НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

“До захисту допущено”
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

“___”_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Цифрові технології в енергетиці”
зі спеціальності *122 “Комп’ютерні науки”*

на тему: **Android-додаток визначення раціону харчування на основі штучного інтелекту**

Виконав:
студент IV курсу, групи ТР-01

ЗДАНЕВИЧ Володимир Романович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник:
доцент, к.т.н., Світлана ШАПОВАЛОВА _____
(посада, вчене звання ,науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент: _____
_____ _____
(посада, вчене звання, науковий ступінь, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль: *старший викладач* Ольга *БЕСПАЛА* _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА_

«__»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

ЗДАНЕВИЧУ Володимиру Романовичу
(прізвище, ім’я, по батькові)

1. Тема роботи **Android-додаток визначення раціону харчування на основі штучного інтелекту**
керівник роботи ***Шаповалова Світлана Ігорівна, к.т.н., доцент***
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 202__ р. №_____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мови програмування Java, Kotlin, Python, СКБД SQLite, Room, середовище розробки Android studio, MVVM

4. Перелік питань, які потрібно розробити:
1) провести аналіз серед наявних аналогів
2) аргументувати вибрані інструменти, технології та алгоритми для реалізації додатку
3) побудувати архітектуру додатку
4) розробити андроїд додаток для правильного раціону харчування

5. Орієнтований перелік ілюстративного матеріалу діаграма прецендентів, що демонструє доступний функціонал різним сутностям системи, приклади роботи з інтерфейсом програмного продукту та рисунки які підкріплюють написане в тексті

6. Дата видачі завдання «15» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи	15.09.23	
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-14.05.24	
7.	Захист програмного продукту	15.05.24	
8.	Передзахист	03.06.24	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-30.06.2024	

Студент _____ **Володимир ЗДАНЕВИЧ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Світлана ШАПОВАЛОВА**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 54 сторінках, містить 4 ілюстрації, 25 джерел в переліку посилань, 1 додаток.

Мета роботи – створення додатку для підбору раціону за заданими параметрами.

Методи та засоби: програмний продукт було розроблено з використанням таких технологій як Kotlin, Java, Python, MVVM, Room, Glide, View Binding, Android Studio.

Результат – андроїд-додаток, який допомагає користувачеві побудувати свій раціон на основі улюблених страв, а також допомагає знаходити нові рецепти та приготувати їх за рецептом.

Ключові слова: Android-додаток, раціон харчування, Kotlin, Java, Python, MVVM.

ANNOTATION

The thesis is made on 52 pages, contains 51 illustrations, 25 sources in the list of references, 1 appendix.

The purpose of the work is to create software for selecting the best diet in order to achieve the goals.

Methods and tools: the software product was developed using technologies such as Kotlin, Java, MVVM, Room, Glide, View Binding, Android Studio.

The result is an android application that helps the user to build their diet based on their favorite dishes, and also helps to find new recipes and cook them according to the recipe.

Keywords: Android application, diet, Kotlin, Java, Python, MVVM.

ЗМІСТ

ВСТУП 7

1 ЗАДАЧА РОЗРОБКИ ДОДАТКА ДЛЯ ПІДБОРУ РЕЦЕПТІВ ЗА ВПОДОБАННЯМИ КОРИСТУВАЧА 9

1.1 Постановка задач 9

1.2 Засоби необхідні для реалізації задач 10

1.3 Аналіз аналогів 10

2 ЗАСОБИ РЕАЛІЗАЦІЇ 12

2.1 Операційна система Андроїд 12

2.2 Середа розробки програмного забезпечення Android Studio 14

2.3 Мова програмування Java 17

2.4 Мова програмування Kotlin 19

2.5 Мова програмування Python 21

2.6 Шаблон проектування архітектури андроїд-додатка 23

2.7 Розширення SQLite -Room 25

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АНДРОЇД-ДОДАТКУ 28

3.1 Розробка андроїд-додатку 28

3.1.1 Діаграма прецедентів 28

3.1.2 Схема взаємодії модулів додатку 31

3.2 Розробка моделі нейронної мережі 32

3.2.1 Попереднє опрацювання набору даних 33

3.2.2 Фільтрація великих рецептів 35

3.2.3 Створення словника 37

3.2.4 Уніфікація наборів символів 39

3.2.5 Перетворення набору даних в датасет TensorFlow..... 42
3.2.6 Розподіл тексту рецептів на цільову та вхідну послідовність .. 42
3.2.7 Групування..... 44
3.2.8 Збірка моделі 46
3.2.10 Тренування моделі 49
4 ВЗАЄМОДІЯ КОРИСТУВАЧА З АНДРОЇД-ДОДАТКОМ..... 51
ВИСНОВКИ..... 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 55
ДОДАТОК А..... 57

ВСТУП

З кожним роком популярність додатків для телефонів тільки зростає. А враховуючи, що найбільшою по чисельності користувачів системою на даний момент є саме операційна система андроїд, то і вибір операційної системи під яку буде написаний додаток пав саме на андроїд.

Актуальність цієї роботи зумовлена необхідністю створення простого у використанні, але ефективного інструментарію, для впровадження нових харчових звичок або збалансування наявного харчування. Боротьба з наслідками ожиріння чи недостачею ваги лагідними та комфортними методами. Адже надлишкова вага призводить до захворювань серця, які займають перше місце по причинах смертності серед населення у всьому світі.

Метою цієї роботи є розробка додатку, який допоможе підібрати рецепти відповідно до харчових звичок користувача, а також пропозицій нових рецептів.

Завданням цієї роботи є:

— проведення аналізу серед наявних аналогів
— аргументація вибраних інструменти, технології та алгоритми для реалізації додатку
— побудова архітектуру додатку
— розробка андроїд додатку для правильного раціону харчування
— розробка та впровадження в додаток методів штучного інтелекту

Ця дипломна робота є досить цінним внеском в медичну сферу, фітнес сферу. Вона вирішує базові проблеми людини у цих сферах, бо без правильного харчування неможливий гарний результат у фітнесі, адже 70 відсотків успіху припадає саме на раціон, а не на вправи, сон чи ще що небудь. Жоден атлет не зміг набрати м’язову масу не збільшивши добове споживання калорій у декілька разів, але при цьому ретельно не спостерігавши за вмістом основних компонентів у раціоні, це і білки, і жири та вуглеводи. Або навпаки неможливо було б схуднути поїдаючи жирну чи їжу з великим вмістом цукру, без дефіцити енергії,

не запустяться біологічні механізми з використання підшкірного жиру.

Таким чином буде зроблено досить серйозний внесок у покращення умов життя людини та її самопочуття що призведе до росту відсотку здорового населення та загального рівня щастя та рівня життя населення.

1 ЗАДАЧА РОЗРОБКИ ДОДАТКА ДЛЯ ПІДБОРУ РЕЦЕПТІВ ЗА ВПОДОБАННЯМИ КОРИСТУВАЧА

У сучасному світі величезна кількість кулінарних рецептів доступна в інтернеті, що надає широкі можливості для кулінарних експериментів та забезпечує різноманітність у харчуванні. Однак, цей великий обсяг інформації також створює певні труднощі для користувачів, які шукають рецепти, що відповідають їхнім конкретним смакам, дієтичним обмеженням або просто бажаним інгредієнтам. Розробка додатка для підбору рецептів за вподобаннями користувача стає необхідною для спрощення цього процесу, забезпечуючи користувачам персоналізований підхід та швидкий доступ до відповідних кулінарних пропозицій.

1.1 Постановка задач

Розробити андроїд-додаток спрямований на вирішення таких задач:

— провести аналіз серед наявних аналогів. Через аналіз наявних технічних рішень буде висвітлено проблему та виявлено слабкі місця інших програмних засобів;

— аргументувати вибрані інструменти, технології та алгоритми для реалізації додатку. Це допоможе обрати найоптимальніші та сумісні програмні інструменти для вирішення поставлених завдань;

— побудувати архітектуру додатку. Це допоможе зробити андроїд-додаток легким у підтримці та доповненні, а також полегшить процес розробки.

— розробити андроїд додаток для правильного раціону харчування

— розробити та впровадити у додаток штучний інтелект. Завдяки штучному інтелекту з’явиться можливість генерувати нові рецепти

спираючись на вподобання користувача

1.2 Засоби необхідні для реалізації задач

Для реалізації побудови архітектури додатку застосовано шаблон проектування MVVM (Model-View-ViewModel).

З метою розробки андроїд додатку для правильного раціону харчування застосовано наступні засоби:

— мови програмування Java, Kotlin, Python

— середа розробки Android Studio

— надбудова ROOM для зручної роботи з базою даних SQLite.

— бібліотеку пайтона TensorFlow та Recipe box

1.3 Аналіз аналогів

Існує багато аналогів андроїд-додатку який буде розроблений в ході цієї дипломної роботи, вони мають сильні та слабкі сторони, але серед них немає єдиного лідера який би був набагато краще за інших представників. Тож порівняймо перших три аналоги, які посідають лідуючі позиції в рейтингах.

Таблиця 1 – Порівняння аналогів

Назва функціональності	MyFitnessPal	Yazio	Lifesum
журнал харчування	v	v	v
сканер штрих-кодів	v		v
рецепти та плани харчування	v	v	v
обчислення кількості макронутрієнтів	v	v	

Кінець таблиці 1

синхронізація з фітнес-додатками			v
гнучкість налаштувань		v	v
аналіз звичок користувача	v	v	v
штучний інтелект			

Як бачимо після аналізу аналогів чітко видно одну і ту саму слабу сторону інших додатків. Не дивлячись на велику кількість переваг усі ці додатки мають бібліотеку рецептів та видають користувачеві заздалегіть приготовлені результати тим самим обмежуючи користувача шаблонами не розкриваючи повного потенціалу.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

В сучасному світі інформаційних технологій вибір засобів реалізації відіграє ключову роль у створенні ефективних та якісних програмних продуктів. Від обраної операційної системи та середовища розробки до мов програмування і архітектурних шаблонів, всі ці компоненти визначають, наскільки успішним буде кінцевий результат. У цьому розділі ми розглянемо основні засоби реалізації, які використовуються для розробки мобільних додатків на платформі Android та які були застосовані для вирішення поставлених задач.

2.1 Операційна система Андроїд

Отже перейдемо до огляду операційної системи. Операційна система Android[1] є однією з найпопулярніших платформ для мобільних пристроїв у світі. Розроблена компанією Android Inc., яка згодом була придбана Google у 2005 році, після чого популярність стрімко почала зростати, таким чином Android стала основною мобільною операційною системою для мільярдів користувачів. Завдяки її відкритому вихідному коду та гнучкості Android став привабливою операційною системою як для розробників, так і для виробників пристроїв. Далі буде розглянуто історію розвитку Android, архітектуру системи, основні функції та її вплив на ринок мобільних технологій.

Почнемо з самого початку, а саме з заснування. Android Inc. була заснована у жовтні 2003 року Енді Рубіном, Річем Майнером, Ніком Сірсом і Крісом Уайтом. На самому початку їх стартап зосереджувався на створенні операційної системи для фотоапаратів, але незабаром вектор розвитку компанії змінився на створення операційної системи для мобільних телефонів. Після придбання компанії Google у 2005 році розвиток операційної системи Android пришвидшився. Перша комерційна версія операційної системи під назвою Android 1.0, була випущена у вересні 2008 року разом із першим пристроєм на

базі Android – HTC Dream. Слід відзначити, що Android пройшов численні стадії розвитку, включаючи великі оновлення системи, такі як Cupcake, Donut, Eclair, та подальші випуски, кожен з яких додавав нові функції та покращення. Кожна нова версія Android має своє кодове ім'я, яке зазвичай є назвою десерту в алфавітному порядку.

Трохи слів про архітектуру Android.

