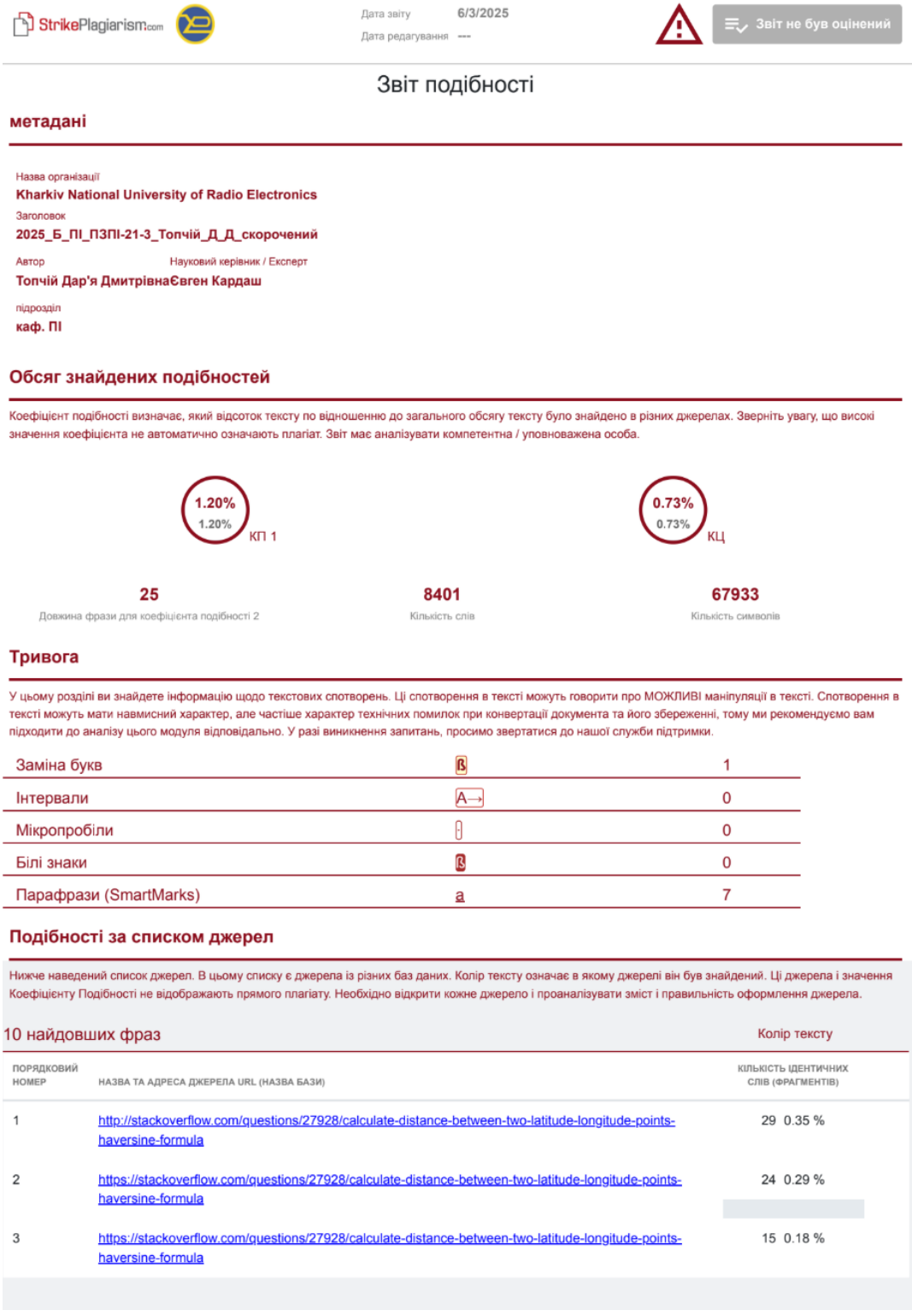
ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/3/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-3_Топчій_Д_Д_скорочений
Автор Науковий керівник / Експерт
Топчій Дар'я ДмитрівнаЄвген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.20% 1.20% КП 1

0.73% 0.73% КЦ

25
Довжина фрази для коефіцієнта подібності 2

8401
Кількість слів

67933
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	¦	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	7

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз | Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	http://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula	29 0.35 %
2	https://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula	24 0.29 %
3	https://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula	15 0.18 %

4	2022_Б_ПІ_ПЗПІ_18_5_Ємельянова_К_О [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	12 0.14 %
5	https://openarchive.nure.ua/bitstream/document/11518/1/2019_M_PI_Zymarev_KD.pdf	11 0.13 %
6	2021_61210000_Kvyk_Dmytro_Ihorovych_85816 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	10 0.12 %

з бази даних RefBooks (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з домашньої бази даних (0.14 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2022_Б_ПІ_ПЗПІ_18_5_Ємельянова_К_О [redacted] 5/30/2024 Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics)	12 (1) 0.14 %

з програми обміну базами даних (0.12 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	2021_61210000_Kvyk_Dmytro_Ihorovych_85816 [redacted] 10/25/2024 National University "Lviv Politechnika" (National University Lviv Politechnika)	10 (1) 0.12 %

з Інтернету (0.94 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula	39 (2) 0.46 %
2	**http://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points-haversine-formula**	**29 (1) 0.35 %**
3	https://openarchive.nure.ua/bitstream/document/11518/1/2019_M_PI_Zymarev_KD.pdf	11 (1) 0.13 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд презентації 1 (рисунок виконано самостійно)

Рисунок Б.2 – Слайд презентації 2 (рисунок виконано самостійно)

Рисунок Б.3 – Слайд презентації 3 (рисунок виконано самостійно)

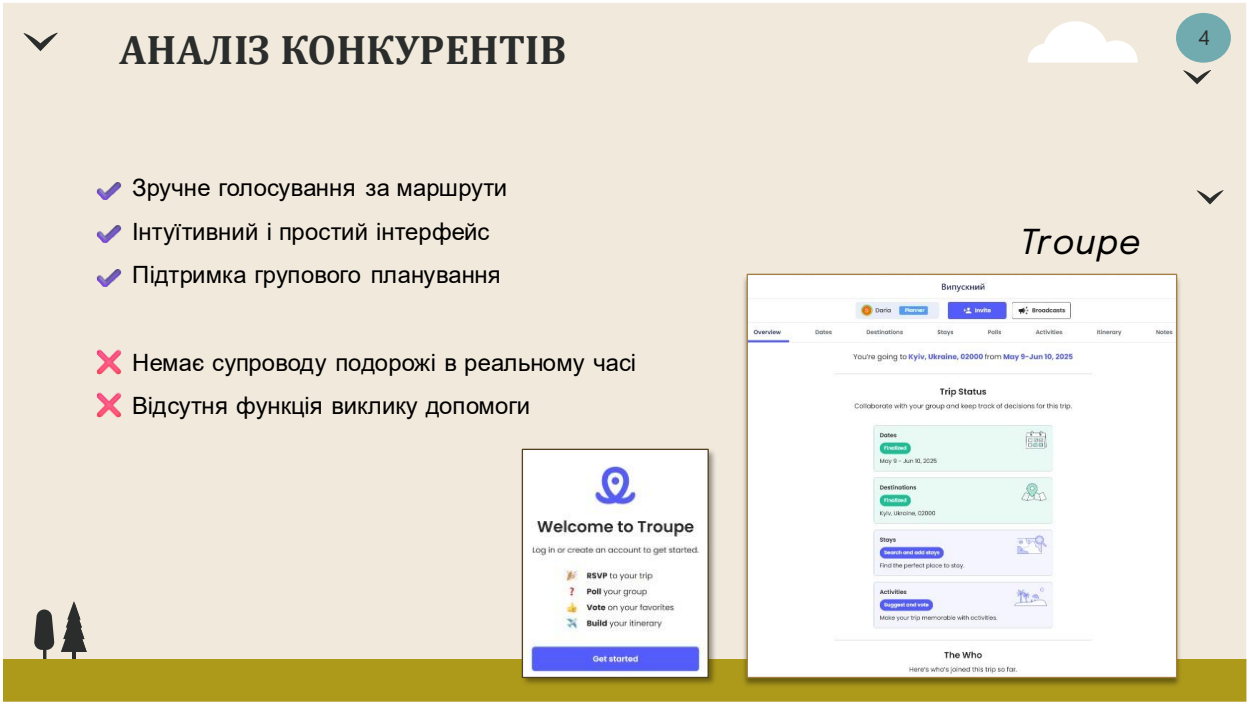


Рисунок Б.4 – Слайд презентації 4 (рисунок виконано самостійно)

АНАЛІЗ КОНКУРЕНТІВ

5

Wanderlog

- ✔ Побудова маршрутів для різних видів активності
- ✔ Високоточні карти з урахуванням рельєфу
- ✔ Синхронізація з пристроями та офлайн-доступ

- ✘ Відсутній обмін повідомленнями між учасниками
- ✘ Немає моніторингу стану учасників
- ✘ Відсутні механізми екстреного реагування

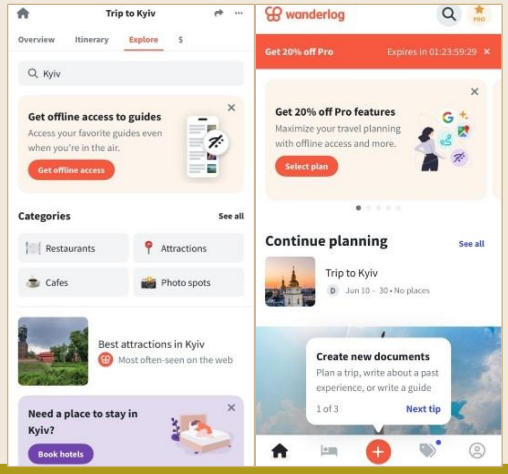


Рисунок Б.5 – Слайд презентації 5 (рисунок виконано самостійно)

ВИСНОВКИ З АНАЛІЗУ КОНКУРЕНТІВ

6

Існуючі сервіси фокусуються переважно на плануванні подорожей, а не на супроводі в реальному часі.

Жодна з популярних систем не забезпечує:

- ✘ відстеження координат усіх учасників у режимі реального часу
- ✘ виклики допомоги або реагування на критичні ситуації

Ключові потреби, які має реалізувати система:

- ✔ Збір та обробка GPS-координат і пульсу кожного учасника
- ✔ Відображення маршруту, статистики, сигналів SOS
- ✔ Комунікація між учасниками та роль організатора
- ✔ Безпечне зберігання даних і швидка передача

Рисунок Б.6 – Слайд презентації 6 (рисунок виконано самостійно)

СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

7

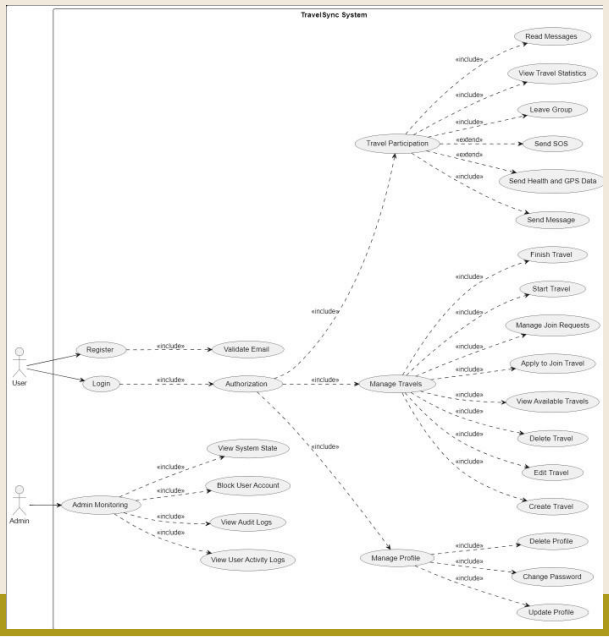


Рисунок Б.7 – Слайд презентації 7 (рисунок виконано самостійно)

