.OnClickListener, EditFoodDialogFragment.DetailActivityNotifyInterface{

    //for logging purpose
    String TAG = this.getClass().getSimpleName();

    private ImageView foodAsset, removeButton, addButton;
    private TextView expirationDateText, sectionText, unitText, categoryText,
quantityText, eatAllButton;

    private int foodId = 0;
    private Food food;
    private boolean exception = false;

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

        foodAsset = (ImageView) findViewById(R.id.detail_food_asset);
        removeButton = (ImageView) findViewById(R.id.detail_remove_button);
        addButton = (ImageView) findViewById(R.id.detail_add_button);
        expirationDateText = (TextView) findViewById(R.id.detail_expiry_date_text);
        sectionText = (TextView) findViewById(R.id.detail_section_text);
        unitText = (TextView) findViewById(R.id.detail_unit_text);
        categoryText = (TextView) findViewById(R.id.detail_category_text);
```

```
        quantityText = (TextView) findViewById(R.id.detail_quantity_text);
        eatAllButton = (TextView) findViewById(R.id.detail_eat_all_button);

        removeButton.setOnClickListener(this);
        addButton.setOnClickListener(this);

        Intent intent = getIntent();

        //populate ui
        if(intent.hasExtra(FOOD_ID)) {
            foodId = intent.getIntExtra(FOOD_ID, 0);

            populateUi(foodId);
        }

        if(intent.hasExtra(EXCEPTION_EXTRA)) exception =
intent.getBooleanExtra(EXCEPTION_EXTRA, false);
    }

    //handle each click event
    @Override
    public void onClick(View view) {
        int id = view.getId();

        switch (id) {

            case R.id.detail_eat_all_button:
                new DBUtils(this).deleteFoodById(foodId);

                if (exception) {//handle case if user navigate to this activity from
widget or notification
                    Intent intent = new Intent(this, MainActivity.class);
                    intent.putExtra(DELETED_FOOD, food);
                    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK |Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(intent);
                }
                else {
                    Intent intent = new Intent();
                    intent.putExtra(DELETED_FOOD, food);
                    setResult(Activity.RESULT_OK,intent);
                    finish();
                }

                break;

            case R.id.detail_remove_button:

                if(food.foodUnit == 0) {
                    if(food.foodQuantity > 0) food.foodQuantity -= 1;
                    else Toast.makeText(this, R.string.warning_negative_case,
Toast.LENGTH_SHORT).show();
                } else {
                    if(food.foodQuantity > 0) food.foodQuantity -= 0.1;
                    else Toast.makeText(this, R.string.warning_negative_case,
Toast.LENGTH_SHORT).show();
                    food.foodQuantity = Math.round( food.foodQuantity * 10.0 ) /
10.0;//avoid floating point arithmetic problem
                }

                new DBUtils(this).updateFood(food);
                populateUi(food.foodId);

                break;
```

```
            case R.id.detail_add_button:

                if(food.foodUnit == 0) {
                    food.foodQuantity += 1;
                } else {
                    food.foodQuantity += 0.1;
                    food.foodQuantity = Math.round( food.foodQuantity * 10.0 ) /
10.0;//avoid floating point arithmetic problem
                }

                new DBUtils(this).updateFood(food);
                populateUi(food.foodId);
                break;
        }
    }

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        switch (id) {
            case R.id.action_edit:
                DialogFragment newFragment = new EditFoodDialogFragment();
                Bundle foodIdBundle = new Bundle();
                foodIdBundle.putInt(FOOD_ID, foodId);
                newFragment.setArguments(foodIdBundle);
                newFragment.show(getSupportFragmentManager(), "edit_food");
                return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onEditFinished(int foodId) {
        populateUi(foodId);
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        navigateToMain();
    }