Android має багатошарову архітектуру, яка включає наступні основні компоненти ядро Linux, Бібліотеки, Runtime, Application Framework, Додатки.

Найнижчій рівень операційної системи це ядро Linux[2]. В основі Android лежить модифіковане ядро Linux, яке забезпечує низькорівневі функції системи, такі як керування пам'яттю, процесами, мережею та іншим обладнанням. Наступним рівнем над ядром йдуть бібліотеки. Android використовує ряд бібліотек, які надають необхідні функції для розробки додатків, включаючи бібліотеки для роботи з мультимедіа, графікою, базами даних та іншими. Далі іде шар під назвою Runtime. Це середовище виконання, яке включає в себе віртуальну машину Dalvik (до Android 5.0) та Android Runtime (ART) для запуску додатків[3]. Після рантайму йде Application Framework. Цей шар забезпечує розробників інструментами та API для створення додатків. Він включає в себе менеджери вікон, ресурсів, активностей та інших компонентів. І останнім ідуть вже додатки. На цьому рівні, користувач взаємодіє з системою. Сюди входять як стандартні додатки, такі як телефон, контакти, браузер, так і додатки, встановлені користувачем.

Що ж про архітектуру поговорили тому саме час згадати про основні функції операційної системи.

Андроїд має широкий перелік функцій, які і роблять його потужною та надзвичайно гнучкою операційною системою. Одна з таких функцій це інтерфейс користувача. Android пропонує настроюваний інтерфейс, тут можна встановити та налаштувати віджети під свій смак, має сповіщення та багатозадачність. Також немаловажним фактором є опція підтримки мультимедіа. Система підтримує відтворення аудіо, відео та графіки різних

форматів, що робить її ідеальною для мультимедійних додатків. Довгий час перевагою андроїда була можливість зовнішнього підключення, та й зараз ця операційна система тримає планку. Android підтримує різні методи підключення, разом з Wi-Fi, Bluetooth, NFC, і мобільні мережі. Неменш важливим для користувачів є камери та датчики. Операційна система дозволяє використовувати камери, мікрофони та іншу гарнітуру для створення додатків доповненої реальності та інших інноваційних рішень. Є у андроїда і централізоване сховище усіх додатків які випускають компанії або програмісти власноруч, воно має назву Google Play Store. Цей магазин додатків дозволяє користувачам легко знаходити, встановлювати та оновлювати додатки без ризиків підхопити віруси.

З огляду на попередню інформацію зрозуміло що така операційна система має вагомий вплив на ринок мобільних технологій. Тож трохи оглянемо цей вплив.

2.2 Середа розробки програмного забезпечення Android Studio

Android Studio це офіційне інтегроване середовище розробки для створення додатків заточене під операційну систему Android. Розроблене компанією Google. Це середовище базується на IntelliJ IDEA від JetBrains і забезпечує розробникам потужні інструменти для створення, тестування і розгортання додатків на різних пристроях Android. Далі буде розглянуто історію розвитку Android Studio, його архітектуру, основні можливості та вплив на процес розробки додатків. Тож почнемо з історії створення

Android Studio було представлено на конференції Google I/O у травні 2013 року як заміну попереднього середовища розробки Eclipse Android Development Tools. Рішення про створення нового середовища розробки було прийняте через необхідність більш інтегрованого і зручного інструменту для розробників, який би надавав більше функціональності та підтримки.

Перша стабільна версія Android Studio 1.0 вийшла у грудні 2014 року. Вона

включала в себе основні функції, необхідні для розробки Android-додатків, такі як редактор коду, візуальний редактор інтерфейсів, емулятор Android і багато інших. З тих пір Android Studio[4] постійно оновлюється, додаючи нові можливості та покращення. Наприклад, у версії 2.0 було додано функцію Instant Run, яка дозволяє значно прискорити процес налагодження.

Розглянемо основні можливості Android Studio. Android Studio пропонує широкий спектр можливостей, що робить його потужним інструментом для розробки додатків. Перша можливість, яку ми розглянемо це вбудовані інструменти для інтелектуального редагування коду. Android Studio забезпечує автозавершення коду, перевірку помилок у реальному часі, рефакторинг коду та інші функції, що значно полегшують процес розробки. Редактор також підтримує живі шаблони (live templates) та аналіз коду, що допомагає уникнути поширених помилок. Також тут представлені потужні інструменти дизайну інтерфейсу. Візуальний редактор дозволяє розробникам швидко створювати та налаштовувати користувацькі інтерфейси, переглядати їх на різних пристроях та роздільних здатностях. Інструменти дизайну включають ConstraintLayout, що дозволяє створювати складні макети з меншими зусиллями. У цій програмі присутня гнучка система збірки під назвою Gradle. Gradle забезпечує гнучкість і можливість налаштування процесу збірки, управління залежностями, а також підтримує автоматизацію рутинних завдань. Gradle також дозволяє розробникам визначати різні конфігурації збірки для різних середовищ, таких як налагодження, тестування та випуск. У цій середі розробки є досить потужний інструмент, який дозволяє одразу запустити програму на обладнанні, яке імітує параметри цільового телефону і називається він Емулятор Android. Вбудований емулятор дозволяє тестувати додатки на різних версіях Android, що значно спрощує процес налагодження та тестування. Емулятор підтримує гарячі оновлення, що дозволяє швидко вносити зміни та тестувати їх без необхідності перезапуску додатка. Є також і інструменти для тестування. Android Studio підтримує написання та виконання різноманітних тестів, включаючи юніт-тести, інтеграційні тести та тести інтерфейсу користувача. Інструменти тестування

включають Espresso для автоматизованого тестування інтерфейсу та Robolectric для емуляції середовища Android у тестах. Є моніторинг продуктивності. Інструменти для аналізу продуктивності додатків дозволяють виявляти проблеми з продуктивністю, витоки пам'яті та інші потенційні проблеми ще на етапі розробки. Android Profiler надає детальну інформацію про використання центрального процесора, пам'яті, мережі та графічного процесора додатком. Для зручності розробникам в середі розробці є інтеграція з Google Cloud. Android Studio дозволяє легко інтегрувати додатки з сервісами Google Cloud, що забезпечує можливості хмарного зберігання, аналітики та інших функцій. Firebase інтеграція дозволяє додавати до додатка аналітику, аутентифікацію, бази даних у режимі реального часу та інші функції.

Тепер поговоримо як же ж такий фундаментальний пул інструментарію повпливав на розробку додатків.

Android Studio значно змінила процес розробки додатків для Android. Ось декілька основних аспектів цього впливу. Підвищила продуктивність. Завдяки потужним інструментам для редагування коду та дизайну інтерфейсу, розробники можуть швидше створювати високоякісні додатки. Інструменти, такі як Instant Run, значно скорочують час на налагодження і тестування. Дала можливість для покращення якості коду. Інструменти для тестування та аналізу продуктивності дозволяють виявляти та виправляти помилки на ранніх етапах розробки, що підвищує загальну якість додатків. Інтеграція з інструментами для безперервної інтеграції дозволяє автоматизувати процеси тестування і збірки, забезпечуючи стабільність і надійність коду. Надала розробникам зручний у використанні інструмент. Інтеграція з системами керування версіями, емулятор Android та інші інструменти роблять процес розробки більш зручним та організованим. Це дозволяє розробникам легко співпрацювати над проектами, слідкувати за змінами та керувати версіями коду. Є можливості для навчання. Android Studio надає багато корисних інструментів та документів, що робить її відмінним вибором для новачків, які тільки починають вивчати розробку під Android. Документація, приклади коду, інтеграція з Kotlin та інші ресурси

допомагають швидко освоїти основи та просунуті техніки розробки. І що не менш важливо вона має інтеграцію з сучасними технологіями. Постійні оновлення Android Studio забезпечують підтримку нових технологій та стандартів, що дозволяє розробникам створювати сучасні та інноваційні додатки. Підтримка нових версій Android, Jetpack компонентів, архітектурних шаблонів та інших сучасних технологій дозволяє створювати додатки, які відповідають найновішим тенденціям.

2.3 Мова програмування Java

Java – це мова програмування загального призначення, яка була створена для забезпечення можливості написання програм, здатних працювати на будь-якому пристрої, незалежно від платформи. Створена компанією Sun Microsystems у 1995 році, Java швидко здобула популярність завдяки своїй незалежності від платформи, безпеці та простоті у використанні. Далі буде розглянуто історію розвитку Java, її архітектуру, основні можливості та вплив на розвиток сучасного програмування.

Тож перейдемо безпосередньо до історії. Розробка Java почалася на початку 1990-х років, коли Джеймс Гослінг і його команда в Sun Microsystems працювали над проектом Green для створення мови, яка могла б працювати на різних платформах. У 1995 році Java була офіційно представлена публіці, і її основною концепцією стала ідея "Write Once, Run Anywhere" (WORA), тобто "Напиши один раз, запускай скрізь". Перший публічний випуск Java (JDK 1.0) відбувся в січні 1996 року, і з того часу мова зазнала численних оновлень. Кожна нова версія додавала нові можливості та покращення, зокрема стандартні бібліотеки, підтримку багатопоточності, покращення безпеки та продуктивності.

У 2009 році Oracle Corporation придбала Sun Microsystems, включаючи Java, і продовжила розвиток мови. Випуски нових версій Java стали регулярними, зокрема, Java 8 (2014)[5], яка внесла значні зміни, такі як лямбда-вирази та Streams API.

Архітектура Java складається з кількох ключових компонентів Java Development Kit, це набір інструментів, необхідних для розробки Java-програм. Включає компілятор (javac), стандартні бібліотеки та середовище виконання Java (JRE). Java Runtime Environment у перекладі означає середовище, необхідне для запуску Java-програм. Включає віртуальну машину Java (JVM) та стандартні бібліотеки. Наступним компонентом джава є Java Virtual Machine[6]. Віртуальна машина, яка виконує байт-код Java. Вона забезпечує незалежність від платформи, оскільки байт-код може виконуватися на будь-якій системі, де встановлено JVM. Також немаловажливу роль відіграють Java API[7]. Вони ж набір бібліотек класів, що надають стандартні функціональні можливості для розробки додатків. Сюди входять пакети для роботи з графічним інтерфейсом, мережами, базами даних, вхідними/вихідними потоками і т.д.

Java володіє численними можливостями, які роблять її однією з найпопулярніших мов програмування, а саме це портативність. Бо завдяки JVM, Java-програми можуть виконуватися на будь-якій платформі без необхідності в перекомпіляції. Основополагаючою парадигмою в джава є об'єктно-орієнтоване програмування (ООП). Java повністю підтримує ООП, що дозволяє створювати добре структуровані та повторно використовувані коди. Також сюди входять багатопоточність. Java підтримує багатопоточне програмування, що дозволяє створювати ефективні та продуктивні програми.

Безпекаи Java має вбудовані механізми безпеки, які забезпечують захист від шкідливого коду та несанкціонованого доступу. Автоматичне керування пам'яттю. Java використовує збір сміття (garbage collection) для автоматичного звільнення невикористовуваної пам'яті, що знижує ризик витоків пам'яті. Також немаловажним фактором є розширюваність. Завдяки широкому набору бібліотек та фреймворків, Java дозволяє легко розширювати можливості додатків.

Java значно вплинула на розвиток сучасного програмування. Вона стала основною мовою для створення корпоративних додатків, веб-сервісів та мобільних додатків. Популярні фреймворки, такі як Spring[8] та Hibernate[9], базуються на Java і широко використовуються у розробці програмного

забезпечення. Крім того, Java є основною мовою для розробки Android-додатків, що робить її важливою для мобільного програмування. Велика кількість корпоративних систем, включаючи банківські системи, системи керування підприємствами та веб-платформи, також використовують Java завдяки її стабільності, безпеці та продуктивності.