АРХІТЕКТУРА СИСТЕМИ

8

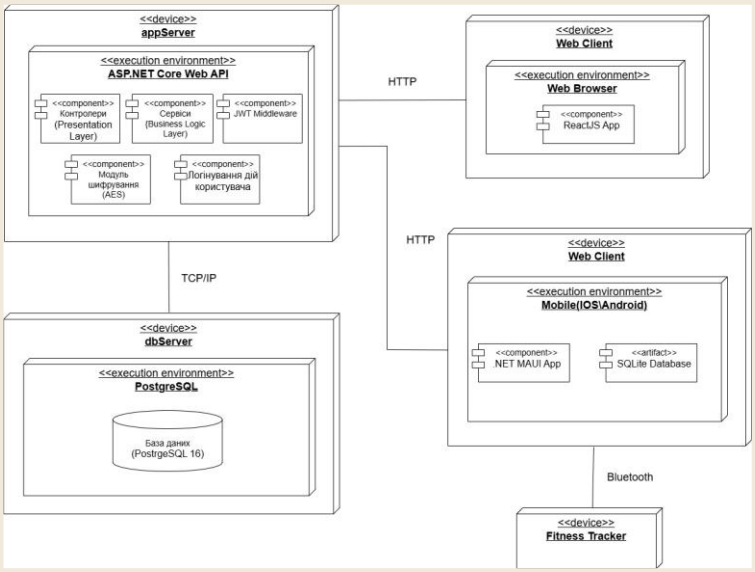


Рисунок Б.8 – Слайд презентації 8 (рисунок виконано самостійно)

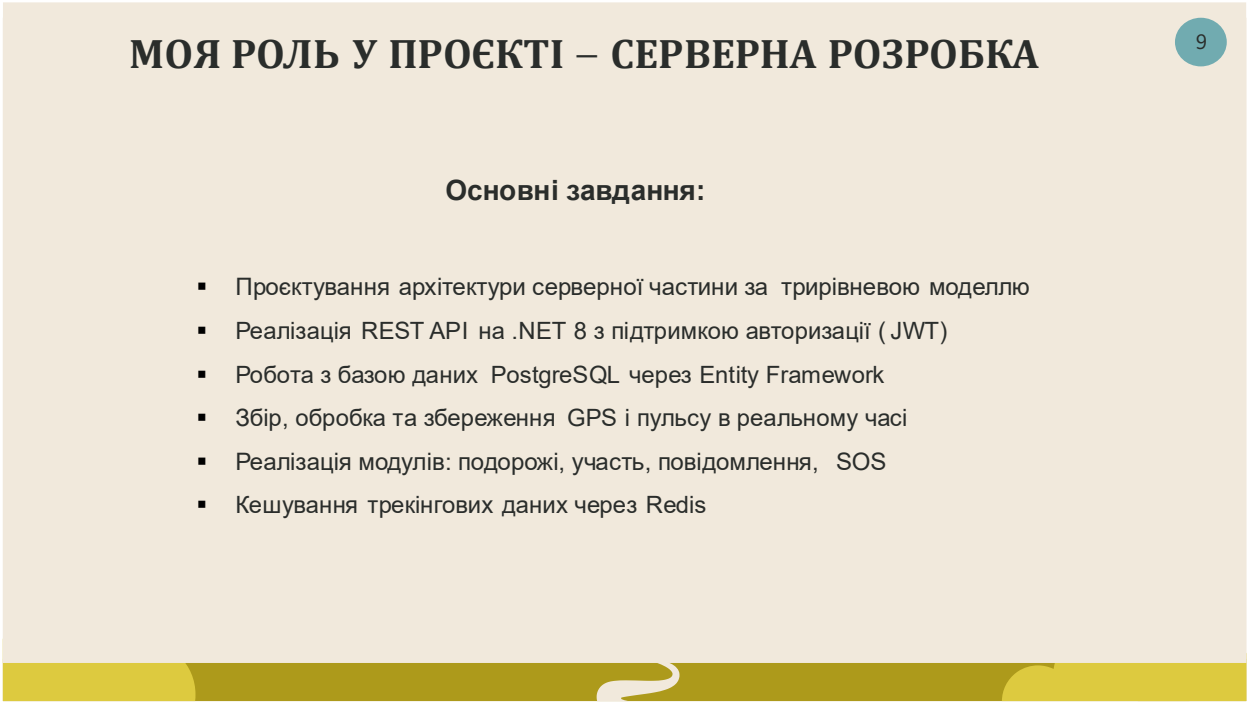


Рисунок Б.9 – Слайд презентації 9 (рисунок виконано самостійно)

Рисунок Б.10 – Слайд презентації 10 (рисунок виконано самостійно)

Рисунок Б.11 – Слайд презентації 11 (рисунок виконано самостійно)

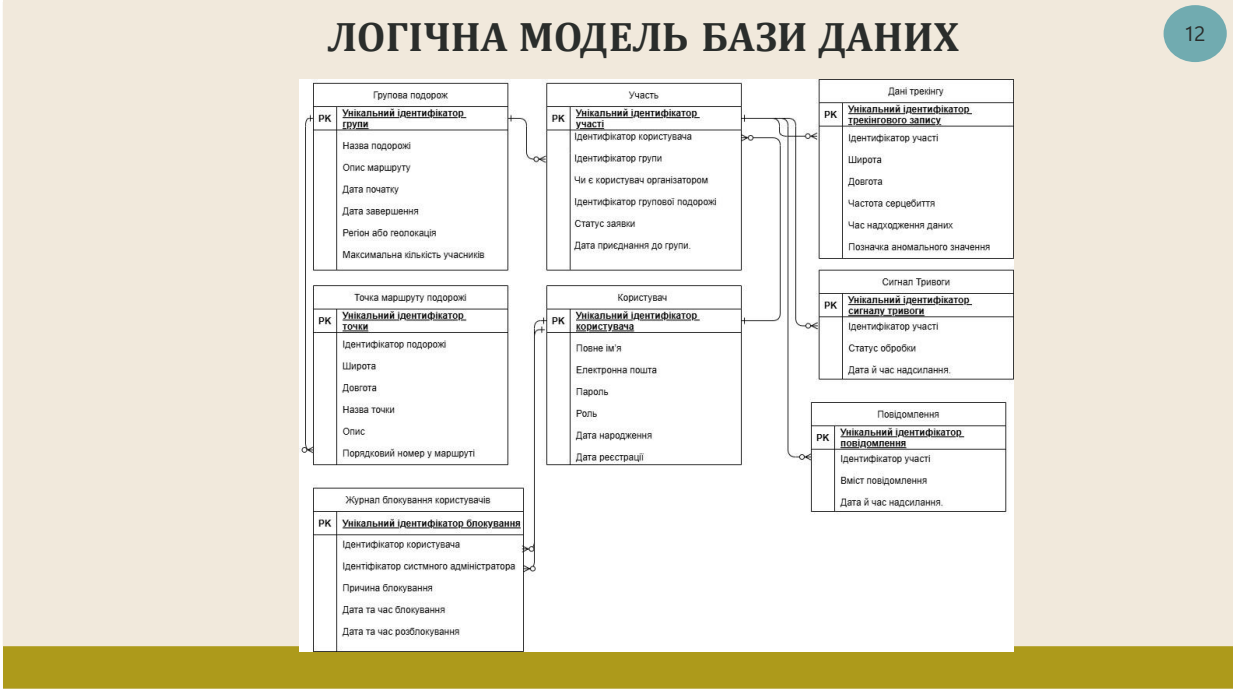


Рисунок Б.12 – Слайд презентації 12 (рисунок виконано самостійно)

ТРЕКІНГ ТА КЕШУВАННЯ

13

Трекінг у реальному часі

- Дані (GPS + пульс) надходять кожні 20 секунд.
- Частота серцебиття згладжується перед збереженням.
- Використовується формула Гаверсина для обчислення дистанції.

Кешування через Redis

- Зберігаються останні 5 трекінгових точок на користувача.
- Кеш використовується для автоматичного виявлення аномалій.

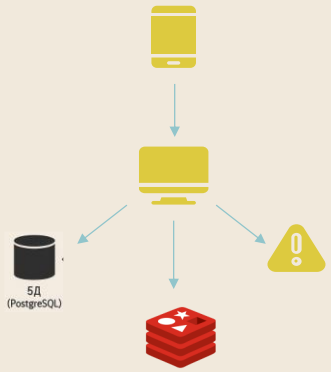


Рисунок Б.13 – Слайд презентації 13 (рисунок виконано самостійно)

ОБРОБКА ДАНИХ В СИСТЕМІ

14

Алгоритми в системі:

1. Алгоритм Гаверсину для обчислення дистанції між точками маршруту.
2. Ковзне середнє для згладжування пульсу для аналізу стану.
3. Автоматичний SOS для виявлення критичних значень пульсу.

Обчислення дистанції (формула Гаверсина)

```
public double CalculateDistanceKm(double
lat1, double lon1, double lat2, double lon2)
{
    const double R = 6371; // радіус Землі в
км
    var dLat = DegreesToRadians(lat2 - lat1);
    var dLon = DegreesToRadians(lon2 - lon1);
    var a = Math.Sin(dLat / 2) *
Math.Sin(dLat / 2) +
            Math.Cos(DegreesToRadians(lat1))
* Math.Cos(DegreesToRadians(lat2)) *
            Math.Sin(dLon / 2) *
Math.Sin(dLon / 2);
    var c = 2 * Math.Atan2(Math.Sqrt(a),
Math.Sqrt(1 - a));
    return R * c;
}
```

Рисунок Б.14 – Слайд презентації 14 (рисунок виконано самостійно)

15

ЗГЛАДЖУВАННЯ ПУЛЬСУ ТА АЛГОРИТМ ЕКСТРЕНОГО ВИКЛИКУ

Згладжування *пульсу*

Зберігаються останні 5 вимірів.

Кожне нове значення додається в кінець, найстаріше видаляється.

```
if (queue.Count == 5)
    queue.Dequeue();
queue.Enqueue(newValue);
return (int)Math.Round(queue.Average());
```

Генерація SOS-*сигналу*

Якщо всі останні точки мають критичні значення пульсу, то створється екстрений виклик.

Підтримується ручна активація SOS через застосунок.

```
if (points.All(p => p.SmoothedHeartRate <
35 || p.SmoothedHeartRate >190))
    await CreateEmergencyCallAsync(...);
```

Рисунок Б.15 – Слайд презентації 15 (рисунок виконано самостійно)

16

БЕЗПЕКА СИСТЕМИ

Шифрування чутливих даних

Геокоординати та пульс кожного учасника шифруються перед збереженням у базу.

Застосовується AES (Advanced Encryption Standard).

Авторизація та автентифікація

Для доступу до захищених API використовується JWT-автентифікація.

Система перевіряє роль користувача перед виконанням критичних дій.

```
{   ValidateIssuerSigningKey = true,
IssuerSigningKey = new
SymmetricSecurityKey(key),
ValidateIsSymmetricSecurityKeysuer = true,
ValidIssuer = _jwtSettings.Issuer,
ValidateAudience = true, ValidAudience =
_jwtSettings.Audience, ClockSkew = TimeSpan.Zero
}
```

Рисунок Б.16 – Слайд презентації 16 (рисунок виконано самостійно)

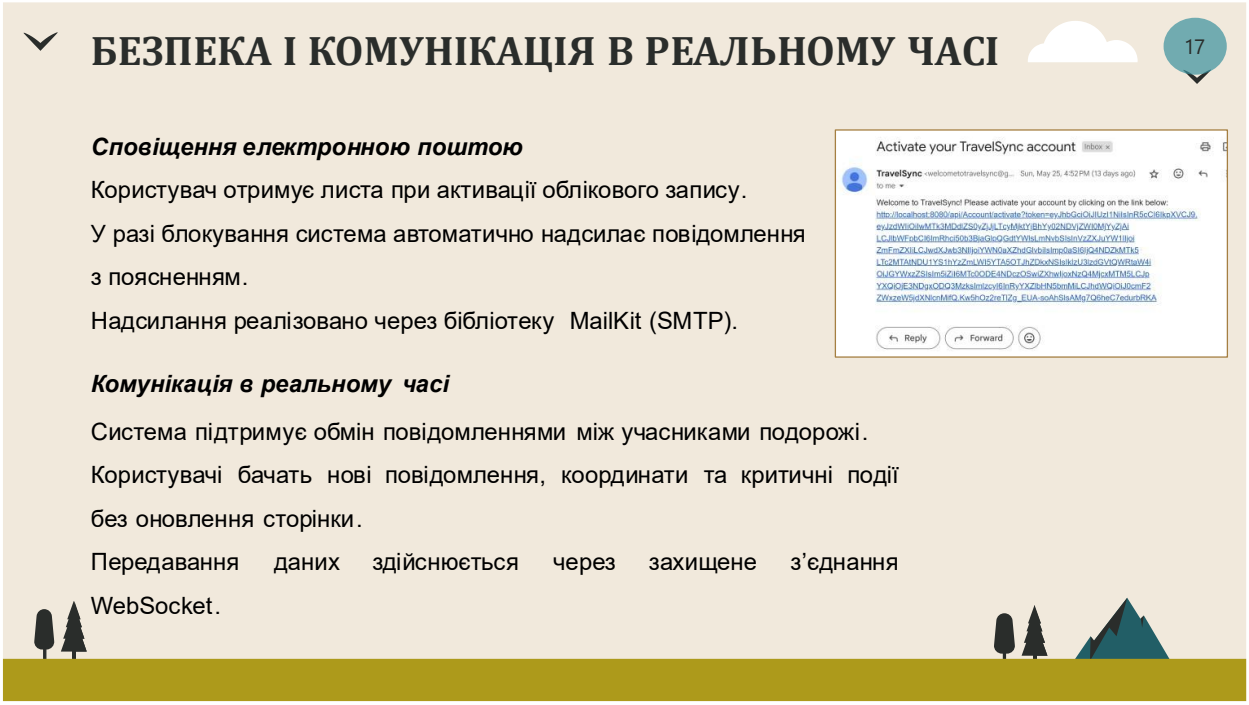


Рисунок Б.17 – Слайд презентації 17 (рисунок виконано самостійно)

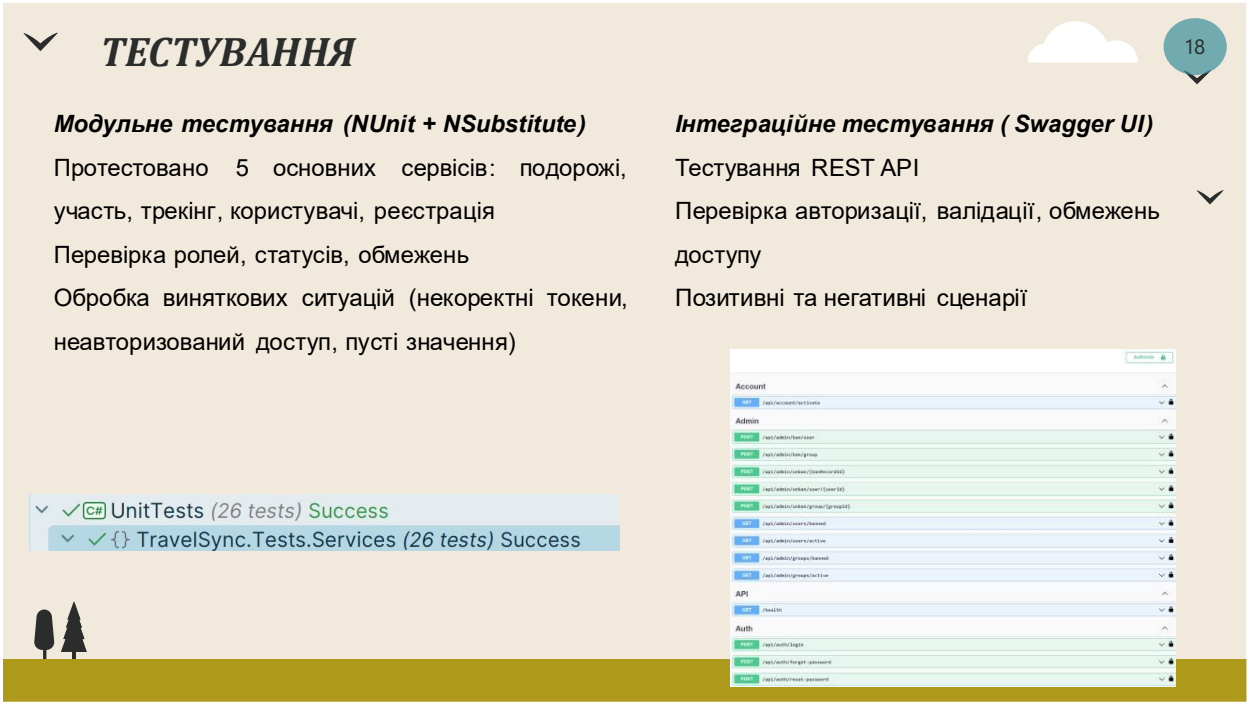
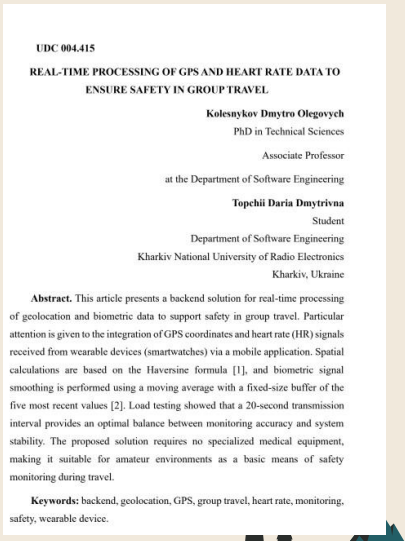


Рисунок Б.18 – Слайд презентації 18 (рисунок виконано самостійно)

АПРОБАЦІЯ

19



UDC 004.415

REAL-TIME PROCESSING OF GPS AND HEART RATE DATA TO ENSURE SAFETY IN GROUP TRAVEL

Kolesnykov Dmytro Olegovych
PhD in Technical Sciences
Associate Professor
at the Department of Software Engineering

Topchii Daria Dmytrivna
Student
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine

Abstract. This article presents a backend solution for real-time processing of geolocation and biometric data to support safety in group travel. Particular attention is given to the integration of GPS coordinates and heart rate (HR) signals received from wearable devices (smartwatches) via a mobile application. Spatial calculations are based on the Haversine formula [1], and biometric signal smoothing is performed using a moving average with a fixed-size buffer of the five most recent values [2]. Load testing showed that a 20-second transmission interval provides an optimal balance between monitoring accuracy and system stability. The proposed solution requires no specialized medical equipment, making it suitable for amateur environments as a basic means of safety monitoring during travel.

Keywords: backend, geolocation, GPS, group travel, heart rate, monitoring, safety, wearable device.

Рисунок Б.19 – Слайд презентації 19 (рисунок виконано самостійно)

ВИСНОВКИ

20

Розроблено серверну частину системи для організації та супроводу групових подорожей.

Забезпечено підтримку обміну даними в реальному часі, захист чутливої інформації, обробку участі, трекінгу, повідомлень та екстрених сигналів.

Система протестована, інтегрована з клієнтськими застосунками та готова до повноцінного використання.

Рисунок Б.18 – Слайд презентації 18 (рисунок виконано самостійно)

ДОДАТОК В

Наукова стаття, подана до X Міжнародної науково-практичної конференції «Science in the Modern World: Innovations and Challenges»

(12–14.06.2025, м. Торонто, Канада)

Рисунок В.1 – Сертифікат участі у міжнародній конференції

UDC 004.415

REAL-TIME PROCESSING OF GPS AND HEART RATE DATA TO ENSURE SAFETY IN GROUP TRAVEL

Kolesnykov Dmytro Olegovych

PhD in Technical Sciences

Associate Professor

at the Department of Software Engineering

Topchii Daria Dmytrivna
Student
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine

Abstract. This article presents a backend solution for real-time processing of geolocation and biometric data to support safety in group travel. Particular attention is given to the integration of GPS coordinates and heart rate (HR) signals received from wearable devices (smartwatches) via a mobile application. Spatial calculations are based on the Haversine formula [1], and biometric signal smoothing is performed using a moving average with a fixed-size buffer of the five most recent values [2]. Load testing showed that a 20-second transmission interval provides an optimal balance between monitoring accuracy and system stability. The proposed solution requires no specialized medical equipment, making it suitable for amateur environments as a basic means of safety monitoring during travel.

Keywords: backend, geolocation, GPS, group travel, heart rate, monitoring, safety, wearable device.

Object of the study. The object of the study is a backend software system designed for real-time processing of geolocation and biometric data collected during group travel. The system focuses on receiving, verifying, analysing, and storing heart rate and GPS coordinates from wearable devices via a mobile application. The study explores the methods and technical implementation required to ensure timely detection of potentially dangerous health conditions during travel without the need for specialized medical equipment.

Relevance of the study. Most existing solutions for travel support focus solely on navigation, without monitoring the physiological state of users. As a result, sudden heart rhythm disturbances, immobility, or signs of fatigue often go unnoticed. While some advanced systems do support biometric monitoring, they

usually require specialized medical equipment or dedicated sensors, making them inaccessible for amateur users. The proposed system combines spatial and biometric data using only a smartwatch and a mobile application, providing an affordable and practical safety tool for group travel.

Purpose of the study. To develop the backend of a software system capable of receiving, processing, validating, and storing spatial and biometric data of participants in group travel in real time.

Materials and methods. The server part of the system is implemented using the ASP.NET 8 framework and the PostgreSQL database management system. The client application transmits GPS coordinates, heart rate values, and timestamps. Spatial data is processed using the Haversine formula [1], while biometric signals are smoothed using a moving average with a fixed window size [2]. Reference ranges for heart rate are based on WHO guidelines [3]: 60–100 bpm at rest and up to 180–190 bpm during physical activity.

Results and discussion. The backend system, built with ASP.NET 8 [4] and PostgreSQL, receives telemetry data over a secure HTTPS connection. Each transmission includes GPS coordinates, heart rate values, and a timestamp. Before processing, the system authenticates the request, verifies the user's participation in the journey, and validates the structure of the incoming data. A centralized result processor handles progress checks, diagnostic messages, and error codes, providing a unified failure handling mechanism at both the business logic and database levels. The full server-side source code is publicly available on GitHub [5].

One of the key parameters affecting monitoring performance and accuracy is the message transmission interval. To determine the optimal frequency, load testing was conducted using a simulation of 1,000 concurrent clients. The k6 utility [6] was used to generate a controlled load on the API. In the test scenario, 1,000 users simultaneously sent messages at predefined intervals (10, 20, and 30 seconds). Each request included GPS coordinates, heart rate data, and a timestamp.

During testing, the system recorded metrics such as the percentage of successfully processed requests, failure rates, and average server response time. The results are shown in Table 1.

Таблиця В.1 – Результати навантажувального тестування

Transmission interval, s	Successful requests, %	Refusals, %	Average response time, ms
10	91,3	8,7	290
20	98,9	1,1	260
30	99,2	0,8	230

As the table shows, a 10-second transmission interval results in a higher failure rate and increased response time. A 30-second interval reduces server load but delays the system's response to physiological changes. Therefore, a 20-second interval was selected as the optimal compromise between responsiveness and performance.

For spatial processing, the system uses a dedicated module that calculates the distance between two route points based on the Haversine formula [1]. The calculations follow these formulas:

$$a = sin^2(\Delta\varphi/2) + cos(\varphi_1) \cdot cos(\varphi_2) \cdot sin^2(\Delta\lambda/2) \quad (1)$$

$$c = 2 \cdot arctan2(\sqrt{a}, \sqrt{(1-a)}) \quad (2)$$

$$d = R \cdot c \quad (3)$$

The theoretical error of the Haversine formula is below 0.5% for distances up to 100 km, which is acceptable for amateur-level monitoring. For short segments (up to 10 meters), the deviation is less than 1 meter [7], as confirmed in test scenarios. Compared to the spherical law of cosines or Euclidean distance, the Haversine formula performs more reliably on short segments [8], which is critical for high-frequency tracking. Although Vincenty's algorithm [9] offers higher

precision, its computational complexity makes it less suitable for real-time applications [10].

The calculated distance between successive GPS points is accumulated for each user within the current journey and stored in the database. This spatial data is later used to reconstruct the route, analyse user activity, identify potentially hazardous zones, and generate personalized recommendations.

Heart rate processing is handled by a pre-smoothing module. Signals from wearable devices often contain artifacts caused by motion, poor sensor contact, or environmental interference. To reduce noise sensitivity, a moving average method with a fixed window size of n = 5 was applied [2], calculated as follows:

$$MA_{\mathrm{t}} = \frac{1}{n} \cdot \sum_{i=t-n+1}^{i=t} x_{\mathrm{i}} \tag{4}$$

where x_i is the heart rate value at time i, MA_t is the smoothed value at time t, and n is the size of the sliding window.

At the implementation level, each user is assigned a separate buffer that stores the five most recent heart rate values. With every incoming request, the buffer is updated, and the smoothed value is independently stored in the database. This approach helps prevent false alerts and ensures system stability even when input data is noisy.

The moving average method was chosen due to its low computational complexity and suitability for real-time processing under limited resource conditions. During testing, it was observed that more complex techniques, such as the Kalman filter, require prior model training and careful parameter tuning, which limits their applicability in amateur settings [11]. By contrast, the moving average method provides stable results without the need for calibration, making it a practical choice for wearable-based health monitoring.

In addition to smoothed values, the system also stores raw heart rate data for further analysis, model training, or verification of boundary events. To detect critical conditions, the system monitors whether heart rate values exceed predefined safety thresholds. According to WHO guidelines [3], the normal resting heart rate is between 60–100 bpm, and up to 180–190 bpm during physical activity. The system sets fixed alert thresholds at 50 and 190 bpm. Exceeding these limits without contextual justification triggers an alert in the internal event log. Future versions of the system are planned to support personalized thresholds based on age, medical history, or user-specific recommendations.

To ensure the integrity of time series data, the system verifies the uniqueness of each incoming message based on the combination of user, journey, and timestamp. This mechanism prevents duplicate entries and preserves the correct chronological order.

To evaluate the effectiveness of telemetry processing, a time series of records was analysed, each containing GPS coordinates and heart rate values. Messages were received at 20-second intervals, capturing both spatial and biometric information from the user's wearable device (see Table 2).

Таблиця В.2 – Телеметричні дані частоти серцевих скорочень і координат під час контрольної подорожі

№	Time, s	Latitude	Longitude	Raw heart rate value	Smoothed heart rate value	Distance from the previous point, m
1	0	49.8397	24.0297	98	98	0
2	20	49.8398	24.0298	105	101,5	13
3	40	49.8400	24.0300	70	91	22
4	60	49.8402	24.0302	175	112	21
5	80	49.8403	24.0303	92	108	11
6	100	49.8404	24.0304	96	107,6	11

7	120	49.8405	24.0305	194	111.4	10
8	140	49.8406	24.0306	198	123.0	10
9	160	49.8407	24.0307	202	151.8	11
10	180	49.8408	24.0308	201	172.2	10
11	200	49.8409	24.0309	195	178.8	10
12	220	49.8410	24.0310	180	195.2	10

The distances between consecutive telemetry points are calculated using the Haversine formula integrated into the spatial processing module. This enables the system to track the accumulated distance travelled by the user, which serves as a variable for further analysis and visualization.

Heart rate smoothing is implemented in real time using the moving average method. The buffer of recent measurements is updated with each new request. This makes it possible to stabilize the heart rate signal without significantly reducing sensitivity to physiological changes. As illustrated in the graph (Figure 1), raw heart rate values exhibit pronounced fluctuations and sharp peaks, while the smoothed curve presents a more gradual and realistic trend aligned with physical effort and distance.

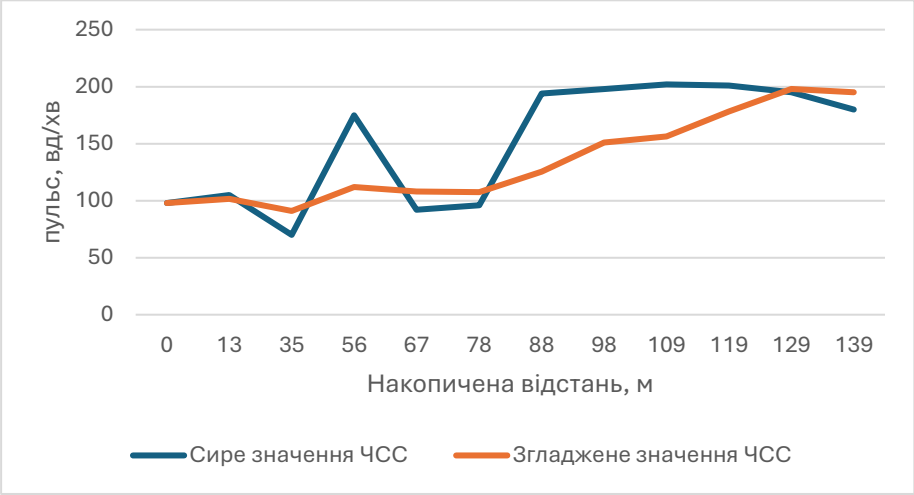


Рисунок В.1 – Зміна ЧСС при зростанні відстані (рисунок виконано самостійно)

The moving average algorithm significantly improves the reliability of biometric monitoring by reducing transient artifacts, including sudden spikes caused by movement or poor sensor contact. As shown in the graph, raw values display sharp peaks, with heart rate reaching 202 beats per minute, which may not indicate actual physiological stress. In contrast, the smoothed signal reflects consistent trends more accurately.