    @Override
    public boolean onSupportNavigateUp(){
        navigateToMain();
        finish();
        return true;
    }

    public void populateUi(int foodId){
        food = new DBUtils(this).getFoodById(foodId);
        Section section = new DBUtils(this).getSectionById(food.sectionId);

        getSupportActionBar().setTitle(food.foodName);
```

```
        foodAsset.setImageResource(FoodUtils.getFoodAsset(food.foodCategory));

expirationDateText.setText(getResources().getString(R.string.expiration_label) + " "
+ food.foodExpireDate);
        sectionText.setText(getResources().getString(R.string.section_label) + " " +
section.sectionName);
        unitText.setText(getResources().getString(R.string.unit_label)  + " " + new
FoodUtils(this).getUnitString(food.foodUnit));
        categoryText.setText(getResources().getString(R.string.category_label)  + " "
+ new FoodUtils(this).getCategoryName(food.foodCategory));
        quantityText.setText(getResources().getString(R.string.quantity_label)  + " "
+ food.foodQuantity);
        if(food.foodUnit == 0)
quantityText.setText(getResources().getString(R.string.quantity_label) + (int)
food.foodQuantity);
        eatAllButton.setOnClickListener(this);
    }

    public void navigateToMain() {
        Intent intent = new Intent(this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK |Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);
    }
}
```

ДОДАТОК В

(обов'язковий)

ПРЕЗЕНТАЦІЙНІ СЛАЙДИ

Хмельницький Національний Університет
Факультет програмування та комп'ютерних
і телекомунікаційних систем
Кафедра інженерії програмного забезпечення

Дипломний проект, на тему:

«Мобільний додаток для обліку термінів придатності продуктів харчування »

Студент: Романчук Сергій Володимирович

Керівник: Праворська Н. І. кандидат педагогічних наук, доцент

ВСТУП

На сьогоднішній день по даним Всесвітньої продовольчої програми (ВВП), на сміттєзвалищі опиняється близько третини усіх продуктів харчування на планеті. Глобальна економіка втрачає через це приблизно трильйон доларів щорічно. На цю проблему впливають різні чинники, такі як готування чи придбання лишньої їжі ресторанами, магазинами та звичайними громадянами, що часто призводить до витікання терміну придатності і власне до викидання.

Для подолання цієї проблеми ВВП пропонує більш детально приділяти увагу на відходи під час приготування для використання в інших рецептах, а також бути більш уважним до купівлі зайвих продуктів.

Але все одно людям часто доводиться зіштовхуватись з проблемою витікання термінів придатності в продуктів і тому їм потрібне зручне і робоче рішення для цього.

АКТУАЛЬНІСТЬ ТЕМИ

Актуальність моєї теми полягає в тому, що я пропоную зручне рішення у вигляді мобільного додатку, який завжди є під рукою, для проблеми контролю термінів придатності продуктів харчування, яке би допомогло не лише зменшити кількість безглуздої трати продуктів на викидання, а й допомогло би зекономити гроші завдяки контролю наявних продуктів.

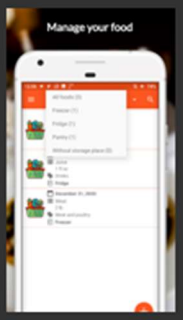
МЕТА ТА ЗАВДАННЯ ДИПЛОМНОГО ПРОЕКТУВАННЯ

Метою проекту є створення мобільного додатку, який би дозволяв швидко та зручно проводити контроль продуктів харчування на їх термін придатності, повідомляючи користувача про витікання терміну придатності.

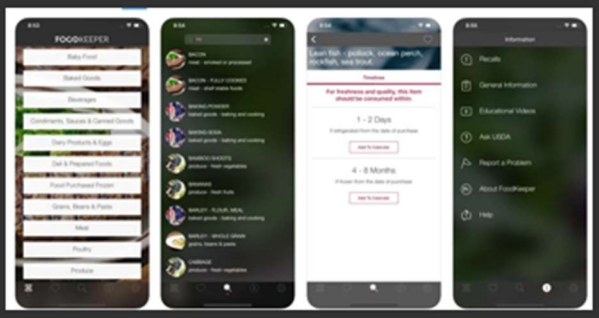
Основними завданнями при виконанні дипломного проекту є:

- визначення та аналіз особливостей предметної області, її суті та основних проблем;
- аналіз наявних на ринку програмних засобів та платформ відповідно предметної області;
- проектування та реалізація програмної системи, яка б відповідала основним вимогам та вирішувала основні проблеми;
- тестування реалізованої системи.

НАЯВНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ



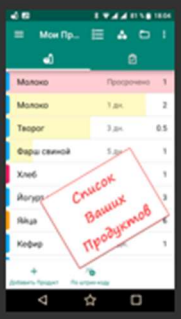
Інтерфейс «Срок годности продуктов питания»



Інтерфейс FoodKeeper

НАЯВНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Інтерфейс Expiring Things



Інтерфейс «Учёт Продуктов»

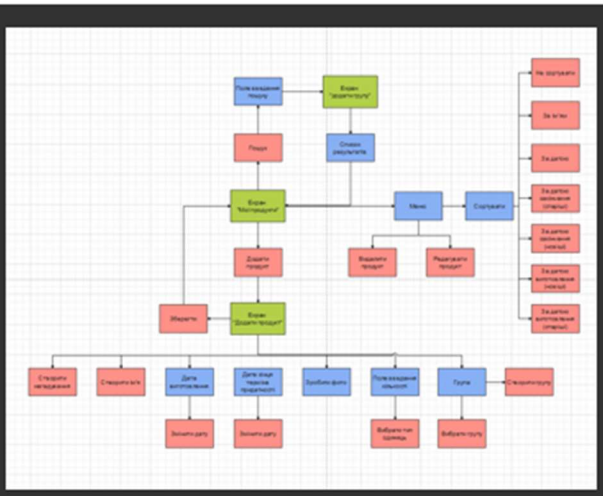
ПРОЕКТУВАННЯ. ВИБІР ПЛАТФОРМИ

Серед доступних варіантів реалізації програми для мобільних платформ я вибрав нативний варіант, а саме реалізацію під операційну систему Android.

Перевагами нативної розробки є висока швидкість відгуку інтерфейсів та розширені можливості стилізації додатку, які доступні ексклюзивно на вибраній операційній системі.

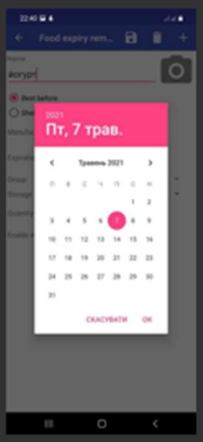
Серед недоліків можна віднести менше охоплення аудиторії.

СХЕМА ПЕРЕХОДІВ МІЖ ЕКРАНАМИ

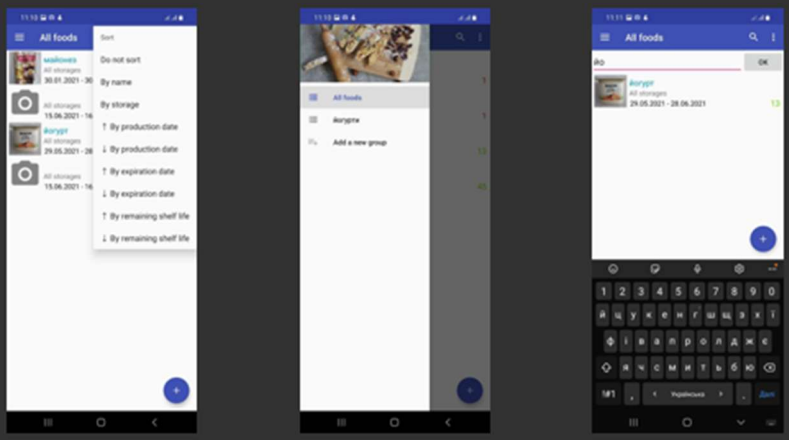


ДІАГРАМИ СТАНІВ

ІНТЕРФЕЙС ДОДАТКУ



ІНТЕРФЕЙС ДОДАТКУ



ВИСНОВКИ

У процесі виконання дипломного проекту був проведений аналіз предметної області, її структурних і функціональних особливостей, також визначено специфіку та особливості розроблення програмного забезпечення для обліку термінів придатності продуктів харчування. Були переглянуті та проаналізовані популярні мобільні додатки для контролю термінів придатності. Під час проектування був проведений аналіз технологій та інструментів, які використовуються для створення Android застосунків. По результатам проведеного аналізу був спроектований зручний та сучасний інтерфейс, який відповідає усім функціональним особливостям та розроблена архітектура додатка. На основі спроектованої архітектури та інтерфейсу, реалізовано мобільний застосунок. Після розробки проведено модульне та функціональне тестування, в результаті якого в розробленому додатку присутні усі функції , які були визначені в технічному завданні і визначено що додаток є готовим для використання.

У результаті роботи був реалізований мобільний додаток для обліку термінів придатності продуктів, завдяки чому було закріплено теоретичні та практичні знання та навички розробки життєвого циклу програмного забезпечення

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Романчука С. В.
Прізвище, ініціали

факультет ПКТС, 4 курс, група ІПЗ-17-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

18.06.2021р
дата

підпис

Mon Jun 21 09:25:41 EEST 2021, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 7.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 94914 Назва: Мобільний додаток для обліку термінів придатності продуктів харчування Додано в БД: 2021-06-21 Автора: С.В. Романчук Керівники: Н.І. Праворська Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	76026	736	12710 (17%)	129 (18%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
92363	Назва: Мобільний додаток для обліку щоденних та щотижневих задач з додаванням тегів та з можливістю створення резервної копії Додано в БД: 2021-06-04 Автора: А. І. Дьоміна Керівники: І. В. Гурман Консультанти: Опоненти:	5051 (7.0%)	96 (13.0%)

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1008337767

Дата перевірки:
21.06.2021 10:39:06 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.06.2021 11:08:16 EEST

ID користувача:
100005589

Назва документа: **Диплом_Романчук_С_В (готовий)_без додатків**

Кількість сторінок: **75** Кількість слів: **12409** Кількість символів: **98930** Розмір файлу: **2.79 MB** ID файлу: **1008408523**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

17.3%
Схожість

Найбільша схожість: **10.1%** з джерелом з Бібліотеки (ID файлу: **1008265198**)

8.95% Джерела з Інтернету	425	Сторінка 77
12.3% Джерела з Бібліотеки	102	Сторінка 83

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1
Підозріле форматування	16 сторінок