2.4 Мова програмування Kotlin

Kotlin — це сучасна мова програмування, розроблена компанією JetBrains, яка вперше була представлена у 2011 році. Назва мови походить від назви острова Котлін у Фінській затоці, де розташована штаб-квартира компанії. Перший стабільний реліз мови відбувся у лютому 2016 року. Kotlin швидко здобув популярність завдяки своїй сумісності з Java та покращеній функціональності, яка усуває багато недоліків Java. Мова розроблена для підвищення продуктивності розробників і забезпечення надійності коду.

Kotlin є статично типізованою мовою програмування, що компілюється у байт-код JVM (Java Virtual Machine). Вона також може компілюватися у JavaScript та рідний код за допомогою Kotlin/Native. Розглянемо основні характеристики Kotlin[10]. Kotlin повністю сумісний з Java, що дозволяє використовувати існуючі бібліотеки та фреймворки Java без жодних проблем. Синтаксис Kotlin значно компактніший у порівнянні з Java, що дозволяє писати менш об'ємний та більш читабельний код. Kotlin має вбудовані механізми для уникнення помилок, пов'язаних з null, які є частою причиною збоїв у програмах на Java. Kotlin підтримує функціональне програмування, дозволяючи використовувати ламбда-функції, вищі порядкові функції та інші функціональні конструкції. Можливість створення розширень для існуючих класів без необхідності вносити зміни у їх вихідний код.

У Kotlin існують два ключових слова для оголошення змінних: val та var. Val використовується для оголошення констант (не змінюваних змінних), а var

— для змінюваних змінних. Функції в Kotlin оголошуються за допомогою ключового слова fun. Вони можуть повертати значення певного типу або бути Unit, що є еквівалентом void у Java. Класи в Kotlin оголошуються за допомогою ключового слова class. Конструктори можуть бути первинними та вторинними, що забезпечує гнучкість при створенні об'єктів. Класи в Kotlin за замовчуванням фінальні (тобто, не можуть бути успадковані). Для дозволу наслідування використовується ключове слово open.

Перевагами мови є сумісність з Java. Це дозволяє розробникам використовувати Kotlin у вже існуючих проектах на Java без необхідності переписувати весь код. Безпечність коду. Механізми, що захищають від помилок, пов'язаних з null, значно знижують кількість збоїв у програмі. Лаконічність. Більш компактний код дозволяє знизити об'єм коду та підвищити його читабельність. Підтримка Android. Google офіційно підтримує Kotlin як мову для розробки під Android, що робить її привабливою для мобільних розробників.

Проте є і недоліки. Низька швидкість компіляції. Хоча продуктивність самого коду на Kotlin є високою, процес компіляції може займати більше часу порівняно з Java. Мала кількість бібліотек. Попри сумісність з Java, все ще бракує нативних бібліотек, створених спеціально для Kotlin. Нова мова. Незважаючи на швидке зростання популярності, Kotlin ще не настільки широко поширений, як Java, що може ускладнити пошук фахівців та навчальні ресурси.

Kotlin використовується у багатьох відомих компаніях та проектах. Наприклад, такі компанії як Google, Amazon, Netflix та Uber активно використовують Kotlin у своїх мобільних додатках. Завдяки офіційній підтримці Android, багато нових проектів обирають Kotlin як основну мову для розробки.

Одним із прикладів успішного використання Kotlin є додаток Trello. Команда розробників вирішила перейти з Java на Kotlin, оскільки це дозволило зменшити кількість коду та зробити його більш читабельним і безпечним. Інший приклад — додаток Evernote, який також використовує Kotlin для мобільної розробки, забезпечуючи стабільну та ефективну роботу додатку.

Kotlin є потужною та сучасною мовою програмування, яка надає розробникам інструменти для створення надійного, безпечного та лаконічного коду. Її сумісність з Java робить перехід на Kotlin безболісним для існуючих проектів, а підтримка з боку великих компаній та спільноти розробників забезпечує подальший розвиток та зростання популярності мови. Попри деякі недоліки, такі як повільніша компіляція та менша кількість нативних бібліотек, переваги Kotlin роблять її привабливою для сучасних проектів у різних галузях програмного забезпечення.

2.5 Мова програмування Python

Python – це високорівнева мова програмування загального призначення, яка відзначається простою синтаксичною структурою та легкістю вивчення. Вона була створена Гвідо ван Россумом і вперше випущена в 1991 році. З того часу Python став однією з найпопулярніших мов програмування у світі завдяки своїй гнучкості, багатству бібліотек та активній спільноті розробників. Мова Python широко використовується в різних сферах, включаючи веб-розробку, аналіз даних, наукові обчислення, автоматизацію процесів, і особливо – в розробці нейронних мереж та машинному навчанні.

Python[11] був створений як мова, яка повинна бути легшою у використанні, ніж існуючі на той час мови, з чіткою та зрозумілою синтаксичною структурою. Від самого початку він був орієнтований на простоту та зручність, що дозволило йому швидко здобути популярність серед розробників. У 2000 році вийшла версія Python 2.0, яка внесла багато покращень і нових функцій. У 2008 році була випущена Python 3.0, яка, хоч і не повністю сумісна з попередніми версіями, принесла значні покращення і нові можливості. З тих пір Python 3 став основною версією, яка активно розвивається і використовується сьогодні.

Python пропонує широкий спектр можливостей для розробників, серед яких:

— Простота та зручність: Легкий для вивчення синтаксис робить Python ідеальним вибором як для початківців, так і для досвідчених програмістів.

— Багата стандартна бібліотека: Python має вбудовану бібліотеку, яка включає модулі для роботи з файлами, інтернет-протоколами, управління даними і багато іншого.

— Підтримка багатьох парадигм програмування: Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування.

— Велика спільнота та екосистема: Активна спільнота та величезна кількість доступних бібліотек роблять Python гнучким інструментом для вирішення різноманітних завдань.

— Міжплатформенність: Python працює на більшості сучасних платформ, включаючи Windows, macOS та Linux.

Однією з найсильніших сторін Python є його популярність у сфері машинного навчання та розробки нейронних мереж. Це зумовлено кількома факторами:

— Бібліотеки для машинного навчання: Python має багато потужних бібліотек, таких як TensorFlow[12], Keras, PyTorch та Scikit-learn, які значно спрощують розробку і тренування нейронних мереж.

— Підтримка обробки даних: Завдяки бібліотекам, таким як NumPy[13], Pandas та Matplotlib[14], Python забезпечує ефективну обробку, аналіз та візуалізацію даних.

— Легкість інтеграції: Python легко інтегрується з іншими мовами програмування і технологіями, що робить його гнучким інструментом для розробки складних систем.

У своїй роботі я використав мову програмування Python для написання нейронної мережі, що дозволило мені ефективно і швидко реалізувати складні алгоритми машинного навчання. Завдяки потужним бібліотекам, таким як TensorFlow та Keras, процес створення нейронної мережі був значно спрощений.

Python забезпечив мене всіма необхідними інструментами для підготовки даних, побудови моделі, її тренування та оцінки. Простота синтаксису та наявність детальної документації зробили цей процес максимально зручним і зрозумілим.

2.6 Шаблон проектування архітектури андроїд-додатка

Шаблон проектування MVVM (Model-View-ViewModel)[15] є одним з найбільш популярних архітектурних шаблонів для розробки програмного забезпечення, особливо у сфері розробки мобільних додатків та десктопних програм. Він був вперше запропонований Джоном Госсманом у 2005 році і став стандартом для багатьох платформ, таких як WPF (Windows Presentation Foundation), Silverlight, Xamarin та інші. Цей шаблон дозволяє чітко розділити логіку додатку та його представлення, що значно спрощує розробку, тестування та підтримку коду. Для наглядності на рисунку 2.1 проілюстровано схему цього патерна.

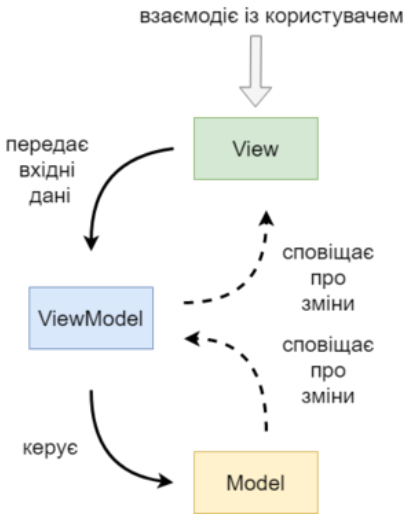

Рисунок 2.1 — Діаграма взаємодії між компонентами шаблону MVVM

MVVM складається з трьох основних компонентів, а саме модель (Model), представлення (View) та представлення-модель (ViewModel). Кожен з цих компонентів має свою роль і відповідальність. Розберемо кожен компонент

більш детально.

Почнемо з першого компоненту, а саме моделі. Модель представляє дані додатку та логіку їх обробки. Це можуть бути як прості об'єкти даних, так і складні бізнес-логіки, які взаємодіють з базою даних або зовнішніми сервісами. Модель не має жодних знань про інтерфейс користувача і працює виключно з даними та бізнес-логікою.

Тепер розглянемо наступний компонент представлення (View). Представлення відповідає за відображення даних та взаємодію з користувачем. Це може бути будь-який UI-компонент, такий як Activity або Fragment в Android, який безпосередньо взаємодіє з користувачем. Представлення не містить бізнес-логіки і працює виключно з відображенням даних та подіями інтерфейсу користувача.

І останнім компонентом є представлення-модель(ViewModel). Це компонент виступає посередником між моделлю та представленням. Вона містить логіку, яка необхідна для підготовки даних для відображення, а також управляє станом інтерфейсу користувача. ViewModel отримує дані від Моделі, обробляє їх і передає до Представлення. Крім того, вона обробляє події інтерфейсу користувача та викликає відповідні методи Моделі.

Цей шаблон проектування має досить багато переваг. Це і розділення відповідальностей. MVVM чітко розділяє логіку додатку та його представлення, що дозволяє легко підтримувати та змінювати код. І тестованість. Оскільки бізнес-логіка зосереджена у ViewModel, її легко тестувати окремо від інтерфейсу користувача. І повторне використання коду. ViewModel можна використовувати повторно у різних частинах додатку або навіть у різних проектах. А також легкість у підтримці. Чітке розділення компонентів спрощує підтримку та розширення коду.

Але не обійшлося і без недоліків. Одним з перших недоліків є складність для новачків. Для розробників, які тільки починають вивчати шаблони проектування, MVVM може здатися складним і незрозумілим. Також це зайвий код. Іноді для реалізації MVVM може знадобитися написати більше коду, ніж

для інших шаблонів, таких як MVC або MVP. А також залежність від фреймворків. У деяких випадках реалізація MVVM вимагає використання специфічних фреймворків або бібліотек, що може збільшити складність проекту.

2.7 Розширення SQLite -Room

Room є однією з основних бібліотек, що входять до складу Android Jetpack, і служить об'єктно-реляційним маппером для роботи з базою даних SQLite[16]. Вона значно спрощує процес збереження, завантаження та управління даними в Android-додатках, забезпечуючи простий та ефективний інтерфейс для розробників. Room надає переваги компіляційних перевірок запитів SQL, що підвищує надійність коду та зменшує кількість помилок.

Розглянемо детально основні компоненти Room[17]. Room складається з трьох основних компонентів Entity (сутність), DAO (об'єкт доступу до даних) та Database (база даних).

Entity представляє таблицю у базі даних. Це клас, який описує структуру таблиці, де кожне поле класу відповідає одному стовпчику таблиці. Entity також може мати додаткові атрибути для налаштування таблиці, такі як первинний ключ, індекси тощо. Цей компонент визначає, як дані будуть зберігатися в базі даних.

DAO є інтерфейсом, що містить методи для доступу до бази даних[18]. Ці методи використовуються для виконання операцій з даними, таких як вставка, оновлення, видалення та запити. DAO забезпечує чітке розділення між логікою додатку та логікою доступу до даних, що спрощує тестування та підтримку коду.

Клас Database є абстрактним класом, що розширює RoomDatabase. Він містить абстрактні методи для доступу до DAO. Клас Database також відповідає за налаштування бази даних, включаючи визначення всіх Entity та версії бази даних. Room використовує цей клас для створення і управління базою даних.

Для використання Room у додатку необхідно виконати декілька кроків. Спочатку додаються відповідні залежності до файлу конфігурації проекту. Далі

створюються класи, що представляють Entity, DAO та Database.

Першим кроком є створення класів, що представляють таблиці бази даних. Ці класи визначають структуру даних, що зберігаються в базі даних. Кожне поле класу відповідає одному стовпчику таблиці, а спеціальні атрибути використовуються для визначення первинного ключа та інших властивостей таблиці.

Наступним кроком є створення інтерфейсів DAO. Ці інтерфейси містять методи для виконання операцій з даними. Методи DAO анотуються спеціальними анотаціями для виконання операцій вставки, оновлення, видалення та запитів. DAO забезпечує зручний спосіб доступу до бази даних без необхідності писати складні SQL-запити вручну.

Останнім кроком є створення класу Database, який розширює RoomDatabase. Цей клас містить абстрактні методи для доступу до DAO та налаштування бази даних. Room використовує цей клас для створення та управління базою даних. Важливо також налаштувати міграцію бази даних у разі зміни її структури.

Room має досить багато переваг. Room надає простий та інтуїтивно зрозумілий інтерфейс для роботи з базою даних SQLite. Вона дозволяє зосередитися на логіці додатку, не витрачаючи багато часу на написання SQL-запитів.

Однією з найбільших переваг Room є компіляційні перевірки запитів SQL. Це означає, що помилки в запитах будуть виявлені під час компіляції, а не під час виконання, що значно зменшує кількість помилок та підвищує надійність коду.

Room легко інтегрується з LiveData та RxJava, що дозволяє зручно працювати з асинхронними даними. Це особливо корисно для додатків, що потребують реагування на зміни даних у реальному часі.

Room забезпечує простий механізм для міграції бази даних при зміні її структури. Це дозволяє легко оновлювати базу даних без втрати даних, забезпечуючи безперервність роботи додатку.

Проте є і недоліки у цієї технології. Для розробників, які не знайомі з ORM, може знадобитися час для освоєння Room. Однак, цей недолік є тимчасовим і зменшується після отримання базових знань та досвіду роботи з бібліотекою.

Room не підтримує деякі складніші функції SQL, такі як повнотекстовий пошук або складні запити з використанням кількох таблиць. Це може обмежити використання Room у проектах, що потребують таких функцій.

Використання анотацій може створювати додатковий оверхед при компіляції, що може вплинути на швидкість розробки. Однак, це також є тимчасовим недоліком, який компенсується підвищенням надійності та зручності коду.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АНДРОЇД-ДОДАТКУ

Особливість додатку полягає у впровадженні нейронної мережі що дало змогу генерувати більш гнучкі та персональні відповіді, а не давати користувачеві заздалегідь підготовлений варіант. Це покращило досвід користування юзерами, через те що вони отримують персоналізовані рішення замість шаблонних як у конкурентів. Використання штучного інтелекту дозволяє генерувати рецепти основуючись на даних введених чи вибраних користувачем, що дозволило генерувати нові смаки та нестандартні поєднання давши юзеру поле для експериментів.

3.1 Розробка андроїд-додатку

Розробка андроїд додатку складається з декількох етапів. Перший етап це формування вимог, побудова діаграми прецедентів для чіткого розуміння повноважень та доступу до функціоналу додатку усіх учасників, що взаємодіють з додатком. Після чого для полегшення роботи над додатком його буде поділено на окремі сегменти і після виконання усіх сегментарних робіт та поєднання їх в єдину систему починається розробка нейронної мережі, яка в кінці від’єднується та органічно поєднується з усіма компонентами системи, з якими нейронна мережа має взаємодіяти. Тож спочатку розглянемо схему будови додатку.

3.1.1 Діаграма прецедентів

Діаграма прецедентів[19] є важливим інструментом у процесі розробки програмного забезпечення, який допомагає візуалізувати взаємодію користувачів із системою. Вона використовується для моделювання функціональних вимог та визначення основних сценаріїв використання додатку. Діаграма прецедентів дозволяє ідентифікувати різні ролі користувачів, які

взаємодіють із системою, та їхні цілі, що допомагає розробникам і бізнес-аналітикам зрозуміти, як саме повинна працювати система для задоволення потреб користувачів.

Основними елементами діаграми прецедентів є актори та прецеденти. Актори представляють зовнішніх учасників системи, які можуть бути користувачами, іншими системами або апаратними засобами. Прецеденти, або сценарії використання, описують функції, які система повинна виконувати, щоб забезпечити досягнення цілей акторів. Зв'язки між акторами та прецедентами показують, які користувачі взаємодіють з якими функціями системи.

Використання діаграм прецедентів у розробці програмного забезпечення має кілька переваг. По-перше, вони допомагають у комунікації між розробниками, бізнес-аналітиками та замовниками, забезпечуючи спільне розуміння функціональних вимог. По-друге, діаграми прецедентів сприяють структуризації процесу розробки, дозволяючи визначити ключові функції системи та їхні взаємозв'язки як це показано на рисунку 3.1. Нарешті, ці діаграми можуть бути корисними під час тестування, оскільки вони забезпечують основу для розробки тестових сценаріїв, які перевіряють, чи відповідає система визначеним вимогам.

Таким чином, діаграма прецедентів є критично важливим інструментом у процесі проектування та розробки програмного забезпечення, який забезпечує чітке уявлення про функціональність системи та її взаємодію з користувачами.

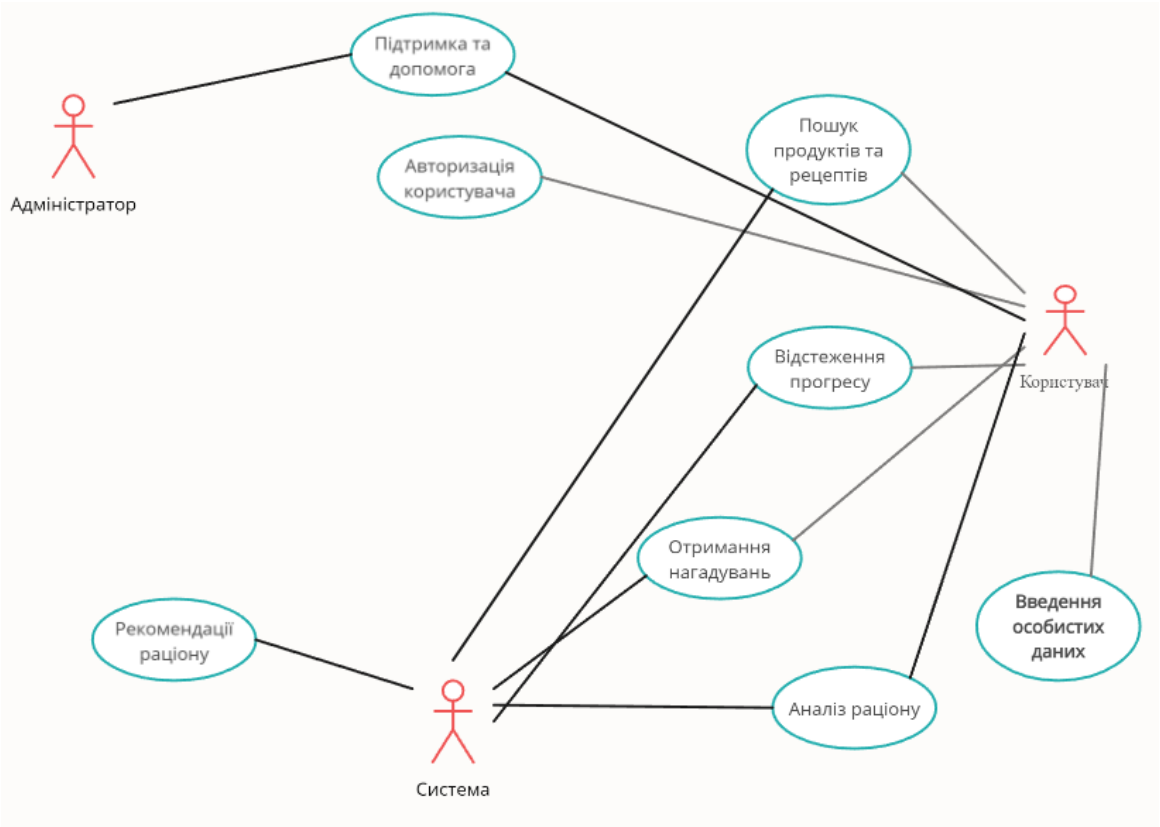

Рисунок 3.1 – Діаграма прецендентів

Прецденти користувача:

— Пошук продуктів та рецептів

— Аналіз раціону

— Відстеження процесу

— Отримування нагадувань

— Введення особистих даних

— Авторизація

— Підтримка та допомога

Прецденти системи:

— Рекомендації раціону

— Аналіз раціону

— Відстеження процесу

— Пошук продуктів та рецептів

— Отримування нагадувань

Прецеденти адміністратора:

— Підтримка та допомога

3.1.2 Схема взаємодії модулів додатку

У цьому додатку було задля зручності роботи над виконанням завдання додаток було поділено на декілька частин. Це модуль UI[20] тобто юзер інтерфейс з яким можуть взаємодіяти користувачі вибираючи функції по душі. За юзер інтерфейсом іде модуль bisness logic[21] де виконуються усі основні задачі. Саме цей модуль пов’язує усі інші модулі разом являючись необхідним проміжним звіном. У цьому кейсі прописана логіка додатка, прописані усі функції кнопок, полів пошуку, полів з рекомендованими блюдами, та інший неменш важливий функціонал. Далі іде провал до блоку AI algoritm[22] бо коли користувач натискає на пошук рецепту страви він через блок бізнес логіки викликає алгоритм штучного інтелекту який і обробляє, а потім видає результат. І після виконання його частини роботи штучний інтелект передає данні через логіку до бази даних яка є основним сховищем даних для додатка. Ця схема взаємодії проілюстрована на рисунку 3.2

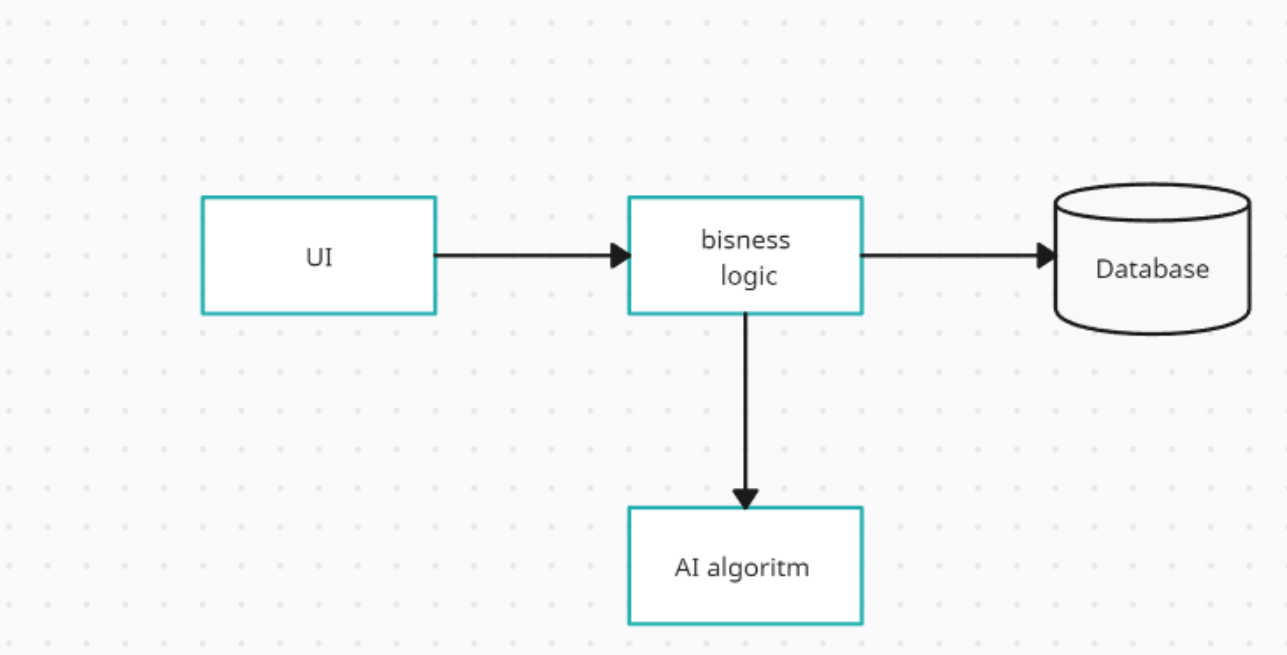

Рисунок 3.2 Схема взаємодії основних компонентів програми

Таким чином вдається максимально оптимізувати роботу додатка адже чим ефективніше та швидше взаємодіють різні компоненти програми тим комфортніше користувачу використовувати додаток, та й висока ефективність є необхідним фактором адже в телефонах набагато менше ресурсів ніж у стільникових комп’ютерів.

3.2 Розробка моделі нейронної мережі

При розробці андроїд-додатку було використано рекурентну нейронну мережу. Рекурентні нейронні мережі є класом глибоких нейронних мереж, які найчастіше застосовують до даних, що ґрунтуються на послідовності, таких як мова, голос, текст. Ключова особливість рекурентних нейронних мереж полягає в тому, що вони мають внутрішню пам'ять, у якій може зберігатися деякий контекст для послідовності який залишився після роботи над моделлю та навчанням її. Наприклад, якщо першим словом послідовності було She, то рекурентна нейронна мережа може запропонувати, що наступним словом буде cooks замість cook, щоб сформувати фразу She cooks, тому що попереднє знання

про перше слово She вже знаходиться у внутрішній пам'яті.

Для будь-якої нейронної мережі треба набір даних, щоб вона могла на них навчитися та коректно виконувати поставлені перед нею завдання. В цій роботі використано Recipe box, через наявність досить великої бази яка налічує більше ста двадцяти п’яти тисяч інгредієнтів, а також рецептів для приготування. З метою навчання залучено бібліотеку пайтона TensorFlow[23]. На рисунку 3.3 проілюстровано імпортування усіх необхідних утиліт.

```
# Бібліотеки для тренування та работи з даними
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import json

# Утиліти
import platform
import time
import pathlib
import os
```

Рисунок 3.3 – Імпортування

Тепер імпортувавши усі необхідні утиліти можна приступати до розробки нейронної мережі шляхом написання коду.

3.2.1 Попереднє опрацювання набору даних

Тепер завантажуємо та розпаковуємо дані після чого об’єднуємо інформацію у одну колекцію. Після проведених дій треба відфільтрувати дані через можливість того, що деякі рецепти не матимуть важливих полів. Цю фільтрацію проведено за допомогою функції зображеної на рисунку 3.4

```python
def recipe_validate_required_fields(recipe):
    required_keys = ['title', 'ingredients', 'instructions']

    if not recipe:
        return False

    for required_key in required_keys:
        if not recipe[required_key]:
            return False

        if type(recipe[required_key]) == list and len(recipe[required_key]) == 0:
            return False

    return True
```

Риснок 3.4 – Функція фільтрації даних

У результаті отримуємо показники проілюстровані на рисунку 3.5

```
Dataset size BEFORE validation 125164
Dataset size AFTER validation 122938
Number of incomplete recipes 2226
```

Рисунок 3.5 – Результат фільтрації

Тож ми отримали сто двадцять дві тисячі дев’ятсот тридцять вісім об’єктів проте модель не може оперувати об’єктами, натомість може оперувати тільки числами тому необхідно конвертувати об’єкти у строки, а після в числа(індекси). Для цього написана функція на рисунку 3.6

```python
def recipe_to_string(recipe):
    #через наявність реклами треба видалити цю строку в датасетах
    noize_string = 'ADVERTISEMENT'

    title = recipe['title']
    ingredients = recipe['ingredients']
    instructions = recipe['instructions'].split('\n')

    ingredients_string = ''
    for ingredient in ingredients:
        ingredient = ingredient.replace(noize_string, '')
        if ingredient:
            ingredients_string += f'• {ingredient}\n'

    instructions_string = ''
    for instruction in instructions:
        instruction = instruction.replace(noize_string, '')
        if instruction:
            instructions_string += f' {instruction}\n'

    return f'{STOP_WORD_TITLE}{title}\n{STOP_WORD_INGREDIENTS}{ingredients_string}{STOP_WORD_INSTRUCTIONS}{instructions_string}'
```

Рисунок 3.6 – Конвертація

Тож відсортувавши та конвертувавши усе що треба можемо вже переходити до першого тестування, яке покаже як працює нейронна мережа. Тож виведемо результат продемонструвавши його на рисунку 3.7

```
Recipe #1
---------
• Slow Cooker Chicken and Dumplings

❖

• 4 skinless, boneless chicken breast halves
• 2 tablespoons butter
• 2 (10.75 ounce) cans condensed cream of chicken soup
• 1 onion, finely diced
• 2 (10 ounce) packages refrigerated biscuit dough, torn into pieces

*

Place the chicken, butter, soup, and onion in a slow cooker, and fill with
enough water to cover.
Cover, and cook for 5 to 6 hours on High. About 30 minutes before serving,
place the torn biscuit dough in the slow cooker. Cook until the dough is no
longer raw in the center.
```

```
Recipe #2
---------
• Awesome Slow Cooker Pot Roast

❖

• 2 (10.75 ounce) cans condensed cream of mushroom soup
• 1 (1 ounce) package dry onion soup mix
• 1 1/4 cups water
• 5 1/2 pounds pot roast

*

In a slow cooker, mix cream of mushroom soup, dry onion soup mix and water.
Place pot roast in slow cooker and coat with soup mixture.
Cook on High setting for 3 to 4 hours, or on Low setting for 8 to 9 hours.
```

Рисунок 3.7 – Перший результат

3.2.2 Фільтрація великих рецептів

Перед початком навчання рекурентної нейронної мережі необхідно зробити всі тексти рецептів однаковими за довжиною. Чим більша ця довжина, тим більше інформації про кожен рецепт братиме участь у навчанні (за умови, що довжину всіх рецептів налаштовано відповідно до найбільшого з них). З іншого боку, довгі послідовності сповільнюють навчання. Також можливо, що

у нас є дев’яносто дев’ять рецептів довжиною в тисяча символів і один рецепт довжиною в п’ять тисяч символів. Зіставлення дев’яноста дев’яти рецептів довжиною п’ять тисяч символів (шляхом додавання символів зупинки в кінці рецептів) з іншими рецептами навряд чи суттєво покращить точність моделі, але точно сповільнить її навчання. Тому ми використаємо код на рисунку 3.8, щоб проаналізувати довжину рецептів у наборі даних і вибрати відповідний

```
recipes_lengths = []
for recipe_text in dataset_stringified:
    recipes_lengths.append(len(recipe_text))

plt.hist(recipes_lengths, bins=50)
plt.show()
```

Рисунок 3.8 – Довжини рецептів

Отримали графік який чітко демонструє на рисунку 3.9 що абсолютна більшість рецептів має розмір менший за п’ять тисяч символів.

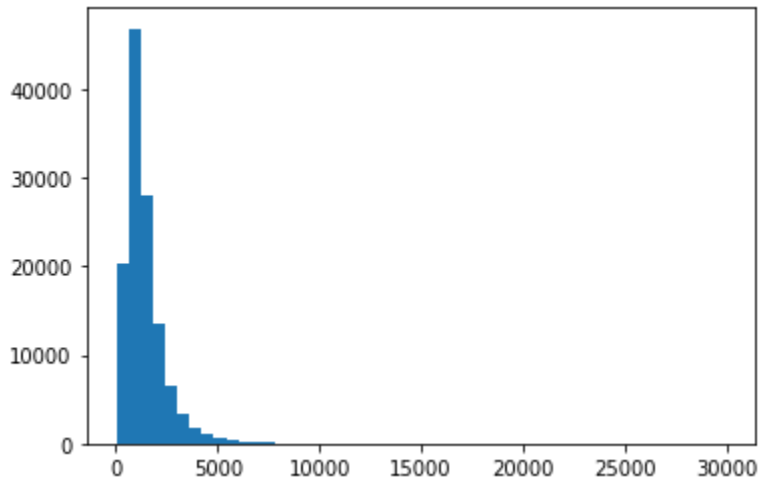

Рисунок 3.9 – Графік розподілу розмірів

З метою точнішого визначення граничного значення подивимось на графік

поближче завдяки коду на рисунку 3.10

```
plt.hist(recipes_lengths, range=(0, 8000), bins=50)
plt.show()
```

Рисунок 3.10 – Більший масштаб графіка

Відповідно і сам графік проілюстровано на рисунку 3.11

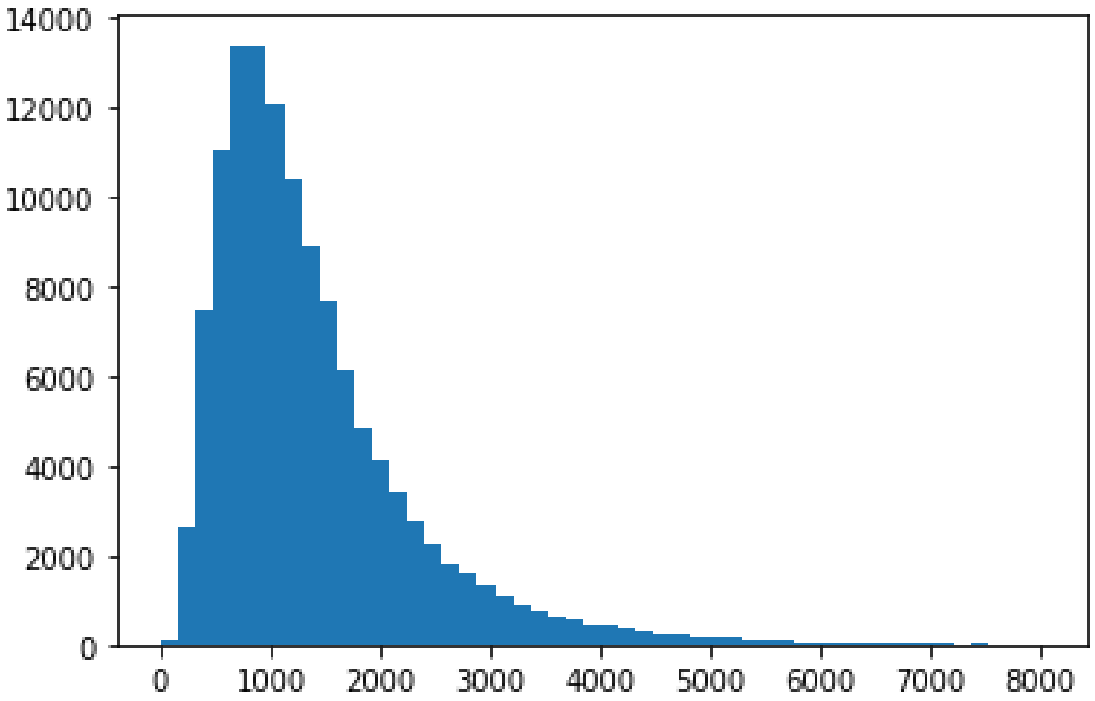

Рисунок 3.11 – Графік розподілу розмірів

Як видно по графіку найоптимальнішим та однорідним датасет буде при розмірі у дві тисячі символів.

3.2.3 Створення словника

Рекурентні нейронні мережі не можуть розуміти символ та слова. Замість цього вони розуміють числа. Тому нам необхідно перетворити текст рецептів (послідовність символів) на послідовність чисел.

Тому в цій роботі використано модель символьного рівня з архітектурою довгої короткочасної пам’яті. Тобто замість того щоб піти стандартним шляхом присвоївши індекси словам, ці індекси будуть присвоєні літерам, бо слова це масиви літер з яких вони складаються. За рахунок цього нейронна мережа буде передбачати індекс наступного символа замість індекса наступного слова.

Невід’ємною частиною генерації тексту рецептів є унікальний стоп-символ, який буде обмежувати генерацію тексту при нових запитах. Саме завдяки цьому символу будуть зберігатися визначена довжина генерації нових рецептів. На рисунку 3.12 можна побачити реалізацію цієї ідеї в коді.

```
STOP_SIGN = '␣'

tokenizer = tf.keras.preprocessing.text.Tokenizer(
    char_level=True,
    filters='',
    lower=False,
    split=''
)
```

Рисунок 3.12 – Демонстрація реалізації стоп-символу

У результаті цих дій отримано наступний набір символів(рисунок 3.13)

```
['␣', ' ', 'e', 'a', 't', 'o', 'n', 'i', 'r', 's', 'l', 'd', 'h', 'c', 'u', 'p', '\n', 'm', 'g', 'b', ',', '.', 'f', 'w', '•', 'k', '1', 'v', 'y', '2', '/', '"', '"', 'S', '4', 'C',
'-', '3', 'x', 'P', '5', '0', '(', ')', 'A', 'B', 'z', 'j', 'F', 'T', 'R', '▫', '❖', '*', 'I', 'M', ';', 'q', 'D', 'W', '8', 'G', '6', 'L', 'H', ':', '7', 'O', '"',
'E', 'K', '9', 'U', 'N', 'V', 'J', '°', 'é', '"', 'Y', 'Q', '*', '!', 'Z', '–', '&', '%', 'ñ', 'è', '"', 'Î', 'X', '?', '¿', '—', 'ç', '#', '½', 'í', '=', '"', 'â', '"',
'¼', '+', '>', '$', '<', 'á', 'ó', 'ú', 'ï', 'É', 'û', ']', '[', 'ü', 'ê', 'à', '_', '\xad', '¾', '’', '', '°', '⁄', 'ä', 'Ú', 'ù', '"', '}', 'ö', '{', 'ì', 'ô', '\x96',
'"', '×', '°', '»', '@', '§', '\', '◊', '‰', '"', '·', '\u202d', '⅛', 'å', 'fl', '`', 'Á', 'ë', '\x97', '\x1a', 'ø', '⅓', '|', 'ư', '\x92', '´', '–', 'Â',
'␣', '¤', '"', '\xa0', 'ơ', 'ă', '̀', '⅞', '€', '~', '\x95']
```

Рисунок 3.13 – Символьний список

Саме на цих символах і буде тренуватися нейронна мережа формувати

майбутні рецепти, спираючись на список інгредієнтів які вводить користувач. Тепер конвертуймо цілу строку символів для перевірки роботоспроможності підходу. Для цього скористаймося функціоналом токенайзера як показано на рисунку 3.14 та отримано результат на рисунку 3.15

```
tokenizer.texts_to_sequences(['  yes'])
```

Рисунок 3.14 – Конвертація строки

```
[[51, 1, 28, 2, 9]]
```

Рисунок 3.15 – Результат

В даній роботі векторизація даних являє собою перетворення масиву строк у числовий вектор який являє собою одномірний масив індексів. Таким чином з’явився набір відповідностей символ-індекс та індекс-символ. Завдяки цьому тепер стає можливою конвертація строк з рецептами у символьну послідовність.

3.2.4 Уніфікація наборів символів

Проте масиви можуть бути різної довжини, що ускладнить роботу з ними у майбутньому тому для подальшої роботи треба привести довжини усіх послідовностей до одного розміру. Тож для початку дізнаємось за допомогою циклу на рисунку 3.16 довжини перших десяти рецептів.

```
for recipe_index, recipe in enumerate(dataset_vectorized[:10]):
    print('Recipe #{} length: {}'.format(recipe_index + 1, len(recipe)))
```

Рисунок 3.16 – Цикл виводу

У результаті виконання ми отримаємо наступний результат на рисунку 3.17.

```
Recipe #1 length: 546
Recipe #2 length: 401
Recipe #3 length: 671
Recipe #4 length: 736
Recipe #5 length: 1518
Recipe #6 length: 740
Recipe #7 length: 839
Recipe #8 length: 667
Recipe #9 length: 1264
Recipe #10 length: 854
```

Рисунок 3.17 – Результат

Як видно з рисунка довжина коливається від чотириста одного до тисяча п’ятсот вісімнадцяти символів що унеможливлює кластеризацію та подальшу роботу з цим. Тому усі рецепти будуть приведені до одного розміру за допомогою встановлення на кінці стоп-символу. Це реалізовано наступним чином(рисунок 3.18).

```
dataset_vectorized_padded_without_stops = tf.keras.preprocessing.sequence.pad_sequences(
    dataset_vectorized,
    padding='post',
    truncating='post',

    maxlen=MAX_RECIPE_LENGTH-1,
    value=tokenizer.texts_to_sequences([STOP_SIGN])[0]
)

dataset_vectorized_padded = tf.keras.preprocessing.sequence.pad_sequences(
    dataset_vectorized_padded_without_stops,
    padding='post',
    truncating='post',
    maxlen=MAX_RECIPE_LENGTH+1,
    value=tokenizer.texts_to_sequences([STOP_SIGN])[0]
)

for recipe_index, recipe in enumerate(dataset_vectorized_padded[:10]):
    print('Recipe #{} length: {}'.format(recipe_index, len(recipe)))
```

Рисунок 3.18 – Реалізація уніфікації

В результаті отримали наступне(рисунок 3.19)

```
Recipe #0 length: 2001
Recipe #1 length: 2001
Recipe #2 length: 2001
Recipe #3 length: 2001
Recipe #4 length: 2001
Recipe #5 length: 2001
Recipe #6 length: 2001
Recipe #7 length: 2001
Recipe #8 length: 2001
Recipe #9 length: 2001
```

Рисунок 3.19 – Результат уніфікації

Після виконаних дій розмір усіх рецептів став дві тисячі один символ, що більше на один символ ніж має бути. Це зроблено спеціально бо далі буде сформовано вхідну послідовність X довжиною дві тисячі символів та цільову послідовність У довжиною в дві тисячі символів, які матимуть зсув в один символ відносно один одного. Після цієї підгонки рецепт виглядає як на рисунку 3.20.

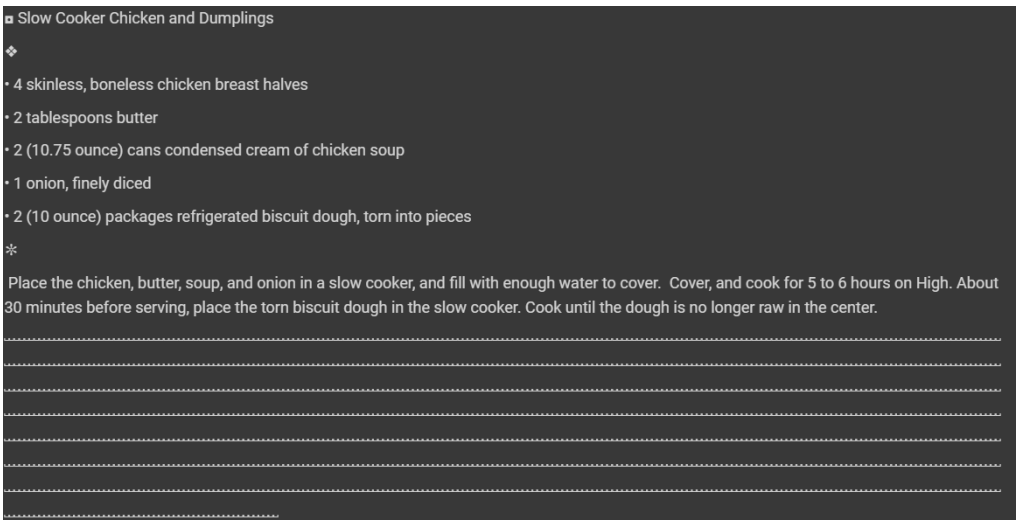

Рисунок 3.20 – Результат після уніфікації

Після генерації рецепту проміжки між останнім символом рецепту та кінцем заповнились символом ␣. Відтак кожен рецепт тепер заповнений одним

чи декількома такими символами.

3.2.5 Перетворення набору даних в датасет TensorFlow.

До цього моменту ми працювали з набором даних як з масивом NumPy. Але цей підхід породжує ряд труднощів при подальшій роботі тому краще трансформувати NumPy масив даних в TensorFlow dataset[24]. Скориставшись API TensorFlow dataset виведемо перший рецепт з набору даних. Результат на рисунку 3.21.

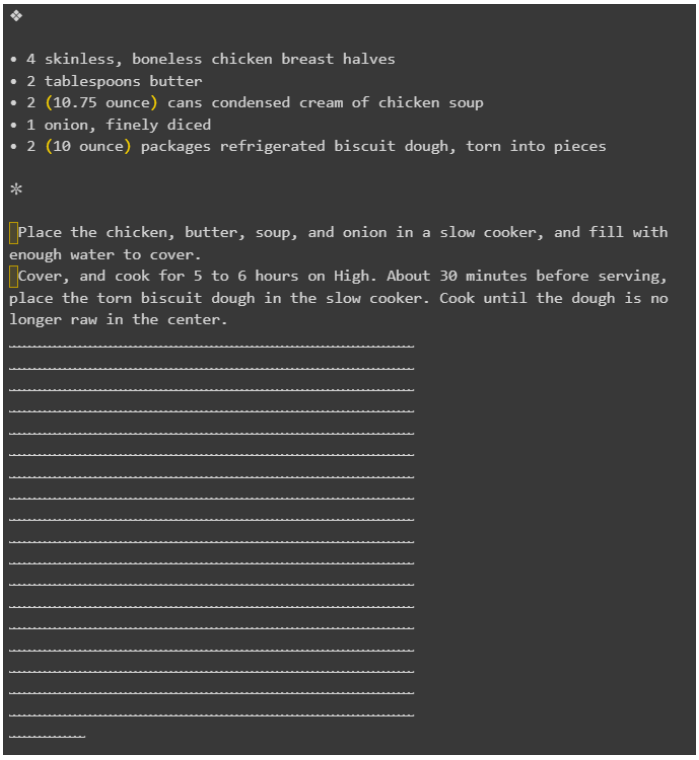

Рисунок 3.21 – Перший рецепт згенерований за допомогою API

3.2.6 Розподіл тексту рецептів на цільову та вхідну послідовність

Штучний інтелект потребує підказок під час навчання. До прикладу, коли подаватиметься на вхід символ H, мережі треба буде сповістити, який символ чекаємо наступним (наприклад це буде e). Отримавши підказки стосовно кожного символу рецепта, модель намагатиметься відшукати залежності між символами.

З цієї причини для послідовності з текстом рецепту необхідно

продублювати та зсунути на один символ, щоб сформувати вхідну та цільову послідовність. Наприклад, якщо довжина послідовності дорівнює 3, натомість текст – Good, то вхідна послідовність буде відповідно Goo, а цільова – ood. Тепер кожен екземпляр даних з датасету являє собою кортеж з двох послідовностей: вхідної та цільової що можна побачити за допомогою коду на рисунку 3.22

```
for input_example, target_example in dataset_targeted.take(1):
    print('Input sequence size:', repr(len(input_example.numpy())))
    print('Target sequence size:', repr(len(target_example.numpy())))
    print()

    input_stringified = tokenizer.sequences_to_texts([input_example.numpy()[:50]])[0]
    target_stringified = tokenizer.sequences_to_texts([target_example.numpy()[:50]])[0]

    print('Input:  ', repr(''.join(input_stringified)))
    print('Target: ', repr(''.join(target_stringified)))
```

Рисунок 3.22 – Код для демонстрації екземпляру

Результат виконання коду на рисунку 3.23

```
Input sequence size: 2000
Target sequence size: 2000

Input:  '▯ S l o w  C o o k e r  C h i c k e n  a n d  D u m p l i n g s
\n \n ❖ \n \n •  4  s k i n l e'
Target:  ' S l o w  C o o k e r  C h i c k e n  a n d  D u m p l i n g s
\n \n ❖ \n \n •  4  s k i n l e s'
```

Рисунок 3.23 - Результат

Кожен індекс цих двох послідовностей буде крок за кроком оброблятися нашою нейронною мережею. На кроці номер нуль модель отримуватиме індекс символу на вхід, і для цього нейронна мережа має передбачити індекс символу який позначає пробіл в якості наступного символу. На наступного кроку модель отримує індекс символу пробіл на вхід і має передбачити індекс символу S` на

виході. Так на кожному наступному кроці на вході моделі буде надходити не тільки новий символ, а також збережений внутрішній стан моделі, що надасть їй змогу брати до уваги не тільки один символ, а й історію кількох попередніх символів що можна побачити на рисунку 3.24.

```
Step  1
  input: 51 ('Q')
  expected output: 1 (' ')
Step  2
  input: 1 (' ')
  expected output: 33 ('S')
Step  3
  input: 33 ('S')
  expected output: 10 ('l')
Step  4
  input: 10 ('l')
  expected output: 5 ('o')
Step  5
  input: 5 ('o')
  expected output: 23 ('w')
Step  6
  input: 23 ('w')
  expected output: 1 (' ')
Step  7
  input: 1 (' ')
  expected output: 35 ('C')
Step  8
  input: 35 ('C')
  expected output: 5 ('o')
Step  9
  input: 5 ('o')
  expected output: 5 ('o')
Step 10
  input: 5 ('o')
  expected output: 25 ('k')
```

Рисунок 3.24 – Демонстрація роботи алгоритму

3.2.7 Групування

У пулі даних знаходиться приблизно десять тисяч рецептів кожен з яких має довжину у дві тисячі символів(рисунок 3.25).

```
<MapDataset shapes: ((2000,), (2000,)), types: (tf.int32, tf.int32)>
```

Рисунок 3.25 – Розмір рецептів

Для наглядності продемонструємо на рисунках 3.26 та 3.27 параметри набору даних.

```
print('TOTAL_RECIPES_NUM: ', TOTAL_RECIPES_NUM)
print('MAX_RECIPE_LENGTH: ', MAX_RECIPE_LENGTH)
print('VOCABULARY_SIZE: ', VOCABULARY_SIZE)
```

3.26 – Код виводу даних

```
TOTAL_RECIPES_NUM:  100212
MAX_RECIPE_LENGTH:  2000
VOCABULARY_SIZE:  176
```

3.27 – результат виконання коду

Якщо під час процесу тренування моделі ми надамо їй повний набір даних, після чого спробуємо розрахувати метод зворотного поширення помилки[25] для всіх рецептів одночасно, то стикнемося з проблемою недостачі пам'ять, через що кожна тренувальна епоха може зайняти занадто багато часу. Щоб уникнути такої ситуації, потрібно поділити набір даних на пакети. Це реалізовано в коді на рисунку 3.28.

```
BATCH_SIZE = 64


SHUFFLE_BUFFER_SIZE = 1000

dataset_train = dataset_targeted
    .shuffle(SHUFFLE_BUFFER_SIZE)
  .batch(BATCH_SIZE, drop_remainder=True)
  .repeat()

print(dataset_train)
```

Рисунок 3.28 - Пакетування

У результаті виконання вищевказаного коду отримуємо в консолі результат який продемонстровано на малюнку 3.29.

```
<RepeatDataset shapes: ((64, 2000), (64, 2000)), types: (tf.int32, tf.int32)>
```

Рисунок 3.29 – Результат пакетування

Можна помітити що тепер кожний екземпляр складається з усе тих же наборів для входящої та цільової послідовності але вже згрупований в пакети по шістдесят чотири штуки. Також це прекрасно видно на рисунку 3.30.

```
1st batch: input_text: tf.Tensor(
[[ 51   1  54 ... 165 165 165]
 [ 51   1  64 ... 165 165 165]
 [ 51   1  44 ... 165 165 165]
 ...
 [ 51   1  69 ... 165 165 165]
 [ 51   1  55 ... 165 165 165]
 [ 51   1  70 ... 165 165 165]], shape=(64, 2000), dtype=int32)

1st batch: target_text: tf.Tensor(
[[  1  54   4 ... 165 165 165]
 [  1  64   5 ... 165 165 165]
 [  1  44   6 ... 165 165 165]
 ...
 [  1  69   3 ... 165 165 165]
 [  1  55   3 ... 165 165 165]
 [  1  70   2 ... 165 165 165]], shape=(64, 2000), dtype=int32)
```

Рисунок 3.30 – Демонстрація групування

3.2.8 Збірка моделі

Перейдемо до повноцінного збору нейронної моделі яка повноцінно генеруватиме текст, а не просто набір букв які навчилася вгадувати. Тож напишемо наступну функцію яка будуватиме модель(рисунок 3.31).

```python
def build_model(vocab_size, embedding_dim, rnn_units, batch_size):
    model = tf.keras.models.Sequential()

    model.add(tf.keras.layers.Embedding(
        input_dim=vocab_size,
        output_dim=embedding_dim,
        batch_input_shape=[batch_size, None]
    ))

    model.add(tf.keras.layers.LSTM(
        units=rnn_units,
        return_sequences=True,
        stateful=True,
        recurrent_initializer=tf.keras.initializers.GlorotNormal()
    ))

    model.add(tf.keras.layers.Dense(vocab_size))

    return model

model = build_model(
  vocab_size=VOCABULARY_SIZE,
  embedding_dim=256,
  rnn_units=1024,
  batch_size=BATCH_SIZE
)

model.summary()
```

Рисунок 3.31

В результаті виконання цієї функції отримано повноцінну модель та для наглядності виведено в консоль у вигляді таблиці(рисунок 3.32)

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_13 (Embedding)     (64, None, 256)           45056
_________________________________________________________________
lstm_9 (LSTM)                (64, None, 1024)          5246976
_________________________________________________________________
dense_8 (Dense)              (64, None, 176)           180400
=================================================================
Total params: 5,472,432
Trainable params: 5,472,432
Non-trainable params: 0
```

Рисунок 3.32

Перед тренуванням взглянемо як працює ненавчена модель та подивимось що вона прогнозує перед тренуванням завдяки коду на рисунку 3.33.

```
    print('Prediction for the 1st letter of the batch 1st sequense:')
print(example_batch_predictions[0, 0])
```

Рисунок 3.33 – предтренування

Отже можемо отримати та подовитися на результати які видає система(рисунок 3.34).

```
Prediction for the 1st letter of the batch 1st sequense: tf.Tensor( [-9.0643829e-03 -1.9503604e-03 9.3381782e-04 3.7442446e-03
-2.0541784e-03 -7.4054599e-03 -7.1884273e-03 2.6014952e-03 4.8721582e-03 3.0045470e-04 2.6016519e-04 -4.1374690e-03
5.3856964e-03 2.6284808e-03 -5.6002503e-03 2.6019611e-03 -1.9491187e-03 -3.1097094e-04 6.3465843e-03 1.4640498e-03
2.4560774e-03 -3.1256995e-03 1.4104056e-03 2.5478401e-04 5.4266443e-03 -4.1188141e-03 3.6904984e-03 -5.8337618e-03
3.6372752e-03 -3.1899021e-05 3.2178329e-03 1.5033322e-04 5.2770867e-04 -8.1920059e-04 -2.2364906e-03 -2.3271297e-03
4.4109682e-03 4.2381673e-04 1.0532180e-03 -1.4208974e-03 -3.2446394e-03 -4.5869066e-03 4.3250201e-04 -4.3490473e-03
3.7889536e-03 -9.2122913e-04 7.8936084e-04 -9.7079907e-04 1.7070504e-03 -2.5260956e-03 6.7904620e-03 1.5470090e-03
-9.4337866e-04 -1.5072266e-03 6.8939931e-04 -1.0795534e-03 -3.1912089e-03 2.3665284e-03 1.7737487e-03 -2.3504677e-03
-6.8649277e-04 9.6421910e-04 -4.1204207e-03 -3.8750230e-03 1.9077851e-03 4.7145790e-05 -2.9846188e-03 5.8050319e-03
-5.6210475e-04 -2.5910907e-04 5.2890396e-03 -5.8653783e-03 -6.0040038e-06 2.3905798e-03 -2.9405006e-03 2.0132761e-03
-3.5594390e-03 4.0282350e-04 4.7719614e-03 -2.4438011e-03 -1.1028582e-03 2.0007135e-03 -1.6961874e-03 -4.2196750e-03
-3.5689408e-03 -4.1934610e-03 -8.5307617e-04 1.5773368e-04 -1.4612130e-03 9.5826073e-04 4.0543079e-04 -2.3562380e-04
-1.5394683e-03 3.6650903e-03 3.5997448e-03 2.2390878e-03 -6.8982318e-04 1.4068574e-03 -2.0531749e-03 -1.5443334e-03
-1.8235333e-03 -3.2099178e-03 1.6660831e-03 1.2230751e-03 3.8084832e-03 6.9559496e-03 5.7684043e-03 3.1751506e-03
7.4234616e-04 1.1971325e-04 -2.7798198e-03 2.1485630e-03 4.0362971e-03 6.4410735e-05 1.7432809e-03 3.2334479e-03
-6.1469898e-03 -2.2205685e-03 -1.0864032e-03 -2.0876178e-07 2.3065242e-03 -1.5816523e-03 -2.1492387e-03 -4.4033155e-03
1.1003019e-03 -9.7132073e-04 -6.3941808e-04 3.0277157e-03 2.9096641e-03 -2.4778468e-03 -2.9532036e-03 7.7463314e-04
2.7473709e-03 -7.6333171e-04 -8.1811845e-03 -1.3959130e-03 3.2840301e-03 6.0461317e-03 -1.3022404e-04 -9.4000692e-04
-2.0096730e-04 3.3895797e-03 2.9710699e-03 1.9046264e-03 2.5092331e-03 -2.0799250e-04 -2.2211851e-04 -3.4621451e-05
1.9962704e-03 -2.3159904e-03 2.9832027e-03 3.3852295e-03 3.4411502e-04 -1.9019389e-03 -3.6734296e-04 -1.4232489e-03
2.6938838e-03 -2.8015859e-03 -5.7366290e-03 8.0239226e-04 -6.2909431e-04 1.1508183e-03 -1.5899434e-04 -5.9326587e-04
-4.1618512e-04 5.2454891e-03 1.2823739e-03 -1.7550631e-03 -3.0120560e-03 -3.8433261e-03 -9.6873334e-04 1.9963509e-03
1.8154597e-03 4.7434499e-03 1.7146189e-03 1.1544267e-03], shape=(176,), dtype=float32)
```

Рисунок 3.34 – Результат не тренованої системи

Для кожного символу на вході моделі масив прикладів пакетних прогнозів містить вектор імовірностей того, який символ може бути наступним. У разі ймовірності у десятого елемента вектора, припустимо дорівнює три десятих, а ймовірність у двадцятого елемента дорівнює одна ціла три десятих, то

для нейронної мережі вибір двадцятого елемента буде пріорітетнішим та бажаним, саме тому його варто обрати як прогнозований наступний елемент.

Однак так як наша ціль це генерація різних рецептів, навіть якщо будуть однакові вхідні слова, то не можна щоразу обирати символ у якого найбільша вірогідність бути наступним. Бо якщо користуватися тільки максимальною вірогідністю то буде щоразу генеруватися один і той же рецепт знову і знову, за умови однакових даних поданих на вхід. Для вирішення цієї проблеми спробуємо використати відбір вибірки за ймовірностями завдяки функції tf.random.categorical(). Це надасть спонтанності та непередбачуваності в передбаченнях моделі. Завдяки цьому з’являється варіативність в генерації рецептів.

3.2.9 Тренування моделі

Компілюємо модель завдяки наступному коду(рисунок 3.35).

```
adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model.compile(
    optimizer=adam_optimizer,
    loss=loss
)
```

Рисунок 3.35 – Компіляція моделі

Тренуватимемо модель п’ятсот епох, причому кожна епоха матиме тисяча п’ятсот кроків. Виконуючи крок модель навчатиметься на пакеті з шістидесяти чотирьох рецептів. Кожного кроку виконано градієнтний спуск для пакету з шестидесяти чотирьох рецептів, де в кожному міститься по дві тисячі символів. Для візуалізації на рисунку 3.36 буде вивід консолі зі звітом по контрольних точках.

```
EPOCHS:           500
INITIAL_EPOCH:    1
STEPS_PER_EPOCH:  1500
```

Рисунок 3.36 – Кількість навчання штучного інтелекту

На рисунку 3.36 чітко видно що система пройшла п’ятсот епох, була одна єдина епоха що ініціювала та було півтори тисячі кроків між епохами. Тож тепер візуалізуймо процес навчання за допомогою графіка. На рисунку 3.37 продемонстрований графік який показує залежність кількості помилок до кількості епох. Так як цей показник постійно падає то можна стверджувати що штучний інтелект вчиться і з часом допускає менше помилок.

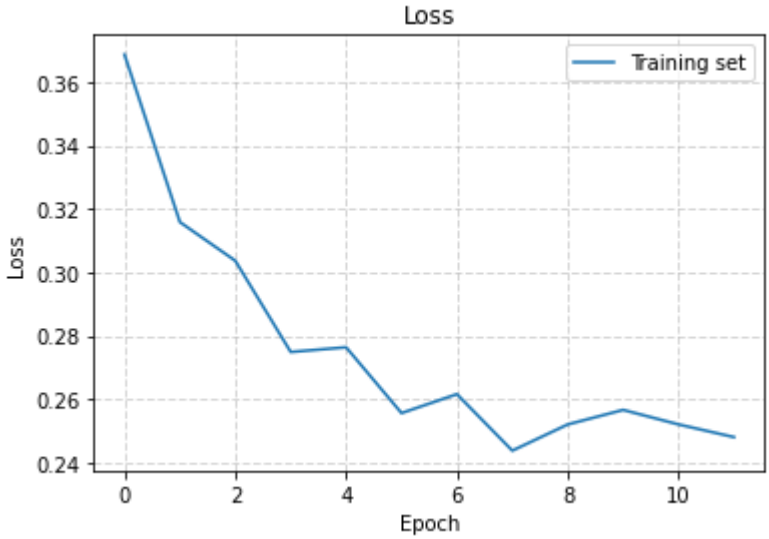

Рисунок 3.37 – Графік залежності кількості помилок від епох

Тепер можна стверджувати що система штучного інтелекту навчена та готова до використання. Тож далі виконано імплементацію цієї системи до основного коду додатку тим самим завершивши розробку програми.

4 ВЗАЄМОДІЯ КОРИСТУВАЧА З АНДРОЇД-ДОДАТКОМ

У попередніх розділах ми детально розглянули всі аспекти андроїд-додатку, зокрема технічні засоби, використані при його проектуванні та створенні. Однак, функціональність додатку та використовувані технології, хоч і важливі, залишаються прихованими від користувача. Вони є невидимими елементами, які забезпечують роботу додатку, але не є тим, з чим безпосередньо взаємодіє користувач. Саме інтерфейс користувача є тією частиною, з якою користувач стикається щоразу, коли відкриває додаток. Інтерфейс - це те, що користувач бачить, відчуває і з чим взаємодіє, і саме він формує перше враження від додатку.

Перше що бачить користувач коли заходить в додаток це головну сторінку. Для наочності нижче буде проілюстровано основні компоненти головної сторінки на рисунку 4.1.

Рисунок 4.1 Головна сторінка

На головній сторінці для зручності розміщено три розділи. Це розділ де можна вписуючи продукти чи назву страви знайти її, це розділ під назвою Популярне, а також розділ категорії. Натиснувши на перший розділ користувач бачить поле для вводу та спливаючий список страв по його запросу. У розділі популярне відображається або страви які по версії алгоритму штучного інтелекту є досить близькими та спорідненими з вподобаннями користувача, або блюда які найбільше за останню добу запитували інші користувачі. Варіанти відображення залежать від вибору користувача та можуть бути змінені у налаштуваннях.

У нижній частині екрану користувачу доступна функція перемикання між вкладками. Натиснувши на вкладку улюблене користувач перейде до частини де відображаються усі страви, які він позначив. Для прикладу було позначено декілька страв які можна буде побачити на рисунку 4.2.

Рисунок 4.2- Улюблене

Для прикладу було вибрано стейки з яловичини та червоної риби, а також гамбургер. З часом цих страв ставатиме все більше та якщо користувач захоче

прибрати якусь з них, він запросто зможе вилучити страву.

Натиснувши в нижній частині екрану на кнопку категорії людина потрапляє на сторінку де зібрані усі категорії страв, кухонь світу та напрямків.

Рисунок 4.3 Категорії

Як видно на рисунку 4.3 у цій вкладці представлені такі категорії як страви для веганів та окремо страви для вегетаріанців. Тут представлені десерти та страви з італійської кухні, повноцінні рецепти сніданків, обідів, вечерь у різних конфігураціях. Це розмаїття дає великі можливості для урізноманітнення раціону користувача.

ВИСНОВКИ

Під час роботи над дипломною роботою було виконано усі поставлені завдання. Основною метою було впровадження нових технологій в підходи до розробки андроїд-додатків, ціллю яких є генерування раціону харчування тим самим розширюючи функціонал та можливості додатків.

Процес розробки додатку охоплював декілька ключових етапів.

1. Проведено аналіз серед наявних аналогів та виявлено головний недолік. Це те що в них не має варіативності, що було в розробленому під час роботи додатку.
2. Побудовано архітектуру додатку та описано програмне забезпечення яке використовувалось під час роботи. Це значно прискорило та облегшило розробку коду та дало можливість на майбутнє розширяти функціонал додатку.
3. Розроблено додаток для підбору правильного харчування на основі побажань та цілей користувача та успішно імплементовано в додаток алгоритми штучного інтелекту.

Розроблений андроїд-додаток має потенціал для розвитку. В майбутньому є можливість впровадити систему оплати та додаткового платного контенту. Також є заділ на майбутнє для розширення функціоналу та введення корисних функцій. Наприклад можна буде впровадити систему кооперації з іншими додатками чи зробити сканер штрих-кодів для автоматичного додавання товарів до кошика або відслідковування товарів які будуть спожиті ще в магазині і відсіювати шкідливі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Develop for Android | Android Developers. *Android Developers*. URL: https://developer.android.com/develop.
2. Учасники проектів Вікімедіа. Ядро Linux – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Ядро_Linux.
3. Android runtime and Dalvik | Android Open Source Project. *Android Open Source Project*. URL: https://source.android.com/docs/core/runtime.
4. Android Studio Jellyfish | 2023.3.1 | Android Developers. *Android Developers*. URL: https://developer.android.com/studio/releases.
5. Java Documentation. *Oracle Help Center*. URL: https://docs.oracle.com/en/java/.
6. The Java Virtual Machine Specification. *Moved*. URL: https://docs.oracle.com/javase/specs/jvms/se8/html/.
7. Java Standard Edition 7 API. *Moved*. URL: https://docs.oracle.com/javase/7/docs/api/.
8. Spring Framework Documentation :: Spring Framework. *Spring | Home*. URL: https://docs.spring.io/spring-framework/reference/index.html.
9. Documentation - 6.5 - Hibernate ORM. *Hibernate*. URL: https://hibernate.org/orm/documentation/6.5/.
10. Kotlin Docs | Kotlin. Kotlin Help. URL: https://kotlinlang.org/docs/home.html.
11. Python Documentation. 3.12.4 Documentation. URL: https://docs.python.org/uk/3/.
12. API Documentation | TensorFlow v2.16.1. TensorFlow. URL: https://www.tensorflow.org/api_docs.
13. NumPy Documentation. NumPy -. URL: https://numpy.org/doc/.
14. Matplotlib documentation. Matplotlib – Visualization with Python. URL: https://matplotlib.org/stable/index.html.
15. ViewModel overview | Android Developers. Android Developers. URL: https://developer.android.com/topic/libraries/architecture/viewmodel.
16. SQLite Documentation. SQLite Home Page. URL: https://www.sqlite.org/docs.html.
17. Room | Jetpack | Android Developers. Android Developers. URL: https://developer.android.com/jetpack/androidx/releases/room.
18. Data Access Object. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/java/technologies/dataaccessobject.html.

19. Учасники проектів Вікімедіа. Діаграма прецедентів – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Діаграма_прецедентів.
20. Учасники проектів Вікімедіа. Дизайн інтерфейсу користувача – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Дизайн_інтерфейсу_користувача.
21. Учасники проектів Вікімедіа. Бізнес-логіка – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Бізнес-логіка.
22. Учасники проектів Вікімедіа. Штучний інтелект – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Штучний_інтелект.
23. Guide | TensorFlow Core. *TensorFlow*. URL: https://www.tensorflow.org/guide.
24. TensorFlow Datasets. *TensorFlow*. URL: https://www.tensorflow.org/datasets.
25. Учасники проектів Вікімедіа. Метод зворотного поширення помилки – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Метод_зворотного_поширення_помилки.

ДОДАТОК А

ТЕКСТ ПРОГРАМНОГО МОДУЛЯ

«Android-додаток визначення раціону харчування
на основі штучного інтелекту»

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ІАТЕ_ЦТЕ_ТР-01_24Б

Аркушів 2

Київ-2024

```
def load_data(quiet=False):

    data_file_list = [
        'recipes_raw_nosource_ar.json',
        'recipes_raw_nosource_epi.json',
        'recipes_raw_nosource_fn.json',
    ]

    combined_data = []

    for data_file in data_file_list:
        data_file_path = f'{CACHE_DIR}/datasets/{data_file}'

        with open(data_file_path) as file:
            json_data = json.load(file)
            json_data_values = list(json_data.values())
            key_list = [key for key in json_data_values[0]]
            key_list.sort()
            combined_data += json_data_values



            if quiet == False:
                print(data_file_path)
                print('========================================')
                print('Number of examples: ', len(json_data_values), '\n')
                print('Example object keys:\n', key_list, '\n')
                print('Example object:\n', json_data_values[0], '\n')
```

```
            print('Required keys:\n')
            print('  title: ', json_data_values[0]['title'], '\n')
            print('  ingredients: ', json_data_values[0]['ingredients'], '\n')
            print('  instructions: ', json_data_values[0]['instructions'])
            print('\n\n')

    return combined_data

raw_data = load_data()
```