МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНФОРМАЦІЙНА СИСТЕМА МОНІТОРИНГУ ФІЗИЧНИХ НАВАНТАЖЕНЬ ВЕЛОСПОРТСМЕНА

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910119

Виконав студент 4-го курсу, групи 401
_____ ***Б. В. Пилипчук***
«20» червня 2023 р.

Керівник: д-р техн. наук, проф.
_____***І. М. Журавська***
«20» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Пилипчуку Богдану Віталійовичу.

1. Тема кваліфікаційної роботи «Інформаційна система моніторингу фізичних навантажень велоспортсмена».

Керівник роботи Журавська Ірина Миколаївна, д-р техн. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: спеціалізовані файли (Data Set), зібрані за допомогою спеціалізованого спортивного обладнання фірми Garmin.

Очікуваний результат: Програмне забезпечення обробки даних для оцінювання фізичних показників велоспортсмена.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- огляд прикладної сфери;

- огляд вже існуючих аналогів систем, призначених для оцінювання фізичних показників велоспортсмена;
- вибір інструментів для реалізації застосунку;
- програмна реалізація застосунку.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Аналіз робочого місця розробника програмного забезпечення за усіма вимогами техніки безпеки»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., канд. техн. наук, доцент	

Керівник роботи __д-р техн. наук, проф. Журавська І. М._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Пилипчук Б. В._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: «Інформаційна система моніторингу фізичних навантажень велоспортсмена»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заявки на затвердження теми та керівників БКР	27.10.2022	27.10.2022	Виконано
2	Отримання завдання на виконання БКР	08.11.2022	08.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	19.11.2022	22.11.2022	Виконано
4	Отримання завдання на переддипломну практику	18.04.2023	18.04.2023	Виконано
5	Проходження переддипломної практики	01.05.2023	14.05.2023	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: аналіз сучасного стану задачі фізичної підготовки велоспортсмена, огляд існуючих технологій, розробка ПЗ	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	Виконано

Розробив студент _____Пилипчук Б. В._____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи д-р техн. наук, проф. Журавська І. М. _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«_08_» _____грудня_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи

студента групи 401 ЧНУ ім. Петра Могили

Пилипчука Богдана Віталійовича

Тема: «Інформаційна система моніторингу фізичних навантажень велоспортсмена»

Бакалаврська кваліфікаційна робота присвячена моніторингу фізичних навантажень велоспортсмена. Дана робота є актуальною тому, що через різні надзвичайні події в світі (пандемії, воєнний стан та ін.) стало важко стежити за своїм фізичним здоров'ям та підтримувати свої фізичні показники. Велика кількість спортивних шкіл з велоспорту олімпійського резерву призупинила свою роботу, тим самим знизився рівень підготовки майбутніх чемпіонів.

Об'єкт роботи (розробки) – процеси моніторингу фізичних навантажень велоспортсмена.

Предмет роботи (розробки) – методи та засоби створення інформаційних систем, що виконують функції обробки даних щодо фізичного навантаження та коригування плану тренувань велоспортсменів.

Мета – оцінювання фізичних навантажень велоспортсмена шляхом обробки статистичних даних та їх подальшої візуалізації засобами розробленого програмного забезпечення (ПЗ).

У першому розділі було розглянуто актуальність створення системи для оцінювання. У другому розділі продемонстровано моделі і прототипи ПЗ. У третьому розділі наводяться необхідні інструменти та описується архітектура застосунку. В четвертому розділі наведено кроки реалізації застосунку та процеси тестування, а також продемонстрована розроблена документація.

В результаті роботи було реалізовано ПЗ для оцінки фізичних навантажень велоспортсмена, створено документацію до нього, а також інтерфейс користувача.

БКР викладена на 64 с. (без додатків), містить 4 розділи, 42 рис., 5 табл., 3 додатки, 27 джерел в переліку посилань.

***Ключові слова**: моніторинг навантажень, аналіз даних, набір даних, велоспортсмен, Python, Big Data, Tkinter, Pandas, Matplotlib, NumPy.*

ABSTRACT

of the Bachelor's Thesis

by student of group 401 in Petro Mohyla Black Sea National University

Pylypchuk Bohdan

"Information system for monitoring physical exertion of a cyclist"

The bachelor's qualification work is devoted to the monitoring the physical exertion of a cyclist via created software. This work is relevant due to various extraordinary events in the world (pandemics, martial law), it has become difficult to monitor one's physical health and maintain one's physical indicators. A large number of children's and youth cycling sports schools of the Olympic reserve have suspended their work, thereby lowering the level of training of our future champions.

The object of research (development) is the processes of monitoring the physical exertion of a cyclist.

The subject of research (development) is methods and means of creating information systems that perform the functions of obtaining data on physical load and adjusting the training plan of cyclists.

The goal is to assess physical exertion of a cyclist by processing statistical data and their subsequent visualization via the developed software.

In the first chapter, the relevance of creating a system for evaluation was considered. In the second section, software models and prototypes are demonstrated. The third section lists the necessary tools and describes the application architecture. In the fourth chapter, the application implementation steps and testing processes are given, and the developed documentation is demonstrated.

As a result of the work performed, the software was implemented for the assessment of the physical loads of the cyclist, the documentation for it, as well as the user interface, have created.

The Bachelor's Thesis contains 64 pages (without appendices), 42 figures, 5 tables, 3 appendices, 27 sources in the reference list.

Keywords: *load monitoring, data analysis, data set, cyclist, Python, Big Data, Tkinter, Pandas, Matplotlib, NumPy.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІС – інформаційна система
МТБ – маунтенбайк
ПЗ – програмне забезпечення
ПК – персональний комп’ютер

FTP – Functional Threshold Power
FTS – Foster's Training Scale
IDE – Integrated Development Environment
KISS – Keep it simple, stupid
RPE – Rating of Perceived Exertion
UML – Unified Modelling Languagef
VAMs – Vertical Ascent Meters per Second

ВСТУП

Велосипедний спорт – один з найпопулярніших олімпійських видів спорту.

Перший чемпіонат світу з велосипедного спорту був проведений у 1893 р.

На перших Олімпійських іграх у 1896 р. велоспорт входив до програми змагань [1].

На початку 80-х років XX століття у США почали розвиватися такі види велоспорту, як МТБ (маунтенбайк, гірський велосипед) та BMX (байк-крос). З 1990 р. проводяться чемпіонати світу з МТБ, а з 1996 р. МТБ включений до програми Олімпійських ігор. Перший чемпіонат світу з BMX проведений в 1982 р., олімпійським видом велоспорту BMX став з 2003 р. і входив до програми Олімпійських ігор 2008 р. в Пекіні.

В сучасному світі, де пандемії, війни та інші надзвичайні події стали частими явищами, підтримка фізичного здоров'я та контроль за фізичними навантаженнями стають все важливішими завданнями. Особливо це стосується спортсменів, які мають тренуватись і підтримувати свої показники незалежно від умов зовнішнього середовища.

У зв'язку з цим, велика увага приділяється розробці програмного забезпечення для моніторингу фізичних навантажень велоспортсменів, що дозволило б покращити якість підготовки наших майбутніх олімпійських призерів.

Метою даної роботи є оцінювання фізичного навантаження велоспортсмена за допомогою розробленого програмного забезпечення шляхом обробки статистичних даних та їх подальшої візуалізації у тому вигляді, що буде зрозумілий людині.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести аналіз предметної області та існуючих аналогів;
– обрати інструменти, необхідні для створення застосунку;
– знайти дані для подальшого оброблення;
– визначити та реалізувати основний функціонал застосунку;
– реалізувати відображення результатів обробки;

– створити документацію.

Об’єктом дослідження являється процес обробки даних (Data Set) щодо оцінювання фізичного навантаження велоспортсмена.

Предметом дослідження є методи та засоби обробки великих обсягів даних (Big Data) за допомогою спеціалізованого програмного забезпечення (ПЗ) для оцінювання та візуалізації фізичного навантаження.

Актуальністю розробки даного програмного забезпечення є те, що завдяки йому тренери зможуть швидше давати оцінку фізичної підготовки спортсмена, що дозволить пришвидшити та покращити коригування загальних відновлювальних процесів для організму спортсмена.

Розробка даного ПЗ є необхідною як акт удосконалення наявних напрацювань з галузі моніторингу та оцінювання фізичних навантажень та структуризація їх у єдину інформаційну систему (ІС), в якій будуть усі наявні методи та інструменти оцінки відновлення здоров’я спортсменів.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення та результати бакалаврської кваліфікаційної роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання – 2022» [2].

Робота складається з вступу, чотирьох фахових розділів, висновків, переліку джерел посилання з 27 джерел та 3 додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний опис фізичні навантаження для велосипедиста

Фізичні навантаження – будь-який довільний рух тіла, що виробляється скелетними м’язами та вимагає енергетичних витрат [3].

Фізичні навантаження є невід'ємною частиною спортивної підготовки спортсменів у будь-якій дисципліні, включаючи велоспорт. Ці навантаження включають в себе різні види тренувань, які спрямовані на розвиток фізичної форми, підвищення витривалості, швидкості, сили та інших фізичних якостей спортсмена.

Фізичні навантаження можуть бути різної інтенсивності та тривалості, тому контроль за ними є дуже важливим завданням для тренерів та спортсменів. Вірний підхід до тренувань дозволяє досягнути максимальних результатів та зменшити ризик отримання травм.

Для оцінювання фізичної підготовки використовуються різні шкали оцінювання, основними із яких є:

– шкала FTP;
– шкала Вамса (VAMs);
– шкала Фостера (FTS);
– шкала Ратінга спроможності (RPE);
– шкала Борхардта.

Для досягнення оптимальних результатів у велоспорті, тренування повинні бути ретельно плановані та контрольовані. При цьому важливо враховувати особливості кожного велосипедиста, його фізичні можливості та мету тренувань.

Для контролю за фізичними навантаженнями спортсменів використовують різні методи, включаючи вимірювання пульсу, рівня кисню та інших фізіологічних показників. Застосування програмного забезпечення для моніторингу фізичних навантажень дозволяє автоматизувати цей процес та зробити його більш точним та ефективним.

1.2 Аналіз шкал для оцінювання фізичного стану спортсмена

Шкала за рівнем показника **FTP** (Functional Threshold Power) зазвичай використовується для оцінювання максимальної потужності, яку велосипедист може виробляти впродовж однієї години без відчуття значного втомлення (рис. 1.1) [4].

Показники ФПМ для велоспорту	(Вт/кг)
Чудово	> 5.04
Відмінно	3.93 – 5.04
добре	2.79 – 3.92
Задовільно	2.23 – 2.78
Непідготовлений	< 2.23

Рисунок 1.1 – Шкала оцінювання максимальної потужності велосипедиста

Дана шкала є важливим показником для велосипедистів, оскільки він вказує на максимальну потужність, яку спортсмен може підтримувати впродовж приблизно години без відчуття значного втомлення, не перевищуючи порогу лактату.

Поріг лактату – це показник, який вказує на рівень молочної кислоти в крові велосипедиста. Коли молочна кислота в крові велика, то це свідчить про перевантаження м'язів та може привести до втоми. Поріг лактату вимірюється в мілімолях на літр крові (ммоль/л).

FTP вимірюється в ватах (W) на кілограм ваги тіла (W/kg) і вказує на максимальну потужність, яку велосипедист може утримувати протягом однієї години без перевантаження та відчуття втоми, що обумовлено аеробними процесами в організмі.

Для вимірювання FTP зазвичай використовують спеціальні тести на велотренажері або вуличному велосипеді. У тесті велосипедист повинен проїхати 20-хвилинну розділку з максимально можливою інтенсивністю, під час тесту потужність та пульс вимірюються регулярно. Потім використовуючи дані з цих тестів, можна розрахувати FTP.

FTP є важливим показником для велосипедистів, оскільки він дозволяє визначити оптимальні зони тренувань та розробляти програми тренувань для підвищення фізичної підготовки та досягнення максимальних результатів.

Шкала Вамса (VAMs) – це метод оцінки фізичної підготовки велосипедистів, який використовується для вимірювання швидкості підйому на велосипеді [5].

VAMs – це скорочення від «Velocita Ascensionale Media», що означає середню підйомну швидкість на італійській мові. Цей метод був розроблений італійським велосипедистом та тренером Жаном Карлосом Вамсом та є популярним серед професійних велосипедистів.

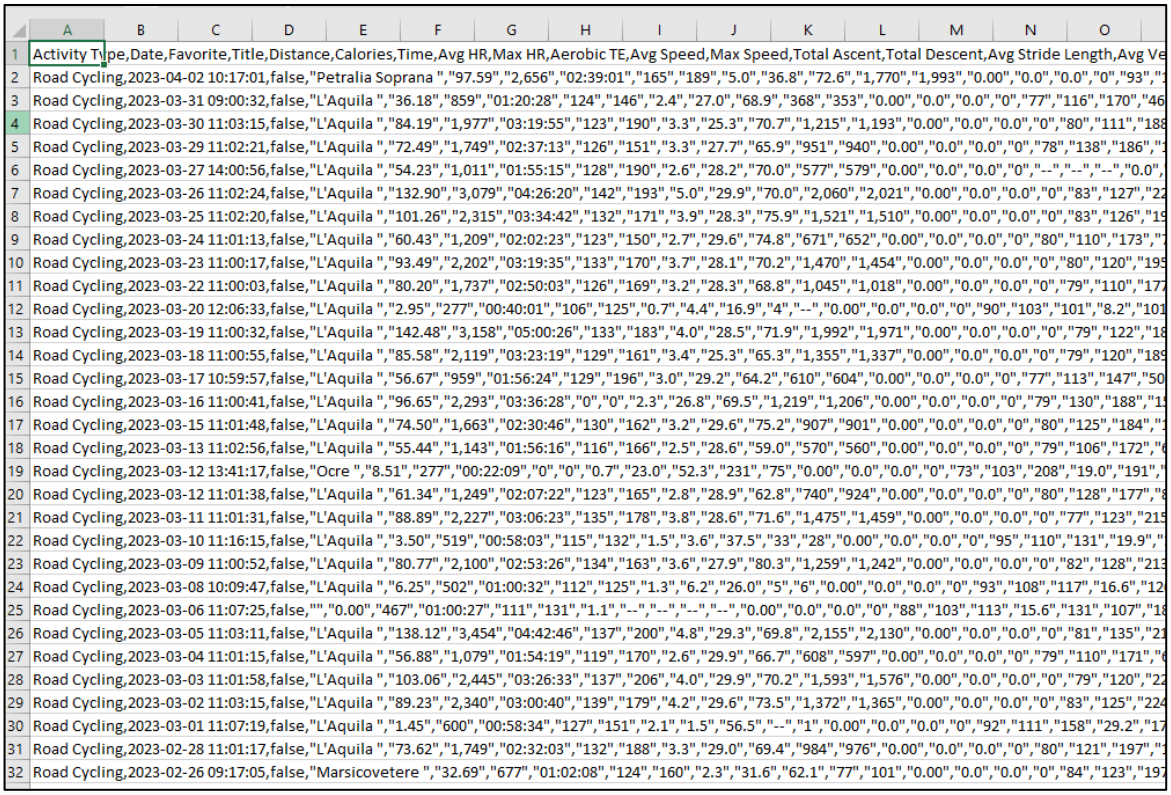
Для вимірювання VAMs потрібно виконати підйом на велосипеді на певну відстань з певним висотним перепадом. За допомогою спеціального обладнання, такого як GPS або велосипедний комп'ютер, вимірюється час, який потрібно, щоб пройти цю відстань. За допомогою формули VAMs вираховується швидкість підйому на велосипеді в метрах на годину.

Оцінка VAMs дає змогу визначити, наскільки швидко велосипедист може підніматися на певну висоту та може використовуватися для порівняння фізичної підготовки між велосипедистами. Цей метод також допомагає велосипедистам визначити свої сильні та слабкі сторони у підйомах та розвивати свою фізичну підготовку залежно від цих результатів.

Шкала Вамса не є стандартною та не використовується всіма велосипедистами, але може бути корисним інструментом для професійних велосипедистів, які бажають покращити свою фізичну підготовку для підйомів на велосипеді.