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Романчук Сергій Володимирович
Тема Мобільний додаток для обліку термінів придатності продуктів харчування

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки 101

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті проведений аналіз предметної області та проблеми, які мають бути вирішені. Також проведений аналіз наявного програмного забезпечення на предмет достатності їх функціоналу для вирішення поставлених задач. На основі отриманих даних були розроблені функціональні вимоги до мобільного додатку. Також розглянуті деталі реалізації мобільного додатку, проведено його тестування, за результатами якого можна вважати, що додаток працює правильно.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект освітнього ступеня «бакалавр» в основному відповідає поставленому завданню як в теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовується актуальність обраної теми дипломного проектування, формуються мета і завдання проектування, описується практична значимість дипломного проекту. У першому розділі виконано аналіз предметної області, охарактеризовані існуючі рішення для вирішення поставленої задачі, викладено опис функціональних вимог до розроблюваного мобільного додатку. У наступних розділах розкриваються деталі проектування мобільного додатку, моделювання інтерфейсу та системи в цілому, а також описана програмна реалізація системи та тестування.

4. Позитивні сторони проекту Тема дипломного проекту є актуальною, оскільки люди часто не слідкують за термінами придатності продуктів і такий додаток міг би їм допомогти контролювати якість своєї їжі. До позитивних сторін реалізації мобільного додатку можна віднести те, що при розробці програмного продукту використовувались новітні технології та актуальні архітектурні рішення.

5. Негативні сторони проекту Розроблюваний додаток є MVP-продуктом, а отже не можна виконати у повній мірі навантажувальне тестування, що означає, що забезпечення стабільної продуктивності при високих навантаженнях залишається лише на теоретичному рівні. Також в меню додатка не вистачає пункту налаштувань для кастомізації інтерфейсу програми.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконане відповідно до теми дипломного проекту. Графічне представлення виконано на задовільному рівні. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про дипломний проект в цілому Дипломний проект заслуговує задовільної оцінки. Матеріал пояснювальної записки структурований та розписаний на рівні, що дозволяє зрозуміти викладений матеріал у рамках тематики дипломного проекту. Графічний матеріал підкріплений інформацією та дозволяє наочно побачити результати виконання дипломного проекту.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект заслуговує оцінки «задовільно»

РЕЦЕНЗЕНТ Лисенко Сергій Миколайович, доктор технічних наук, доцент кафедри комп'ютерної інженерії та системного програмування (КІСП) ХНУ

"22" червня 2021 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів ідентичності схожості:

Назва: «Мобільний додаток для обліку термінів придатності продуктів харчування»

Автор: Романчук Сергій Володимирович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Праворська Наталія Іванівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломного проекту системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання, відомість документів), у структурі змісту, назвах розділів підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформлені посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/схожості, складає 17,3% і адресується до 425 джерел з Інтернет та 102 джерел з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ Н. І. Праворська

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк