м

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

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення комп’ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання*

Виконав: студент IV курсу, групи *ІП-72 Зоренко Вікторія Василівна* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *асистент, Сарнацький В.В.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *ст. викладач Вітковська І.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

_____ _____
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

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

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення комп’ютеризованих систем*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2021 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Зоренко Вікторії Василівані
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання*»

керівник проєкту Сарнацький Владислав Віталійович, *ас.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “11” травня 2021 р. №1139-с

2. Термін подання студентом проєкту «08» *червня 2021 року*

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: загальні положення, аналіз предметної області і опис її, аналіз успішних IT-проєктів, аналіз вимог до програмного забезпечення
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, безпека даних
3) Аналіз якості та тестування програмного забезпечення
4) Інструкція користувача

5. Перелік графічного матеріалу

1) Схема структурна компонентів програмного забезпечення

2) Схема структурна діяльності

3) Схема бази даних

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «14» *березня 2021 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*10.03.2021*	
2.	*Аналіз існуючих методів розв'язання задачі*	*24.03.2021*	
3.	*Постановка та формалізація задачі*	*31.03.2021*	
4.	*Аналіз вимог до програмного забезпечення*	*3.04.2021*	
5.	*Алгоритмізація задачі*	*5.04.2021*	
6.	*Моделювання програмного забезпечення*	*25.04.201*	
7.	*Обґрунтування використовуваних технічних засобів*	*27.04.201*	
8.	*Розробка архітектури програмного забезпечення*	*1.04.2021*	
9.	*Розробка програмного забезпечення*	*7.04.2021*	
10.	*Налагодження програми*	*13.05.2021*	
11.	*Виконання графічних документів*	*14.05.201*	
12.	*Оформлення пояснювальної записки*	*15.05.201*	
13.	*Подання ДП на попередній захист*	*17.05.2021*	
14.	*Подання ДП рецензенту*		
15.	*Подання ДП на основний захист*	*08.06.2021*	

Студент _____ Вікторія Зоренко
(підпис)

Керівник _____ Владислав Сарнацький
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-7203.045490.02.81	Пояснювальна записка	74	
3	А4	КПІ.ІП-7203.045490.03.81	Технічне завдання	12	
4	А4	КПІ.ІП-7203.045490.04.51	Програма та методика тестування	6	
5	А3	КПІ.ІП-7203.045490.81.99.СС	Схема структурна компонентів програмного забезпечення	1	
6	А3	КПІ.ІП-7203.045490.81.99.СС	Схема структурна діяльності	1	
7	А3	КПІ.ІП-7203.045490.81.99.СБД	Схема бази даних	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-7203.045490.01.90		
Розробн.		Зоренко В.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Сарнацький В.В					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-72	
Н/контр.		Вітковська І.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка до дипломної роботи: 78 сторінки, 27 рисунків, 31 таблиць, 7 посилань.

Мета роботи: Проєктування та розробка мобільного застосунку для підтримки здорового способу життя на базі машинного навчання.

В першому розділі проведено аналіз вимог до програмного забезпечення, наведені функціональні та нефункціональні вимоги, наведено постановку завдання.

В другому розділі виконане моделювання програмного забезпечення, наведені його архітектура та подробиці реалізації.

В третьому розділі розроблені процеси тестування даного програмного забезпечення, а також проведений його аналіз якості.

В четвертому розділі розроблені процеси розгортання програмного забезпечення, та наведена інструкція користувача.

МОБІЛЬНИЙ ЗАСТОСУНОК, ЗДОРОВИЙ СПОСІБ ЖИТТЯ, МАШИННЕ НАВЧАННЯ.

ANNOTATION

Explanatory note to the thesis: 78 pages, 27 figures, 31 tables, 7 links.

Purpose: Besigning and development a mobile application to support a healthy lifestyle based on machine learning algorithms.

The first section analyzes the software requirements, provides functional and non-functional requirements, sets the task.

The second section simulates the software, its architecture and implementation details.

The third section develops the testing processes of this software, as well as its quality analysis.

The fourth section develops software deployment processes and provides user instructions

MOBILE APPLICATION, HEALTHY LIFESTYLE, MACHINE LEARNING.

Пояснювальна записка
до дипломного проєкту

на тему: МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ НА БАЗІ МАШИННОГО НАВЧАННЯ

Київ – 2021 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Tensorflow [1] – платформа машинного навчання з відкритим вихідним кодом.

Django [2] - високорівневий веб-фреймворк Python, що містить набір готових інструментів для конструювання сучасного та захищеного веб-застосунку. Безкоштовний та з відкритим кодом.

Celery [3] - це розподілена система для обробки величезної кількості повідомлень, що містить набір інструментів, необхідними для підтримки такої системи. Ця черга завдань зосереджена на обробці в режимі реального часу, а також підтримує планування завдань.

Redis [4] - сховище структури даних в пам'яті, що використовується як база даних, кеш-пам’ять та посередник повідомлень.

UWsgi [5] – сервер для хостингу веб-застосунків, створений для запуску python-додатків через протокол wsgi.

MVC [6] – патерн програмування, що передбачає реалізацію додатку у вигляді тришарової архітектури (модель, представлення, контролер).

ВСТУП

У сучасному світі дуже гостро постає питання здорового способу життя та підтримки морального стану. Вже більше як рік у світі панує пандемія і людство обмежене у своїх ресурсах та діях. Це стосується як і високо розвинутих країн, так і країн третього світу. Ситуація у світі змушує людей обмежувати контакти, пересування і відповідно відчувати стрес. Життя у постійній напрузі, хвилюванні та нервах стало нормою для більшості. Якість життя населення не є такою як минулого року і це помітно. На жаль, сучасна медицина не може спрогнозувати, які наслідки COVID-19 буде мати на людство у моральному плані. Саме тому кожен самостійно повинен підтримувати свій фізичний та моральний стан на гарному рівні, задля кращого майбутнього. На цей момент життя багатьох схоже на «День бабака», коли наче і відбувається щось протягом дня, але дні одноманітні. Ми можемо спостерігати як змінилось життя сучасної людини з соціально-активного до більш уособленого та домашнього. Більшість людей працюють вдома, ведуть малоактивний спосіб життя і користуються послугами доставками. Це має негативний вплив на організм та моральний стан. Зараз найбільш гостро постає питання у контролі свого здоров'я та способів його покращення. Існують різні додатки де можна стежити за станом свого здоров'я, але мало з них включають ще моральний аспект життя та соціальне життя, хоча це ключова складова дійсно збалансованого способу життя, адже гідно з визначенням Всесвітньої організації охорони здоров'я: здоров'я - це стан повного фізичного, психічного та соціального благополуччя, а не лише відсутність хвороб та вад. Мій застосунок буде поєднувати у собі можливість відстежувати фізичний і моральний стан. Завдяки цьому кожен користувач не проґавить важливий аспект життя і зможе покращити якість свого життя.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Машинне навчання – підгалузь штучного інтелекту, яка дозволяє програмному забезпеченню застосовувати статистичні прийоми для можливості “навчатись” (покращувати продуктивність у певних задачах) з даних, без явного програмування їх.

На початку свого існування у 1950-х роках машинне навчання застосовувалось для вдосконалення статичних методів, пізніше почали використовувати і розробляти прості алгоритми. Лише у 1990-х роках машинне навчання почали використовувати для обробки, аналізу даних та “навчання” на них.

У сучасному світі за допомогою машинного навчання можна вирішувати не лише аналітичні, фізичні чи фінансові задачі, а цілком буденні від яких залежить життя кожної людини. Цьому і присвячена дана дипломна робота.

1.2 Аналіз предметної області і опис її

Здоровий спосіб життя є невіддільною частиною існування кожної людини. Визначальною особливістю здорового способу життя є те ще він допомагає підтримувати організм людини у працездатному стані довший проміжок часу. Основними складовими здоров’я є фізичне здоров’я, емоційне здоров’я, духовне здоров’я, інтелектуальне здоров’я, екологічне здоров’я та соціальне здоров’я (рисунок 1.1).

Рисунок 1.1 – Складові здоров’я

Для того аби чіткіше зрозуміти що саме мається на увазі розглянемо визначення кожної складової:

Фізичне здоров’я: підтримка здорового тіла за допомогою регулярних фізичних вправ, правильного харчування, повноцінного сну та уникання шкідливих звичок.

Емоційне здоров’я: Будьте в контакті зі своєю емоційною присутністю та усвідомлюйте власні думки та почуття та почувайтесь комфортно. Емоційне самопочуття покладається на те, щоб висловлювати свої думки та відчуття та бути здатними поглинати думки інших.

Духовне здоров’я: маючи відчуття того, що життя має сенс і має мету, і що ми керуємось у своїй подорожі. Духовне благополуччя полягає у тому, щоб охопити метафізичне і вийти за межі фізичної сфери існування та переживань.

Інтелектуальне здоров’я: можливість брати участь у жвавій взаємодії з навколишнім світом. Інтелект – це згинання м’язів розуму та відкриття розуму. Інтелектуальна істота - це продовження навчання, розв'язання проблем, обробка та творчість. Інтелектуальне оздоровлення передбачає спілкування з іншими на загальному рівні.

Екологічне здоров’я: оточуючи себе здоровим робочим та життєвим середовищем, вільним від небезпек, і зосередженим на збереженні всіх природних ресурсів та ролі, яку ми відіграємо в покращенні навколишнього середовища. Оздоровлення навколишнього середовища - це повага до природи та навколишнього середовища та отримання особистого задоволення від нашого оточення.

Соціальне здоров’я: соціальне благополуччя - це стосунки, взаємодія та комунікація з іншими. Соціальне оздоровлення - це також відчуття комфорту у власній шкірі, щоб мати змогу внести свій внесок і залучитись до здорового середовища життя. Включення людей у всі аспекти нашого життя рівнозначно соціальному оздоровленню.

Для того аби індивідуум міг сказати, що він повністю здоровий потрібно мати прогрес і слідкувати за кожним аспектом свого життя. На цей момент існує багато способів підтримувати здоровий спосіб життя: займатись у спортзалі, відвідувати басейн, вживати здорову, збалансовану їжу. Для цього люди звертаються до послуг персональних тренерів, дієтологів, друзів чи просто сідають у пошуках інформації за опрацювання інформації з мережі Інтернет. Саме тому на ринку зараз є багато додатків, котрі допомагають відслідковувати режим сну, режим харчування, режим пиття води та інші. Через високий попит кількість таких додатків зростає кожного року. Безсумнівно ці додатки допомагають людині досягати вищого рівня усвідомленості свого здоров’я життя, себе та стилю свого життя. У сучасному ритмі життя, коли потрібно бути постійно у русі і вирішувати багато задач одночасно дані додатки суттєво допомагають, адже людський фактор ніхто не скасовував. У багатозадачності люди часто забувають про базові потреби і потім можуть мати непоправні проблеми з здоров’ям. Консультуючись у різних спеціалістів, особа намагається вирішити ці проблеми, проте людям властиво помилятись, що може призвести до непоправних наслідків, у той час, як рекомендації «кваліфікованих» тренерів, дієтологів можуть здаватись

коректними. Часто зустрічається практика групових рекомендацій по харчуванню, режиму дня, занять спортом, режиму пиття води. Якщо розглядати практику групових занять спортом, то дана практика має низку мінусів таких як: недостатня увага для однієї особи, ігнорування особливостей організму, розпорошування уваги на групу. Коли мова йде про харчування, режим пиття води то невиключно, що можливе ігнорування індивідуальних особливостей організму, побажань та режиму. Неоднозначним, але плюсом такого є менше вартість за годину спеціаліста, адже він вже працює не індивідуально, а на колектив, відповідно менше уваги приділяючи кожному. Більш ефективним є підхід індивідуальний, коли спеціаліст працює з клієнтом тет-а-тет і приділяє всю свою увагу клієнту. Це дає змогу усвідомити, звернути увагу і приділити її певній проблемі. Також індивідуальний підхід розглядає запит кожної людини окремо, а не підлаштовується під групу. Це дає змогу швидше досягнути бажаного результату. Така система дає змогу людям не лише підібрати те що відходить їм індивідуально, а й вносити зміни незалежно від інших. Мінусом такого підходу є його вартість, адже спеціаліст буде приділяти увагу лише 1 особі.

1.3 Аналіз успішних IT-проектів

Існують різні програмні рішення для підтримки здорового способу життя на ринку. Дані застосунки мають місце і для платформ Android і для платформ iOS. Найпопулярніші з них пропонують своїм користувачам різний спектр можливостей. Основний функціонал цих додатків, як і вони самі, будуть розглянуті далі.

1.3.1 Аналіз відомих технічних рішень

На даний момент на ринку є різні додатки, що допомагають слідкувати за прогресом у звичках, слідкувати за харчуванням або кількістю випитої води чи зроблених вправ. Поділити їх на категорії доволі важко, адже вони мають

різний функціонал і різну мету. Умовно можна виділити такі сфери реалізації та використання додатків:

- Трекери харчування;
- Трекери фізичної активності;
- Трекери сну.

Звісно, це не є повний список місць використання таких додатків, адже існує багато продуктів на ринку. Використання таких додатків допомагає у поліпшенні та більшому контролі свого життя. Вони можуть стати у нагоді при активному способі життя, коли не вистачає часу на всі необхідні справи. Здебільшого такі додатки мають функціонал нагадувань, який можна адаптувати під себе. Таким чином програма відправить сповіщення про те що потрібно випити води, поїсти чи зробити певну фізичну активність. На рисунках 1.2, 1.3 та 1.4 приклади сповіщень від програм аналогів.

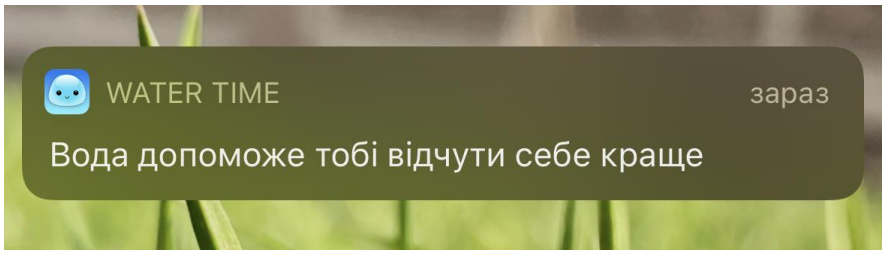


Рисунок 1.2 – Приклад сповіщення про необхідність випити води

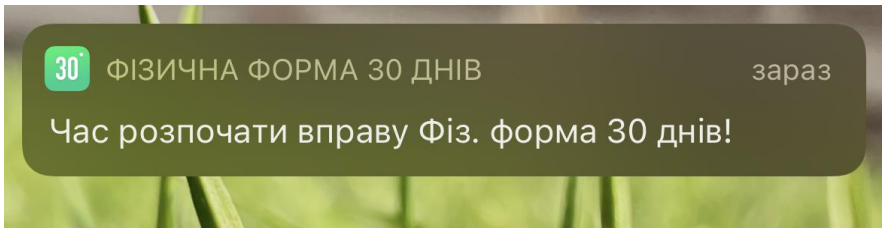


Рисунок 1.3 – Приклад сповіщення про тренування

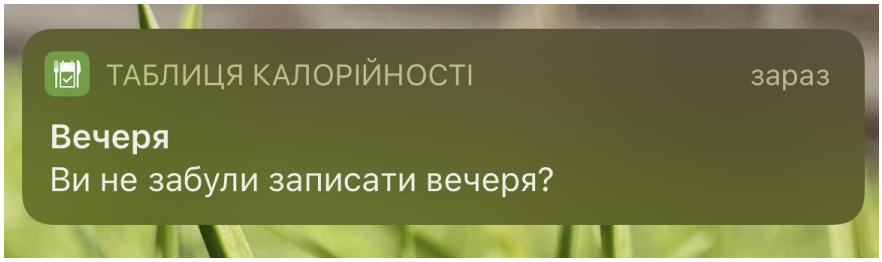


Рисунок 1.4 – Приклад сповіщення про заповнення раціону харчування

Ці додатки мають допомагають вести більш усвідомлений та контрольований спосіб життя. Проте вони мають свої недоліки про, які поговоримо нижче.

1.3.2 Аналіз відомих програмних продуктів

а) Way of life - Habit Tracker – додаток для відстеження звичок. Даний додаток допомагає відстежувати прогрес нових та існуючих звичок. Інтерфейс даного додатку є зручним та має кольорове відображення поточного стану прогресу по звичці. Також у додатку реалізована можливість ставити нові цілі, проводити аналітику успіхів. Зручна особливість – можливість сортувати звички за категоріями. Додаток надає можливість пройти коротку інструкцію при першому вході, що є зручно і полегшує знайомлення користувача з функціоналом. Але додаток не має функції додавання приміток незалежних від звичок. Також додаток виконаний українською мовою, що є зручним для користувачів, котрі не знають іноземних мов, проте турбуються про свої звички. На рисунку 1.5 наведений скріншот з додатку де показаний функціонал відстеження звички.

Рисунок 1.5 – Скріншот з додатку

б) Water Time – додаток для відстеження режиму пиття води. Допомагає поступово досягнути мети у кількості випитої води. Додаток має систему нагадувань, котрі можна налаштувати відповідно до ваших побажань. Великим плюсом додатку є можливість додавання різного об'єму випитої води. На цьому його функціонал закінчується, на жаль. На рисунку 1.6 наведений

головний екран додатку з певним функціоналом. На рисунку 1.7 наведені можливі варіанти напоїв, котрі пропонує додаток.

Рисунок 1.6 – Скріншот з додатку

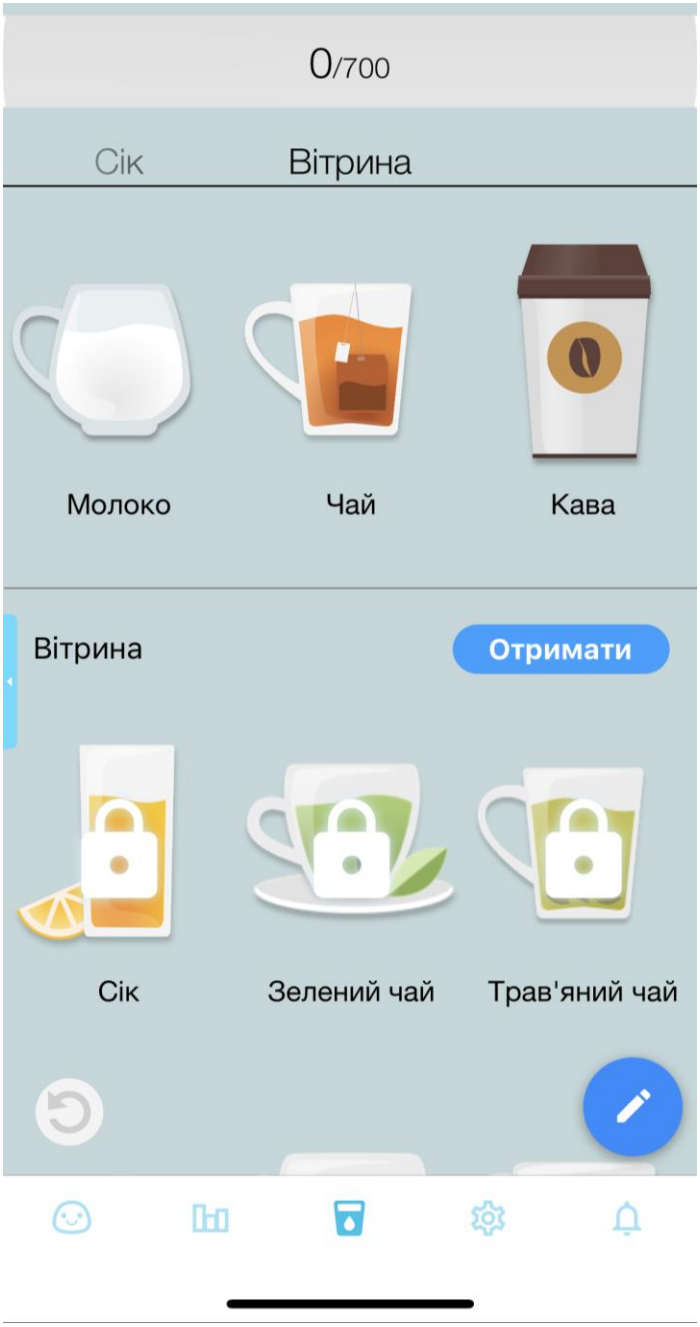


Рисунок 1.7 – Скріншот з додатку

в) Фізична форма 30 днів – додаток для занять спортом. У ньому ви можете обрати якими саме тренуваннями плануєте займатись. Його суть у заняттi спортом щодня, але з невеликою навантаженням. Додаток має систему нагадувань, котрі можна налаштувати відповідно до ваших побажань. Проте даний додаток взагалі не враховує режиму харчування та пиття води, що є не

зручним для користувачів з щільним графіком. На рисунку 1.8 зображений головний екран додатку. На рисунку 1.9 зображено розподіл тренувань.

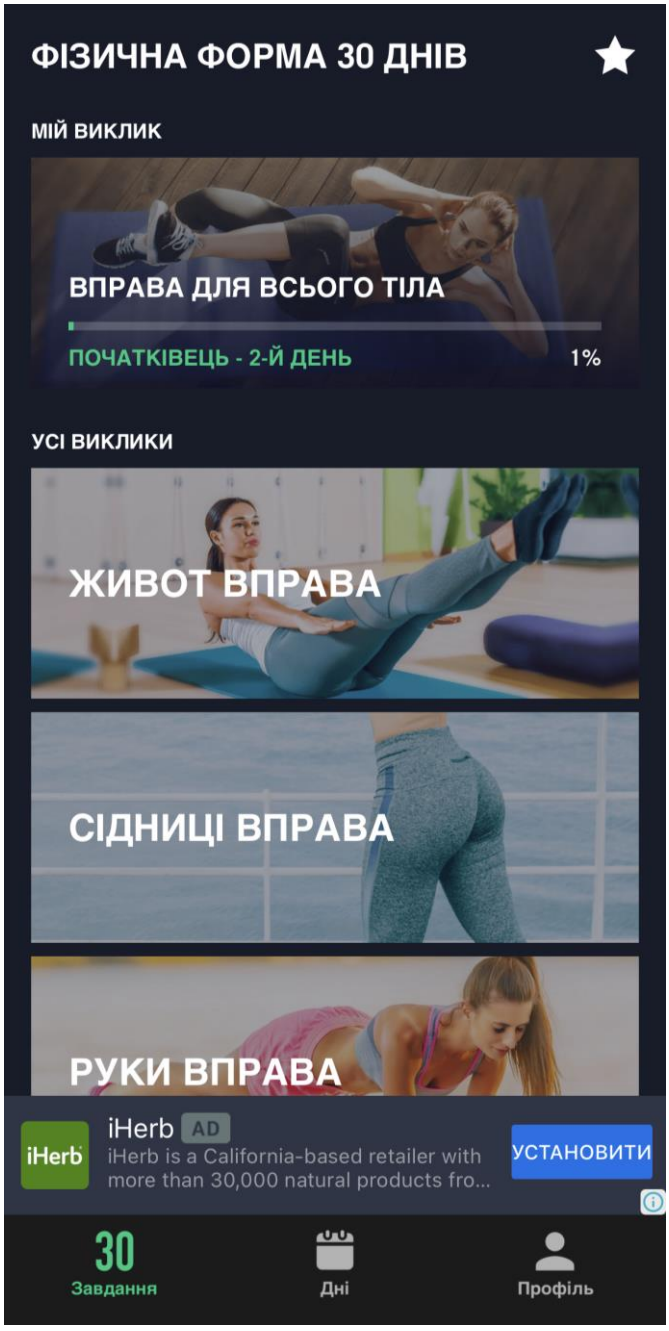


Рисунок 1.8 – Скріншот з додатку

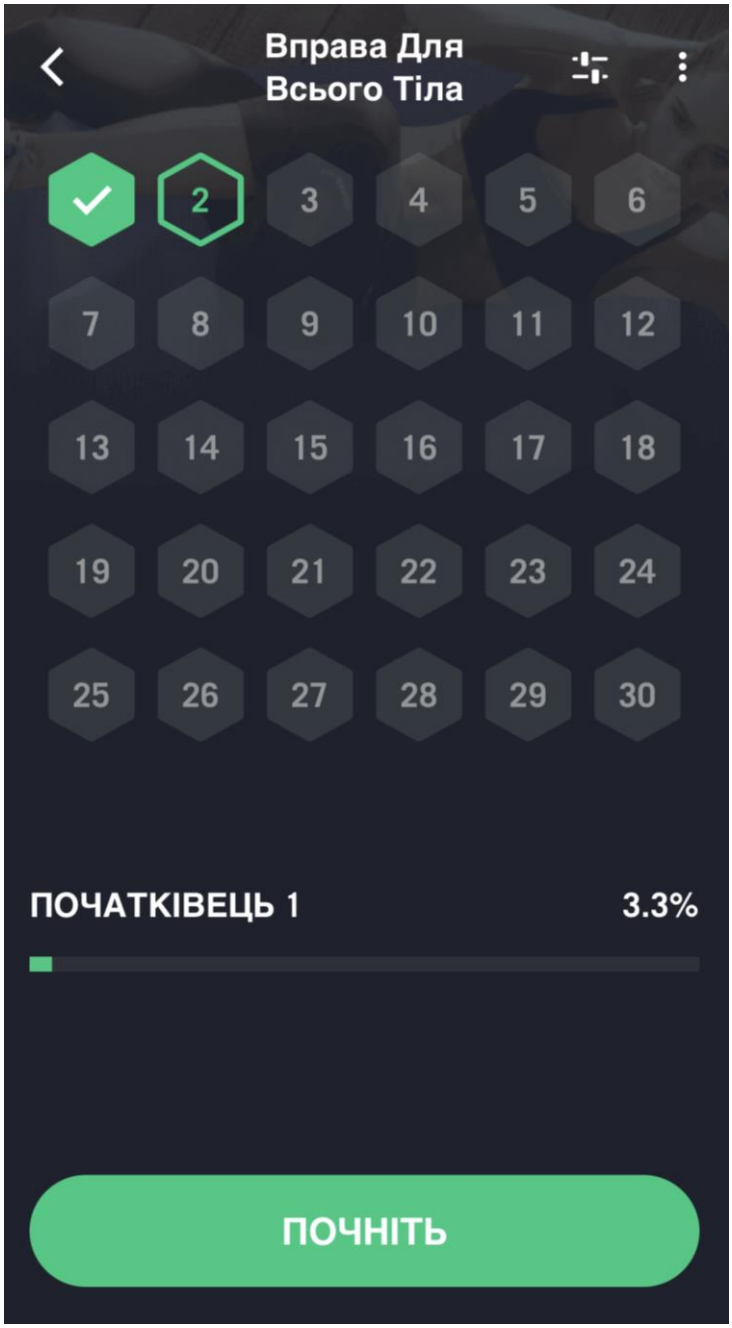


Рисунок 1.9 – Скріншот з додатку

г) Таблиця калорійності – додаток для контролю раціону харчування. У ньому є можливість записувати всі продукти, страви, які були спожиті протягом доби. Функціонал дозволяє розділити ці страви на різні проміжки часу. Додаток має систему нагадувань, котрі можна налаштувати відповідно до ваших побажань. Також є можливість записувати Нотатки. Проте система самостійно не пропонує меню на певний період. Користувач може самостійно

обрати рецепти, за якими хоче харчуватись, але мати це автоматично він не може. На рисунку 1.10 наведений головний екран додатку. На рисунку 1.11 наведений екран відображення спожитого за день.

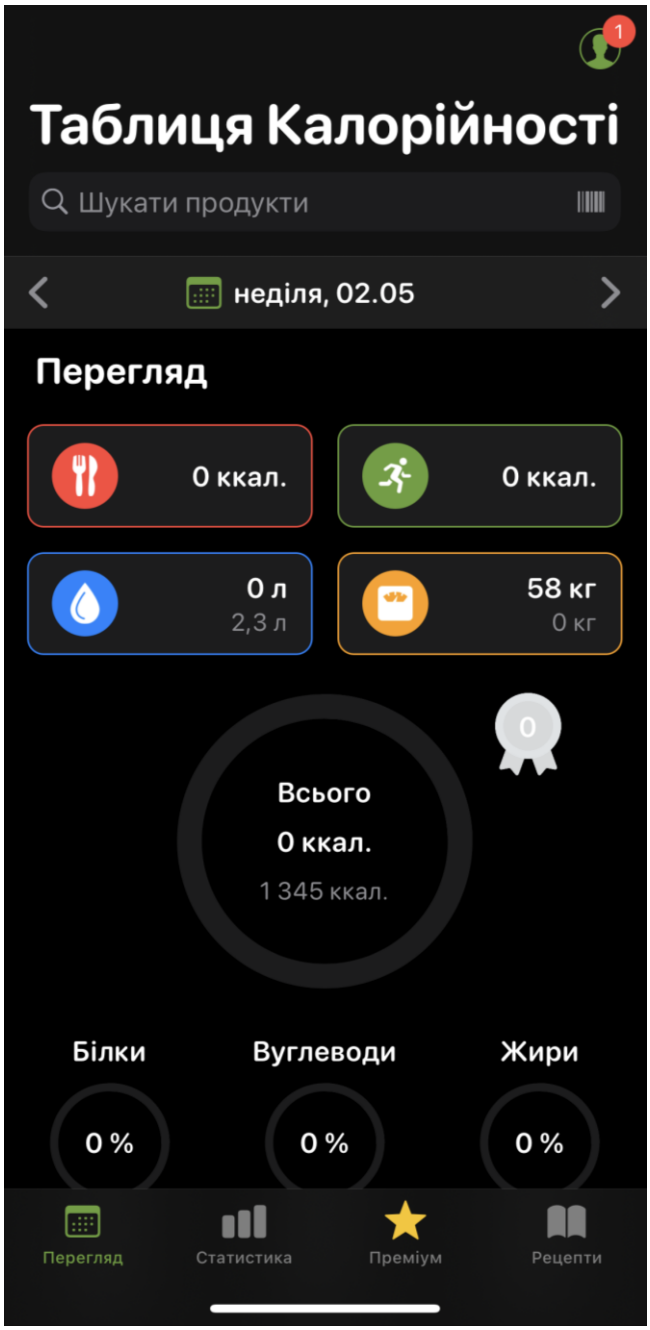


Рисунок 1.10 – Скріншот з додатку

Рисунок 1.11 – Скріншот з додатку

1.4 Аналіз вимог до програмного забезпечення

1.4.1 Функціональні вимоги

Основною і єдиною дійовою особою у додатку є користувач. Об’єктами, над якими виконуються основні операції є нотатник, меню на тиждень, розклад фізичної активності, розклад пиття води, щоденник емоційного здоров’я.

Неавторизовані користувачі – це будь-які відвідувачі додатку, котрі не зареєструвались або не авторизувались у системі.

Користувачі – це відвідувачі додатку, котрі авторизувались у системі.

Нотатник – функціонал, який дозволяє вести записи самопочуття, змін стану організму, побажання і тому подібне.

Меню – запропоноване системою автоматичне меню, яке базується на даних користувачів таких як вік, стать, спосіб життя та їх побажань.

Розклад фізичної активності - запропонований системою набір фізичних вправ, котрі допоможуть підтримувати фізичну активність.

Розклад пиття води – автоматично розрахований і індивідуально налаштований графік та об’єм рідини котру користувач споживає протягом доби.

Щоденник емоційного здоров’я – набір вправ, запитань та порад для покращення свого морального та психоемоційного стану. Розробка індивідуального меню, розкладу фізичних вправ, розкладу пиття води, редагування даних функцій під кожного користувача, аналіз його побажань, стану є основними функціями даного додатку.

Користувач може спілкуватись з системою виставляючи оцінки стравам, вправам та проходячи короткі опитування. Таким чином система буде розуміти індивідуальні потреби користувача та адаптуватись саме під нього.

Як саме користувач може використовувати систему можна побачити на Use Case діаграмі (рисунок 1.12). Також у таблиці 1.1 можна прочитати функціональні вимоги сформовані на основі Use Case діаграми. У таблиці 1.2 наведені варіанти використання системи. Побачити наскільки функціональні вимоги покриті варіантами використання можна на матриці залежності (таблиці 1.3).

Рисунок 1.12 – Модель варіантів використання

Таблиця 1.1 – Функціональні вимоги

№	Назва функції	Код	Призначення функції	Актор	Пріоритет
1.	Вхід у систему	req_1	Авторизація користувачів у системі	Неавторизований користувач	Високий
2.	Зареєструватися	req_2	Реєстрація користувачів у системі	Неавторизований користувач	Високий
3.	Перегляд меню	req_3	Перегляд страв на день користувачем	Користувач	Високий
4.	Редагування меню	req_4	Редагування меню користувачем самостійно	Користувач	Високий
5.	Перегляд фізичних вправ	req_5	Відображення фізичної активності, яка пропонується користувачеві	Користувач	Високий
6.	Перегляд щоденника емоційного здоров’я	req_6	Відображення записів користувача, виконаних вправ та функцій для поліпшення морального здоров’я	Користувач	Високий
7.	Перегляд анкети користувача	req_7	Перегляд користувачем даних про самого себе	Користувач	Високий

Продовження таблиці 1.1

8.	Редагування даних користувача	req_8	Можливість користувача редагувати дані про себе	Користувач	Високий
9.	Проходження вправ у щоденнику емоційного здоров’я	req_9	Проходження користувачем вправ для поліпшення свого емоційного стану	Користувач	Середній
10	Додавання нотаток	req_10	Додавання нотаток користувачем, до певної активності	Користувач	Середній
11	Перегляд розкладу пиття води	req_11	Відображення кількості випитої води та поставленої цілі	Користувач	Високий

Таблиця 1.2 – Варіанти використання

№	Варіант використання	Код	Опис
1.	Індивідуальний підбір харчування	uc_1	Індивідуальний підбір харчування системою для користувача
2.	Індивідуальний підбір фізичних вправ	uc_2	Індивідуальний підбір фізичних вправ для користувача
3.	Індивідуальний підбір режиму пиття води	uc_3	Індивідуальний підбір режиму пиття води для користувача
4.	Контроль над емоційним здоров’ям	uc_4	Користувач може відслідковувати свій емоційний стан

Продовження таблиці 1.2

5.	Оцінювання харчування/режиму пиття/режиму фізичних вправ	uc_5	Оцінювання підібраних системою розкладів та систем
6	Вхід у систему	uc_6	Користувач може ввійти у систему на особисту сторінку

Таблиця 1.3 – Матриця залежності між функціональними вимогами та варіантами використання

	uc_1	uc_2	uc_3	uc_4	uc_5	uc_6
req_1						■
req_2						■
req_3	■				■	
req_4	■				■	
req_5		■			■	
req_6				■	■	
req_7					■	■
req_8						■
req_9				■	■	
req_10	■	■	■	■	■	
req_11			■		■	

1.4.2 Розроблення нефункціональних вимог

Нефункціональними вимогами для даного мобільного застосунку є

– інтуїтивний візуальний інтерфейс;

– стійкість до відмов на високому рівні;

– висока швидкість завантаження та видачі результатів запитів;
– наочне відображення помилок.

Вони повинні забезпечити найкращий досвід користування програмою користувачами.

1.4.3 Постановка комплексу завдань модулю

1.4.3.1 Призначення розробки

Дана розробка призначена для підбору індивідуального способу харчування, комплексу вправ, режиму пиття води та контролю над психоемоційним станом особи. Також вона надає можливість у будь-який момент часу відредагувати налаштування індивідуально, що дасть сигнал системі про зміну алгоритму підбору вище зазначених режимів.

1.4.3.2 Задачі та цілі розробки

Метою даної розробки є розробка мобільного застосунку для легшого способу досягнення здорового способу життя, за допомогою різного функціоналу. А саме автоматично підібраного режиму та раціону харчування, комплексу фізичних вправ, автоматично підібраного режиму пиття води та контролю за психоемоційним станом людини. Також дана розробка надає можливість зайнятим людям не відволікатись від нагальних справ, бо додаток включає у себе систему нотифікацій, які можна налаштувати індивідуально.

Задачею розробки є створення мобільного додатку для ведення здорового способу життя, який містить у собі:

– індивідуальний підбір харчування;
– індивідуальний підбір фізичних вправ;
– індивідуальний підбір режиму пиття води;
– контроль над емоційним здоров’ям;
– оцінка автоматично підібраних режимів та подальше коригування індивідуально.

1.5 Висновки по розділу

Виходячи з усього вище описаного, можна дійти висновку, що предметна область даної розробки потребує оптимізації та покращення. На ринку вже існують додатки, що певною мірою допомагають розв'язувати питання здорового способу життя, проте їх функціонал є обмеженим і вони не мають інтеграції одне з одним. З цього можна дійти висновку, що спроба об’єднати складові в одному додатку принесе користь суспільству та зробить користувачів на кілька кроків ближче до здорового життя.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

У цьому розділі розглянемо варіанти поведінки користувачів у системі, опишемо детально кожен сценарій.

Основним бізнес-процесом даного мобільного застосунку є контроль за раціоном харчування. Алгоритм за яким це буде відбуватись зображений на рисунку 2.1.

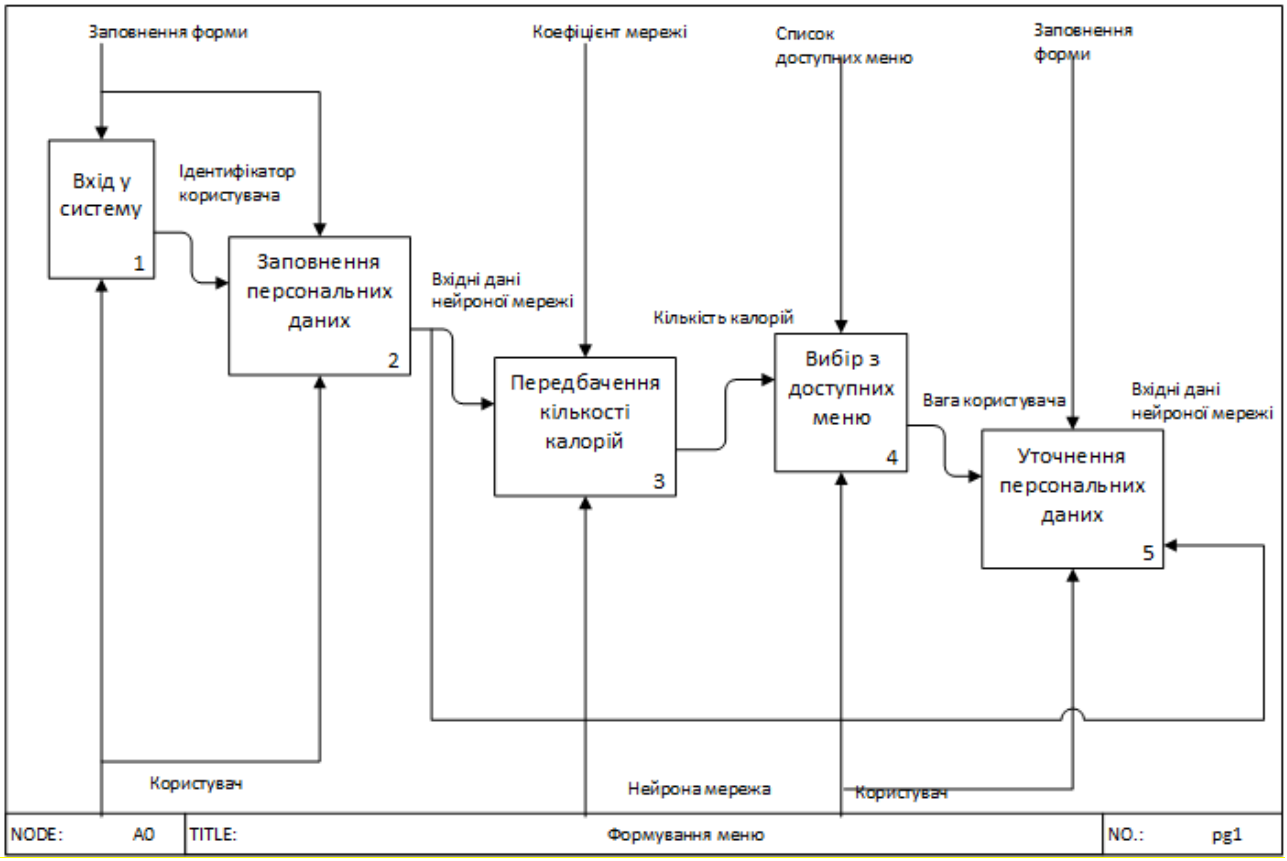


Рисунок 2.1 – Схема структурна бізнес-процесу

а) першим кроком є заповнення форми користувачем. Після цього автоматично створюється ідентифікатор користувача та він є залогіненим у систему;

б) наступним кроком система очікуватиме від користувача введення таких даних як вік, стать, стиль життя та вага. Це необхідно для розрахунку раціону харчування;

в) далі система самостійно обробляє ці дані і робить прогноз для кількості калорій за допомогою машинного навчання;

г) після цього користувачу будуть запропоновані маню на вибір і він зможе обрати де яке до вподоби;

д) якщо протягом знаходження на меню користувачу не будуть подобатись підібрані варіанти страв, він може прибрати їх з меню вказавши причину і потім система запропонує користувачу нову страву.

У даному додатку є менш суттєві процеси:

– реєстрація користувача відбувається за допомогою звичайної реєстраційної форми;

– кожен користувач може змінювати вподобання прямо на сторінці меню, не заповнюючи додаткові форми.

2.2 Архітектура програмного забезпечення

Дане програмне забезпечення складається з декількох основних модулів, котрі наведені на рисунку 2.2.

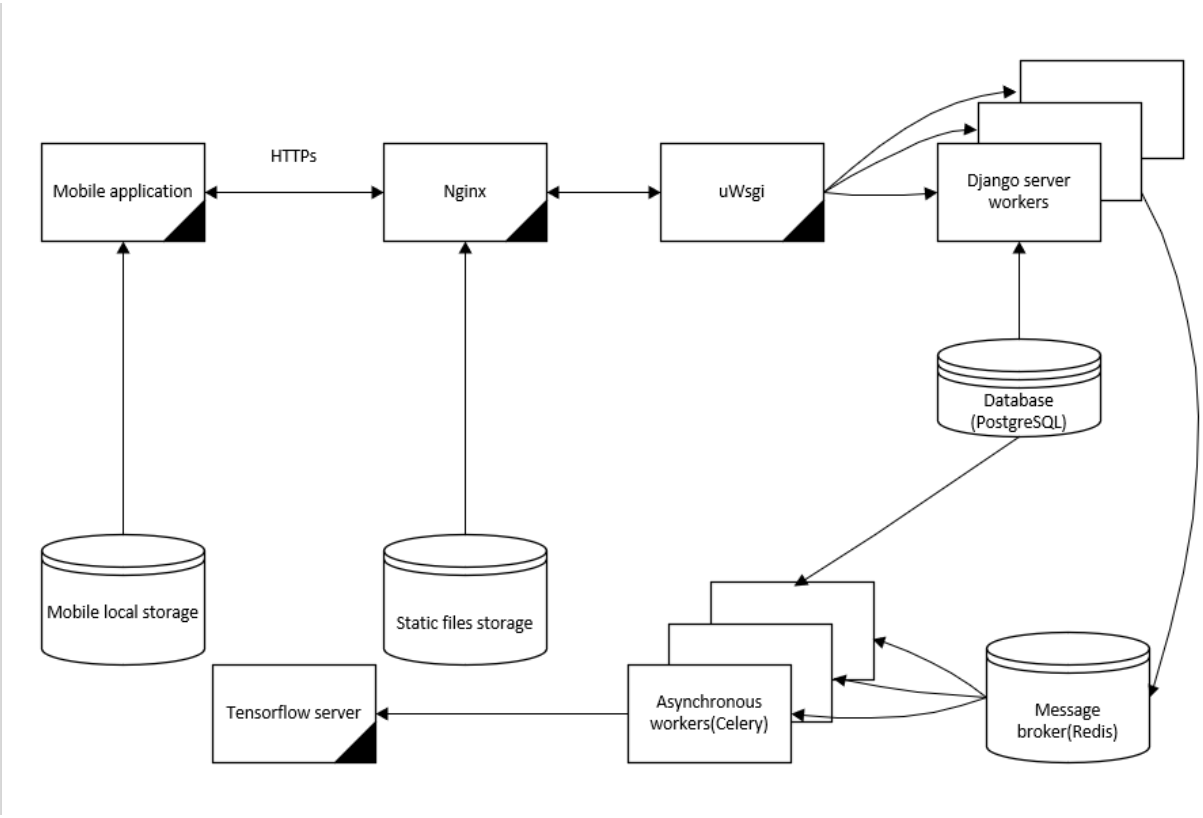


Рисунок 2.2 – Архітектурна схема програмного забезпечення

Розглянемо взаємодію вище наведеної схеми. У таблиці 2.1 наведений опис призначення кожного з наявних модулів програмного забезпечення.

Таблиця 2.1 – Опис модулів програмного забезпечення

Модуль	**Призначення**
Mobile application	Надання візуального інтерфейсу взаємодії з програмою для клієнта.
Nginx	Розповсюдження статичного контенту (зображення, текстові файли тощо).
uWsgi	Сервер для перенаправлення вхідних запитів декільком процесам на основному сервері.

Продовження таблиці 2.1

Django server worker	Процес основного серверу, що відповідає за обробку запитів клієнта та взаємодію з базою даних. У ньому міститься вся бізнес логіка застосунку.
Database	Реляційна база даних, призначена для структурованого збереження даних застосунку.
Message broker	Сервер, що відповідає за прийом асинхронних повідомлень постановки їх у чергу та передачу відповідним процесам на виконання.
Asynchronous workers	Процеси для виконання асинхронних завдань від основного серверу, що не можуть бути виконані у процесі обробки синхронного запиту користувача.
Tensorflow server	Сервер на якому запущена модель машинного навчання, який приймає на вхід повідомлення, для яких має бути виконане передбачення, і передаються дані відправнику.
Mobile local storage	Програмний інтерфейс для доступу до файлового сховища мобільного пристрою.
Static file storage	Файлове сховище для статичного контенту.

Алгоритм взаємодії модулів між собою наступний: користувач заходить у мобільний застосунок у якому використовуються дані з мобільного пристрою для визначення чи є користувач авторизованим у системі чи ні. Далі мобільний застосунок робить HTTP запит на Nginx сервер за допомогою HTTPs протоколу, що забезпечує безпеку даних, які надсилаються. На цьому етапі Nginx сервер визначає які саме дані хоче отримати клієнт, якщо клієнт хоче отримати статичні дані, то він віддає їх з відповідного файлового сховища, якщо клієнт хоче отримати дані, оброблені основним сервером то запит передається далі, а саме на uWsgi сервер. На цьому сервері дані переформатовуються у бінарний формат, сервер вирішує який з процесів основного сервера вільний і передає дані на цей процес. Всі процеси основного сервера мають доступ до бази даних. Якщо обробка запиту клієнта може бути виконана за короткий час, то вона здійснюється синхронно у окремому потоці і результат повертається клієнту у зворотному напрямку. Якщо запит клієнта передбачає взаємодію з моделлю машинного навчання, то процес основного сервера запаковує необхідні для моделі машинного навчання дані в асинхронне завдання. Дане завдання надсилається брокеру повідомлень, де воно стає у чергу на виконання. Як тільки один з процесів для виконання цих завдань звільняється, як і я після диплому, йому на виконання потрапляє перше асинхронне завдання з черги. Оскільки дані процеси виконуються за допомогою CPU, а для роботи моделі машинного навчання потрібен GPU і можливість працювати з CUDA, то сама модель запускається на окремому сервері Tensorflow. Взаємодія вищезазначеного сервера із процесами для виконання асинхронних завдань відбувається шляхом синхронних HTTP запитів.

2.2.1 База даних.

Для того аби розробити базу даних для цього додатку було використано СКБД PostgreSQL 10. На рисунку 2.3 можна побачити детальну структуру бази даних, яка була використана у даному проєкті. Короткий опис бази даних знаходиться у таблиці 2.2.

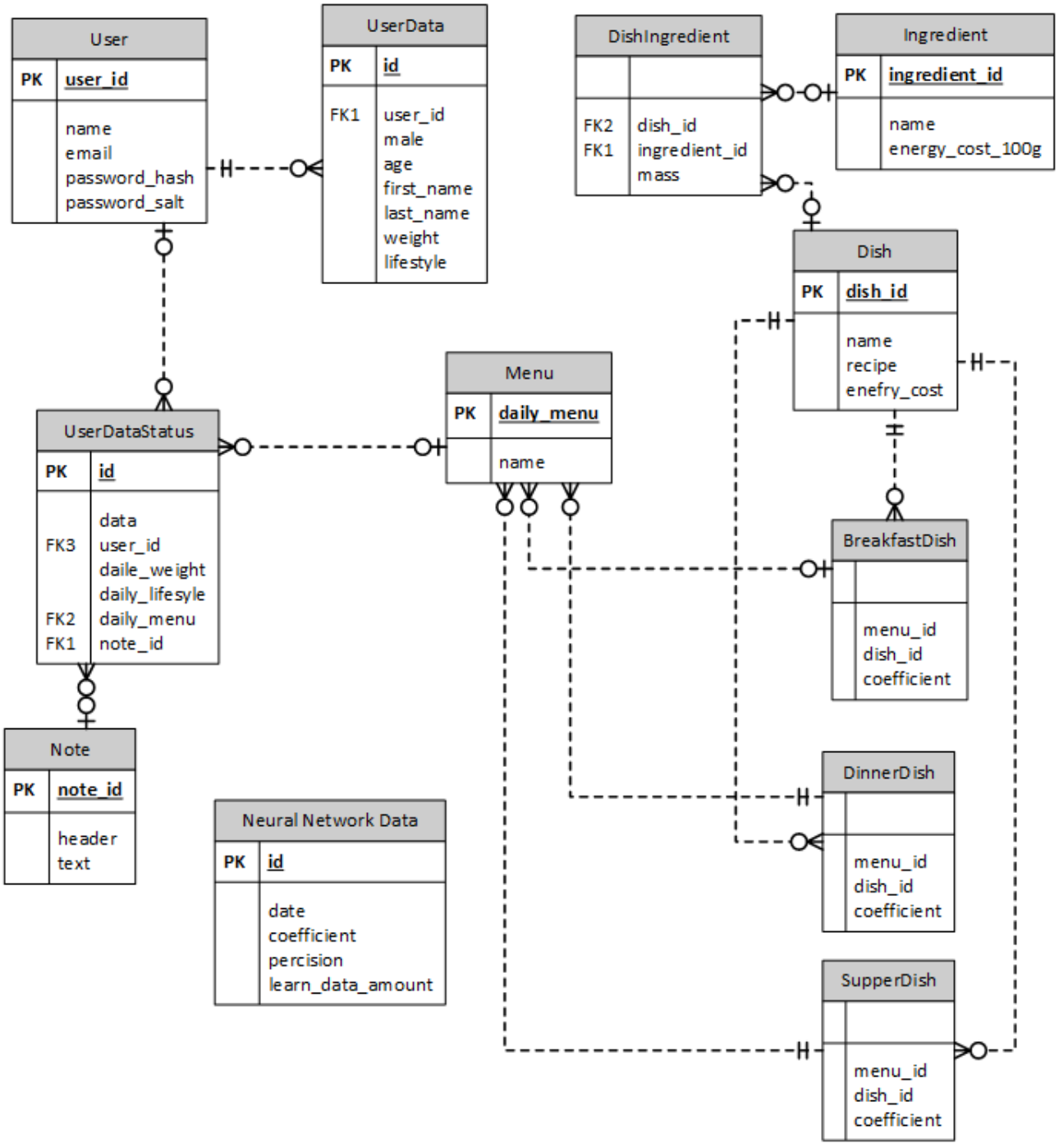


Рисунок 2.3 – Схема бази даних

Таблиця 2.2 – Короткий опис таблиць БД

Назва таблиці	Опис таблиці
User	Дані для логіну користувачів у системі
UserData	Таблиця з даними користувачів такі як стать, вага, вік, стиль життя, ім'я та прізвище
UserdDataStatus	Таблиця для зберігання щоденних даних користувачів
Note	Таблиця для зберігань нотаток користувача
Menu	Таблиця для зберігання щоденного меню користувача
BreakfastDish	Таблиця з усіма можливими сніданками
DinnerDish	Таблиця з усіма можливими обідами
SupperDiah	Таблиця з усіма можливими вечерями
Dish	Таблиця з усіма стравами та їх рецептами
Ingredient	Таблиця всіх інгредієнтів які використовуються користувачем
DishIngredient	Таблиця інгредієнтів по стравам
NeuralNetworkData	Таблиця для зберігання даних нейронної системи

Далі йде почерговий опис кожної таблиці у базі даних.

Таблиця 2.3 – Службові поля БД (наявні у кожній таблиці)

Поле	Тип	Призначення
*_id	UUID	Унікальний ідентифікатор запису в таблиці, *_ позначає що спочатку може йти додатковий опис поля

Таблиця 2.4 – Опис таблиці User

Поле	Тип	Призначення
name	Varchar(100)	Ім’я користувача
email	Varchar(100)	Логін, який користувач буде використовувати для входу у систему
password_hash	Varchar(255)	Хеш від паролю
password_salt	Varchar(255)	Сіль до паролю

Таблиця 2.5 – Опис таблиці UserData

Поле	Тип	Призначення
user_id	UUID	Зовнішній ключ до таблиці USER
male	Varchar(100)	Стать користувача
age	INT	Сік користувача
first_name	Varchar(255)	Ім’я користувача
last_name	Varchar(255)	Прізвище користувача
weight	INT	Вага користувача
lifestyle	Varchar(255)	Стиль життя користувача

Таблиця 2.6 – Опис таблиці UserDataStatus

Поле	Тип	Призначення
date	DATE	Дата запису
user_id	UUID	Зовнішній ключ до таблиці USER
daily_weight	INT	Денна вага користувача
daile_lifestyle		Денний стиль життя користувача
daily_menu	INT	Денне меню користувача
note_id	UUID	зовнішній ключ до таблиці NOTE

Таблиця 2.7 – Опис таблиці Note

Поле	Тип	Призначення
header	Varchar(255)	Заголовок нотатки
text	Varchar(255)	Текст нотатки

Таблиця 2.8 – Опис таблиці Menu

Поле	Тип	Призначення
name	Varchar(255)	Назва меню
daily_menu		Первинний ключ таблиці

Таблиця 2.9 – Опис таблиці DishIngredient

Поле	Тип	Призначення
dish_id	UUID	Зовнішній ключ на таблицю DISH

Продовження таблиці 2.9

ingredient_id	UUID	Зовнішній ключ на таблицю Ingredient
mass	INT	вага продукту для цієї страви

Таблиця 2.10 – Опис таблиці Ingredient

Поле	**Тип**	**Призначення**
ingredient_id	UUID	Первинний ключ таблиці
name	Varchar(100)	Назва продукту
energy_cost_100g	INT	Енергетична вага на 100 гр

Таблиця 2.11 – Опис таблиці Dish

Поле	**Тип**	**Призначення**
dish_id	UUID	Первинний ключ таблиці
name	Varchar(100)	Назва страви
recipe	Varchar(255)	Рецепт страви
energy_cost	INT	Енергетична цінність страви

Таблиця 2.12 – Опис таблиці BreakfastDish

Поле	**Тип**	**Призначення**
menu_id	UUID	Зовнішній ключ на таблицю Menu

Продовження таблиці 2.12

dish_id	UUID	Зовнішній ключ на таблицю Dish
coefficient	INT	Коефіцієнт для обрахунку калорій для користувача

Таблиця 2.13 – Опис таблиці SupperDish

menu_id	UUID	Зовнішній ключ на таблицю Menu
dish_id	UUID	Зовнішній ключ на таблицю Dish
coefficient	INT	Коефіцієнт для обрахунку калорій для користувача

Таблиця 2.14 – Опис таблиці DinnerDish

menu_id	UUID	Зовнішній ключ на таблицю Menu
dish_id	UUID	Зовнішній ключ на таблицю Dish
coefficient	INT	Коефіцієнт для обрахунку калорій для користувача

Таблиця 2.15 – Опис таблиці NeuralNetworkData

Поле	Тип	Призначення
id	UUID	Унікальний ідентифікатор запису
date	DATE	Дата запису
coefficients	FILE	Модель нейронної мережі
precision	FLOAT	Точність моделі нейронної мережі
learn_data_amount	INT	Кількість даних на основі яких була навчена модель

2.2.2 Серверна частина.

Серверна частина написана за архітектурою REST API. Варто пам’ятати, що більшість архітектурних стилів відповідає певним архітектурним обмеженням. Нище опишемо які саме обмеження має REST API:

а) Розділення відповідальності між компонентами. Це означає що компоненти можуть еволюціонувати незалежно, адже сервер і клієнт чітко розділені;

б) Відсутність стану. Вся інформація, яка необхідна для обробки запиту знаходить безпосередньо у записі. Тобто сервер не знає про стан клієнтів і не має запам’ятовувати послідовність запитів;

в) Кешування. Дані, які передаються сервером повинні мати інформацію чи можна їх кешувати і як довго;

г) Шари абстракції. Дане обмеження дозволяє зменшити складність компонентів.

Так як весь додаток складається з модулів, було поставлено задачу зробити їх максимально ізольованими одне від одного для того аби мати змогу використовувати їх на інших проєктах. Також такий підхід зменшує ризики, що при зміні одного модуля можливі змінити у іншому. Тобто шанс вивести систему з ладу, зміною одного єдиного модуля менший.

У таблиці 2.16 наведений опис модулів створеної системи.

Таблиця 2.16 – Модулі серверної частини ПЗ

Модуль	**Опис**
Auth	Даний модуль відповідає за реєстрацію, авторизацію, вихід з системи користувача. Також у список задач цього модуля входить аутентифікація запитів клієнта.
Menus	У задачі даного модуля входять робота з даними меню користувача, додавання цих даних та збереження налаштувань побажань користувача відносно змісту меню.
Stats	Даний модуль відповідає за формування статистики для користувача базуючись на спожитих користувачем калорій.
Users	Даний модуль відповідає за зміну та збереження персональних даних користувача на базі яких виконується машинне навчання.
Utils	Сервісний модуль призначений для збереження функцій та класів, що використовуються іншими модулями застосунку.

Продовження таблиці 2.16

Django_backend	Головний модуль серверного застосунку. Відповідає за налаштування застосунку, навігацію запитів користувача між іншими модулями.

2.2.3 Мобільний застосунок

Для розробки мобільного застосунку був використаний фреймворк React Native. Він дозволяє компілювати код написаний мовою Javascript у нативний код Android або iOS застосунку. Цей підхід дозволяє розробляти код для цих двох платформ одночасно, тим самим забезпечуючи швидший процес розробки програмного забезпечення та можливість реалізації на двох платформах одночасно. Дане програмне забезпечення експлуатує патерн проєктування MVC (model-view-controller). Як видно з рисунку 2.4 даний патерн передбачає у собі три складові системи: модель даних, модель керування та презентація даних.

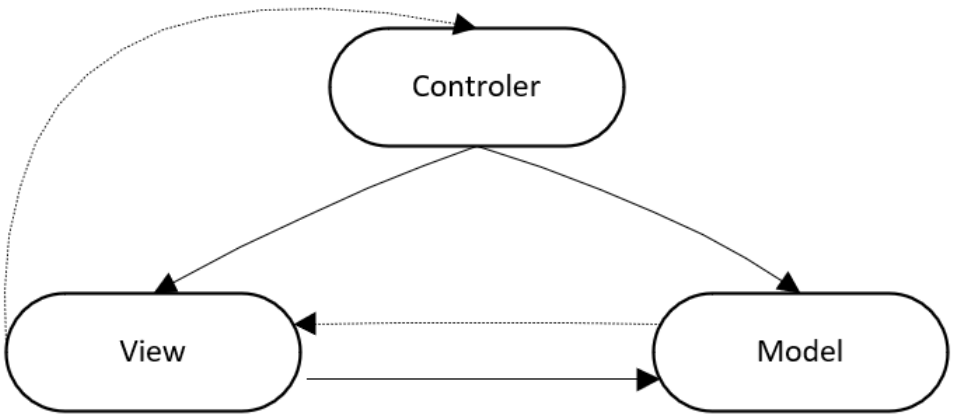


Рисунок 2.4 – Діаграма взаємодії між компонентами патерна

Такий підхід дозволяє відокремити користувацький інтерфейс від даних моделі, що дозволяє при змінах інтерфейсу користувача мінімально впливати на модель даних, у той час як зміни у моделі даних будуть мати також мінімальний вплив на інтерфейс користувача.

Тобто основні задачі кожного компонента можна описати як наступні.

а) Модель даних – дані, методи роботи з ними. Також у функції цієї складової входить реагування та адаптація під запити.

б) Презентація даних – наочне відображення інформації, візуалізація її.

в) Модель керування – забезпечення зв’язку між користувачем та системою, саме тут відбувається контроль запитів та даних користувача.

У даному застосунку модель даних представлена у вигляді класів, поля який повторюють поля відповідних таблиць у базі даних, та методів що отримуються ці дані з серверного застосунку у форматі JSON та форматують їх, ініціалізують ними дані класи.

Модель керування даних представляє собою набір методів що реалізуються перетворення даних на шляху від моделі до презентації даних. До даних перетворень входять валідація даних та їх форматування для правильного відображення у візуальній частину клієнтського застосунку.

До шару презентації даних входять класи, що описують зовнішній вигляд візуальних елементів, що відображаються на екрані.

Аналогічно до серверної частина дана частина має модульну структуру. Для кращого розуміння роботи наведено таблицю 2.17 з описів модулів та їх призначеннями.

Таблиця 2.17 - Опис модулів клієнтської частини застосунку

Модуль	**Опис**
Components	Даний модуль являє собою набір універсальних графічних компонент, які використовуються для різних графічних представлень мобільного застосунку.
Core	Набір універсальних інструментів для використання іншими модулями серед, яких є функції валідатори та адаптери даних для графічної частини.
Models	Набір класів призначених для роботи з даними отриманих з серверної частини.
Screens	Класи, що описують графічне представлення сторінок інтерфейсу мобільного застосунку.
Services	Це Singleton-класи які містять набір функцій призначених для обробки даних моделей, інтерфейс доступу до ресурсів серверної частини. Також вони зберігають всі проміжні дані отримані з серверної частини.

2.3 Конструювання програмного забезпечення

Далі будуть наведені особливості конструювання кожного з модулів програмного забезпечення.

2.3.1 Серверна частина

Для серверної частини було розроблене API що має структуру описану в таблиці 2.18.

Таблиця 2.18 – Структура API

Адреса API	Вид запиту	Призначення
/auth/login	POST	Авторизація користувача у системі.
/auth/login/refresh	POST	Повторна авторизація користувача у системі у випадку коли його JWT токен вичерпав свій термін дії.
/auth/register	POST	Додавання нового користувача до системи.
/auth/change_password	POST	Зміна пароля авторизованим користувачем.
/auth/profile	GET	Запит на отримання даних профіля користувача.
/auth/update_profile	PUT	Оновлення даних профілю користувача.

Продовження таблиці 2.18

/auth/logout	POST	Запит для виходу користувача з системи.
/user/data	GET/POST	Додавання та редагування персональних даних користувача на основі яких здійснюється машинне навчання.
/user/data/today	GET/POST	Додавання та перегляд показників користувача, які мали місце бути на сьогоднішній день.
/user/data/history	GET	Перегляд історії змін показників користувача.
/user/data/statistics	GET	Отримання статистики зі змін показників користувача.
/predict	POST	Запит на виконання нейронною мережею передбачення очікуваної кількості калорій рекомендованої до споживання.

Продовження таблиці 2.18

/choose_menu	POST	Вибір персонального меню зі списку обраних та його адаптація під результати передбачення нейронної мережі.
/user/preferences	POST	Встановлення побажань відносно складових меню, що пропонуються користувачу.
/menus	GET	Отримання списку меню.

Дані API ресурси були реалізовані у даному проєкті із використанням бібліотеки Django REST Framework.

2.3.2 Клієнтська частина

Як зазначалось раніше написання клієнтської частини застосунку здійснювалось з застосуванням патерну MVC. Далі будуть наведені особливості конструювання кожного шару даної архітектури (презентація даних, контролер та модель).

Кожен клас шару презентації даних відповідає окремому екрану мобільного застосунку. Повний перелік цих класів наведений у таблиці 2.19.

Таблиця 2.19 – Опис класів шару презентації даних

Назва класу	**Опис**
HomeScreen	Перше вікно що бачить користувач. На ньому зображуються елементи управління авторизаціє користувача.

Продовження таблиці 2.19

LoginScreen	Вікно для введення логіну та паролю користувачем перед авторизацією.
RegisterScreen	Вікно додавання даних нового користувача.
ForgotPasswordScreen	Вікно відновлення паролю.
DashBoardScreen	Вікно для відображення основних показників користувача та його поточного обраного меню.
ProcessingScreen	Вікно завантаження додатку. З’являється коли клієнт очікує на відповідь від сервера у процесі машинного навчання.
EditMenuScreen	Вікно для редагування та вибору персонального меню.
StatisticScreen	Вікно для відображення статистики споживання калорій користувача за останні 30 днів.
QuestionScreen	Вікно з запитаннями що дозволяє виявити побажання користувача відносно вибору меню.
EditDataScreen	Вікно для додавання та редагування персональних даних користувача.

До шару контролеру даних належать класи описані в таблиці 2.20

Таблиця 2.20 – Опис класів шару контролеру даних

Назва класу	Опис
APIService	Singleton-клас, що призначений для взаємодії з основними ресурсами API серверу. Він надає інтерфейс для доступу до кожного з ресурсів.
AuthService	Singleton-клас, що зберігає дані необхідні для авторизації користувача у системі (JWT токен та токен для відновлення попереднього токену). Надає інтерфейс для здійснення всіх дій пов’язаних з авторизацією користувача.
UserService	Singleton-клас, який містить методи для зміни та додавання персональних даних користувача, а також тимчасово зберігає поточний стан цих даних.
MenuService	Singleton-клас, що надає інтерфейс для вибору персонального меню та налаштування побажань до списку меню, що пропонується.
StatisticService	Singleton-клас, який виконує обробку даних для подальшого відображення у статистиці користувача.
Validators	Статичний клас, що містить методи для валідації полів даних що заповнюються користувачем.

Продовження таблиці 2.20

Storage	Клас що надає інтерфейс для роботи з файловими сховища мобільного пристрою. Використовується для тимчасового зберігання сесії авторизованого користувача.

До шару моделі даних належать класи адаптери, що перетворюють JSON дані отримані від серверної частини в об’єкти мови Javascript, які зберігаються і обробляються іншими шарами мобільного застосунку.

2.3.1 Налаштування Nginx серверу

Nginx сервер був налаштований таким чином щоб віддавати статичні дані напряму з дискового сховища, а решту запитів передавати на основний сервер. Це дозволяє зменшити навантаження на даний сервер, а також у майбутньому змінювати джерело з якого видається даний контент без необхідності вносити зміни у основний сервер застосунку. Конфігурація Nginx сервера наведена на рисунку 2.5.

```
server {
    listen 80;
    server_name localhost;

    gzip on;
    gzip_proxied expired no-cache no-store private auth;
    gzip_types text/plain text/css text/xml text/javascript application/x-javascript application/json;

    client_max_body_size 700m;
    proxy_ignore_client_abort on;

    location ~ ^/static/ {
        root /app/;
        autoindex on;

        access_log off;
    }

    location ~ ^/(auth|user|predict|choose_menu|menus)/ {
        uwsgi_pass      django:3001;

        uwsgi_param     QUERY_STRING       $query_string;
        uwsgi_param     REQUEST_METHOD     $request_method;
        uwsgi_param     CONTENT_TYPE       $content_type;
        uwsgi_param     CONTENT_LENGTH     $content_length;

        uwsgi_param     REQUEST_URI        $request_uri;
        uwsgi_param     PATH_INFO          $document_uri;
        uwsgi_param     DOCUMENT_ROOT      $document_root;
        uwsgi_param     SERVER_PROTOCOL    $server_protocol;
        uwsgi_param     REQUEST_SCHEME     $scheme;
        uwsgi_param     HTTPS              $https if_not_empty;

        uwsgi_param     REMOTE_ADDR        $remote_addr;
        uwsgi_param     REMOTE_PORT        $remote_port;
        uwsgi_param     SERVER_PORT        $server_port;
        uwsgi_param     SERVER_NAME        $server_name;

        uwsgi_read_timeout 900s;
        uwsgi_send_timeout 900s;
    }
}
```

Рисунок 2.5 - Конфігурація Nginx серверу

2.3.2 Налаштування uWsgi серверу

uWsgi сервер був налаштований таким чином щоб обслуговувати одночасно п’ять процесів модуля основного сервера. Таким чином ми досягаємо зменшення вірогідності впливу відмови одного з процесів або неочікуваного його припинення на роботу всього додатку.

Скріншот файлу конфігурації uWsgi серверу зображений на рисунку 2.6.

```
[uwsgi]
chdir=/app/src
protocol=uwsgi
module=django_backend.wsgi:application
max-requests=500
master=True
socket=127.0.0.1:3001
processes=5
pidfile=/tmp/django_backend.pid
vacuum=True
daemonize=/app/logs/uwsgi.log
post-buffering=1
env = DJANGO_SETTINGS_MODULE=django_backend.settings
```

Рисунок 2.6 - файл конфігурації uWsgi серверу

2.3.3 Налаштування процесів для виконання асинхронних завдань

Для виконання асинхронних завдань в даному проєкті була застосована бібліотека Celery. Для неї були написані функції для асинхронного виконання завдань з передбачення нейронною мережею рекомендованого споживаного калоражу користувача. Результати виконання цих завдань зберігаються у тій самі базі даних яка використовується сервером. Асинхронні завдання надсилаються на виконання Celery спеціальним брокером повідомлень, що відслідковує доступність кожного окремо взятого процесу і за необхідності ставить ці завдання в FIFO чергу. У якості брокеру повідомлень було обрана бібліотека Redix.

2.3.4 Побудова та навчання моделі нейронної мережі

Оскільки побудова нейронної мережі з нуля є процесом доволі трудомістким і потребує тривалого часу для його налагодження, то для реалізації нейронної мережі була обрана платформа для машинного навчання Tensorflow. Вона містить у собі готові реалізації найбільш поширених шарів нейронних мереж, методів оптимізації та оцінки тощо.

Нейронна мережа має перетворювати вхідні параметри статі, віку, ваги та способу життя користувача у єдине дійсне число - рекомендовану до

споживання кількість калорій на день. Тобто, треба вирішити задачу регресії, одного з видів навчання з учителем. В якості вхідних даних до нейронної мережі маємо набір ознак об’єкту (персональні дані користувача).

Взаємозв’язок перелічених вище ознак із очікуваними результатами, який треба знати для навчання з учителем, був взятий із дослідження під назвою «Human energy requirements: Report of a Joint FAO/WHO/UNU Expert Consultation» [7]. В ньому дослідники вивели залежність показників людини від кількості калорій, необхідної їй для споживання.

Модель нейронної мережі складається з 4-х послідовних шарів (рисунок 2.7).

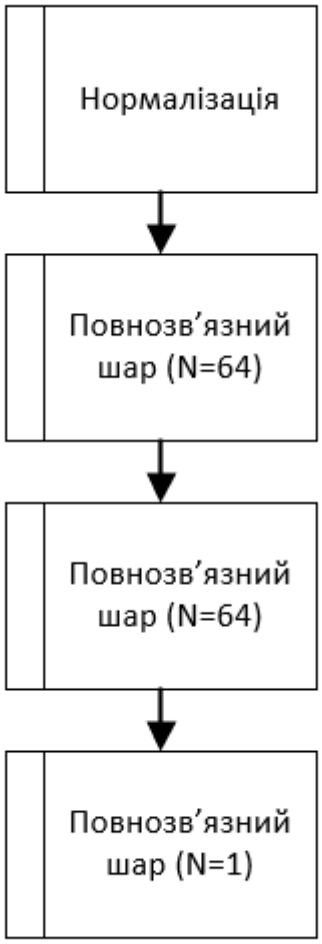


Рисунок 2.7 - Шари нейронної мережі

Шар нормалізації необхідний для того щоб вирівняти діапазони значень вхідних параметрів. Це робиться для того, аби вони мали еквівалентний вплив на результат регресії. Два наступні шари є повнозв'язними шарами, які виконують роль моделювання складної нелінійної функції, що перетворює матрицю вхідних параметрів. Для них використовується функція активації relu, яка просто відкидає негативні значення. Задачею останнього шару є зменшення розмірності матриці вхідних параметрів до єдиного числа - кількості калорій.

Для навчання цієї моделі вхідні дані були розділені у співвідношенні 80 до 20 на дані для тренування моделі і дані для її тестування. Тренована модель зберігається на сервері у вигляді бінарного файлу.

2.3.5 Розгортання моделі нейронної мережі

Для розгортання моделі нейронної мережі була використана бібліотека Tensorflow Serving, що дозволяє надсилати дані цій моделі у вигляді REST-запитів та синхронно отримувати дані регресії чи класифікації у відповідь. Також вона має вбудовану підтримку CUDA, тобто здатна запускатись на відеокартах, що суттєво пришвидшує роботу моделі. Даний факт є вирішальним у виборі технології, оскільки є дуже важливим у роботі навантаженого додатку.

2.4 Безпека даних

У сучасному світі безпека даних є однією з ключових вимог до будь-якого додатку, особливо коли говориться про здоров'я. Було використано фреймворк Django, у ньому вже є певні механізми захисту даних і це було однією з причин обрання саме цієї технології.

Паролі є одним з слабким місць користувачі, адже багато використовує один і той самий на багатьох ресурсах. Тому для зберігання паролів був використаний вбудований Django алгоритм PBKDF2 з хешем SHA256. Рішення не обирати інший алгоритм та хеш ще обумовлений рекомендацією саме цього

методу Національним інститутом стандартів і технології (NIST). PBKDF2 перетворює будь-які дані які є у паролі на ключ, що є псевдовипадковим. Такий метод забезпечує великий час зворотного підбору і робить його можливим при повному переборі всіх можливих комбінацій паролів.

Аутентифікацію користувач проходить вже через JWT-токен-ключа, що має обмеження по часу для кожного користувача. Цей ключ буде передаватись користувачем на сервер під час сесії. У іншому випадку користувач не зможе ввійти у систему, код помилки буде 401 Unauthorized.

Захист від найбільш розповсюджений атак таких як: SQL-ін’єкції, SSL/HTTPS та HOST вже реалізована у фреймворку Django.

2.5 Висновки по розділу

Цей розділ був присвячений описанню процесу моделювання програмного забезпечення, розробці архітектури та нюансам реалізації. Приділялась увага також безпеці даних у мобільний застосунка з клієнт-серверною архітектурою.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Протягом розробки програмного забезпечення дефекти, баги, помилки виникають на кожній стадії від написання документації до підтримки. Вони можуть бути різного виду, важкості та пріоритету, але причини виникнення варіюватись настільки не будуть.

Основні причини виникнення дефектів:

- не правильна реалізація ПЗ;
- експлуатація ПЗ невідповідним чином;
- не вичерпне тестування;
- тестування без контексту

Ціна кожного дефекту це гроші і час, які будуть витрачені на усунення цієї проблеми. Якщо ж дефект виявлено вже у продукті, який доступний загалу бізнес може втратити клієнтів. Для запобігання негативних сценаріїв потрібно проводити тестування з самого початку розробки ПЗ.

На даний момент стандарт IEEE 829 є найбільш повним та включаючим у себе стратегію тестування продукту, фактори ризики, які з нею пов’язані і кроки для тестування. Саме він був обраний для написання плану тестування цього продукту.

3.2 Опис процесів тестування

Нище представлений тест-план з стратегією тестування, основними ролями та методами тестування.

3.2.1 Ідентифікатор тест-плану

Test Plan for Zorenko`s Diploma

3.2.2 Вступ до тест-плану

Даний мобільний застосунок написаний на React Native. У проєкті є такі модулі:

- модуль реєстрації;
- модуль входу/виходу з системи та відновлення паролю;
- модуль облікового запису клієнта;
- головний модуль (створення, редагування меню).

3.2.3 Об’єкт тестування

- Модуль реєстрації: версія 0.1.0;
- Модуль входу/виходу з системи та відновлення паролю: версія 0.1.0;
- Модуль облікового запису клієнта: версія 0.2.0;
- Головний модуль (створення, редагування меню) : версія 0.1.0.

3.2.4 Компоненти, що тестуються

- Реєстрація клієнта;
- Вхід до системи клієнта;
- Вихід з системи клієнта;
- Відновлення паролю клієнтом;
- Пошук меню;
- Пошук страв з використанням фільтрації.

3.2.5 Компоненти, що не тестуються

- Система нотаток.

3.2.6 Підхід до тестування

Тестування дипломного проєкту відбуватиметься на різних стадіях та складатиметься з юніт-, інтеграційного, системного тестування та тестування критеріїв користувача.

Починати тестування заплановано з юніт-тестування, тобто тестувати кожен модуль окремо. Дане тестування можна проводити як методом білого ящика так і методом чорного ящика. Тести будуть виконуватись у ручному режимі і не будуть автоматизовані. Крім тестування кожного модуля окремо слід проводити інтеграційне тестування. Це дозволить перевірити чи коректно працюють модулі між собою і чи не виникає проблем на цій стадії. Оскільки інтеграційне тестування проводиться для перевірки роботи між модулями, воно буде проводитись по мірі написання коду і додавання нових модулів.

Коли додаток буде вже повністю реалізований потрібно проводити системне тестування, а саме: функціональне тестування, нефункціональне тестування та тестування інтерфейсу.

– Під час проведення функціонального тестування буде перевіряється чи відповідає система визначеним умовам методом чорного ящика. У разі виявлення невідповідності програма повинна бути допрацьована та випущена нова версія у якої все тестування буде проведено наново;

– Під час нефункціонального тестування додаток буде перевірений на зручність користування, інтуїтивно зрозумілий інтерфейс та дизайн. Під тестуванням дизайну мається на увазі відповідність вимогам та можливість користуватись без виникнення проблем з функціоналом. У разі виявлення невідповідності програма повинна бути допрацьована та випущена нова версія у якої все тестування буде проведено наново;

– Тестування навантаження також буде проведене через те що системою одночасно може користуватись багато користувачів і задача розробника забезпечити коректну роботу систему в умовах великого попиту. У

разі виявлення невідповідності програма повинна бути допрацьована та випущена нова версія у якої все тестування буде проведено наново;

– Тестування критеріїв користувач полягає у перевірці виконання бізнес вимог або як ще можна сказати варіантів використання системи користувачем. У разі виявлення невідповідності програма повинна бути допрацьована та випущена нова версія у якої все тестування буде проведено наново.

3.2.7 Критерії тестування

а) Система готова: Всі тести пройдені і помилок не знайдено. Система готова до релізу;

б) Система потребує мінімального доопрацювання: Система задовольняє 90% і більше відсотків тестів. Це означає що система відпрацьовує вірно у більшості сценаріїв, але потребує ще доопрацювання;

в) Система потребує доопрацювання: Система задовольняє більшу частину тестів і підходить під більшу частину сценаріїв використання. Проте потребує ще доопрацювання;

г) Система не готова: Система не підтримує більшість бізнес-процесів, завершує роботу самостійно і не дозволяє користувачам працювати з нею.

3.2.8 Умови початку та кінця тестування:

а. Початок тестування:

1) Додаток описаний на папері;

2) Схема бази даних та архітектура вже намальовані та мають візуальне представлення;

3) Реалізована частина програмного коду.

б. Завершення тестування:

1) Додаток відповідає вимогам бізнес-процесу;

2) Реалізовані всі сценарії використання;

3) Додаток перестає підтримуватись розробником.

3.2.9 Тест-кейси за якими буде проводитись тестування

Таблиця 3.1 – Тест-кейс до req_1

Назва	Авторизація користувачів у системі
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач на сторінці авторизації
Кроки для відтворення	1. Натисніть кнопку ‘Login’ 2. Введіть дані у поля 3. Натисніть кнопку підтвердження
Очікуваний результат	1. Кнопка активна, користувач перейшов на сторінку логіну 2. Поля активні, дані введені, користувач їх бачить 3. Кнопка активна, користувач перейшов на сторінку профілю

Таблиця 3.2 – Тест-кейс до req_2

Назва	Реєстрація користувачів у системі
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач на сторінці реєстрації

Продовження таблиці 3.2

Кроки для відтворення	1. Натисніть кнопку ‘Sign up’ 2. Введіть дані у поля 3. Натисніть кнопку підтвердження
Очікуваний результат	1. Кнопка активна, користувач перейшов на сторінку реєстрації 2. Поля активні, дані введені, користувач їх бачить 3. Кнопка активна, користувач перейшов на сторінку профілю

Таблиця 3.3 – Тест-кейс до req_3

Назва	Перегляд страв на день користувачем
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач залогінений.
Кроки для відтворення	1. Натисніть кнопку ‘Menu’
Очікуваний результат	1. Кнопка активна, користувач перейшов на меню. Користувач бачить запропоноване меню на день

Таблиця 3.4 – Тест-кейс до req_4

Назва	Редагування меню користувачем самостійно
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач залогінений.

Продовження таблиці 3.4

Кроки для відтворення	1. Натисніть кнопку ‘Menu’ 2. Натисніть кнопку з зображенням олівця 3. Змініть 1 позицію з меню на запропонований варіант 4. Натисніть кнопку ‘Save’
Очікуваний результат	1. Кнопка активна, користувач перейшов на меню. Користувач бачить запропоноване меню на день 2. Кнопка активна. Користувач бачить форму редагування 3. Користувач бачить запропоновані варіанти страв та може їх вибрати 4. Кнопка активна, дані збережені

Таблиця 3.5 – Тест-кейс до req_5

Назва	Відображення фізичної активності, яка пропонується користувачеві
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач на сторінці авторизації
Кроки для відтворення	1. Натисніть кнопку ‘Activity’

Продовження таблиці 3.5

Очікуваний результат	1. Кнопка активна, користувач перейшов на сторінку активностей та бачить запропоновані варіанти вправ.

Таблиця 3.6 – Тест-кейс до req_8

Назва	Можливість користувача редагувати дані про себе
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач на сторінці авторизації
Кроки для відтворення	1. Натисніть на зображення 2. Натисніть кнопку з зображенням олівця 3. Змініть якісь дані про себе 4. Натисніть кнопку ‘Save’
Очікуваний результат	1. Користувач перейшов на сторінку профілю та бачить дані про себе 2. Кнопка активна. Користувач бачить форму редагування 3. Користувач може змінити дані про себе та бачить зміни 4. Кнопка активна, дані збережені. Якщо користувач змінював вагу, то кількість ккал перерахована

Таблиця 3.7 – Тест-кейс до req_7

Назва	Перегляд користувачем даних про самого себе
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач на сторінці авторизації
Кроки для відтворення	1. Натисніть на зображення .
Очікуваний результат	1. Користувач перейшов на сторінку профілю та бачить дані про себе.

Таблиця 3.8 – Тест-кейс до req_11

Назва	Відображення кількості випитої води та поставленої цілі.
Важливість	Високий
Передумови	Додаток у спішно встановлений та запущений. Користувач на сторінці авторизації.
Кроки для відтворення	1. Натисніть кнопку ‘Menu’ 2. Натисніть кнопку ‘Water’
Очікуваний результат	1. Кнопка активна, Користувач бачить запропоноване меню на день. 2. Кнопка активна. Користувач перенаправлений на сторінку з необхідною кількістю випитої води.

3.3 Висновки до розділу

У даному розділі був розроблений тест план для тестування додатку “Your favorite menu”. У тест плані прописані критерії тестування, тест-кейси, критерії початку тестування, критерії кінця тестування, який функціонал буде тестуватись, який ні. Також описаний підхід до тестування.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

У даному розділі наведена інструкція для користувача для користування додатком.

1. Для входу у програму потрібно натиснути іконку додатка (рисунок 4.1);

Рисунок 4.1 - Іконка додатку

2. Після натиснення на іконку відображається сторінка входу, де користувач може або ввійти у систему якщо вже має акаунт або створити акаунт (рисунок 4.2);

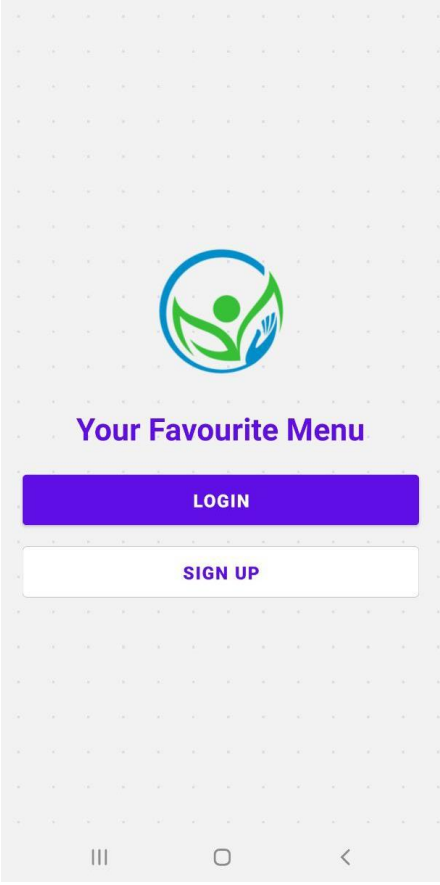


Рисунок 4.2 - Сторінка входу додатка

3. Далі користувач, який зайшов вперше у додаток має натиснути кнопку “Sign up”. Після цього він потрапить на сторінку реєстрації де він має ввести свої дані для входу у додаток (рисунок 4.3);

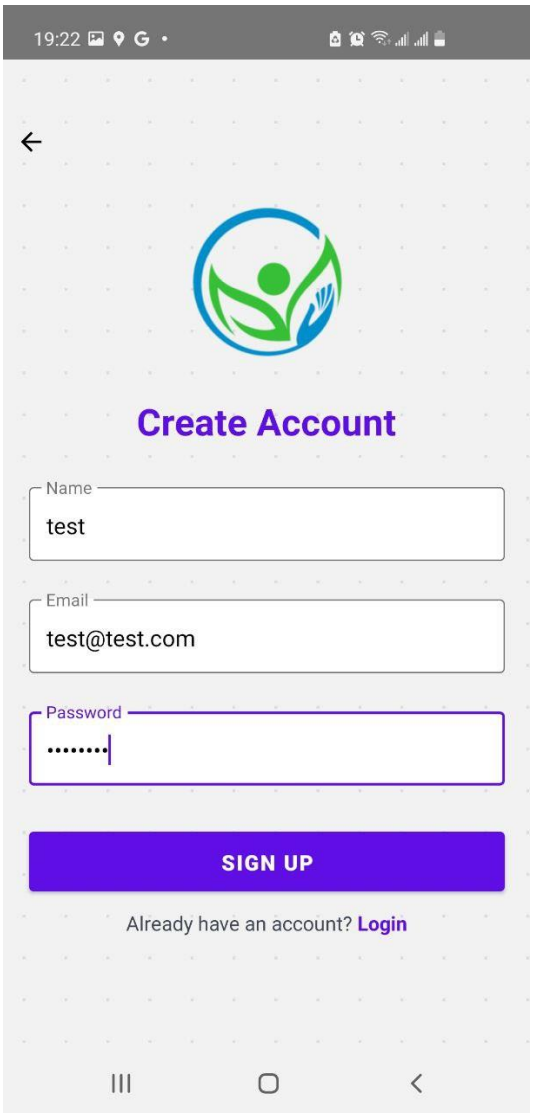


Рисунок 4.3 - Сторінка створення акаунту

4. Після створення реєстрації акаунту у системі користувач автоматично перейде на головну сторінку екрану де він має натиснути кнопку “Add Personal data” (рисунок 4.4);

Рисунок 4.4 - Головна сторінка додатку, до введення персональних даних

5. На сторінці заповнення персональних даних користувач має ввести свої дані, котрі будуть оброблені нейронною мережею. Після введення даних користувач має натиснути кнопку “Save & Process Data” (рисунок 4.5);

Рисунок 4.5 - Сторінка введення персональних даних

6. Після натиснення кнопки “Save & Process Data” користувач побачить сторінку обробки даних (рисунок 4.6);

Рисунок 4.6 - Сторінка обробки даних

7. Після обробки даних користувач автоматично перенаправляється на головну сторінку додатку, де відображається його персональні дані та меню на тиждень, якщо воно вже було обране (рисунок 4.7);

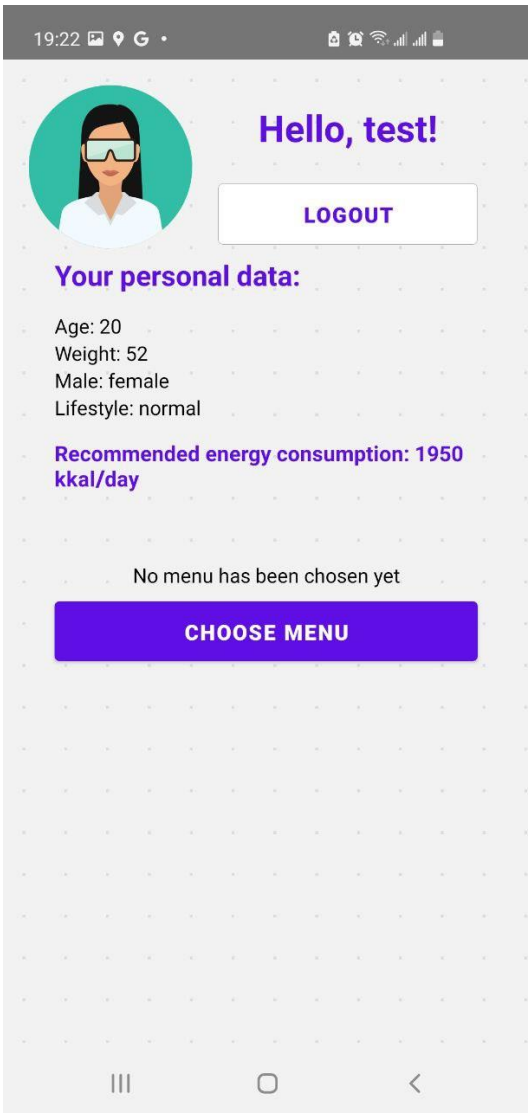


Рисунок 4.7 - Головна сторінка додатку з введеними даними та до обрання меню

8. Після цього користувач має натиснути кнопку “Choose Menu” та обрати меню з запропонованих системою (рисунок 4.8);

Рисунок 4.8 - Сторінка обрання меню

9. Якщо користувач вже має акаунт у систему він повинен натиснути кнопку “Login” та бути перенаправленим на сторінку логіну (рисунок 4.9);

Рисунок 4.9 - Сторінка логіну

10. Після введення даних для входу у систему та натиснення кнопки “Login” користувач перейде на головну сторінку меню (рисунок 4.10).

Рисунок 4.10 - Головна сторінка меню з введеними даними та вибраним меню.

ВИСНОВКИ

Розглядаючи теоретичні матеріали по темі даного дипломного проєкту , були детально вивчені та розглянуті проблеми з здоровим способом життя українців за останні роки, проблемам надлишкової ваги була приділена більша частина дослідження.

Проєктуючи програмне забезпечення були розроблені база даних, серверний та клієнтський додаток та сервер з нейронною мережею, пов’язані в єдину архітектурну систему.

Тестуючи програмне забезпечення, були виявлені та влагоджені можливі ситуації, котрі можуть призвести до некоректної роботи мобільного додатку. Базуючись на цих ситуаціях, була розроблена система обробки даних ситуацій.

Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання є результатом даного дипломного проєкту. Цей застосунок можна використовувати для контролю за своїм раціоном харчування.

ПЕРЕЛІК ПОСИЛАНЬ

1) Tensorflow [Електронний ресурс]: Tensorflow – 2021. – Режим доступу: https://www.tensorflow.org/.

2) Django [Електронний ресурс]: Django. – 2021. – Режим доступу: https://www.djangoproject.com/.

3) Celery [Електронний ресурс]: Celery - Distributed Task Queue – 2018. – Режим доступу: https://docs.celeryproject.org/en/stable/.

4) Redis [Електронний ресурс]: Redis – 2021. – Режим доступу: https://redis.io/.

5) uWsgi [Електронний ресурс]: The uWSGI project – 2016. – Режим доступу: https://uwsgi-docs.readthedocs.io/en/latest/.

6) Design Patterns - MVC Pattern [Електронний ресурс]: Tutorialspoint – 2021. – Режим доступу: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm.

7) Report of a Joint FAO/WHO/UNU Expert Consultation - Human energy requirements [Стаття]: Food and nutrition technical report series – 2001.

UNICHECK

Ім'я користувача:
Попенко Володимир Дмитрович

ID перевірки:
1008230370

Дата перевірки:
08.06.2021 15:13:02 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2021 16:32:03 EEST

ID користувача:
77149

Назва документа: **Zorenko_bachelor_ip72**

Кількість сторінок: **54** Кількість слів: **7905** Кількість символів: **57659** Розмір файлу: **1.05 MB** ID файлу: **1008303709**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.29%
Схожість

Найбільша схожість: **5.62%** з джерелом з Бібліотеки (ID файлу: **1000037970**)

4.17% Джерела з Інтернету	59	Сторінка 56
7.01% Джерела з Бібліотеки	142	Сторінка 56

0% Цитат

Не знайдено жодних цитат

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	10 сторінок

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2021 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ НА БАЗІ МАШИННОГО НАВЧАННЯ

Технічне завдання

КПІ.ІП-7203.045490.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ В.В. Сарнацький

Нормоконтроль:

_____ І.І. Вітковська

Виконавець:

_____В.В. Зоренко

Київ – 2021 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання.

Галузь застосування:

Наведене технічне завдання поширюється на розробку мобільного застосунку [КПІ.ІП-7203.045490], котрий використовується для регулювання режиму харчування людини і надання їй порад щодо складання раціону харчування та виконання фізичних вправ. Даний додаток знадобиться як людям, які страждають від зайвої / недостатньої ваги та просто людям які хочуть підтримувати здоровий спосіб життя.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосування є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання адаптивних порад щодо режиму харчування, складання індивідуального меню в залежності від параметрів людини та її способу життя а також рекомендованого комплексу індивідуальних вправ в залежності від побажань користувача.

Метою даної розробки є спрощення процесу слідкування за власним тілом, підтримка здорового способу життя із нормованим фізичним навантаженням та необхідною кількістю споживаних калорій. Дана розробка позбавляє необхідності у власному фітнес-тренері як для людей, що бажають підтримувати здоровий спосіб життя, так і для людей які страждають від зайвої чи недостатньої ваги.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача системи:

- авторизація в системі за допомогою форми входу та / або Google-акаунту;
- редагування особистих даних;
- заповнення опитування про спосіб життя користувача;
- заповнення опитування про бажаний рівень фізичного навантаження;
- визначення норми споживаних калорій на основі передбачення алгоритму градієнтного спуску;
- генерація рекомендованого меню на основі норми споживаних калорій та побажань користувача;
- генерація рекомендованого комплексу вправ на основі норми споживаних калорій та побажань користувача;
- відслідковування дотримання даних рекомендацій і відповідних змін параметрів тіла.
- наочна аналітика змін параметрів тіла з часом користування даним додатком.

4.1.1.2 Для адміністратора системи:

- Редагування даних у системі за допомогою стандартного веб-інтерфейсу Django;
- Уточнення моделей машиного навчання на основі даних користувачів системи для підвищення точності і якості прогнозів.

4.1.2 Розробку виконати на платформі Android (мобільний застосунок) та Linux Ubuntu 18 (серверна частина).

4.1.3 Додаткові вимоги:

– забезпечити надійність та безпеку персональних даних користувачів;

– забезпечити безперебійний зв’язок мобільного додатку із сервером.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

Вимоги до обслуговування не пред‘являються.

4.3.3 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не пред‘являються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах (сервер) та Android-сумісних мобільних пристроях (мобільний додаток).

4.4.2 Мінімальна конфігурація технічних засобів:

Конфігурація сервера:

Тип процесору: AWS Graviton (2 виділених віртуальних ядра).

Об‘єм ОЗП: 8 Гб.

Тип: AWS ECS t2.large.

Примітка: на даному залізі повинно бути запущено 2 сервери: 1 основний і 1 дублюючий.

Конфігурація мобільного пристрою:

Будь-який мобільний пристрій, що відповідає наступним мінімальним параметрам:

Версія ОС: Android 8.

Об‘єм ОЗП: 4 Гб.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix (сервер) та Android (клієнт).

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: HTTP-запити на сервер з форматом даних типу JSON і методами GET, POST, PUT, DELETE а також ввід користувача.

4.5.3 Результати повинні бути представлені в наступному форматі: відповіді на HTTP-запити у вигляді JSON, візуальний інтерфейс мобільного додатку.

4.5.4 Програмне забезпечення повинно бути створеним на мові Python 3.6 із використанням Django, Django REST Framework (серверна частина) та Dart із використанням фреймворку Flutter (мобільний додаток). СУБД – Postgres.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 80 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Керівництво адміністратора

5.3.5 Програма та методика тестування

5.4 Графічна частина повинна бути виконана на аркушах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структурна компонентів структур даних

5.4.2 Схема структурна класів програмного забезпечення

5.4.3 Креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	01.02.2021	
2.	Розробка технічного завдання	15.02.2021	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	01.03.2021	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	10.03.2021	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму …)
5.	Програмна реалізація програмного забезпечення	20.03.2021	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	20.04.2021	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	01.05.2021	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	10.05.2021	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	20.05.2021	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2021 р.

Мобільний застосунок для підтримки здорового способу життя на базі машиного навчання

Опис програми

КПІ.ІП-7203.045490.04.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ В. В. Сарнацький

Нормоконтроль:

_____ І. І. Вітковська

Виконавець:

_____В. В. Зоренко

Київ – 2021 року

Тексти програмного коду

Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання

(Найменування програми (документа))

DVD-R

(Вид носія даних)

17 арк, 56 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2021

```
from django.db import models
from django.core.validators import MinValueValidator

from utils.models import BaseModel


class Menu(BaseModel):
    name = models.CharField(max_length=255)
    breakfast = models.ForeignKey('menus.Dish', on_delete=models.CASCADE,
related_name='breakfast_menus')
    dinner = models.ForeignKey('menus.Dish', on_delete=models.CASCADE,
related_name='dinner_menus')
    supper = models.ForeignKey('menus.Dish', on_delete=models.CASCADE,
related_name='supper_menus')
    recipe = models.TextField(null=True, blank=True)

    @property
    def is_vegetarian(self):
        return self.breakfast.is_vegetarian and self.dinner.is_vegetarian and
self.supper.is_vegetarian

    @property
    def is_spicy(self):
        return self.breakfast.is_spicy and self.dinner.is_spicy and
self.supper.is_spicy

    @property
    def is_lactose_free(self):
        return self.breakfast.is_lactose_free and self.dinner.is_lactose_free
and self.supper.is_lactose_free

    @property
    def energy_cost_kcal(self):
        return self.breakfast.energy_cost_kcal + self.dinner.energy_cost_kcal +
self.supper.energy_cost_kcal

    def __str__(self):
        return f"{self.name} - {self.energy_cost_kcal} kcal"


class Dish(BaseModel):
    name = models.CharField(max_length=255)
    is_vegetarian = models.BooleanField(null=True, blank=False, default=False)
    is_spicy = models.BooleanField(null=True, blank=False, default=False)
    is_lactose_free = models.BooleanField(null=True, blank=False, default=False)

    @property
    def energy_cost_kcal(self):
        return self.ingredients.aggregate(
            energy_cost_kcal=models.Sum(models.F('weight') *
models.F('ingredient__energy_cost_kcal_100g') / 100)
        )['energy_cost_kcal']

    def __str__(self):
        return f"{self.name} - {self.energy_cost_kcal} kcal"


class DishIngredient(BaseModel):
    dish = models.ForeignKey('menus.Dish', on_delete=models.CASCADE,
related_name='ingredients')
    ingredient = models.ForeignKey('menus.Ingredient', on_delete=models.CASCADE,
related_name='dish_ingredients')
```

```
    weight = models.FloatField(validators=[MinValueValidator(0)])


class Ingredient(BaseModel):
    name = models.CharField(max_length=255)
    energy_cost_kcal_100g = models.FloatField()

    def __str__(self):
        return f"{self.name} - {self.energy_cost_kcal_100g} kcal/100g"




import uuid
from django.db import models
from django.conf import settings
from django.core.validators import MaxValueValidator, MinValueValidator
from django.contrib.auth.models import AbstractUser

from utils.models import BaseModel


class User(AbstractUser):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)


class UserData(BaseModel):
    GENDER_MALE = 'male'
    GENDER_FEMALE = 'female'
    GENDER_CHOICES = (
        (GENDER_MALE, 'Male'),
        (GENDER_FEMALE, 'Female'),
    )

    LIFESTYLE_SEDENTARY = 'sedentary'
    LIFESTYLE_LIGHT_ACTIVITY = 'light'
    LIFESTYLE_MODERATE_ACTIVE = 'moderate'
    LIFESTYLE_ACTIVE = 'active'
    LIFESTYLE_VIGOROUS = 'vigorous'
    LIFESTYLE_VIGOROUSLY_ACTIVE = 'vigorously_active'
    LIFESTYLE_CHOICES = (
        (LIFESTYLE_SEDENTARY, 'Sedentary'),
        (LIFESTYLE_LIGHT_ACTIVITY, 'Light activity'),
        (LIFESTYLE_MODERATE_ACTIVE, 'Moderate active'),
        (LIFESTYLE_ACTIVE, 'Active'),
        (LIFESTYLE_VIGOROUS, 'Vigorous'),
        (LIFESTYLE_VIGOROUSLY_ACTIVE, 'Vigorously active'),
    )

    user = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='data')
    gender = models.CharField(max_length=10, choices=GENDER_CHOICES,
default=GENDER_MALE)
    age = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(99)])
    weight = models.DecimalField(max_digits=5, decimal_places=2,
validators=[MinValueValidator(0), MaxValueValidator(200)])
    lifestyle = models.CharField(max_length=20, choices=LIFESTYLE_CHOICES,
default=LIFESTYLE_MODERATE_ACTIVE)
    recommended_calories_intake =
models.FloatField(validators=[MinValueValidator(0)], null=True, blank=True)
    menu = models.ForeignKey('menus.Menu', on_delete=models.SET_NULL, null=True,
blank=True)
```

```
    @property
    def menu_weight_coefficient(self):
        return self.recommended_calories_intake / self.menu.energy_cost_kcal if
self.menu and self.recommended_calories_intake else None


class UserDailyStatus(BaseModel):
    user = models.ForeignKey('users.User', on_delete=models.CASCADE,
related_name='daily_statuses')
    date = models.DateField(auto_now_add=True)
    weight = models.DecimalField(max_digits=5, decimal_places=2,
validators=[MinValueValidator(0), MaxValueValidator(200)])
    lifestyle = models.CharField(max_length=20,
choices=UserData.LIFESTYLE_CHOICES, default=UserData.LIFESTYLE_MODERATE_ACTIVE)
    real_calories_intake = models.FloatField(validators=[MinValueValidator(0)],
null=True, blank=True)
    menu = models.ForeignKey('menus.Menu', on_delete=models.SET_NULL, null=True,
blank=True)

    class Meta:
        unique_together = ['user', 'date']


class UserMenuPreferences(BaseModel):
    user = models.OneToOneField(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='menu_preferences')
    is_vegetarian = models.BooleanField(null=True, default=False)
    is_spicy = models.BooleanField(null=True, default=False)
    is_lactose_free = models.BooleanField(null=True, default=False)
    breakfast_percentage = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(100)], blank=True, default=30)
    dinner_percentage = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(100)], blank=True, default=40)
    supper_percentage = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(100)], blank=True, default=30)

import uuid

from django.db import models
from django.utils.translation import ugettext_lazy as _


class BaseModel(models.Model):
    """
    Setups default class representation and adds created_date and
    modified_date database fields.
    """
    id = models.UUIDField(_("id"), primary_key=True, default=uuid.uuid4,
editable=False)
    created_date = models.DateTimeField(_("created date"), auto_now_add=True)
    modified_date = models.DateTimeField(_("modified date"), auto_now=True)

    class Meta:
        abstract = True

    @property
    def is_new(self):
        return not self.created_date
```

```
export class DishModel {
  name: string;
  energy_cost: number;
  recipe: {ingredient: string; mass: number}[];

  public constructor({
    name = '',
    energy_cost = 0,
    recipe = [],
  }: {
    name?: string;
    energy_cost?: number;
    recipe?: {ingredient: string; mass: number}[];
  }) {
    this.name = name;
    this.energy_cost = energy_cost;
    this.recipe = recipe;
  }
}

export default class MenuModel {
  name: string;
  coefficient: number;
  breakfast: DishModel;
  dinner: DishModel;
  supper: DishModel;

  public constructor({
    name = '',
    coefficient = 1,
    breakfast = new DishModel({}),
    dinner = new DishModel({}),
    supper = new DishModel({}),
  }: {
    name?: string;
    coefficient?: number;
    breakfast?: DishModel;
    dinner?: DishModel;
    supper?: DishModel;
  }) {
    this.name = name;
    this.coefficient = coefficient;
    this.breakfast = breakfast;
    this.dinner = dinner;
    this.supper = supper;
  }
}




export default class UserModel {
  username: string;
  email: string;
  password: string;
  first_name: string;
  last_name: string;

  public constructor({
      username = '',
      email = '',
      password = '',
      first_name = '',
      last_name = ''
```

```
  }: { username?: string, email?: string, password?: string, first_name?:
string, last_name?: string }) {
    this.username = username;
    this.email = email;
    this.password = password;
    this.first_name = first_name;
    this.last_name = last_name;
  }
}



export default class UserDataModel {
  male: 'male' | 'female';
  weight: number;
  age: number;
  lifestyle: 'passive' | 'normal' | 'active' | 'sport';

  public constructor({
    male = 'male',
    weight = 0,
    age = 0,
    lifestyle = 'normal',
  }: {
    male?: 'male' | 'female';
    weight?: number;
    age?: number;
    lifestyle?: 'passive' | 'normal' | 'active' | 'sport';
  }) {
    this.male = male;
    this.age = age;
    this.weight = weight;
    this.lifestyle = lifestyle;
  }
}
import UserModel from '../models/user.model';
import storage from '../core/storage';
import {strToHashCode} from '../core/utils';
import UserDataModel from '../models/userData.model';
import MenuModel from "../models/menu.model";

export default class APIService {
  public constructor() {}

  public register(user: UserModel): Promise<void> {
    const userId = strToHashCode(user.email).toString();

    return storage
      .save({
        key: 'user',
        id: userId,
        data: user,
      })
      .then(() => {
        storage.save({
          key: 'userId',
          data: userId,
        });
        return;
      });
  }

  public login(email: string): Promise<UserModel> {
```

```
    const userId = strToHashCode(email).toString();

    return storage
      .load({
        key: 'user',
        id: strToHashCode(email).toString(),
      })
      .then(ret => {
        storage.save({
          key: 'userId',
          data: userId,
        });
        return ret;
      });
  }

  public logout(): Promise<void> {
    return storage.remove({
      key: 'userId',
    });
  }

  public setUserData(userData: UserDataModel): Promise<void> {
    return storage.load({key: 'userId'}).then(userId => {
      return
  }

  public setUserMenu(userMenu: MenuModel): Promise<void> {
    return storage.load({key: 'userId'}).then(userId => {
      return storage.save({
        key: 'userMenu',
        id: userId,
        data: userMenu,
      });
    });
  }
}
```

```
import {BehaviorSubject} from 'rxjs';

import storage from '../core/storage';
import {consts} from '../core/consts';


export default class AuthService {
    public accessTokenSubject = new BehaviorSubject();
    public refreshTokenSubject = new BehaviorSubject();

    public constructor() {
        storage.load({key: 'accessToken'}).then(accessToken => {
            this.accessTokenSubject.next(accessToken);
        });
        storage.load({key: 'refreshToken'}).then(accessToken => {
            this.refreshTokenSubject.next(accessToken);
        });
    }

    public login(username: string, password: string): Promise<void> {
        const requestOptions = {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
```

```
            body: JSON.stringify({username, password}),
        };

        return fetch(`${consts.HOSTNAME}/auth/login`, requestOptions)
            .then(this.handleResponse)
            .then(value => this.updateToken(value.access, value.refresh));
    }

    public logout(): Promise<void> {
        const requestOptions = {
            method: 'POST',
            headers: {'Content-Type': 'application/json', ...this.authHeaders},
        };
        return fetch(`${consts.HOSTNAME}/auth/logout`, requestOptions)
            .then(this.handleResponse)
            .then(() => this.removeToken());
    }

    public handleResponse(response: Response) {
        return response.text().then(text => {
            const data = text && JSON.parse(text);
            if (!response.ok) {
                if ([401, 403].indexOf(response.status) !== -1) {
                    return this.refreshToken();
                }
                const error = (data && data.message) || response.statusText;
                return Promise.reject(error);
            }
            return data;
        });
    }

    public get authHeaders(): {[key: string]: string} {
        if (this.accessTokenSubject.value) {
            return { Authorization: `Bearer ${this.accessTokenSubject.value}` };
        } else {
            return {};
        }
    }

    private updateToken(access: string, refresh: string): Promise<void> {
        this.accessTokenSubject.next(access);
        this.refreshTokenSubject.next(refresh);
        return storage.save({key: 'accessToken', data: access,})
            .then(() => storage.save({key: 'refreshToken', data: refresh}));
    }

    private refreshToken(): Promise<void> {
        const requestOptions = {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({refresh: this.refreshTokenSubject.value}),
        };
        return fetch(`${consts.HOSTNAME}/auth/login`, requestOptions)
            .then((response: Response) => {
                return response.text().then(text => {
                    const data = text && JSON.parse(text);
                    if (!response.ok) {
                        if ([401, 403].indexOf(response.status) !== -1) {
                            return this.removeToken();
                        }
                        const error = (data && data.message) ||
response.statusText;
                        return Promise.reject(error);
```

```
                    }
                    return data;
                });
            })
            .then(value => this.updateToken(value.access, value.refresh));
    }

    private removeToken(): Promise<void> {
        this.accessTokenSubject.next(null);
        this.refreshTokenSubject.next(null);
        return storage.remove({key: 'accessToken'})
            .then(() => storage.remove({key: 'refreshToken'}));
    }
}
```

```
import React from 'react';

import APIService from './api.service';
import UserService from './user.service';

export const apiService = new APIService();
export const userService = new UserService();

export const state = {
  apiService,
  userService,
};

export const ServiceContext = React.createContext(state);
import storage from '../core/storage';
import UserModel from '../models/user.model';
import UserDataModel from '../models/userData.model';
import MenuModel from "../models/menu.model";

export default class UserService {
  private _userId: string | undefined;
  private _user: UserModel | undefined;
  private _userData: UserDataModel | undefined;
  private _userMenu: MenuModel | undefined;

  public constructor() {}

  public get userId(): Promise<string> {
    return this.getUserId();
  }

  public get user(): Promise<UserModel> {
    return this.userId.then(id => this.getUser(id));
  }

  public get userData(): Promise<UserDataModel> {
    return this.userId.then(id => this.getUserData(id));
  }

  public get userMenu(): Promise<MenuModel> {
    return this.userId.then(id => this.getUserMenu(id));
  }

  private getUserId(): Promise<string> {
    return storage.load({key: 'userId'}).then(ret => {
      this._userId = ret;
```

```
      return ret;
    });
  }

  private getUser(userId: string): Promise<UserModel> {
    return storage.load({key: 'user', id: userId}).then(ret => {
      this._user = ret;
      return ret;
    });
  }

  private getUserData(userId: string): Promise<UserDataModel> {
    return storage.load({key: 'userData', id: userId}).then(ret => {
      this._userData = ret;
      return ret;
    });
  }

  private getUserMenu(userId: string): Promise<MenuModel> {
    return storage.load({key: 'userMenu', id: userId}).then(ret => {
      this._userMenu = ret;
      return ret;
    });
  }
}
```

```
import React, {memo, useContext, useEffect, useState} from 'react';
import {
  FlatList,
  Image,
  ScrollView,
  StyleSheet,
  Text,
  View,
} from 'react-native';
import Background from '../components/Background';
import Header from '../components/Header';
import Button from '../components/Button';
import {Navigation} from '../types';
import {ServiceContext} from '../services';
import {theme} from '../core/theme';

type Props = {
  navigation: Navigation;
};

const Dashboard = ({navigation}) => {
  const {apiService, userService} = useContext(ServiceContext);
  const [user, setUser] = useState();
  const [userData, setUserData] = useState();
  const [userMenu, setUserMenu] = useState();

  const _onLogoutPressed = () => {
    apiService.logout().then(() => {
      navigation.navigate('HomeScreen');
    });
  };

  userService.user
    .then(value => {
      setUser(value as any);
```

```
    })
    .catch(() => {
      navigation.navigate('HomeScreen');
    });

  useEffect(() => {
    const unsubscribe = navigation.addListener('focus', payload => {
      userService.userData.then(value => {
        setUserData(value as any);
      });
      userService.userMenu.then(value => {
        setUserMenu(value as any);
      });
    });

    return unsubscribe;
  }, [navigation]);

  return (
    <Background>
      <View
        style={{
          flexDirection: 'row',
          flex: 1,
          width: '100%',
          flexGrow: 0,
          flexShrink: 0,
          flexBasis: 100,
        }}>
        <Image source={require('../assets/avatar.png')} style={styles.avatar} />
        <View style={{flexGrow: 1, alignItems: 'center', marginHorizontal: 20}}>
          <Header>Hello, {user ? user.username : ''}!</Header>
          <Button mode="outlined" onPress={_onLogoutPressed}>
            Logout
          </Button>
        </View>
      </View>
      <View style={{flexGrow: 1, width: '100%'}}>
        {userData ? (
          <View>
            <View style={{margin: 20}}>
              <Text
                style={{
                  fontSize: 20,
                  color: theme.colors.primary,
                  fontWeight: 'bold',
                  paddingVertical: 14,
                }}>
                Your personal data:
              </Text>
              <View>
                <Text>Age: {userData.age}</Text>
              </View>
              <View>
                <Text>Weight: {userData.weight}</Text>
              </View>
              <View>
                <Text>Male: {userData.male}</Text>
              </View>
              <View>
                <Text>Lifestyle: {userData.lifestyle}</Text>
              </View>
              <View>
                <Text
```

```
                    style={{
                      fontSize: 15,
                      color: theme.colors.primary,
                      fontWeight: 'bold',
                      paddingVertical: 14,
                    }}>
                    Recommended energy consumption: 1950 kkal/day
                  </Text>
                </View>
              </View>
              {userMenu ? (
                <View style={{margin: 20}}>
                  <View style={styles.menuItem}>
                    <Text style={styles.menuTitle}>{userMenu.name}</Text>
                    <View>
                      <Text style={styles.dishTitle}>
                        Breakfast: {userMenu.breakfast.name}
                      </Text>
                      <View>
                        <Text>Recipe:</Text>
                        <FlatList
                          nestedScrollEnabled
                          keyExtractor={(item, index) => item.ingredient + index}
                          style={{width: '100%'}}
                          data={userMenu.breakfast.recipe}
                          renderItem={({item}) => (
                            <Text>
                              {item.ingredient}:{' '}
                              {item.mass * userMenu.coefficient}g
                            </Text>
                          )}
                        />
                      </View>
                    </View>
                    <View>
                      <Text style={styles.dishTitle}>
                        Dinner: {userMenu.dinner.name}
                      </Text>
                      <View>
                        <Text>Recipe:</Text>
                        <FlatList
                          nestedScrollEnabled
                          keyExtractor={(item, index) => item.ingredient + index}
                          style={{width: '100%'}}
                          data={userMenu.dinner.recipe}
                          renderItem={({item}) => (
                            <Text>
                              {item.ingredient}:{' '}
                              {item.mass * userMenu.coefficient}g
                            </Text>
                          )}
                        />
                      </View>
                    </View>
                    <View>
                      <Text style={styles.dishTitle}>
                        Supper: {userMenu.supper.name}
                      </Text>
                      <View>
                        <Text>Recipe:</Text>
                        <FlatList
                          nestedScrollEnabled
                          keyExtractor={(item, index) => item.ingredient + index}
                          style={{width: '100%'}}
```

```
                        data={userMenu.supper.recipe}
                        renderItem={({item}) => (
                          <Text>
                            {item.ingredient}:{' '}
                            {item.mass * userMenu.coefficient}g
                          </Text>
                        )}
                      />
                    </View>
                  </View>
                </View>
                <Button
                  mode="contained"
                  onPress={() => navigation.navigate('EditMenuScreen')}>
                  Change menu
                </Button>
              </View>
            ) : (
              <View style={{margin: 20}}>
                <Text style={{alignSelf: 'center'}}>
                  No menu has been chosen yet
                </Text>
                <Button
                  mode="contained"
                  onPress={() => navigation.navigate('EditMenuScreen')}>
                  Choose menu
                </Button>
              </View>
            )}
          </View>
        ) : (
          <View style={{marginTop: 200}}>
            <Text style={{alignSelf: 'center'}}>
              No personal data has been added yet
            </Text>
            <Button
              mode="contained"
              onPress={() => navigation.navigate('EditDataScreen')}>
              Add personal data
            </Button>
          </View>
        )}
      </View>
    </Background>
  );
};

const styles = StyleSheet.create({
  avatar: {
    width: 128,
    height: 128,
    marginBottom: 12,
  },
  menuItem: {
    borderColor: '#AAA',
    borderStyle: 'solid',
    borderWidth: 1,
    borderRadius: 5,
    marginVertical: 5,
    padding: 5,
  },
  menuTitle: {
    fontSize: 15,
    color: theme.colors.primary,
```

```
    fontWeight: 'bold',
    paddingVertical: 5,
  },
  dishTitle: {},
});

export default Dashboard;
```

```
import React, {memo, useContext, useState} from 'react';
import {Navigation} from '../types';
import {ServiceContext} from '../services';
import Background from '../components/Background';
import {StyleSheet, Text, View} from 'react-native';
import UserDataModel from '../models/userData.model';
import TextInput from '../components/TextInput';
import {RadioButton} from 'react-native-paper';
import Button from '../components/Button';
import {ageValidator, weightValidator} from '../core/utils';

type Props = {
  navigation: Navigation;
};

const EditData = ({navigation}: Props) => {
  const {apiService, userService} = useContext(ServiceContext);
  const [userData, setUserData] = useState(new UserDataModel({}));
  const [male, setMale] = useState({value: userData.male, error: ''});
  const [weight, setWeight] = useState({
    value: userData.weight.toString(),
    error: '',
  });
  const [age, setAge] = useState({value: userData.age.toString(), error: ''});
  const [lifestyle, setLifestyle] = useState({
    value: userData.lifestyle,
    error: '',
  });

  const _onApplyPressed = () => {
    const ageError = ageValidator(age.value);
    const weightError = weightValidator(weight.value);

    if (ageError || weightError) {
      setAge({...age, error: ageError});
      setWeight({...weight, error: weightError});
      return;
    }

    const data = new UserDataModel({
      male: male.value,
      weight: +weight.value,
      age: +age.value,
      lifestyle: lifestyle.value,
    });

    apiService
      .setUserData(data)
      .then(() => {
        navigation.navigate('ProcessingScreen');
      }).catch((e) => {
        alert(e);
    });
```

```
};

const onNumberChanged = (text: string, stateFn: (text) => void) => {
  let newText = '';
  let numbers = '0123456789';

  for (let i = 0; i < text.length; i++) {
    if (numbers.indexOf(text[i]) > -1) {
      newText = newText + text[i];
    } else {
      alert('please enter numbers only');
    }
  }
  stateFn({value: newText, error: ''});
};

return (
  <Background>
    <TextInput
      label="Age"
      returnKeyType="next"
      value={age.value}
      onChangeText={text => onNumberChanged(text, setAge.bind(this))}
      error={!!age.error}
      errorText={age.error}
      keyboardType="numeric"
      maxLength={2}
    />
    <TextInput
      label="Weight"
      returnKeyType="next"
      value={weight.value}
      onChangeText={text => onNumberChanged(text, setWeight.bind(this))}
      error={!!weight.error}
      errorText={weight.error}
      keyboardType="numeric"
      maxLength={3}
    />
    <View style={styles.rbgroup}>
      <RadioButton.Group
        onValueChange={newValue => setMale({value: newValue, error: ''})}
        value={male.value}>
        <Text>Choose your gender:</Text>
        <View style={styles.rbgroupitem}>
          <RadioButton value="male" />
          <Text>Male</Text>
        </View>
        <View style={styles.rbgroupitem}>
          <RadioButton value="female" />
          <Text>Female</Text>
        </View>
      </RadioButton.Group>
    </View>
    <View style={styles.rbgroup}>
      <RadioButton.Group
        onValueChange={newValue => setLifestyle({value: newValue, error: ''})}
        value={lifestyle.value}>
        <Text>Choose your lifestyle:</Text>
        <View style={styles.rbgroupitem}>
          <RadioButton value="passive" />
          <Text>Passive</Text>
        </View>
        <View style={styles.rbgroupitem}>
          <RadioButton value="normal" />
```

```
            <Text>Normal</Text>
          </View>
          <View style={styles.rbgroupitem}>
            <RadioButton value="active" />
            <Text>Active</Text>
          </View>
          <View style={styles.rbgroupitem}>
            <RadioButton value="sport" />
            <Text>Sport</Text>
          </View>
        </RadioButton.Group>
      </View>
      <Button mode="contained" onPress={_onApplyPressed}>
        Save & Process Data
      </Button>
    </Background>
  );
};

const styles = StyleSheet.create({
  rbgroup: {
    width: '100%',
    borderColor: '#AAA',
    borderStyle: 'solid',
    borderWidth: 1,
    borderRadius: 5,
    marginVertical: 5,
    padding: 5,
  },
  rbgroupitem: {
    flexDirection: 'row',
    alignItems: 'center',
  },
});

export default memo(EditData);
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2021 р.

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ПІДТРИМКИ ЗДОРОВОГО СПОСОБУ ЖИТТЯ НА БАЗІ МАШИННОГО НАВЧАННЯ

Програма та методика тестування

КПІ.ІП-7203.045490.05.51

“ПОГОДЖЕНО”

Керівник проекту:

_____ В.В. Сарнацький

Нормоконтроль:

_____ І.І. Вітковська

Виконавець:

_____В.В. Зоренко

Київ – 2021 року

ЗМІСТ

1 ОБ’ЄКТ ТЕСТУВАННЯ

Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання, котрий являє собою мобільний клієнт-серверний застосунок, написаний та створений на базі веб – фреймворку Django та мобільного фреймворку React Native.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є перевірка відповідності поставленим функціональним та нефункціональним вимогам системи.

До функціональних вимог було віднесено:

- авторизація користувача у систему
- реєстрація нового користувача у систему
- зміна пароля авторизованим користувачем
- перегляд меню авторизованим користувачем
- ввід даних для обробки нейронною мережею авторизованим користувачем
- внесення вподобань щодо меню авторизованим користувачем

До нефункціональних вимог було віднесено:

- інтуїтивно зрозумілий інтерфейс
- зручність у переході між екранами додатку
- приємний дизайн додатку

3 МЕТОДИ ТЕСТУВАННЯ

Методи які були застосовані для тестування проєкту Your favorite menu з:

а) unit-тестування;

б) інтеграційне тестування;

в) модульне тестування

г) системне тестування

1) функціональне тестування;

2) навантажувальне тестування;

д) UI/UX тестування

е) UAT (User Acceptance Testing).

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

а) Unit-тестування.

Даний вид тестування є автоматизованим за допомогою засобів Django. Також налаштований процес запуску тестів автоматично під час кожної збірки проєкту завдяки Jenkins CI. Виконується розробником.

б) Інтеграційне тестування.

Автоматизоване завдяки засобам Mocha. Також налаштований процес запуску тестів автоматично під час кожної збірки проєкту завдяки Jenkins CI. Виконується розробником.

в) Модульне тестування

Автоматизоване завдяки засобам Django. Також налаштований процес запуску тестів автоматично під час кожної збірки проєкту завдяки Jenkins CI. Виконується розробником.

г) Системне тестування.

1) Функціональне тестування.

Проводиться лише після успішного проходження попередніх видів тестування. Якщо попередні етапи не пройдені проєкт відправляється на доопрацювання. Проводиться в ручному режимі.

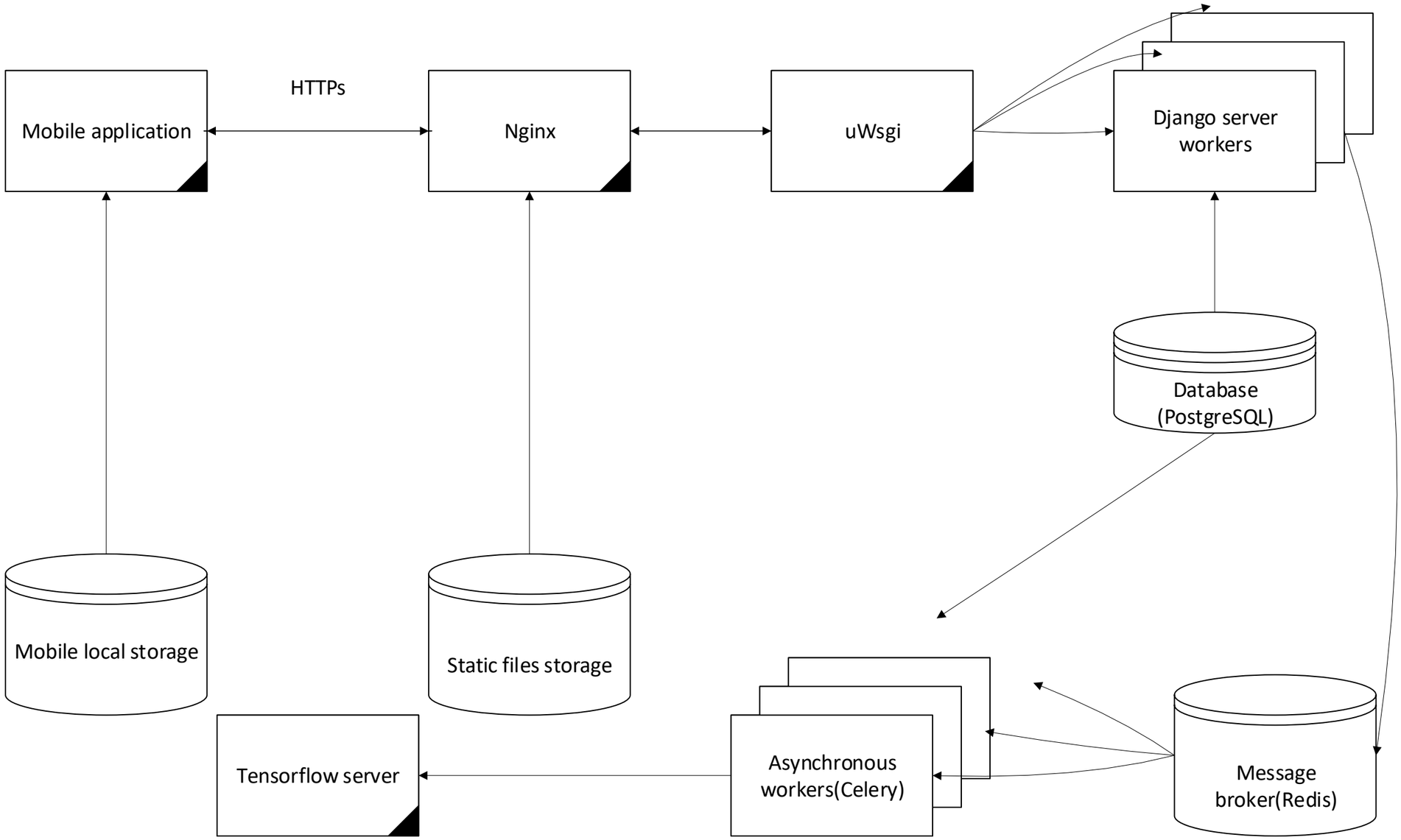
2) Навантажувальне тестування.

Проводиться лише після успішного проходження попередніх видів тестування. Якщо попередні етапи не пройдені проєкт відправляється на доопрацювання. Проводиться в автоматизованому режимі.

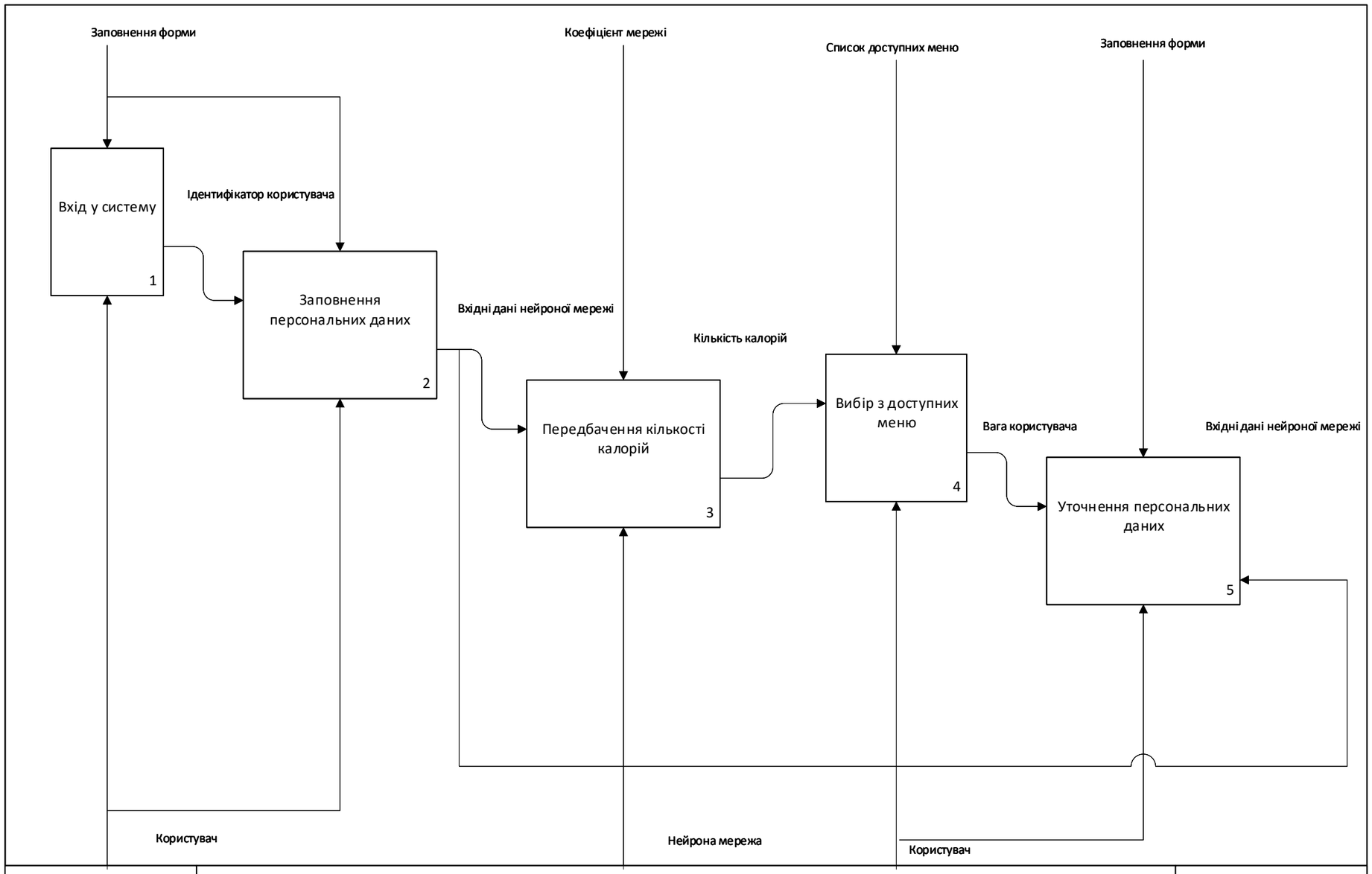
д) UAT.

Виконується клієнтом. У разі невдачі йде повернення до системного тестування, система має бути доопрацьована. У разі проходження цього етапу – перехід до релізу. Проводитиметься в ручному режимі.

КПІ.ІП-7203.045490.81.99.СС

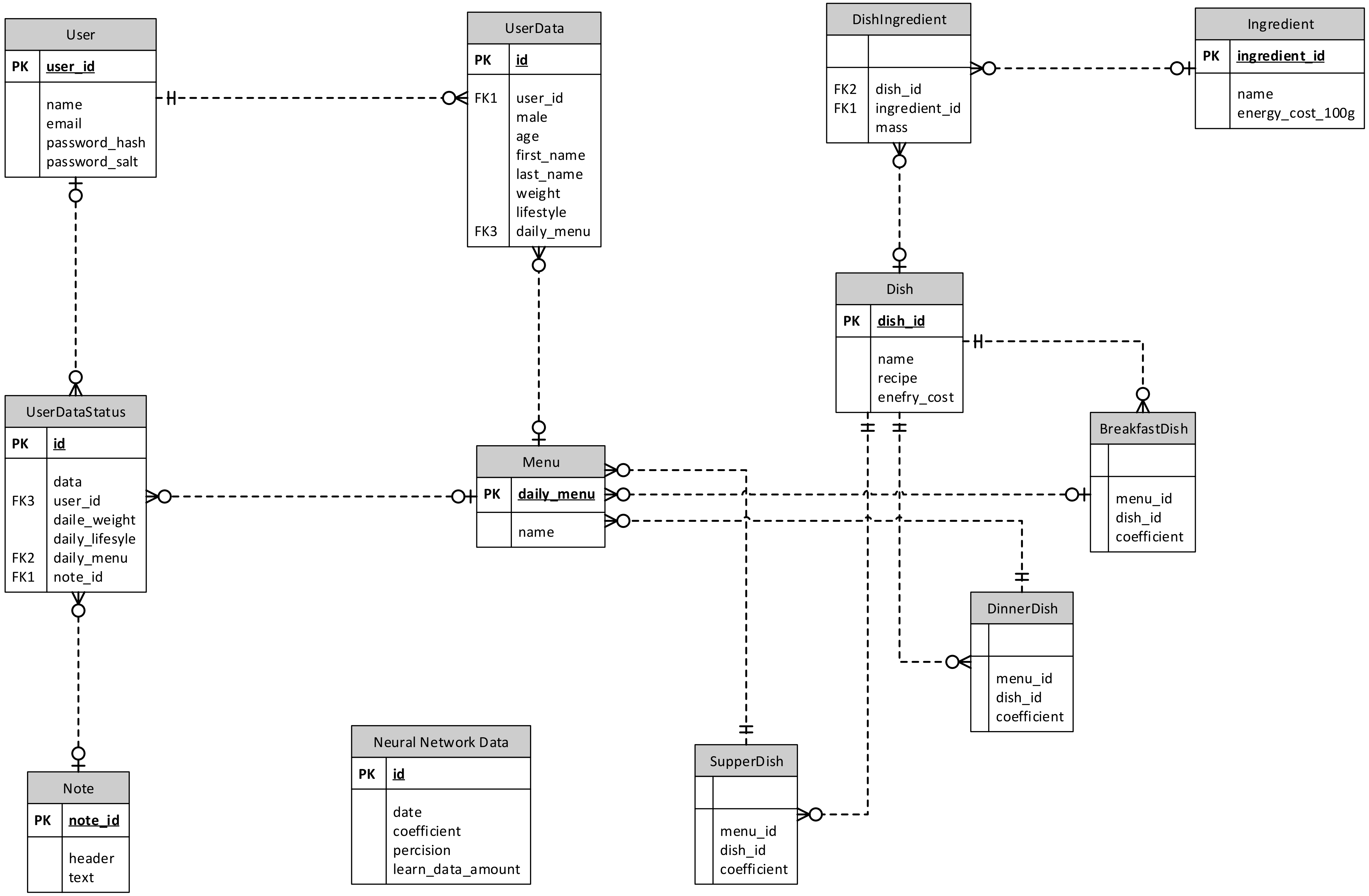


					КПІ.ІП-7203.045490.81.99.СС	
					Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Зоренко В.В.				Аркуш / Аркушів
Перевірив		Сарнацький В.В.			Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-72
Т. кон.						
Н. кон.		Вітковська І.І.				
Затвердив		Сарнацький В.В.				



						КПІ.ІП-7203.045490.81.99.СС
					Схема структурна діяльності	Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Зоренко В.В.				Аркуш / Аркушів
Перевірив		Сарнацький В.В.				
Т. кон.						
					Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-72
Н. кон.		Вітковська І.І.				
Затвердив		Сарнацький В.В.				

КПІ.ІП-7203.045490.81.99.СБД



Зм.	Арк.	№ документа	Підпис	Дата	КПІ.ІП-7203.045490.81.99.СБД		
Розробив		Зоренко В.В.			Схема бази даних	Літера	Маса	Масштаб
Перевірив		Сарнацький В.В.				Аркуш	Аркушів
Т. кон.							
Н. кон.		Вітковська І.І.			Мобільний застосунок для підтримки здорового способу життя на базі машинного навчання	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-72	
Затвердив		Сарнацький В.В.					