1.3 Огляд та аналіз наявних аналогів обробки даних

Також можна помітити, що доволі незручно проводити тестування, оскільки в науковій і статистичній діяльності зазвичай використовують файли з розширенням *.csv* [6], з якими можна працювати за допомогою Microsoft Excel, але це є доволі незручним через те, що значення розділені комами, і тому ускладнюється робота з ними як з матрицею. На рис. 1.2 наведено вміст файлу, до якого з обладнання для вимірів, записуються необхідні показники спортсмена, що потім використовуються для аналізу фізичної підготовки.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Activity Type,Date,Favorite,Title,Distance,Calories,Time,Avg HR,Max HR,Aerobic TE,Avg Speed,Max Speed,Total Ascent,Total Descent,Avg Stride Length,Avg Ve														
2	Road Cycling,2023-04-02 10:17:01,false,"Petralia Soprana ","97.59","2,656","02:39:01","165","189","5.0","36.8","72.6","1,770","1,993","0.00","0.0","0.0","0","93","1														
3	Road Cycling,2023-03-31 09:00:32,false,"L'Aquila ","36.18","859","01:20:28","124","146","2.4","27.0","68.9","368","353","0.00","0.0","0.0","0","77","116","170","46														
4	Road Cycling,2023-03-30 11:03:15,false,"L'Aquila ","84.19","1,977","03:19:55","123","190","3.3","25.3","70.7","1,215","1,193","0.00","0.0","0.0","0","80","111","188														
5	Road Cycling,2023-03-29 11:02:21,false,"L'Aquila ","72.49","1,749","02:37:13","126","151","3.3","27.7","65.9","951","940","0.00","0.0","0.0","0","78","138","186","1														
6	Road Cycling,2023-03-27 14:00:56,false,"L'Aquila ","54.23","1,011","01:55:15","128","190","2.6","28.2","70.0","577","579","0.00","0.0","0.0","0","--","--","--","0.0",														
7	Road Cycling,2023-03-26 11:02:24,false,"L'Aquila ","132.90","3,079","04:26:20","142","193","5.0","29.9","70.0","2,060","2,021","0.00","0.0","0.0","0","83","127","22														
8	Road Cycling,2023-03-25 11:02:20,false,"L'Aquila ","101.26","2,315","03:34:42","132","171","3.9","28.3","75.9","1,521","1,510","0.00","0.0","0.0","0","83","126","19														
9	Road Cycling,2023-03-24 11:01:13,false,"L'Aquila ","60.43","1,209","02:02:23","123","150","2.7","29.6","74.8","671","652","0.00","0.0","0.0","0","80","110","173","1														
10	Road Cycling,2023-03-23 11:00:17,false,"L'Aquila ","93.49","2,202","03:19:35","133","170","3.7","28.1","70.2","1,470","1,454","0.00","0.0","0.0","0","80","120","195														
11	Road Cycling,2023-03-22 11:00:03,false,"L'Aquila ","80.20","1,737","02:50:03","126","169","3.2","28.3","68.8","1,045","1,018","0.00","0.0","0.0","0","79","110","17														
12	Road Cycling,2023-03-20 12:06:33,false,"L'Aquila ","2.95","277","00:40:01","106","125","0.7","4.4","16.9","4","--","0.00","0.0","0.0","0","90","103","101","8.2","101														
13	Road Cycling,2023-03-19 11:00:32,false,"L'Aquila ","142.48","3,158","05:00:26","133","183","4.0","28.5","71.9","1,992","1,971","0.00","0.0","0.0","0","79","122","18														
14	Road Cycling,2023-03-18 11:00:55,false,"L'Aquila ","85.58","2,119","03:23:19","129","161","3.4","25.3","65.3","1,355","1,337","0.00","0.0","0.0","0","79","120","189														
15	Road Cycling,2023-03-17 10:59:57,false,"L'Aquila ","56.67","959","01:56:24","129","196","3.0","29.2","64.2","610","604","0.00","0.0","0.0","0","77","113","147","50														
16	Road Cycling,2023-03-16 11:00:41,false,"L'Aquila ","96.65","2,293","03:36:28","0","0","2.3","26.8","69.5","1,219","1,206","0.00","0.0","0.0","0","79","130","188","1														
17	Road Cycling,2023-03-15 11:01:48,false,"L'Aquila ","74.50","1,663","02:30:46","130","162","3.2","29.6","75.2","907","901","0.00","0.0","0.0","0","80","125","184","1														
18	Road Cycling,2023-03-13 11:02:56,false,"L'Aquila ","55.44","1,143","01:56:16","116","166","2.5","28.6","59.0","570","560","0.00","0.0","0.0","0","79","106","172","6														
19	Road Cycling,2023-03-12 13:41:17,false,"Ocre ","8.51","277","00:22:09","0","0","0.7","23.0","52.3","231","75","0.00","0.0","0.0","0","73","103","208","19.0","191",														
20	Road Cycling,2023-03-12 11:01:38,false,"L'Aquila ","61.34","1,249","02:07:22","123","165","2.8","28.9","62.8","740","924","0.00","0.0","0.0","0","80","128","177","8														
21	Road Cycling,2023-03-11 11:01:31,false,"L'Aquila ","88.89","2,227","03:06:23","135","178","3.8","28.6","71.6","1,475","1,459","0.00","0.0","0.0","0","77","123","215														
22	Road Cycling,2023-03-10 11:16:15,false,"L'Aquila ","3.50","519","00:58:03","115","132","1.5","3.6","37.5","33","28","0.00","0.0","0.0","0","95","110","131","19.9","														
23	Road Cycling,2023-03-09 11:00:52,false,"L'Aquila ","80.77","2,100","02:53:26","134","163","3.6","27.9","80.3","1,259","1,242","0.00","0.0","0.0","0","82","128","213														
24	Road Cycling,2023-03-08 10:09:47,false,"L'Aquila ","6.25","502","01:00:32","112","125","1.3","6.2","26.0","5","6","0.00","0.0","0.0","0","93","108","117","16.6","12														
25	Road Cycling,2023-03-06 11:07:25,false,"","0.00","467","01:00:27","111","131","1.1","--","--","--","--","0.00","0.0","0.0","0","88","103","113","15.6","131","107","18														
26	Road Cycling,2023-03-05 11:03:11,false,"L'Aquila ","138.12","3,454","04:42:46","137","200","4.8","29.3","69.8","2,155","2,130","0.00","0.0","0.0","0","81","135","21														
27	Road Cycling,2023-03-04 11:01:15,false,"L'Aquila ","56.88","1,079","01:54:19","119","170","2.6","29.9","66.7","608","597","0.00","0.0","0.0","0","79","110","171","6														
28	Road Cycling,2023-03-03 11:01:58,false,"L'Aquila ","103.06","2,445","03:26:33","137","206","4.0","29.9","70.2","1,593","1,576","0.00","0.0","0.0","0","79","120","22														
29	Road Cycling,2023-03-02 11:03:15,false,"L'Aquila ","89.23","2,340","03:00:40","139","179","4.2","29.6","73.5","1,372","1,365","0.00","0.0","0.0","0","83","125","224														
30	Road Cycling,2023-03-01 11:07:19,false,"L'Aquila ","1.45","600","00:58:34","127","151","2.1","1.5","56.5","--","1","0.00","0.0","0.0","0","92","111","158","29.2","17														
31	Road Cycling,2023-02-28 11:01:17,false,"L'Aquila ","73.62","1,749","02:32:03","132","188","3.3","29.0","69.4","984","976","0.00","0.0","0.0","0","80","121","197","1														
32	Road Cycling,2023-02-26 09:17:05,false,"Marsicovetere ","32.69","677","01:02:08","124","160","2.3","31.6","62.1","77","101","0.00","0.0","0.0","0","84","123","197														

Рисунок 1.2 – Приклад вмісту csv-файлу

Приклади інструментів для роботи з цим типом файлів будуть наведені в наступному розділі.

1.3.1 Вебзастосунок Garmin connect

Сайт Garmin connect [7] представляє з себе вебформу з можливістю графічної візуалізації даних з обладнання для зняття показників та характеристик спортсмена (рис. 1.3).

Рисунок 1.3 – Візуалізація даних вебзастосунком Garmin

Переваги:

1) доступність з будь-яких пристроїв;

2) не потрібно встановлювати додаткового ПЗ.

Недоліки:

1) відсутність налаштування застосунку для моніторингу та відсутність вибору методів/шкал для аналізу;

2) надаються лише чисельне значення без пояснення;

3) відсутність ведення статистики та зберігання приміток.

1.3.2 Застосунок Endomondo

Застосунок Endomondo розроблений задля відстежування численних фітнес-атрибут і доступний для скачування в Play Market (Android) [8] або App Store(iOS) [9]. Програмне забезпечення може допомогти проаналізувати продуктивність і запропонувати покращення. Інтерфейс застосунку наведений на рис. 1.4.

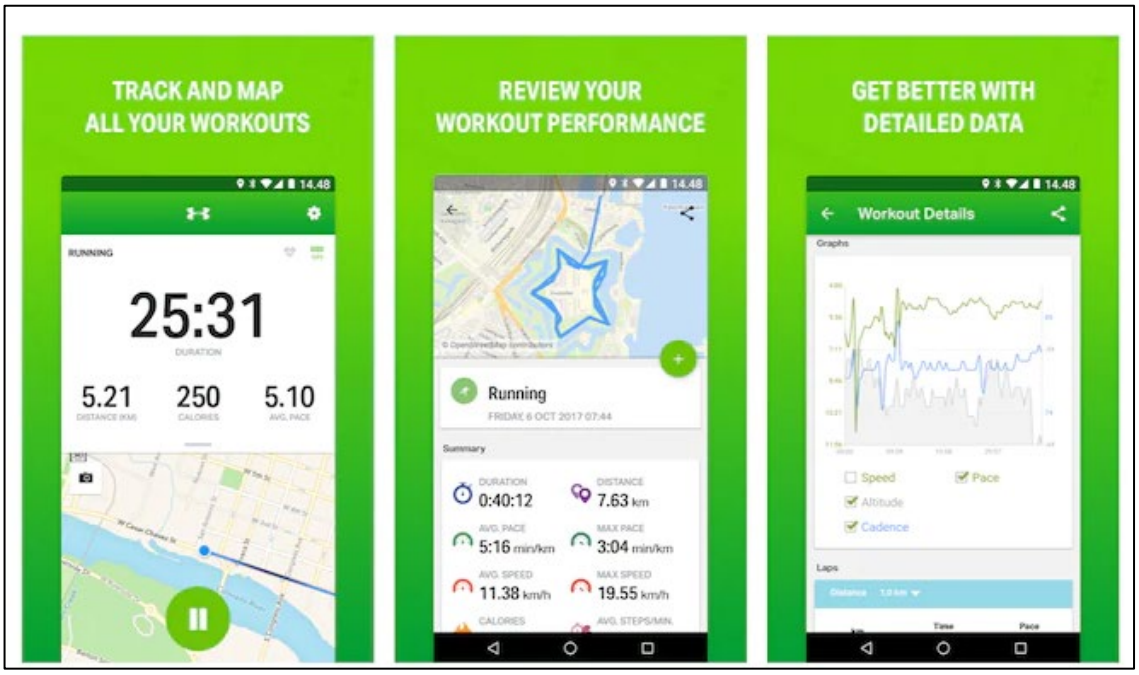


Рисунок 1.4 – Інтерфейс застосунку Endomondo [8-9]

Переваги:

1) візуалізації даних;

2) ведення статистики;

3) можливість допомогти проаналізувати продуктивність і запропонувати покращення.

Недоліки:

1) застосунок більше підходить для аматорів, які хочуть просто підтримувати своє здоров’я в тонусі;

2) відсутність доступу тренерів до застосунку та даних (показників);

3) відсутність аналізу показників саме для велосипедистів.

1.4 Особливості даних, зібраних під час тренування спортсменів

Big Data – це дуже великі та складні набори даних, які не можуть бути оброблені або проаналізовані за допомогою традиційних технік обробки даних [10].

Три особливості великих даних:

– **обсяг** даних має серйозне значення, при роботі з великими даними доводиться обробляти великі обсяги неструктурованих даних з низькою щільністю. Це можуть бути дані невідомої цінності як, наприклад, транзакції, соціальні мережі, обладнання з підтримкою датчиків тощо;

– **швидкість** отримання даних також є значно більшою, наприклад, у застосунках, які працюють у режимі реального часу, що означає, що їх треба обробляти швидко, щоб отримати корисну інформацію. Наприклад, це можуть бути дані, зібрані з мережі датчиків або дані з соціальних мереж, де важлива швидкість реакції;

– **різноманітність** відноситься до багатьох типів доступних даних. І якщо традиційні типи даних були структуровані і вписувались в реляційну базу даних, то з появою великих даних дані з’являються в нових неструктурованих типах, наприклад такі як текст, відео та аудіо, що вимагають додаткової попередньої обробки для отримання значення та підтримки метаданих.

Також великі дані використовуються в машинному навчанні, що дозволяє іноді знаходити нові закономірності в даних або дозволяє швидше та якісніше їх обробляти.

1.5 Специфікація вимог до створюваного проєкту

1.5.1 Призначення та межі проєкту

Призначення ІС (застосунку), для якої розробляється програмне забезпечення

Система призначена для обробки статистичних даних та їх подальшої візуалізації задля полегшення оцінювання фізичної підготовки та корегування тренувань велоспортсмена.

Погодження, що ухвалені в програмній документації

Було погоджено, що для роботи з даними будуть використовуватись бібліотеки Pandas та NumPy, оскільки вони надають більш продвинуті структури даних та методи для роботи з ними.

Межі проєкту ПЗ

Проєкт має реалізовувати функції, що були узгоджені при проєктуванні ПЗ та вирішувати головні недоліки застосунків-аналогів, а саме: відсутність візуалізації даних та більш глибокого аналізу фізичної підготовки спортсмена.

1.5.2 Загальний опис

Сфера застосування

Створюваний проєкт має використостовуватись тренерами ДЮСШ та училищами фізичної культури.

Загальна структура і склад ІС

Система має складатись з прошарку для роботи з даними, а також інтерфейсу, що поєднував би візуал з функціоналом.

1.5.3 Вимоги до інформаційного забезпечення

Джерела і зміст вхідної інформації (даних)

Джерелами є відкриті дані отримані зі спеціального обладнання Garmin.

Вимоги до способів організації, збереження та ведення інформації

Інформація має зберігатись в нереляціонній базі даних або у відповідних *.csv* файлах, що використовуються у науковій сфері.

1.5.4 Вимоги до програмного забезпечення

Архітектура ІС

Програмне забезпечення ІС має складатися з двох ланок: клієнтський застосунок та прошарок роботи з даними.

Мова і технологія розробки ПЗ

Мовою розробки було обрано Python як для розрахунків, так і для реалізації програмного інтерфейсу з використанням відповідних бібліотек.

1.5.5 Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

Користувацький інтерфейс має задовольняти стандартні UI та UX - вимоги.

1.5.6 Властивості програмного забезпечення

Доступність

Від ПЗ вимагається доступність для будь-якого користувача, в якого є наявні бібліотеки та дані, що можуть оброблятись засобами ІС.

Супроводжуваність

Комплекс застосунків має бути легко супроводжуваним та придатним для подальшої модифікації чи доповнення.

Переносимість

Програмне забезпечення має бути кросплатформеним. Клієнтські застосунки можуть бути використані будь-яким ПК.

Надійність

Створюваний проєкт має належним чином оброблювати помилки та припиняти свою роботу у разі виникнення нефатальних помилок та демонструвати

їх у сприятливій для людського читання формі. Потенційно небезпечні для ІС помилки не мають бути показані користувачам.

Безпека

Дані з досліджень мають бути надійно захищені, відповідно до статті 11 Закону України «Про захист персональних даних» [11].

Висновки до розділу 1

В результаті написання першого розділу було наведено основну інформацію, що необхідна для більш повного та поглибленого розуміння зазначеної теми. Описано загальну інформацію щодо методів аналізу фізичної підготовки спортсменів.

Проведено аналіз наявних шкал для оцінювання аналізу фізичної підготовки, робота з якими в подальшому дозволить проводити більш глибокий та інтенсивний аналіз даних задля їх обробітку та надання відповідних результатів оцінювання.

Наведено одні з основних систем-аналогів, що в певній мірі реалізовують функціонал розроблюваної ІС, але при цьому мають низку критично важливих недоліків, вирішення яких є однією з задач дипломної роботи.

Також надано основну інформацію для розуміння того, чим являються «Big Data» і чому робота з ними відрізняється від роботи зі звичайними даними.

Сформовано специфікацію вимог до розроблюваного програмного забезпечення задля більш чіткого представлення та опису його поведінки.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Ключові особливості обробки Big Data

Архітектура програмного забезпечення для роботи з великими даними – це складне завдання. Необхідно враховувати багато факторів, наприклад обсяг, різноманітність і швидкість даних, а також конкретні потреби організації. Однак є деякі загальні принципи, яких можна дотримуватися, щоб створити успішну архітектуру програмного забезпечення для Big Data.

Однією з найважливіших міркувань є масштабованість. Системи великих даних повинні мати можливість обробляти великі обсяги даних, не перевантажуючись. Це означає використання масштабованих технологій, таких як розподілена обробка.

Ще один важливий аспект – це гнучкість. Системи великих даних повинні мати можливість обробляти різні типи даних, включаючи структуровані, напівструктуровані та неструктуровані дані. Це означає використання гнучких технологій.

Нарешті, системи великих даних повинні мати можливість обробляти дані в реальному часі. Це стає все більш важливим, оскільки все більше даних генерується в режимі реального часу. Це означає використання технологій, які можуть швидко обробляти дані, наприклад Numpy.

Big Data зазвичай швидко оновлюються та неструктуровані, тому інструменти для роботи з ними відрізняються від традиційних статистичних систем. Крім того, дані часто є невзаємопов’язаними, що дуже ускладнює подальшу обробку.

На рис. 2.1 показано архітектуру програми для роботи з Big Data, яка адаптована до вимог програмного забезпечення з аналізу даних для оцінки фізичних показників [12].

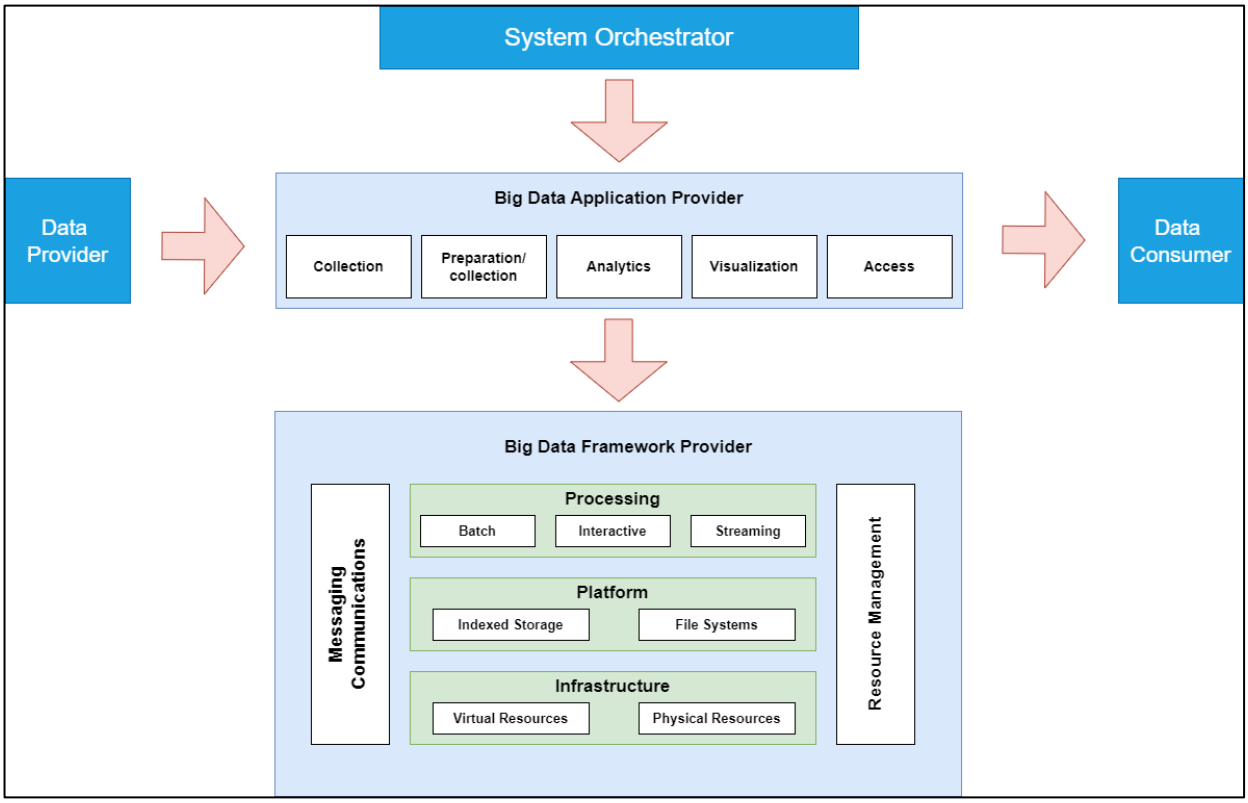


Рисунок 2.1 – Архітектура ПЗ для роботи з Big Data

Опис головних компонентів схеми наведеної вище :

– System Orchestrator гарантує, що різні програми, дані та компоненти інфраструктури працюють разом;

– Data Provider надає нові дані, що надходять з різних джерел, у систему великих даних для перетворення їх у оброблювані Data Set-и [13];

– Big Data Application Provider містить бізнес-логіку та функціональні можливості, необхідні для перетворення даних у цінні знання за допомогою п’яти основних дій:

1) збір;
2) підготовка;
3) аналітика;
4) візуалізація;
5) доступ.

– Big Data Framework Provider має ресурси та сервіси для зберігання та обробки даних;

– Data Consumer використовує інтерфейси сервісів, наданих Big Data Application Provider для доступу до необхідної інформації.

Також у майбутньому планується подальша розробка застосунку, удосконалення існуючих методів та додавання нових функцій, збір інформації та її глибока обробка, тому було обрано саме цю архітектуру ПЗ. Оскільки вона надає всі можливості для масштабованості.

2.2 Основні інструменти моделювання ПЗ

UML – представляє собою стандартну мову моделювання, призначену для опису та проєктування складних систем, зокрема програмного забезпечення [14]. Її головна мета полягає в забезпеченні стандартизованого способу візуалізації дизайну системи. Головним завданням UML є створення нормалізованого засобу комунікації між розробниками та замовниками, що працюють над проєктом. Розробники використовують UML для наступних цілей:

– вказівки щодо послідовності дій команди;

– демонстрація необхідних функцій для розробки;

– спрямування окремих задач для розробників та команди в цілому;

– встановлення критеріїв для моніторингу та оцінки продуктів і діяльності проєкту.

Використання UML в розробці програмного забезпечення є дуже важливим, оскільки головною перевагою є стандартизація мови, що дозволяє легко читати та розуміти створені діаграми, а також навіть генерувати код з діаграми класів за допомогою спеціальних інструментів [15].

В даній роботі будуть наведені конкретні приклади UML – діаграм, такі як діаграми класів (рис. 2.2), діаграма потоків даних та діаграми використання. Використання всіх цих діаграм разом дозволить створити більш повне та чітке уявлення про створюваний продукт.

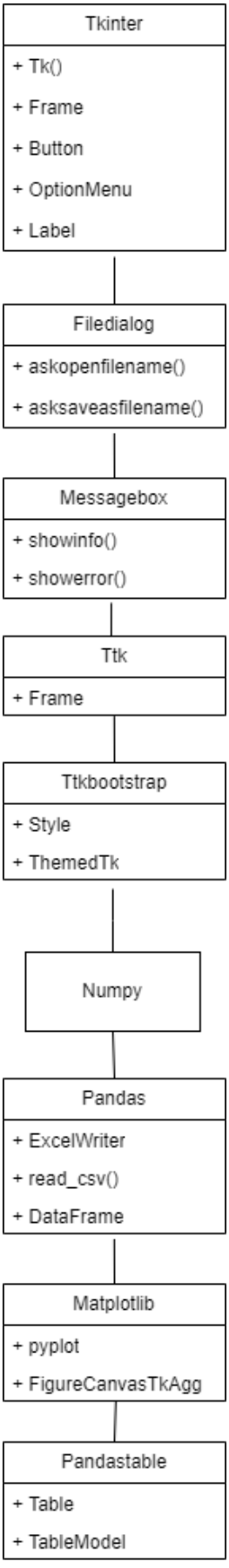


Рисунок 2.2 – UML-діаграма класів

Ця діаграма показує взаємозв'язок між різними класами, використовуваними в коді. Класи залежать від модулів «Tkinter», «Filedialog», «Messagebox», «Ttk», «Ttkbootstrap», «Numpy», «Pandas», «Matplotlib» і «Pandastable», вони виконують різні функції для створення графічного інтерфейсу користувача, завантаження та обробки даних, побудови графіків та аналізу даних за шкалами.

2.3 Розробка діаграми використання

Після розгляду поставлених задач та проведених оцінок застосунків-аналогів, було сформовано можливості використання застосунку користувачем, що зазначені на рис. 2.3.

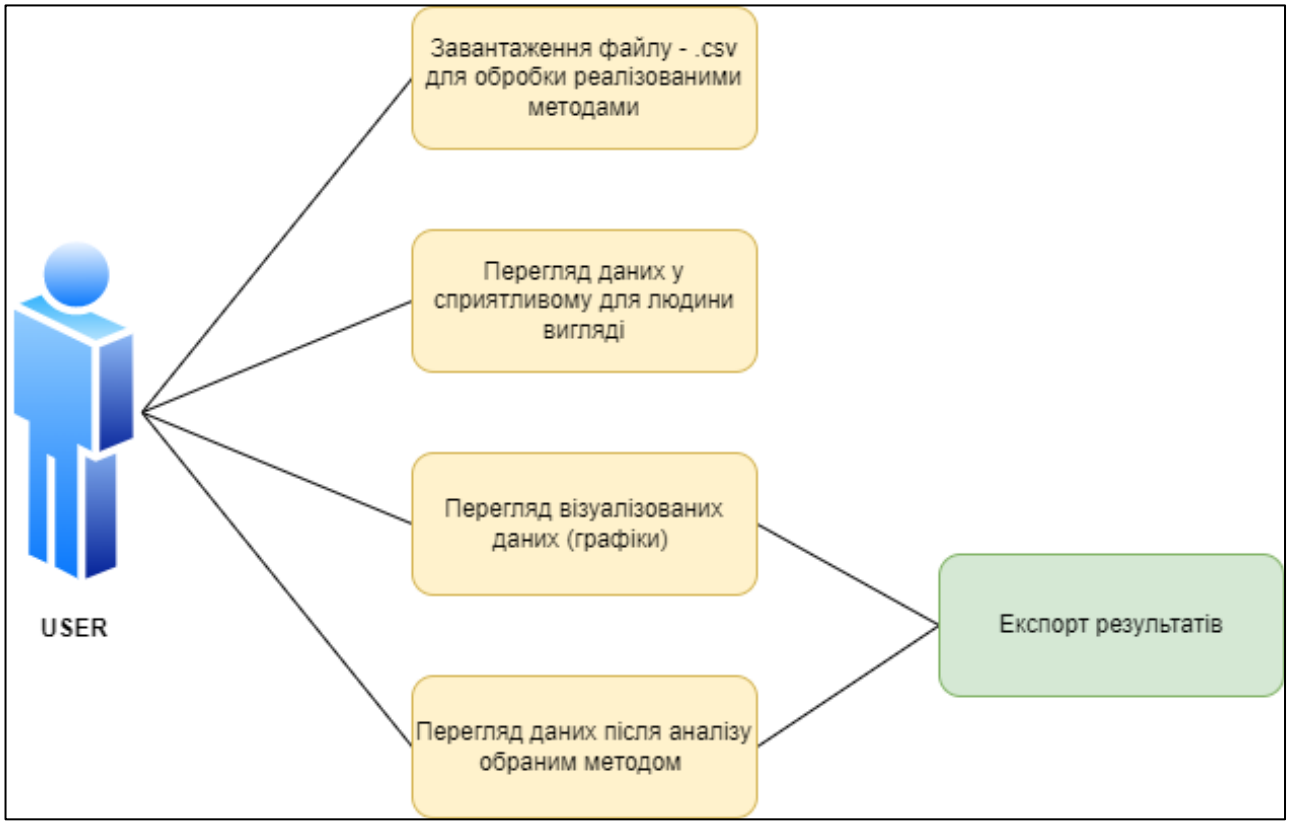


Рисунок 2.3 – Діаграма використання розроблюваного застосунку

Задля надання більш детальної інформації щодо можливостей взаємодії користувача з застосунком, було створено табл. 2.1–2.5 з описом кожної з подій.

Таблиця 2.1 – Завантаження файлу для обробки

Назва	Завантаження файлу для обробки
ID	1
Короткий опис	Користувач вибирає власний файл, який у подальшому може бути оброблений реалізованими методами з обробки даних
Головні актори	Користувач
Другорядні актори	Відсутні
Передумови	Наявність у користувача файлу з відповідним розширенням для подальшої обробки ПЗ
Постумови	Система завантажує обраний файл та проводить відповідні обробки даних
Основний сценарій	1) Користувач за допомогою відповідного меню обирає файл з власної файлової системи. 2) Система завантажує файл. 3) Система проводить необхідні дії для подальшої роботи обробки даних.
Альтернативні сценарії	У разі виникнення помилки ІС повідомляє про неї, інформуючи в залежності від того, на якому етапі помилка виникла. Попередньо передбачені помилки, що можуть виникнути на даному етапі: 1) помилка при завантаженні файлу; 2) помилка при його обробці.

Таблиця 2.2 – Перегляд даних у сприятливому для людини вигляді

Назва	Перегляд даних у адаптованому для людського перегляду форматі
ID	2
Короткий опис	Користувач переглядає вміст обраного файлу у табличному вигляді
Головні актори	Користувач
Другорядні актори	Відсутні
Передумови	Було обрано файл для перегляду і він успішно завантажений
Постумови	Вміст успішно оброблено для подальшого представлення і користувач його переглянув.
Основний сценарій	1) Користувач за допомогою відповідного меню переходить на сторінку представлення файлу. 2) Вміст файлу приводиться до шаблонного вигляду. 3) Відбувається візуалізація вмісту.
Альтернативні сценарії	У разі виникнення помилки ІС повідомляє про неї, інформуючи в залежності від того, на якому етапі помилка виникла. Попередньо передбачені помилки, що можуть виникнути на даному етапі: 1) помилка при обробці файлу; 2) помилка під час приведення його до шаблонного вигляду; 3) помилка під час візуалізації.

Таблиця 2.3 – Перегляд візуалізованих даних

Назва	Перегляд візуалізованих даних (графіків)
ID	3
Короткий опис	Користувач переглядає вміст обраного файлу у вигляді графіку
Головні актори	Користувач
Другорядні актори	Відсутні
Передумови	Було обрано файл для перегляду і він успішно завантажений
Постумови	Вміст файлу успішно оброблено для подальшого створення графіку та користувач їх переглянув
Основний сценарій	1) Користувач за допомогою відповідного меню переходить на сторінку з графіком. 2) Вміст файлу оброблюється для подальшого графічного представлення. 3) Відбувається візуалізація графіку.
Альтернативні сценарії	У разі виникнення помилки ІС повідомляє про неї, інформуючи в залежності від того, на якому етапі помилка виникла. Попередньо передбачені помилки, що можуть виникнути на даному етапі: 1) помилка при обробці файлу; 2) помилка під час обробки даних; 3) помилка під час візуалізації.

Таблиця 2.4 – Отримання результатів аналізу для оцінювання фізичної підготовки

Назва	Отримання результатів аналізу для оцінювання фізичної підготовки
ID	4
Короткий опис	Користувач переглядає результат обробки наданих даних
Головні актори	Користувач
Другорядні актори	Відсутні
Передумови	Було обрано файл для обробки і він успішно завантажений та оброблений
Постумови	Вміст успішно оброблено для подальшого аналізу та розрахунків. Результати переглянуті користувачем
Основний сценарій	1) Користувач за допомогою відповідного меню переходить на сторінку представлення результатів оцінки. 2) Відбувається обробка даних. 3) Відбувається візуалізація оброблених даних.
Альтернативні сценарії	У разі виникнення помилки ІС повідомляє про неї, інформуючи в залежності від того, на якому етапі помилка виникла. Попередньо передбачені помилки, що можуть виникнути на даному етапі: 1) помилка при обробці даних; 2) помилка під час візуалізації.

Таблиця 2.5 – Експорт результатів оцінювання

Назва	Експорт результатів оцінювання
ID	5
Короткий опис	Користувач експортує результати оцінювання
Головні актори	Користувач
Другорядні актори	Відсутні
Передумови	Було обрано файл для обробки, і він успішно завантажений, оброблений та оцінений
Постумови	Користувач успішно експортував результати проведеного оцінювання
Основний сценарій	1) Користувач натискає на кнопку експорту результатів оцінювання. 2) Відбувається експорт даних за вказаним шляхом.
Альтернативні сценарії	У разі виникнення помилки ІС повідомляє про неї, інформуючи в залежності від того, на якому етапі помилка виникла. Попередньо передбачені помилки, що можуть виникнути на даному етапі: 1) помилка під час експорту.

2.4 Розробка макету дизайну інтерфейсу застосунку

Велике значення для застосунку має ретельно продуманий дизайн, оскільки від його лаконічності, простоти та зрозумілості залежить зручність використання кінцевими користувачами продукту. З огляду на те, що розробка інтерфейсів не є новим напрямом у сфері інженерії ПЗ, були узагальнені найкращі практики (англ. Best Practices) протягом цього часу. Основні з них включають [16; 17]:

1) кольорова палітра не повинна бути тьмяною: текст повинен бути чітко видимим і вирізнятись, елементи не мають зливатись між собою, щоб користувач міг легко їх відрізнити;

2) KISS (Keep it simple, stupid) [18] – це принцип проєктування та розробки, який закликає до того, щоб зберігати рішення простим та зрозумілим. Його суть полягає у тому, щоб уникати складнощів, які можуть виникнути в процесі розробки, та спрощувати процес розуміння та використання продукту або системи. Цей принцип особливо важливий у сфері розробки програмного забезпечення, де складність систем може приводити до помилок та недоліків. Простота розробки та використання може поліпшити якість та ефективність продукту, а також сприяти більшій його розповсюдженості серед користувачів;

3) при використанні термінології важливо бути уважним і враховувати розмір словника, яким користується конкретний користувач, оскільки нерозуміння певних термінів може призвести до негативного користувацького досвіду та витрати зайвого часу на пошук значень слів;

4) елементи повинні виконувати завдання, які від них очікуються;

5) інтерфейс повинен бути пристосованим до користувача і до середовища і бути адаптованим під задачі, чи то вебсторінка, гра, чи то застосунок з наукового аналізу;

6) елементи повинні бути згруповані за своєю сутністю. Наприклад, функції роботи з файлами мають бути поєднані у спеціальному меню з назвою *File*;

7) шрифти мають бути легкими для читання на всіх типах екранів та пристосованими до різних роздільних здатностей;

8) візуальне розбиття має велику роль при роботі з ПЗ, в якого існує кілька окремих частин в інтерфейсі з різним функціоналом, тому важливим є структурування цих частин і їх відокремлення одне від одного.

Підсумувавши і прийнявши до уваги зазначені принципи, було розроблено мінімалістичні інтерфейси, зображені на рис. 2.3–2.7.

На рис. 2.3 зображено головне вікно застосунку, на якому наявна базова інформація про те, як користуватись застосунком та наведено список реалізованих методик оцінювання фізичної підготовки. В головному меню буде наведена документація для користувача, в якій будуть пояснюватись можливості створеного

програмного забезпечення та інструкція з використання реалізованих функцій оцінювання, що допоможе швидше зорієнтуватись та почати роботу із оцінювання.

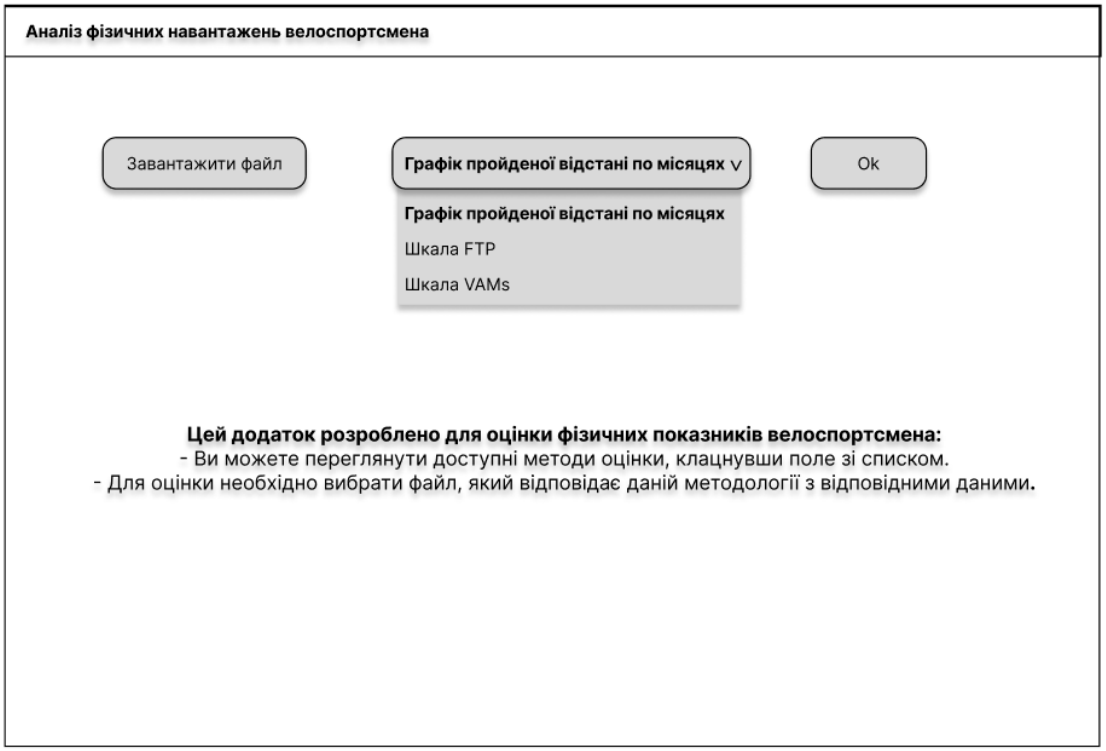


Рисунок 2.2 – Головне вікно застосунку

Оскільки застосунок універсальний і може застосовуватись для різних велоспортсменів, то було обрано рішення реалізувати вибір користувачем файлу з особистими показниками для подальшого аналізу.

На рис. 2.4 зображена сторінка з табличним представленням даних з обраного файлу. Оскільки роздільником в файлах *.csv* зазвичай є кома, то людині складно сприймати ці дані візуально, через що й було ухвалене рішення з розробки даного інтерфейсу.

Також через використання файлів з великим об’ємом було обрано рішення з реалізації їх відображення з більш малим конкретним обсягом, що буде легким для подальшого рендеру. Задля переходу між обраними проміжками додано 2 повзунки для скролінгу списку даних.

Рисунок 2.3 – Сторінка візуалізації даних з файлу

На рис. 2.5 зображена сторінка з представленням даних у вигляді графіків. На ній будуть зображені візуалізовані дані усього масиву дистанцій спортсмена із прикріпленням до часових проміжків, задля полегшення відстеження загального навантаження відповідно до пройденої дистанції.

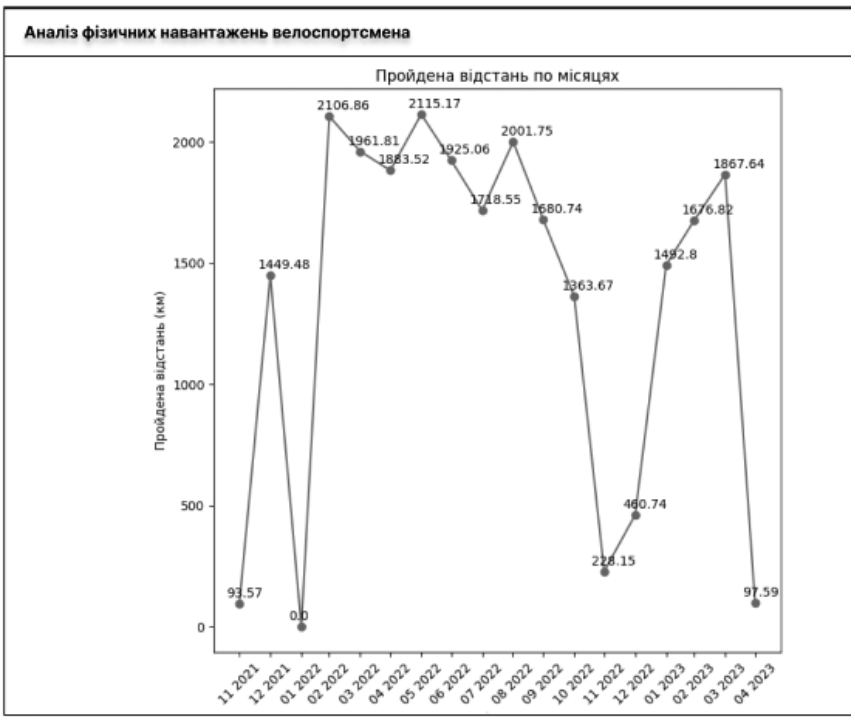
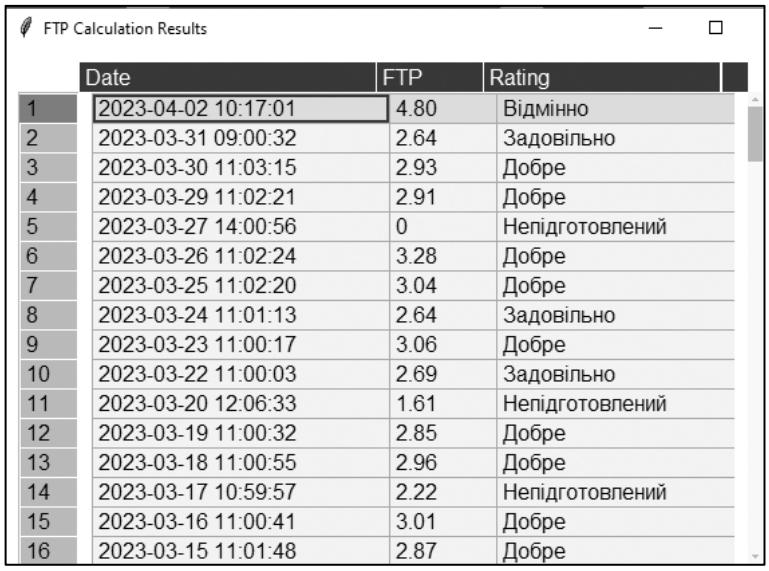


Рисунок 2.4 – Сторінка представлення даних

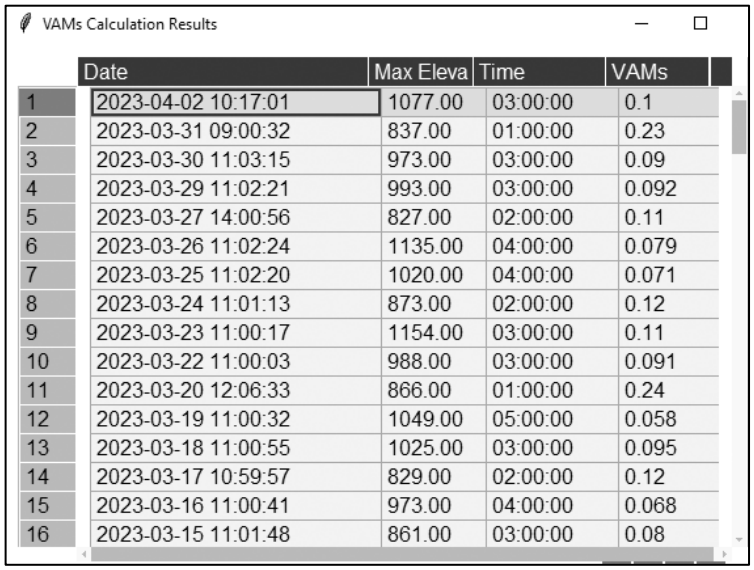
На рис. 2.6 представлена сторінка оцінки фізичної підготовки велоспортсмена за шкалою FTP. Дані будуть представлені у табличному вигляді, показуючи результати по кожному дню тренувань.



	Date	FTP	Rating
1	2023-04-02 10:17:01	4.80	Відмінно
2	2023-03-31 09:00:32	2.64	Задовільно
3	2023-03-30 11:03:15	2.93	Добре
4	2023-03-29 11:02:21	2.91	Добре
5	2023-03-27 14:00:56	0	Непідготовлений
6	2023-03-26 11:02:24	3.28	Добре
7	2023-03-25 11:02:20	3.04	Добре
8	2023-03-24 11:01:13	2.64	Задовільно
9	2023-03-23 11:00:17	3.06	Добре
10	2023-03-22 11:00:03	2.69	Задовільно
11	2023-03-20 12:06:33	1.61	Непідготовлений
12	2023-03-19 11:00:32	2.85	Добре
13	2023-03-18 11:00:55	2.96	Добре
14	2023-03-17 10:59:57	2.22	Непідготовлений
15	2023-03-16 11:00:41	3.01	Добре
16	2023-03-15 11:01:48	2.87	Добре

Рисунок 2.6 – Сторінка оцінки за шкалою FTP

На рис. 2.7 представлена сторінка оцінки фізичної підготовки велоспортсмена за шкалою VAMs. Вона має виглядає аналогічно попередній сторінці та має такий же функціонал.



	Date	Max Eleva	Time	VAMs
1	2023-04-02 10:17:01	1077.00	03:00:00	0.1
2	2023-03-31 09:00:32	837.00	01:00:00	0.23
3	2023-03-30 11:03:15	973.00	03:00:00	0.09
4	2023-03-29 11:02:21	993.00	03:00:00	0.092
5	2023-03-27 14:00:56	827.00	02:00:00	0.11
6	2023-03-26 11:02:24	1135.00	04:00:00	0.079
7	2023-03-25 11:02:20	1020.00	04:00:00	0.071
8	2023-03-24 11:01:13	873.00	02:00:00	0.12
9	2023-03-23 11:00:17	1154.00	03:00:00	0.11
10	2023-03-22 11:00:03	988.00	03:00:00	0.091
11	2023-03-20 12:06:33	866.00	01:00:00	0.24
12	2023-03-19 11:00:32	1049.00	05:00:00	0.058
13	2023-03-18 11:00:55	1025.00	03:00:00	0.095
14	2023-03-17 10:59:57	829.00	02:00:00	0.12
15	2023-03-16 11:00:41	973.00	04:00:00	0.068
16	2023-03-15 11:01:48	861.00	03:00:00	0.08

Рисунок 2.5 – Сторінка оцінки за шкалою VAMs

Наведені вище прототипи майже повністю зображують майбутній інтерфейс застосунку, він буде мати мінімалістичний дизайн та повністю відповідати нашим потребам.

Висновки до розділу 2

В результаті написання другого розділу БКР було наведено більш докладну інформацію щодо роботи з Big Data поза контекстом конкретної мови та модулів.

Також описано основний інструмент розробки діаграм, що використовується у сфері розробки ПЗ, а саме UML. Наведено створену діаграму використання розроблюваного продукту з детальним описом кожної з подій у вигляді таблиці, в якій подана основна інформація про прецедент, що складається з передумов та постумов, основного та альтернативних сценаріїв.

Наведено основну інформацію, яка є необхідною для створення зручних і легких графічних інтерфейсів. Розроблено макети графічного інтерфейсу застосунку згідно з усіма пунктами, зазначеними в відповідному підрозділі. Для кожного із макетів докладно описано його значення та особливості, які необхідно врахувати при його реалізації.

3 ТЕХНІЧНЕ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Пошук набору даних для аналізу

Для аналізу та оцінювання фізичних показників велоспортсмена потрібно знайти файл *.csv* з записаними даними зі спеціального обладнання та датчиків, таких як датчик каденсу [19] датчики пульсу [20] та велокомп'ютер [21]. Або можливо розробити свої модулі та датчики для збору необхідних даних та показників для подальшого аналізу.

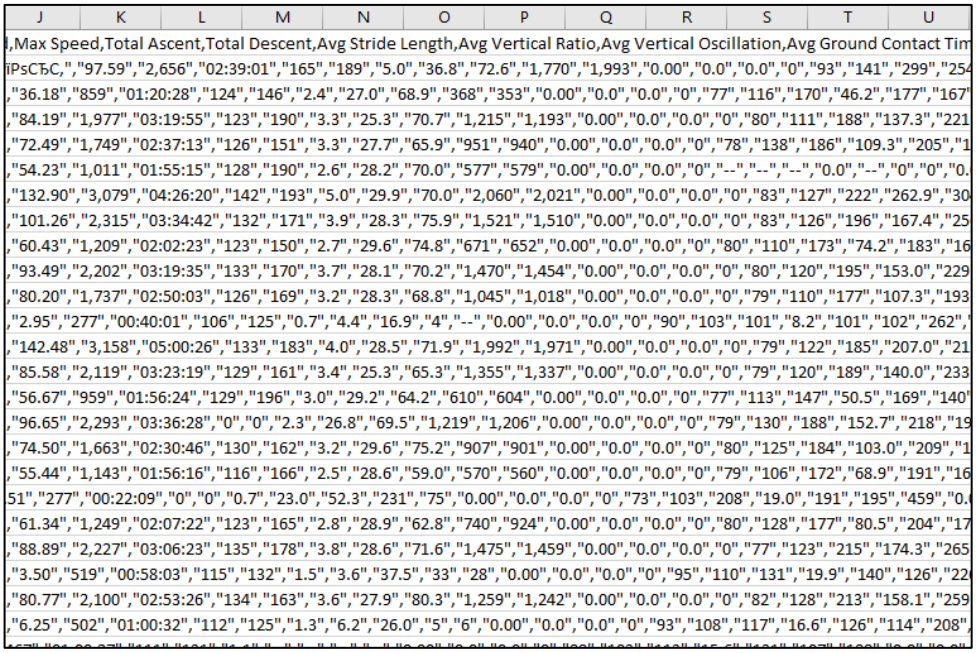
Оскільки основною задачею даної кваліфікаційної роботи є саме обробка даних, буде використано вже існуюче рішення на ринку – обладнання фірми Garmin та вебзастосунку Garmin Connect.

Garmin – відомий виробник спортивних гаджетів, таких як годинники, фітнес-браслети та GPS-пристрої, які фіксують різноманітні дані під час активності. Ці дані, такі як серцевий ритм, відстань, швидкість та багато інших параметрів, збираються і зберігаються вебзастосунком Garmin Connect. Цей застосунок надає користувачам зручний спосіб синхронізації своїх спортивних досягнень та даних про здоров'я. При вивченні цих наборів даних важливо враховувати їх якість, точність та релевантність для конкретних цілей аналізу. Завдяки обладнанню Garmin та вебзастосунку Garmin Connect можливо отримати доступ до цінної інформації шляхом експорту файлу з даними з сайту, який після обробки та аналізу розробленим ПЗ допоможе в розумінні свого спортивного прогресу та прийнятті більш обґрунтованих рішень щодо покращення своїх тренувань та здоров'я.

Також у майбутньому, планується відійти від залежності у використанні модулів від побічних виробників обладнання та розробити власні спеціалізовані модулі та датчики. Тому у основі розроблюваного ПЗ лежать такі можливості, як масштабованість та гнучкість, які у подальшому допоможуть розвивати та оновлювати функціонал та можливості застосунку та проєкту в цілому.

3.2 Опис отриманих даних

Отримані дані є файлами з розширенням *.csv*, один з яких зображений на рис. 3.1.



J	K	L	M	N	O	P	Q	R	S	T	U
,Max Speed,Total Ascent,Total Descent,Avg Stride Length,Avg Vertical Ratio,Avg Vertical Oscillation,Avg Ground Contact Tim											
iPsCЂC,","97.59","2,656","02:39:01","165","189","5.0","36.8","72.6","1,770","1,993","0.00","0.0","0.0","0","93","141","299","254											
,"36.18","859","01:20:28","124","146","2.4","27.0","68.9","368","353","0.00","0.0","0.0","0","77","116","170","46.2","177","167"											
,"84.19","1,977","03:19:55","123","190","3.3","25.3","70.7","1,215","1,193","0.00","0.0","0.0","0","80","111","188","137.3","221											
,"72.49","1,749","02:37:13","126","151","3.3","27.7","65.9","951","940","0.00","0.0","0.0","0","78","138","186","109.3","205","1											
,"54.23","1,011","01:55:15","128","190","2.6","28.2","70.0","577","579","0.00","0.0","0.0","0","--","--","--","0.0","--","0","0","0.											
,"132.90","3,079","04:26:20","142","193","5.0","29.9","70.0","2,060","2,021","0.00","0.0","0.0","0","83","127","222","262.9","30											
,"101.26","2,315","03:34:42","132","171","3.9","28.3","75.9","1,521","1,510","0.00","0.0","0.0","0","83","126","196","167.4","25											
,"60.43","1,209","02:02:23","123","150","2.7","29.6","74.8","671","652","0.00","0.0","0.0","0","80","110","173","74.2","183","16											
,"93.49","2,202","03:19:35","133","170","3.7","28.1","70.2","1,470","1,454","0.00","0.0","0.0","0","80","120","195","153.0","229											
,"80.20","1,737","02:50:03","126","169","3.2","28.3","68.8","1,045","1,018","0.00","0.0","0.0","0","79","110","177","107.3","193											
,"2.95","277","00:40:01","106","125","0.7","4.4","16.9","4","--","0.00","0.0","0.0","0","90","103","101","8.2","101","102","262",											
,"142.48","3,158","05:00:26","133","183","4.0","28.5","71.9","1,992","1,971","0.00","0.0","0.0","0","79","122","185","207.0","21											
,"85.58","2,119","03:23:19","129","161","3.4","25.3","65.3","1,355","1,337","0.00","0.0","0.0","0","79","120","189","140.0","233											
,"56.67","959","01:56:24","129","196","3.0","29.2","64.2","610","604","0.00","0.0","0.0","0","77","113","147","50.5","169","140"											
,"96.65","2,293","03:36:28","0","0","2.3","26.8","69.5","1,219","1,206","0.00","0.0","0.0","0","79","130","188","152.7","218","19											
,"74.50","1,663","02:30:46","130","162","3.2","29.6","75.2","907","901","0.00","0.0","0.0","0","80","125","184","103.0","209","1											
,"55.44","1,143","01:56:16","116","166","2.5","28.6","59.0","570","560","0.00","0.0","0.0","0","79","106","172","68.9","191","16											
51","277","00:22:09","0","0","0.7","23.0","52.3","231","75","0.00","0.0","0.0","0","73","103","208","19.0","191","195","459","0.(											
,"61.34","1,249","02:07:22","123","165","2.8","28.9","62.8","740","924","0.00","0.0","0.0","0","80","128","177","80.5","204","17											
,"88.89","2,227","03:06:23","135","178","3.8","28.6","71.6","1,475","1,459","0.00","0.0","0.0","0","77","123","215","174.3","265											
,"3.50","519","00:58:03","115","132","1.5","3.6","37.5","33","28","0.00","0.0","0.0","0","95","110","131","19.9","140","126","22(											
,"80.77","2,100","02:53:26","134","163","3.6","27.9","80.3","1,259","1,242","0.00","0.0","0.0","0","82","128","213","158.1","259											
,"6.25","502","01:00:32","112","125","1.3","6.2","26.0","5","6","0.00","0.0","0.0","0","93","108","117","16.6","126","114","208",											

Рисунок 3.1 – Вигляд отриманого файлу

Тут зібрані дані велосипедиста протягом тренувального сезону (12 місяців). В даному файлі записані показники: Activity Type, Date, Favorite, Title, Distance, Calories, Time, Avg HR, Max HR, Aerobic TE, Avg Speed, Max Speed, Total Ascent, Total Descent, Avg Stride Length, Avg Vertical Ratio, Avg Vertical Oscillation, Avg Ground Contact Time, Avg Bike Cadence, Max Bike Cadence, Normalized PowerB (NPB), Training Stress ScoreB, Max Avg Power (20 min), Avg Power, Max Power, Grit, Flow, Total Strokes, Avg Swolf, Avg Stroke Rate, Total Reps, Dive Time, Min Temp, Surface Interval, Decompression, Best Lap Time, Number of Laps,Max Temp, Avg Resp, Min Resp, Max Resp, Moving Time, Elapsed Time, Min Elevation, Max Elevation.

3.3 Вибір мови програмування

Майже всі мови програмування мають можливість працювати з Big Data, проте швидкість та зручність написання програмного забезпечення значно відрізняється залежно від конкретної мови. У цьому контексті Scala, Python, R та Go є найбільш популярними та передовими мовами програмування. Кожна з них має потужність у виконанні завдань з обробки даних.

Отже, для правильного вибору мови програмування необхідно провести аналіз переваг та недоліків кожної з них. Після перегляду оцінок та порівнянь цих мов було прийнято рішення скористатись мовою програмування Python. Існує кілька ключових особливостей, які зумовили такий вибір, включаючи:

а) розгортання Python є дуже простим, оскільки в більшості дистрибутивів, заснованих на операційній системі Linux, Python встановлюється за замовчуванням. Навіть на Windows його можна легко встановити за допомогою вбудованого магазину застосунків;

б) швидкість вивчення базового функціоналу бібліотек є вищою відносно інших мов, тому інтегрування їх в код стає більш швидким, від чого сама розробка прототипу робочого програмного забезпечення потребує значно менше часу;

в) Python має велику кількість модулів, що спрощує процес розробки та тестування програмного забезпечення;

г) завдяки наявності різних бібліотек у Python можна реалізувати різноманітні види застосунків, включаючи вебзастосунки та програми, спеціалізовані для певних операційних систем;

д) Python використовує динамічну типізацію, що дозволяє реалізовувати конструкції, які можуть бути неможливими або складними для написання у мовах зі статичною типізацією. Також, за необхідності, можна вказувати типи даних у певних ділянках коду, що сприяє уникненню проблем при використанні реалізованих функцій.

Серед недоліків Python можна зазначити, що в порівнянні з деякими іншими мовами він може бути менш ефективним при виконанні певних розрахункових

завдань. Хоча це зазвичай не помітно, при обробці великих обсягів розрахунків або масивів даних це може створити деякі труднощі. Однак, такі проблеми можна вирішити за допомогою різних модулів, які можна встановити і використовувати у коді. Ці модулі дозволяють покращити швидкість роботи Python завдяки інтеграції з мовами C++ або C.

3.4 Інструменти, необідні для реалізації застосунку

3.4.1 Бібліотека Pandas

Pandas – це бібліотека для аналізу, обробки і маніпуляції з даними. Основними перевагами даної бібліотеки є [22]:

1) швидкий та ефективний клас DataFrame для роботи з даними;
2) гнучка зміна та обробка даних;
3) легкі вставка та видалення стовбців із структури даних;
4) інструменти для читання та запису даних у різних форматах, наприклад, *.csv*, текстові файли, Microsoft Excel, бази даних SQL;
5) розумне вирівнювання даних та інтегрована обробка відсутніх даних;
6) високопродуктивне злиття та поєднання наборів даних;

Окремо необхідно відмітити високу оптимізацію, що зумовлена написанням критичних частин коду на C++ або C.

З урахуванням отриманих наборів даних, ця бібліотека є ідеальним варіантом для зручної, швидкої та простої роботи з ними. Вона надає зручні методи для обробки таблиць, а також можливість безпосереднього читання й запису файлів за допомогою цієї бібліотеки.

3.4.2 Бібліотека Matplotlib

Matplotlib – це потужна бібліотека для створення графіків у Python, яка дозволяє створювати статичні, анімовані та інтерактивні візуалізації [24]. Вона легко інтегрується з NumPy та різними графічними бібліотеками, такими як Tkinter

або PyQt. За допомогою цієї бібліотеки можна побудувати різноманітні типи графіків, включаючи:

а) базові: звичайний графік (рис. 3.2), стовбуровий, крюковий;

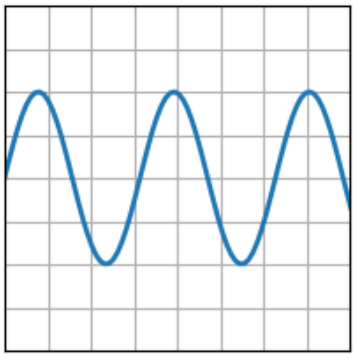


Рисунок 3.2 – Звичайний графік

б) графіки статистики: гістограма (рис. 3.3), скрипковий, пиріг;

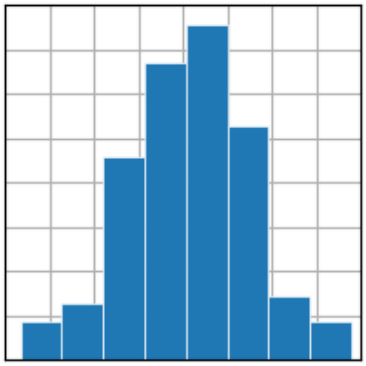


Рисунок 3.3 – Гістограма

в) ділянки масивів і полів: контурний (рис. 3.4), зубчатий, сагайдак, сітковий;

Рисунок 3.4 – Контурний графік

г) неструктуровані координати: триконтур (рис. 3.5), приколір.

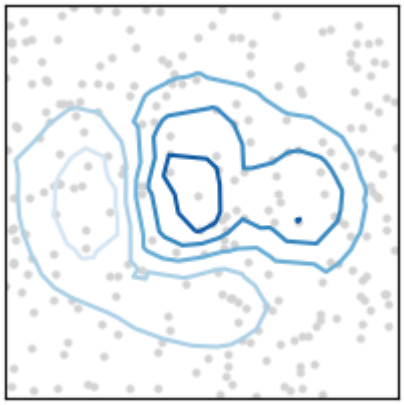


Рисунок 3.5 – Триконтур

Така кількість різноманітних функцій для візуалізації дозволить швидко адаптуватись до вхідних даних без необхідності встановлення сторонніх бібліотек, тому вибір даного модулю є доволі аргументованим.

3.4.3 Бібліотека NumPy

NumPy – це бібліотека, яка розширює стандартні структури даних шляхом додавання багатовимірних масивів та матриць, а також надає велику колекцію математичних функцій для роботи з цими масивами [23].

Більшість переваг цієї бібліотеки співпадають з описаною раніше бібліотекою Pandas, оскільки вони добре взаємодіють і часто використовуються разом для досягнення максимальної ефективності та швидкості. Легко конвертувати структури даних з однієї бібліотеки в структуру іншої, що забезпечує гнучкість коду. Основна потужність NumPy, зокрема об'єкт ndarray, який представляє багатовимірний масив, часто використовується при роботі з DataFrame. Використання вбудованих математичних функцій у NumPy спрощує роботу з таблицями, оскільки в Pandas реалізовано сумісність, що дозволяє легко та швидко обчислювати необхідні показники та застосовувати перетворення до рядків або стовбців таблиць.

3.5 Бібліотека для створення графічних інтерфейсів Tkinter

Tkinter – вбудований пакет Python, який дозволяє створювати настільні графічні інтерфейси. Він забезпечує можливість розробки кросплатформних програм, які працюватимуть на багатьох операційних системах, таких як Unix-подібні, MacOS та Windows.

Архітектура цього модулю є досить модульною, оскільки вона складається з декількох окремих модулів, кожен з яких має свій функціонал і власну офіційну документацію [25].

Tkinter є значно простішим для побудови малих настільних застосунків на відміну від основного його конкурента PyQt [26], який надає більш просунуті можливості для роботи з оформленням та стилізацією. Попри це для реалізації користувацького інтерфейсу було обрано саме Tkinter, оскільки:

а) він є безкоштовним для будь-якого комерційного використання і є відкритим вихідним кодом;

б) його легко зрозуміти та опанувати через невеликий об’єм бібліотеки;

в) він є вбудованим в Python за замовчуванням.

Це основні переваги даного пакету, але в нього також присутні певні недоліки, але вони є не критичними в розробці даного ПЗ. Одним з ключових мінусів Tkinter часто визначають дизайн, але після останніх оновлень він став виглядати краще і тому його можна відкинути. На рис. 3.6 зображено приклад дизайну застосунку. Можна помітити, що даний дизайн є доволі непоганим і мінімалістичним.

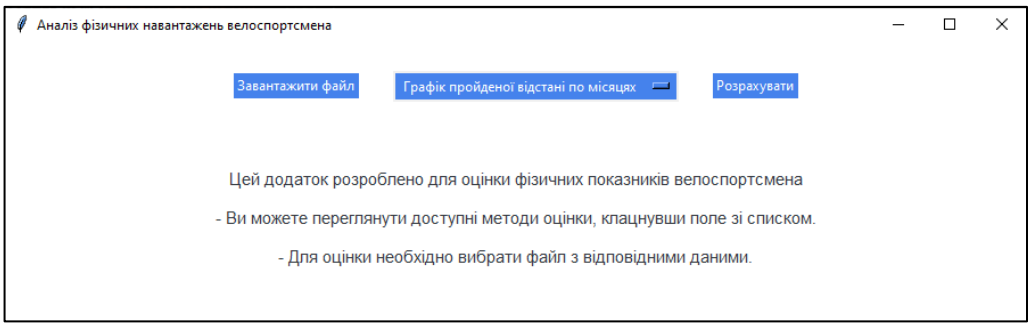


Рисунок 3.6 – Приклад дизайну застосунку з використанням Tkinter

Хоча незначним, але ще одним недоліком Tkinter є відсутність інструменту дизайну, що вимагає ручного визначення всіх елементів інтерфейсу. Проте, на щастя, існують різні бібліотеки, які можуть перетворити дизайн інтерфейсу, створений у програмі Figma [27], в реальний дизайн. Це може значно спростити та прискорити процес розробки дизайну для застосунків.

3.6 Розробка архітектури застосунку

Утримання балансу між простотою і зрозумілістю коду та гнучкістю архітектури є досить складним завданням. При розробці застосунків на мові програмування Python, правильна архітектура відіграє важливу роль у створенні ефективного і легкозмінного програмного рішення. У цьому підрозділі ми розглянемо деякі ключові аспекти розробки даного застосунку на Python.

Модульність є одним з основних принципів при розробці архітектури застосунку. Вона дозволяє розділити функціонал на окремі модулі, які можуть бути незалежно розроблені, тестовані та підтримувані. У Python модульність можна досягти за допомогою пакетів і модулів. Пакети дозволяють групувати модулі разом за спільною функціональністю, що спрощує організацію проєкту.

Правильна структура проєкту відіграє важливу роль у зручній розробці та підтримці коду. Зазвичай, варто розділити код на логічні компоненти, такі як модулі, пакети або сервіси. Кожен компонент повинен мати чітко визначену відповідальність і не залежати від деталей реалізації інших компонентів. Один з поширених шаблонів структури проєкту у Python – «модулі(функції)-проєкту» Цей шаблон спирається на розділення коду на модулі (файли) та функції (або класи) всередині цих модулів. Кожен модуль містить пов'язані функції або класи, що виконують певні завдання або відповідають за певні аспекти функціонування програми, які наведені в діаграмі на рис. 3.8.

Основна ідея полягає в тому, щоб розділити функціональність програми на окремі модулі, які виконують специфічні завдання (рис. 3.7). Це дозволяє покращити читабельність, підтримку і розширюваність проєкту.

```
def plot_distance_by_month():...

def FTP():...

def export_to_excel(data):...


def VAMs():...

def export_to_excel(data):
    save_path = filedialog.asksaveasfilename(defaultextension='.xlsx')
    if save_path:
        with ExcelWriter(save_path) as writer:
            data.to_excel(writer, index=False)
        messagebox.showinfo("Export Successful", "Data exported to Excel successfully.")
```

Рисунок 3.7 – Розбиття коду на модулі(функції)

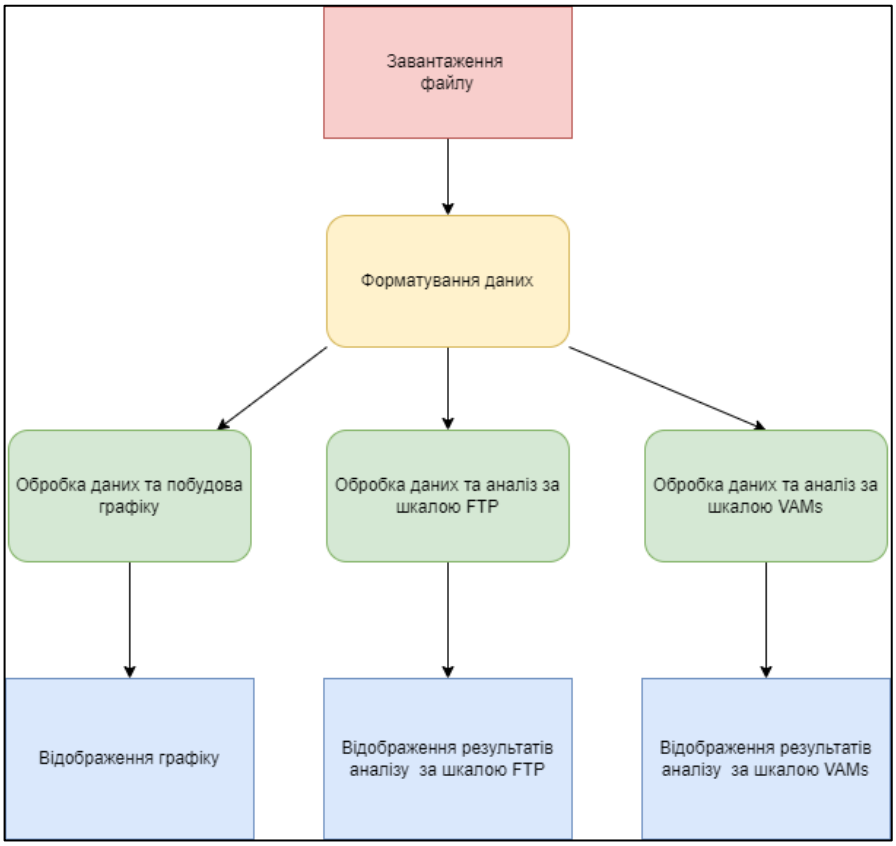


Рисунок 3.7 – Діаграма потоків даних

Саме таких принципів та шаблону структури проєкту було дотримано під час розробки застосунку з моніторингу фізичних навантажень велоспортсмена.

Висновки до розділу 3

У результаті виконання роботи над третім розділом кваліфікаційної роботи було детально описано джерело наборів даних та наведено ті проблеми, з якими довелось зіштовхнутись при їх пошуку та отриманні.

Також було описано отримані набори даних та пояснено ключові особливості роботи з ними. Було наведено інформацію про ключові інструменти, використання яких є необхідним і критично важливим для успішного виконання кваліфікаційної роботи.

Надано основну інформацію щодо бібліотеки, які будуть використовуватись для реалізації графічного інтерфейсу та функціоналу застосунку. Наведено їх плюси та мінуси, а також продемонстровано графічні елементи, які використовуються.

Описано шаблон розробки проєкту, згідно з яким буде розроблятись застосунок та наведено певні переваги в реалізації, що спричинені обраними рішеннями.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

4.1 Розробка рівня роботи з даними

Основною задачею даного прошарку є різноманітна робота та оперування даними, аби не ускладнювати розробку інтерфейсу зайвою логікою. Вирішення досягається шляхом створення функцій – методів для кожного методу оцінки фізичної підготовки, які б містили усі необхідні методи для читання, запису, перетворення даних тощо. Загальна логіка даного шару прописана в базовому класі і перевизначається у дочірніх класах. При цьому можна легко створити різні допоміжні функції всередині похідних методів, які б були локально корисними, без залежності від зовнішніх факторів.

В результаті реалізації обробки даних, було створено функції, що відповідають за кожну окрему методологію оцінки і працюють в залежності від очікуваних даних.

Першою і основною функцією є читання даних, як зображено на лістингу рис. 4.1.

```
def browse_file():
    global file_path
    file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
    print(file_path)

    if file_path:
        try:
            data = pd.read_csv(file_path)
            show_table(data)  # Відображення таблиці у фреймі
            messagebox.showinfo("Успіх", "Файл успішно завантажено")
        except:
            messagebox.showerror("Помилка",
                                 "Не вдалося завантажити файл. Перевірте, чи файл має
розширення .csv та спробуйте ще раз.")

browse_button = tk.Button(frame, text="Завантажити файл",
command=browse_file)
browse_button.grid(row=0, column=0, padx=15, pady=15)
```

Рисунок 4.1 – Лістинг функції читання даних

Дана функція являється загальним інтерфейсом, що містить в собі основні функції для роботи з читанням даних. Дана функція *browse_file()* є обробником події для кнопки «Завантажити файл». Вона викликається при натисканні кнопки і має наступний функціонал:

1) оголошується змінна *file_path* як глобальна, щоб її значення було доступне за межами функції;

2) з'являється діалогове вікно вибору файлу *(filedialog.askopenfilename)* з обмеженням типів файлів на файлові розширення .csv. Користувач може вибрати файл, який відповідає цим обмеженням. Вибраний шлях до файлу зберігається у змінній *file_path;*

3) перевіряється, чи був вибраний файл (значення *file_path* не є порожнім рядком);

4) якщо файл був вибраний, виконується наступне:

– використовуючи бібліотеку *pandas (pd),* файл зчитується у змінну *data* за допомогою функції *read_csv();*

– викликається функція *show_table(data),* яка відображає таблицю з даними у вікні програми або фреймі;

– з'являється діалогове вікно повідомлення *(messagebox.showinfo)* з повідомленням «Успіх» та текстом «Файл успішно завантажено».

5) якщо сталася помилка при завантаженні файлу (наприклад, файл не має розширення *.csv* або має неправильний формат), виконується наступне:

– з'являється діалогове вікно повідомлення про помилку *(messagebox.showerror)* з повідомленням «Помилка» та текстом «Не вдалося завантажити файл. Перевірте, чи файл має розширення *.csv* та спробуйте ще раз.».

Наступною важливою функцією є *def format_file* представлена у додатку А. Дана функція призначена для форматування файлу з даними. Вона приймає один аргумент – об'єкт DataFrame під назвою *df*. Видаляється певний набір стовпців з датафрейму *df*, які не потрібні для реалізації методів оцінки фізичної підготовки. Стовбці, що видаляються, мають такі назви: *Activity Type, Favorite, Aerobic TE, Avg*

Stride Length, Avg Vertical Ratio, Avg Vertical Oscillation, Avg Ground Contact Time, Grit, Flow, Total Strokes, Avg. Swolf, Avg Stroke Rate, Total Reps, Dive Time, Min Temp, Surface Interval, Decompression, Best Lap Time, Number of Laps, Max Temp, Avg Resp, Min Resp, Max Resp, Moving Time, Elapsed Time, Min Elevation. Ці стовбці перераховані у вигляді списку рядків і виключаються з датафрейму. В результаті цих операцій датафрейм *df* змінюється, що пришвидшує роботу застосунку.

Ключовими ж являються кінцеві функції для роботи зі шкалами. Нижче наведено код функції для розрахунку фізичної підготовки за **шкалою FTP**. В ній реалізовані всі необхідні методи для роботи з даною шкалою, а саме функція оцінки фізичної підготовки спортсмена, візуалізація даних та їх експорт. Дана функція FTP() виконує розрахунок FTP (функціональний поріг) на основі даних з файлу, вибраного користувачем (рис. 4.2), та відображає результати розрахунку у вікні програми.

```
def FTP():
    """
    Фунуція обробки даних для аналізй за шкалою FTP
    """
    data = pd.read_csv(file_path)
    data = format_file(data)
    data['Avg Power'] = data['Avg Power'].str.replace(',', '').astype(float)  # 
Перетворення на float
    data['FTP'] = data['Avg Power'] / 60 * 0.95

    # Додавання нового стовбця для рейтингу FTP на основі значення FTP
    conditions = [
        (data['FTP'] < 2.23),
        (data['FTP'] >= 2.23) & (data['FTP'] <= 2.78),
        (data['FTP'] >= 2.79) & (data['FTP'] <= 3.92),
        (data['FTP'] >= 3.93) & (data['FTP'] <= 5.04),
        (data['FTP'] > 5.04)
    ]
    ratings = ['Непідготовлений', 'Задовільно', 'Добре', 'Відмінно', 'Чудово']
    data['Rating'] = np.select(conditions, ratings, default='Unknown')
```

```
    # Створення нового вікна верхнього рівня для результатів розрахунку FTP
    ftp_window = tk.Toplevel(root)
    ftp_window.title("FTP Calculation Results")

    # Створення рамки у вікні верхнього рівня за допомогою менеджера геометрії сітки
    table_frame = tk.Frame(ftp_window)
    table_frame.grid(padx=10, pady=10)

    # Створення віджету таблиці та відображення результпту обчислення FTP
    table = Table(table_frame, dataframe=data[['Date', 'FTP', 'Rating']], showtoolbar=True, showstatusbar=True)
    table.grid(row=0, column=0, padx=10, pady=10)
    table.show()

    # Створення кнопки експорту
    export_button = tk.Button(ftp_window, text="Export to Excel", command=lambda: export_to_excel(data))
    export_button.grid(row=1, column=0, pady=10)

def export_to_excel(data):
    save_path = filedialog.asksaveasfilename(defaultextension='.xlsx')
    if save_path:
        with ExcelWriter(save  path) as writer:
            data.to_excel(writer, index=False)
        messagebox.showinfo("Export Successful", "Data exported to Excel successfully.")
```

Рисунок 4.2 – Лістинг функції розрахунку FTP (функціонального порогу)

Основний опис дій функції:

1) зчитується файл з даними за допомогою функції *pd.read_csv()* і зберігається у змінну *data;*

2) викликається функція *format_file(data),* яка форматує дані, виконуючи певні операції зі стовбцями датафрейму *data*. Оновлений датафрейм знову зберігається у змінну *data*;

3) виконується обробка стовбця даних. Кома в значеннях замінюється на порожній рядок за допомогою методу *str.replace(),* а потім значення

перетворюються на числовий тип даних *(float)* за допомогою методу *astype()* та виконується збереження значень;

4) розраховується значення FTP;

5) додається новий стовбець «Rating» для оцінки FTP на основі значення FTP. Використовується функція *np.select()* для вибору відповідного рейтингу залежно від діапазону FTP. Рейтинги та умови розрахунку відповідають діапазонам FTP, і значення «Unknown» використовується як значення за замовчуванням, якщо жодна з умов не співпадає;

6) створюється нове вікно верхнього рівня *(tk.Toplevel)* для відображення результатів розрахунку FTP;

7) створюється фрейм всередині вікна за допомогою менеджера геометрії *'grid' (table_frame.grid(padx=10, pady=10));*

8) створюється віджет таблиці *(Table)* з бібліотеки **tkinter** у фреймі table_frame і відображаються результати розрахунку FTP з обраними стовпцями «Date», «FTP» та «Rating». Таблиця показує панель інструментів та рядок стану. Розташовується за допомогою *table.grid();*

9) відображається таблиця за допомогою table.show();

10) створюється кнопка експорту до Excel (Export to Excel), яка викликає функцію *export_to_excel(data)* при натисканні. Кнопка розташовується у вікні FTP за допомогою *export_button.grid()*.

Функція для роботи зі шкалою **VAMs** є майже аналогічна, її код наведено у додатку Б, але з певними незначними особливостями. Дана функція VAMs() виконує обробку даних для аналізу за шкалою VAMs (Vertical Ascent Meters per Second) на основі даних з файлу, вибраного користувачем. Результати обчислень відображаються у вікні програми.

Основний опис дій функції:

1) зчитується файл з даними за допомогою функції pd.read_csv() і зберігається у змінну data;

2) викликається функція *format_file(data)*, яка форматує дані, виконуючи певні операції зі стовпцями датафрейму *data*. Оновлений датафрейм знову зберігається у змінну *data*;

3) виконується обробка стовпця «*Max Elevation*». Кома в значеннях замінюється на порожній рядок за допомогою методу *str.replace()*, а потім значення перетворюються на числовий тип даних (*float*) за допомогою методу *astype()*. Результати зберігаються назад у стовпець *«Max Elevation»*;

4) стовпець *«Time»* перетворюється на тип *datetime*, використовуючи *pd.to_datetime()*, з форматом часу *«%H:%M:%S»*. Потім значення округлюються до найближчої години за допомогою методу *round()* та *dt.time*. Оновлені значення зберігаються назад у стовпець *«Time»;*

5) створюється стовпець *«Seconds»*, де обчислюється тривалість часу у секундах на основі значення стовпця *«Time»*. Для цього використовується метод *apply()* та *datetime.timedelta()*. Результати зберігаються у стовпець *«Seconds»;*

6) обчислюється значення VAMs (метри вертикального підйому на секунду) шляхом ділення значення *«Max Elevation»* на значення *«Seconds»*. Результати зберігаються у новому стовпці *«VAMs»;*

7) створюється нове вікно верхнього рівня *(tk.Toplevel)* для відображення результатів розрахунку VAMs;

8) створюється фрейм *table_frame* у вікні верхнього рівня, використовуючи менеджер геометрії *grid();*

9) створюється віджет таблиці *(Table)* та відображаються результати обчислення VAMs за допомогою передачі відповідних стовпців датафрейму *data* у параметр *dataframe* конструктора *Table*. Віджет таблиці розміщується у фреймі *table_frame* за допомогою *table.grid()*. Панель інструментів та рядок стану також відображаються, оскільки параметри *showtoolbar=True* та *showstatusbar=True*;

10) викликається метод *table.show()*, щоб відобразити таблицю з результатами;

11) створюється кнопка експорту до Excel з текстом «Export to Excel», яка викликає функцію *export_to_excel(data)* при натисканні. Кнопка розташовується у вікні *vams_window* за допомогою *export_button.grid()*.

Функція *export_to_excel(data)* виконує експорт даних у форматі Excel *(*.xlsx)*. Функція відкриває діалогове вікно для вибору шляху збереження файлу за допомогою *filedialog.asksaveasfilename()*. Якщо шлях вибраний, то дані зберігаються у файл Excel за допомогою *data.to_excel()*. Після успішного експорту виводиться повідомлення про успішне експортування за допомогою *messagebox.showinfo()*.

4.2 Реалізація сторінок програми

Для реалізації основного вікна застосунку був створений клас, що є наслідником від *tkinter.Tk*. Він містить в собі контейнер для відображення сторінок, розміщення елементів в яких відбувається за допомогою сітки *(grid)*. Даний клас наведений на лістингу рис. 4.3.

```
# Створюємо вікно з заданим розміром та розташуванням
root = tk.Tk()
root.title("Аналіз фізичних навантажень велоспортсмена")

root.geometry("900x700+{}+{}".format(int(root.winfo_screenwidth()/2 - 400),
int(root.winfo_screenheight()/2 - 300)))

# Створюємо фрейм
frame = ttk.Frame(root)
frame.pack(padx=10, pady=10)
```

Рисунок 4.3 – Лістинг коду реалізації основного вікна застосунку

Завдяки сітковій розмітці можна легко створити дизайн, який буде відповідати розробленим макетам.

Результатом став фрейм, зображений на рис. 4.4.

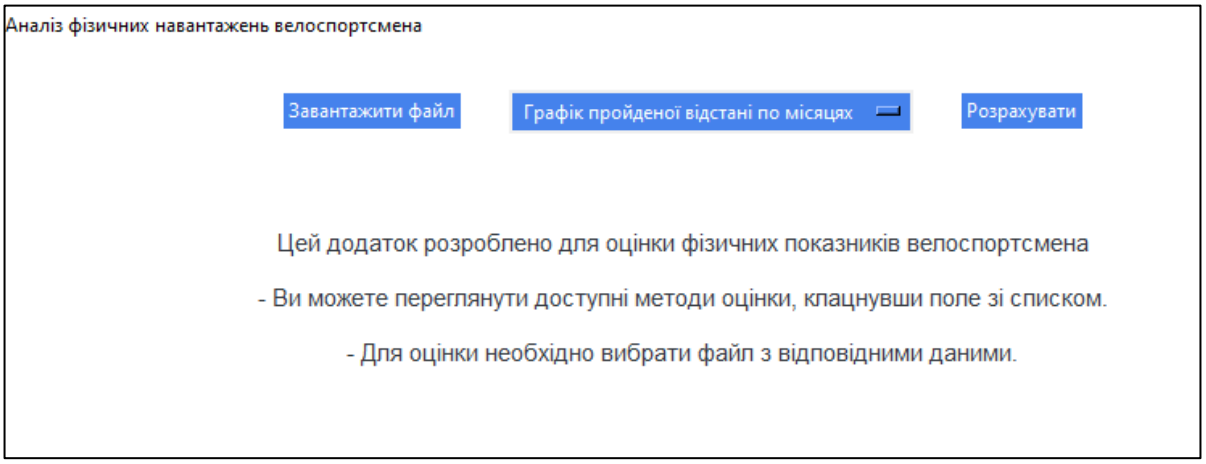


Рисунок 4.4 – Головний фрейм застосунку

Важливим тут є пояснення маршрутизація в застосунку. Перехід на новий фрейм відбувається завдяки виклику в методі елементу відповідної функції для відображення, передавши необхідні параметри. Приклад коду, що виконує даний перехід наведено на лістингу рис. 4.5.

```
# Створення нового вікна верхнього рівня для результатів розрахунку FTP
    ftp_window = tk.Toplevel(root)
    ftp_window.title("FTP Calculation Results")

    table_frame = tk.Frame(ftp_window)
    table_frame.grid(padx=10, pady=10)
```

Рисунок 4.5 – Лістинг коду маршрутизації в застосунку

Також було реалізовано метод вибору варіанту списку з *combobox* (рис. 4.6).

```
def ok_button():
    if variable.get() == "Графік пройденої відстані по місяцях":
        plot_distance_by_month()
    elif variable.get() == "Шкала FTP":
        FTP()
    elif variable.get() == "Шкала VAMs":
        VAMs()
    else:
        root.destroy()
ok_button = tk.Button(frame, text="Розрахувати", command=ok_button)
ok_button.grid(row=0, column=2, padx=15, pady=15)
```

Рисунок 4.6 – Лістинг коду вибору варіанту списку з combobox

Дана функція необхідна для того, щоб можна було за значенням із *combobox,* що використовується для обрання методу оцінки, виконати обраний метод. Можна було також зробити глобальний словник, але тут є певна проблема, що не дозволить реалізувати все через те, що елементи на початку інтерпретованого файлу нічого не знають про елементи після них. Це можна виправити імпортом анотацій з модулю *__future__*, але це спрацює лише всередині функцій. Поза ними дані маніпуляції будуть неуспішними і IDE буде видавати помилку, оскільки посилання на функцію чи клас є невизначеним.

Наступним фреймом є таблична візуалізація (за можливістю) обраних даних, зображений на рис. 4.7.

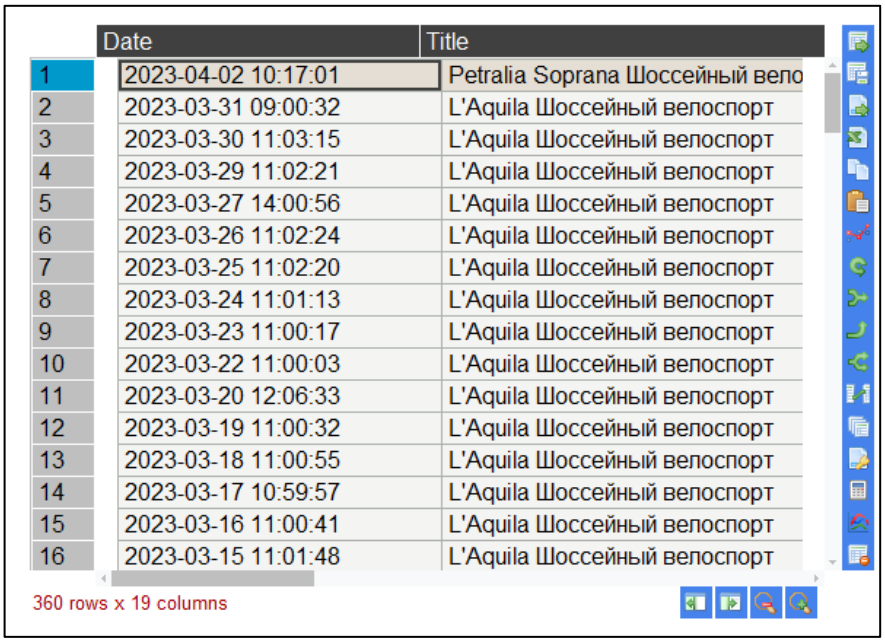


Рисунок 4.7 – Фрейм з табличною візуалізацією даних

Фрейм з візуалізацією табличних даних, обраних користувачем виконується одразу після вибору файлу з даними для аналізу.

Наступним фреймом є візуалізація даних з попереднього фрейму, зображений на рис. 4.8. Під зображенням реалізовано меню для роботи з ним, головною функцією в якому є саме можливість експорту зображення.

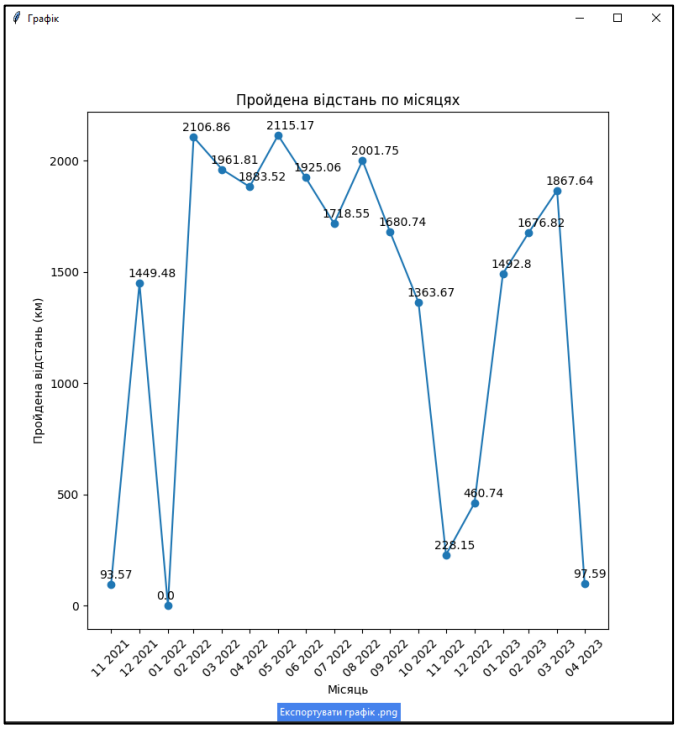


Рисунок 4.8 – Фрейм з візуалізацією даних

Візуалізація відбувається завдяки класам *Figure* та *FigureCanvasTkAgg* бібліотеки *matplotlib*. Перший відповідає за власне фігуру та її зображення, а другий дозволяє отримати віджет для подальшого відображення в фреймі. Приклад суміщення роботи цих класів для отримання результату, зображеного на рис. 4.8 наведено на лістингу нижче. Також на ньому вказано як необхідно додавати зображення до фрейму. Відбувається це через отримання віджету завдяки функції *get_tk_widget()* наведеної на рис. 4.9.

```
# Створюємо нове вікно (frame) для відображення графіка
    graph_window = tk.Toplevel(root)
    graph_window.title('Графік')

    # Створюємо об'єкт FigureCanvasTkAgg для відображення графіка в
новому вікні
    canvas = FigureCanvasTkAgg(fig, master=graph_window)
    canvas.draw()
    canvas.get_tk_widget().pack()
```

```
    # Додавання кнопки експорту графіку
    def export_graph():
        save_path = filedialog.asksaveasfilename(defaultextension='.png')
        if save_path:
            fig.savefig(save_path)
            print("Графік експортовано у форматі .PNG")

    export_button = tk.Button(master=graph_window, text="Експортувати
графік .png", command=export_graph)
    export_button.pack()
```

Рисунок 4.9 – Лістинг коду функції *get_tk_widget()*

Головними фреймами є фрейми з результатами оцінювання спортсмена, зображений на рис. 4.10 та рис. 4.11.

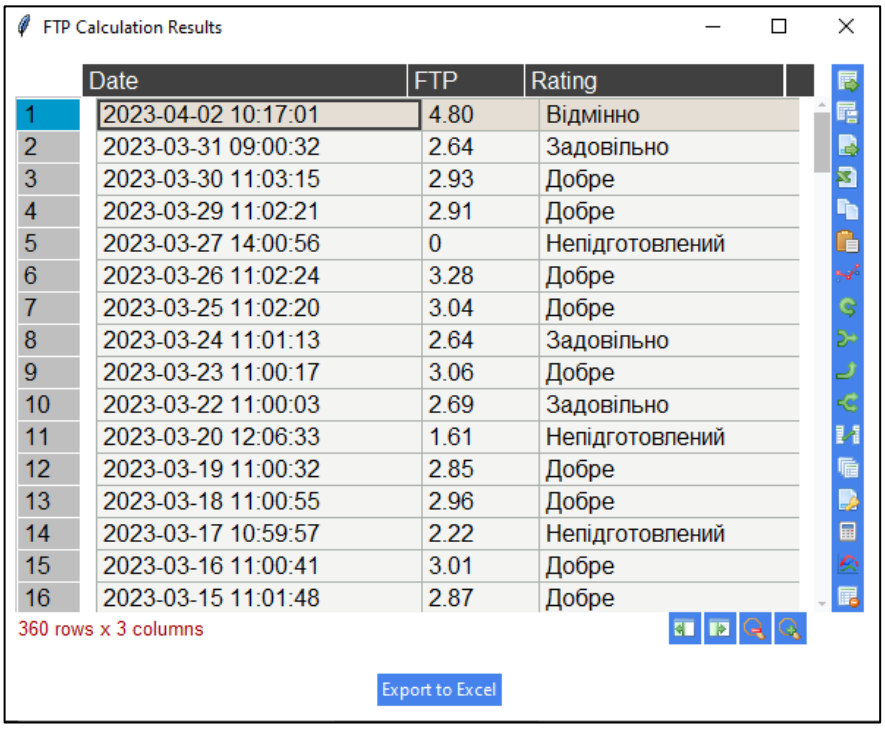


Рисунок 4.10 – Фрейм з результатами оцінювання за шкалою FTP

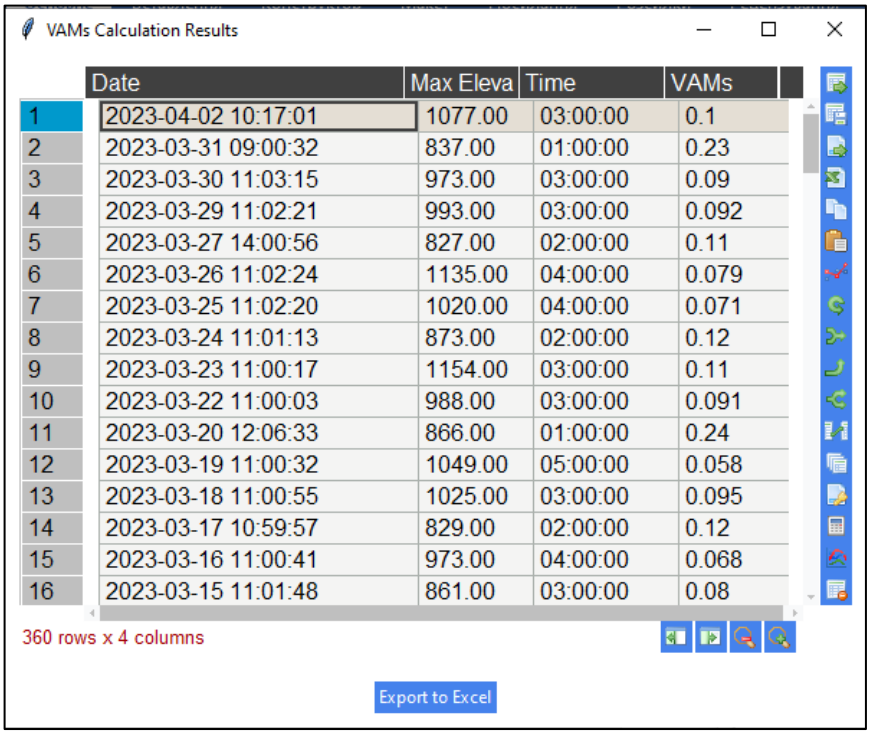


Рисунок 4.11 – Фрейм з результатами оцінювання за шкалою VAMs

Також для кожного з методів оцінки, було додано можливість експорту результатів оцінки фізичної підготовки спортсмена.

4.3 Тестування ПЗ

Під час тестування програмного забезпечення його працездатність перевірялась, розділено відповідно до тих помилок, які може викликати користувач, і тих, що можуть бути всередині функцій. Зі сторони користувача основною можливою помилкою є обрання файлу. На даному етапі можуть виникнути такі помилки:

– файл є порожнім;
– файл не підходить для обробки через відсутність необхідних колонок;
– проблеми з кодуванням;
– неправильний тип файлу.

Відповідно до кожної з цих імовірних помилок було реалізовано виведення необхідної інформації про помилку. Лістинг коду та приклад обробки помилки наведено на рис. 4.12–4.13.

```
if file_path:
        try:
            data = pd.read_csv(file_path)
            show_table(data)  # Відображення таблиці у фреймі
            messagebox.showinfo("Успіх", "Файл успішно завантажено")
        except:
            messagebox.showerror("Помилка",
                                 "Не вдалося завантажити файл. Перевірте, чи файл має розширення .csv та спробуйте ще раз.")
```

Рисунок 4.12 – Лістинг коду обробки помилки

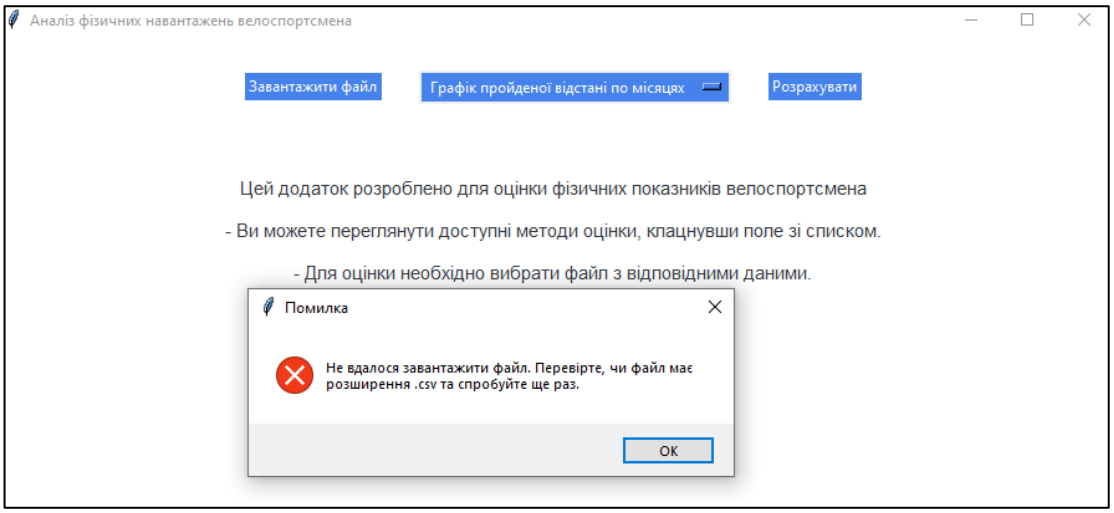


Рисунок 4.13 – Приклад повідомлення про оброблену помилку

Зі сторони коду помилковість могла бути пов’язаною з неправильними обрахунками і відображенням елементів, але після пильного огляду та проведення декількох сеансів тестування недоліків виявлено не було. На рис. 4.14 наведено приклад повідомлення про успішне завантаження файлу для подальшої обробки.

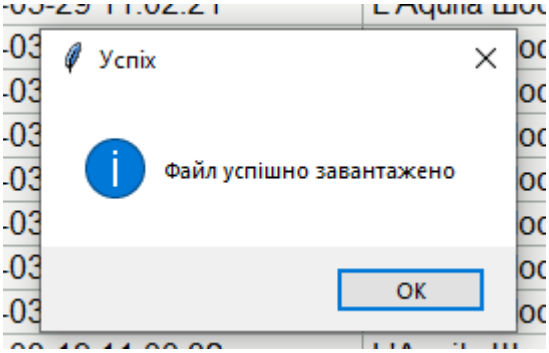


Рисунок 4.14 – Приклад повідомлення про успішне завантаження файлу

4.4 Створення документації

Важливими елементами документації коду є правильні назви та позначення типів. Це часто дозволяє одразу зрозуміти що робить функція і не витрачати зайвий час на перегляд текстової документації, яка, зазвичай, є більше доповненням, аніж основою. В Python є таке поняття як *docstring* – рядковий літерал, що пишеться одразу після визначення функції та обрамлюється потрійними лапками, причому неважливо одинарними чи подвійними. Зазвичай в них пишуть загальний опис функції, описують параметри та наводять приклади використання. Завдяки цьому розвинені IDE по типу Pycharm дозволяють легко переглядати документацію при наведенні на функцію.

Також важливою частиною гарної документації є коментарі, які мають бути доречними, наприклад, як на лістингу рис. 4.15.

```
# Створення випадаючого списку
options = ["Графік пройденої відстані по місяцях", "Шкала FTP", "Шкала
VAMs"]
variable = tk.StringVar(frame)
variable.set(options[0])
dropdown = tk.OptionMenu(frame, variable, *options)
dropdown.config(width=35)
dropdown.grid(row=0, column=1, padx=15, pady=15)

# Створюємо кнопку "Розрахувати" та її обробник
def ok_button():
    if variable.get() == "Графік пройденої відстані по місяцях":
        plot_distance_by_month()
    elif variable.get() == "Шкала FTP":
        FTP()
    elif variable.get() == "Шкала VAMs":
        VAMs()
    else:
        root.destroy()
```

Рисунок 4.15 – Лістинг коду з прикладом коментарів

Це дозволяє легко пояснити важливість того чи іншого коду або принцип його роботи. Часто це необхідно при побудові складних виразів, які людині складно зрозуміти, хоча за можливістю краще розбити їх на більш прості компоненти.

В розробленому застосунку функції мають свої docstring там, де це є необхідним для розуміння її задач або ж параметрів. Для кожного необхідного для пояснення обраного рішення рядка написано коментар, що описує причини виконання поставлених задач саме таким чином.

Проведено типізацію даних де це є необхідним задля усунення помилок при роботі з функціями шляхом чітких обмежень параметрів. Приклад типізації наведено на лістингу рис. 4.16.

```
def VAMs():
    """
    Фунуція обробки даних для аналізу за шкалою VAMs
    """
    data = pd.read_csv(file_path)
    data = format_file(data)
    data['Max Elevation'] = data['Max Elevation'].str.replace(',', '').astype(float)
# Перетворення на float
    # Округлення значення часу
    data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S')
    data['Time'] = data['Time'].dt.round('H').dt.time
```

Рисунок 4.16 – Лістинг коду типізації

Документація користувача є в певній мірі інтегрованою в застосунок шляхом підбору правильних назв кнопок з відповідними функціями та додання інформаційних полів там де це є необхідним.

4.5 Розгортання ІС

Задля зручності перенесення та розгортання проєкту на кінцевій машині користувача, було створено «*.exe» файл. Мова програмування Python має можливість конвертувати код у бінарний файл («*.exe», «*.bat»), в якому знаходиться інтерпретований код, що може бути виконаний незалежно від

наявності встановленої мови програмування чи деяких бібліотек. Щоб реалізувати цю можливість, потрібно встановити бібліотеку pyinstaller та застосувати наведену на рис. 4.17 команду для створення файлу у форматі «*.exe» без додаткових операцій.

Рисунок 4.17 – Процес створення «main.py» файлу

Після завершення генерації «main.exe» у директорії проєкту створюється папка з відповідним файлом (рис. 4.18).

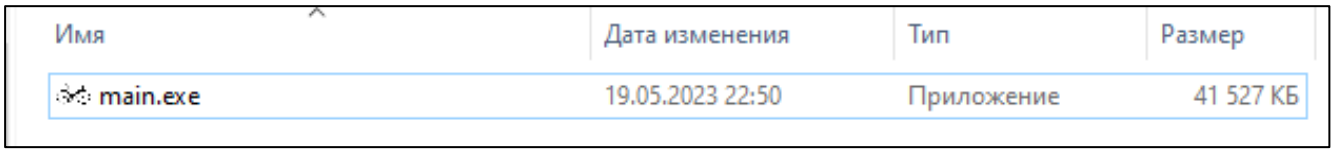


Рисунок 4.18 – Готовий «main.py» файлу

Даний файл має повний набір функціональних можливостей і працює без будь-яких обмежень, що дозволяє його без проблем переносити та працювати з програмою на різних пристроях, без потреби в будь-яких додаткових налаштуваннях або змінах в коді.

Також, для зручності розробника, було створено репозиторій на GitHub, звідки можна вивантажити вихідний код для подальшого оновлення та додавання нових функцій. Щоб розгорнути проєкт з репозиторію GitHub, слід дотриматися наступних кроків:

1) клонувати репозиторій – для завантаження вихідного коду потрібно з офіційного репозиторію проєкту вивантажити всі необхідні файли та встановити інтерпретатор Python версії 3.9 (усі необхідні специфікації вказані у файлі Readme.txt). Це можна зробити за допомогою команди, наведеної на рис. 4.19;

```
git clone <URL репозиторію>
```

Рисунок 4.19 – Клонування репозиторію

2) перейти до директорії проєкту – за допомогою команди «**cd**» (рис. 4.20) перейти до директорії, яка була створена під час клонування репозиторію;

```
cd project-directory
```

Рисунок 4.20 – Клонування репозиторію

3) встановити залежності проєкту – файл requirements.txt містить список залежностей, на рис. 4.21 наведено вміст файлу. За допомогою команди «***pip install - r requirements.txt***» встановити залежності;

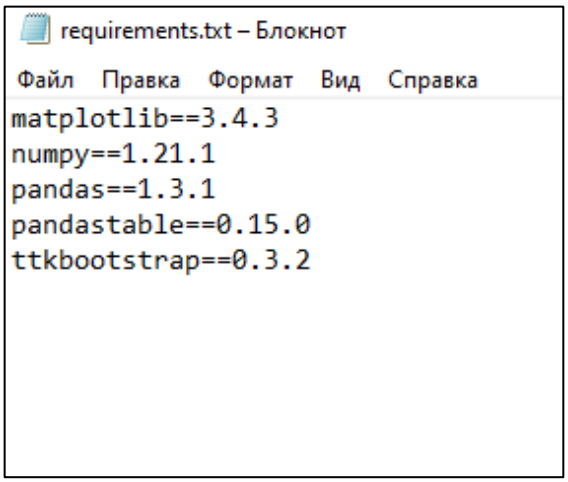


Рисунок 4.21 – Вміст файлу requirements.txt

4) після всіх налаштувань користувачеві треба запустити через командний рядок скрипт, ввівши наступну команду (рис. 4.22).

```
python main.py
```

Рисунок 4.22 – Команда для запуску застосунку

Мова Python знаходить сама інтерпретатор із змінних системного оточення, а другим параметром передає ім’я скрипту, який потрібно запустити. Після цих дій застосунок повністю готовий до роботи.

Висновки до розділу 4

В результаті виконання роботи над четвертим розділом бакалаврської кваліфікаційної роботи було описано розробку та реалізацію різних аспектів застосунку, а саме рівню роботи з даними та інтерфейсу. Надано інформацію про ключові моменти та особливості даних підрозділів.

Наведено необхідні для повного розуміння етапу розробки та кінцевого програмного забезпечення лістинги, а також описано їх важливість. Пояснено особливості роботи з *Tkinter* та чому було обрано саме такі проєктні рішення.

Описано процес успішного тестування програмного забезпечення як зі сторони користувача, так і зі сторони розробника та наведено інформацію про відстеження помилок та повідомлення про них користувачу.

Наведено докладну інформацію про формування документації застосунку. Описано основні інструменти та механізми, які використовувались при її створенні.

Продемонстровано кінцевий дизайн застосунку та надано інформацію про кожний із фреймів, які в ньому використовуються. А також описано процес розгортання ІС на інших пристроях для користувача та наведено покрокову інструкцію клонування вихідного коду з репозиторія GitHub для подальшого удосконалення та роботи з ним.

ВИСНОВКИ

Результати аналізу досліджень предметної сфери моніторингу фізичних навантажень велоспортсмена та збору й обробки відповідних даних показують, що впровадження цифрових технологій для оцінки фізичної підготовки за різними методиками (FTP, Фостера, VAMs, Ратінга, Борхардта) може суттєво змінити уявлення про ефективність інформаційних систем моніторингу навантажень велоспортсмена. Отримані результати підтверджують доцільність подальшого розвитку алгоритмів та програмного забезпечення для аналізу масивів великих даних у даній сфері.

В ході виконання бакалаврської роботи було розроблена інформаційна система та програмне забезпечення для оцінювання фізичних навантажень велоспортсмена шляхом обробки статистичних даних та їх подальшої візуалізації у тому вигляді, що буде зрозуміло людині.

В основу розробки було покладено можливість масштабованості даного застосунку та проєкту в цілому. У майбутньому планується подальша розробка програми, удосконалення існуючих методів та додавання нових функцій, збір інформації та її глибока обробка. Також основним вдосконаленням, буде розробка власних модулів та датчиків для збору фізичних показників та відхід від використання стороннього обладнання.

В БКР було описано значущість даної роботи та необхідність розвитку обраної сфери дослідження, яка полягає у покращені фізичної підготовки, здоров’я спортсменів та піднятті рівня підготовки наших майбутніх чемпіонів.

Для досягнення поставленої мети виконано наступні завдання:

– проведено аналіз предметної області та існуючих аналогів;
– обрано інструменти, необхідні для створення застосунку;
– знайдено та отримано дані для подальшого оброблення;
– визначено та реалізовано основний необхідний функціонал застосунку;
– реалізовано відображення результатів обробки та взаємодію з ними;
– розроблено документацію.

Робота пройшла апробацію під час XXV Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 07–11 листопада 2022 р.).

Публікації. Основні положення та результати бакалаврської кваліфікаційної роботи опубліковані у збірнику матеріалів XXV Всеукраїнської науково-практичної конференції «Могилянські читання – 2022».

За результатами бакалаврської роботи опубліковано 1 друковану працю.

Робота складається з вступу, чотирьох фахових розділів, висновків, переліку джерел посилання з 27 джерел, трьох додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Reeth, Daam Van (2022-10-28). The Economics of Professional Road Cycling. Springer Nature. pp. 363–385. ISBN 978-3-031-11258-4.

2. Пилипчук Б. В., Журавська І. М. Моніторинг фізичних навантажень велоспортсмена. Інтелектуальні інформаційні системи: тези Всеукр. наук.-практ. конф., Миколаїв. 07-11 листопада 2022 р. С. 47–49.

3. Global Recommendations on Physical Activity for Health, 2009. World Health Organization. Geneva, Switzerland. Accessed 13/07/2018. URL: http://www.who.int/ncds/prevention/physical-activity/en/ (Last accessed: 02.05.2023).

4. What is FTP and why does it matter for cyclists?. BikeRadar. URL: https://www.bikeradar.com/advice/fitness-and-training/what-is-ftp-and-why-it-matters-for-cyclists/ (Last accessed: 03.05.2023).

5. De Marchi D., Schena F. (2007). The evaluation of mountain bike cross-country performance. International Journal of Sports Physiology and Performance, 2(4), P. 360-372.

6. C. Tapsai, "Information Processing and Retrieval from CSV File by Natural Language," 2018 IEEE 3rd International Conference on Communication and Information Systems (ICCIS), Singapore, 2018, pp. 212-216, doi: 10.1109/ICOMIS.2018.8644947.

7. Garmin connect – Free Online Fitness Community. URL: https://sso.garmin.com/portal/sso/uk-UA/sign-in?clientId=GarminConnect&service=https://connect.garmin.com/modern/activities (Last accessed: 05.05.2023).

8. Endomondo Sports Tracker. (IOS) URL: https://apps.apple.com/us/app/endomondo/id333210180 (Last accessed: 05.05.2023).

9. Endomondo Sports Tracker (Android). URL: https://endomondo-sports-tracker.ru.uptodown.com/android (Last accessed: 05.05.2023).

10. D. Arruda and N. H. Madhavji, "Towards a big data requirements engineering artefact model in the context of big data software development projects: Poster extended abstract," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 2017, pp. 4725-4726, doi: 10.1109/BigData.2017.8258521.

11. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI (Редакція від 27.10.2022). Стаття 11. Підстави для обробки персональних даних. URL: https://kodeksy.com.ua/pro_zahist_personal_nih_danih/statja-11.htm (Last accessed: 08.05.2023).

12. L. Xiaofeng and L. Jing, "Research on Big Data Reference Architecture Model," 2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 2020, pp. 205-209, doi: 10.1109/ICAIBD49809.2020.9137451.

13. Contributor T. What is data set? Definition from TechTarget. URL: https://www.techtarget.com/whatis/definition/data-set (Last accessed: 01.05.2023).

14. S. Wu, "The Coexistence of Sequence Diagrams and Collaboration Diagrams in Unified Modeling Language," 2011 Fourth International Symposium on Knowledge Acquisition and Modeling, Sanya, China, 2011, pp. 184-187, doi: 10.1109/KAM.2011.56.

15. Visual Paradigm Community Circle. URL: https://circle.visual-paradigm.com/docs/code-engineering/instant-generator/ (Last accessed: 10.05.2023).

16. DevOps Best Practices. URL: https://www.atlassian.com/devops/what-is-devops/devops-best-practices (Last accessed: 10.05.2023).

17. DevOps Best Practices Every Developer Should Know. Spacelift. URL: https://spacelift.io/blog/devops-best-practices (Last accessed: 12.05.2023).

18. L. Sheppard, T. W. Lumpkin and T. Kelly, "A KISS Approach to Embedded Health Management," 2006 IEEE Autotestcon, Anaheim, CA, USA, 2006, pp. 174-180, doi: 10.1109/AUTEST.2006.283622.

19. Велодатчик – датчик частоти обертання педалей (каденсу). URL: https://garmin.ua/catalog/datchiki_antenny_raznoe/bike-speed-sensor2-cadence-sensor2/#description (Last accessed: 15.05.2023).

20. Classification of Garmin heart rate monitors: analysis, tips, recommendations. URL: https://prostobzor.com/garmin-hrms-rating/ (Last accessed: 15.05.2023).

21. Cyclocomputer. URL: https://velokyiv.com/qaview/28 (Last accessed: 15.05.2023).

22. Getting started – pandas documentation. URL: https://www.w3schools.com/python/pandas/default.asp (Last accessed: 17.05.2023).

23. NumPy Introduction. Python documentation. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp (Last accessed: 17.05.2023).

24. Matplotlib: Visualization with Python. URL: https://matplotlib.org/ (Last accessed: 17.05.2023).

25. Tkinter – Інтерфейс Python до Tcl/Tk. Python documentation. URL: https://docs.python.org/uk/3/library/tkinter.html (Last accessed: 17.05.2023).

26. PyQt Documentation. URL: https://doc.qt.io/qtforpython-6/ (Last accessed: 17.05.2023).

27. Gonzalez, Robbie (July 25, 2017). "Figma Wants Designers to Collaborate Google-Docs Style". WIRED. WIRED. Archived from the original on October 20, 2020. Retrieved June 1, 2020.

ДОДАТОК А

Базовий клас для обробників даних

```
    def format_file(df):
        df = df.drop([
            'Activity Type', 'Favorite', 'Aerobic TE', 'Avg Stride Length', 'Avg Vertical Ratio',
            'Avg Vertical Oscillation', 'Avg Ground Contact Time', 'Grit', 'Flow', 'Total Strokes',
            'Avg. Swolf', 'Avg Stroke Rate', 'Total Reps', 'Dive Time', 'Min Temp', 'Surface Interval', 'Decompression',
            'Best Lap Time', 'Number of Laps', 'Max Temp', 'Avg Resp', 'Min Resp', 'Max Resp', 'Moving Time',
            'Elapsed Time', 'Min Elevation'
        ], axis=1)
        df.replace("--", 0, inplace=True)
        # Виконуємо операцію з розділенням значення на дві частини та вибираємо першу частину
        df['Time'] = df['Time'].apply(lambda x: x.split('.')[0])
        return df
    ef browse_file():
        global file_path
        file_path = filedialog.askopenfilename(filetypes=[("CSV files", "*.csv")])
        print(file_path)  # Виводимо шлях до файлу у консоль

        if file_path:
            try:
                data = pd.read_csv(file_path)
                show_table(data)  # Відображаємо таблицю у фреймі
                messagebox.showinfo("Успіх", "Файл успішно завантажено")
            except:
                messagebox.showerror("Помилка",
                                     "Не вдалося завантажити файл. Перевірте, чи файл має розширення .csv та спробуйте ще раз.")
```

ДОДАТОК Б
Код функції для роботи зі шкалою FTP

```
def VAMs():
    """
    Фунуція обробки даних для аналізу за шкалою VAMs
    """
    data = pd.read_csv(file_path)
    data = format_file(data)
    data['Max Elevation'] = data['Max Elevation'].str.replace(',', '').astype(float)  # Видаліть коми та перетворіть на float

    # Округленя значення часу
    data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S')
    data['Time'] = data['Time'].dt.round('H').dt.time

    # Розрахунок VAM на основі тривалості часу в секундах
    data['Seconds'] = data['Time'].apply(lambda x: datetime.timedelta(hours=x.hour, minutes=x.minute, seconds=x.second).total_seconds())
    data['VAMs'] = data['Max Elevation'] / data['Seconds']

    # Створення нового вікна верхнього рівня для результатів розрахунку VAM
    vams_window = tk.Toplevel(root)
    vams_window.title("VAMs Calculation Results")

    # Створення рамки у вікні верхнього рівня за допомогою менеджера геометрії сітки
    table_frame = tk.Frame(vams_window)
```

```
        table_frame.grid(padx=10, pady=10)

        # Створення віджету таблиці та відобразіть результати обчислення VAM
        table = Table(table_frame, dataframe=data[['Date', 'Max Elevation', 'Time', 'VAMs']], showtoolbar=True, showstatusbar=True)
        table.grid(row=0, column=0, padx=10, pady=10)
        table.show()

        # Створення кнопки експорту
        export_button = tk.Button(vams_window, text="Export to Excel", command=lambda: export_to_excel(data))
        export_button.grid(row=1, column=0, pady=10)

    def export_to_excel(data):
        save_path = filedialog.asksaveasfilename(defaultextension='.xlsx')
        if save_path:
            with ExcelWriter(save_path) as writer:
                data.to_excel(writer, index=False)
            messagebox.showinfo("Export Successful", "Data exported to Excel successfully.")
```

ДОДАТОК В

Матеріали апробації роботи

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

«МОГИЛЯНСЬКІ ЧИТАННЯ – 2022:
Досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»

XXV Всеукраїнська науково-практична конференція

ТЕЗИ

Комп'ютерні науки.

Технічні науки

Миколаїв, 7–11 листопада 2022 року

Миколаїв – 2022

УДК 007:343.304

Пилипчук Б. В.,
бакалаврант,
Журавська І. М.,
д-р техн. наук, професор, в.о. завідувача кафедри КІ,
ЧНУ ім. Петра Могили, Миколаїв, Україна

МОНІТОРИНГ ФІЗИЧНИХ НАВАНТАЖЕНЬ ВЕЛОСПОРТСМЕНА

Останнім часом, через різні надзвичайні події в світі (пандемії, воєнний стан та ін.), стало важко стежити за своїм фізичним здоров'ям та підтримувати свої фізичні показники. Велика кількість дитячо-юнацьких спортивних шкіл з велоспорту олімпійського резерву призупинила свою роботу, тим самим знизився рівень підготовки наших майбутніх чемпіонів.

Об'єкт дослідження (розробки): процеси моніторингу фізичних навантажень велоспортсмена.

Предмет дослідження (розробки): методи та засоби створення інформаційних систем, що виконують функції отримання даних щодо фізичного навантаження та коригування плану тренувань велоспортсменів у реальному часі.

Для створення системи з коригування навантажень велоспортсмена потрібно отримати дані каденсу та пульсу під час виконання будь яких фізичних навантажень для подальшого коригування тренером в онлайн-режимі.

Рисунок 1 – Дані частоти обертів педалей

На теперішній час існуючі інформаційні системи (ІС) велотрекінгу дозволяють визначати загальну відстань, пройдену за час тренування, історію маршруту, швидкість та ін., експортувати результати обліку біометричних показників спортсмена у вигляді GPX, TCX-файлів зі спеціалізованого обладнання Endomondo, Garmin та ін.

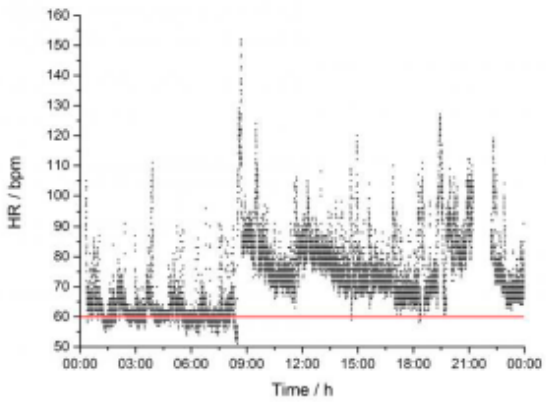


Рисунок 2 – Дані частоти серцебиття

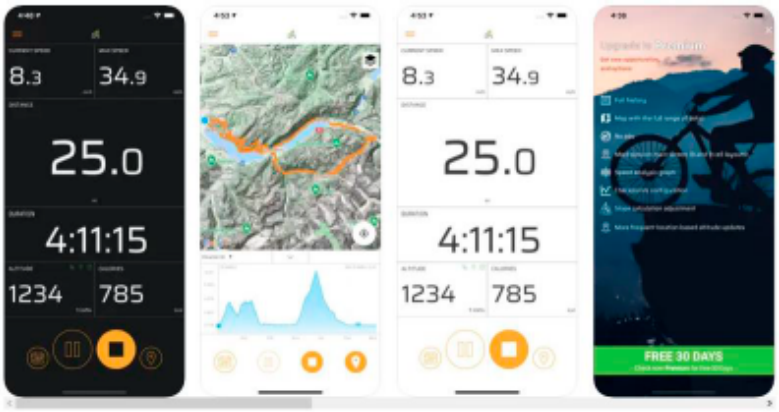


Рисунок 3 – Знімки екрану iPhone з результатами моніторингу параметрів спортивної активності при тренуванні велоспортсмена

Зазначені результати потім передаються тренеру для коригування останнім режиму наступних тренувань. Перевагами існуючих ІС моніторингу велотренувань є безкоштовність, сумісність з рядом пристроїв Android та iOS (смартфони, планшети), персоналізація зовнішнього вигляду застосунку (тема, колір), мінімалістичний дизайн, можливість використання різних мов тощо.

Але існуючі рішення не надають можливості в реальному часі координувати дії спортсмена (вносити зміни у план тренувань), що може привести до незворотної втрати фізичної форми спортсмена через понадмірні навантаження або, навпаки, недосягнення необхідних показників для успіху у змаганнях відповідного рейтингу. Тому доцільно розробляти та запроваджувати для відповідних ІС додаткові

програмні модулі, що використовують телекомунікаційні можливості Garmin-обладнання.

Для реалізації поставленої мети може використовуватись нижче-зазначені технології та методи.

Методи:

а) парсинг даних з Garmin connect.

Технології:

а) мова програмування Python;

б) додаткові бібліотеки мови програмування Python:

1) Requests – це модуль Python, який можна використовувати для відправки всіх видів HTTP-запитів;

2) Beautifulsoup – це бібліотека Python для отримання даних з файлів HTML і XML. Дає можливість навігації, пошуку та зміни дерева розбору;

3) Pandas – це бібліотека Python для обробки та аналізу структурованих даних.

Такий підхід надасть можливість тренеру здійснювати онлайн-підтримку велоспортсменів у реальному часі, що збільшить їх потенціал та покращить фізичну підготовку та досягнення спортивних результатів.

УДК 004.02

Сіденко Є. В.,
канд. техн. наук, доцент, в.о. завідувача кафедри ІІС,
Кондратенко Г. В.,
канд. техн. наук, доцент, доцент кафедри ІІС,
ЧНУ ім. Петра Могили, Миколаїв, Україна

РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ВИБОРУ МОБІЛЬНИХ ПРИСТРОЇВ НА ОСНОВІ МЕТОДІВ ПРИЙНЯТТЯ РІШЕНЬ

Метою даної роботи є формування рекомендацій з вибору мобільних пристроїв на основі методів прийняття рішень. Актуальність дослідження визначається складністю формування оцінки та вибору мобільного пристрою через різноманіття та багатогранність відомої інформації про можливі альтернативи, складністю виділення найбільш важливих та впливових критеріїв для подальшого прийняття рішень.

Розглянемо задачу з вибору мобільних пристроїв (Xiaomi Mi 11 Ultra, Oppo Find X3 Pro, Huawei P50 Pro, Google Pixel 6 Pro, Vivo X70 Pro+,