Over the entire control interval, the smoothed heart rate steadily increased in accordance with the distance travelled and the expected physical exertion. This demonstrates that the algorithm is sensitive to real physiological changes, while effectively filtering out random noise. A single instance of the smoothed heart rate exceeding the critical threshold of 190 beats per minute at 139 meters was enough to activate an SOS alert. Although this conservative approach prioritizes user safety, future versions of the system may support more flexible rules, such as requiring repeated threshold violations to avoid false alerts.

The combined use of the Haversine algorithm and the moving average method is functionally well-grounded. The Haversine algorithm ensures reliable spatial measurements necessary for route reconstruction, speed estimation, and activity monitoring. At the same time, the moving average method guarantees a stable and clear biometric signal, resistant to short-term artifacts and noise. Both algorithms are lightweight in terms of computational load, which enables their efficient use in real-time mobile environments with limited processing resources.

The system demonstrates a high level of reliability in detecting critical physiological deviations under constrained conditions, confirming its applicability in real-world group travel scenarios.

Conclusion. This article presents the implementation of the server-side component of a monitoring system for group travel participants. The solution provides real-time processing of geolocation and biometric data using only a mobile application, wearable devices, and a cloud backend. It does not require specialized

medical or communication equipment, which makes it accessible for non-professional use.

All core computing mechanisms are integrated into the backend logic, ensuring system stability, performance, and compatibility with the limited resources of mobile environments. The Haversine formula was used to calculate the distance travelled with high precision, including on short route segments. Biometric data is processed using the moving average method with a fixed-size buffer, which effectively reduces noise and enhances monitoring accuracy.

Load testing confirmed that a 20-second data transmission interval offers an optimal balance between responsiveness and server performance. The system can identify potentially dangerous conditions, such as heart rate anomalies or immobility, based on a combined analysis of geolocation and biometric inputs. All telemetry data is stored in a way that preserves time series integrity and complies with confidentiality requirements [12].

Further development of the system will focus on personalized biometric analysis, machine learning integration, and connection with external emergency response services.

REFERENCES

1. Van Brummelen, Glen Robert. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry* . – Princeton University Press, 2013. – ISBN 9780691148922. – Access mode: https://press.princeton.edu/books/paperback/9780691148922/heavenly-mathematics (accessed 05.06.2025). – Title from the screen.

2. Zhang, Zhiyong et al. *Use Moving Average Filter to Reduce Noises in Wearable PPG During Continuous Monitoring* . – ResearchGate. – Access mode:

https://www.researchgate.net/publication/311257038 (accessed 05.06.2025). – Title from the screen.

3. Harvard Health Publishing. *What Your Heart Rate is Telling You* . – Access mode: https://www.health.harvard.edu/heart-health/what-your-heart-rate-is-telling-you (accessed 05.06.2025). – Title from the screen.

4. Microsoft Learn. *ASP.NET Core Identity* . – Access mode: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity (accessed 05.06.2025). – Title from the screen.

5. Topchii, D. GitHub Repository: TravelSync – Real-Time Group Travel Monitoring System [Electronic resource]. – Access mode: https://github.com/dariatopchii/TravelSync (accessed 06.06.2025). – Title from the screen.

6. Grafana Labs. *API Load Testing with k6* [Electronic resource]. – Access mode: https://grafana.com/docs/k6/latest/testing-guides/api-load-testing/ (accessed 05.06.2025). – Title from the screen.

7. Azdy, R. A., & Darnis, F. *Use of Haversine Formula in Finding Distance Between Temporary Shelter and Waste End Processing Sites*. // Journal of Physics: Conference Series. – 2020. – Vol. 1500, No. 1. – DOI: 10.1088/1742-6596/1500/1/012104. – Title from the screen. – Accessed: 06.06.2025.

8. Sinnott, R. W. Virtues of the Haversine. // *Sky and Telescope*. – 1984. – Vol. 68, No. 2. – P. 159. – Title from the screen. – Accessed: 06.06.2025.

9. Vincenty, T. (1975). *Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations*. Survey Review, 23(176), 88–93. https://doi.org/10.1179/sre.1975.23.176.88

10. Karney, C. F. F. Algorithms for Geodesics. // *Journal of Geodesy*. – 2013. – Vol. 87, No. 1. – P. 43–55. – DOI: 10.1007/s00190-012-0578-z. – Title from the screen. – Accessed: 06.06.2025.

11. Sun, Y., & Thakor, N. (2016). *Photoplethysmography Revisited: From Contact to Noncontact, From Point to Imaging*. IEEE Transactions on Biomedical Engineering, 63(3), 463–477. DOI: 10.1109/TBME.2015.2476337.

12. PostgreSQL Documentation. *Data Encryption Options* [Electronic resource]. – Access mode: https://www.postgresql.org/docs/current/encryption-options.html (accessed 05.06.2025). – Title from the screen.

ДОДАТОК Г

Специфікація вимог до програмного продукту

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук

Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмна система TravelMate

Студент гр. ПЗПІ-21-3

Топчій Д.Д

Студент гр. ПЗПІ-21-3

Черкасова Е.С

Студент гр. ПЗПІ-21-3

Фомичов А.С.

Харків

2025

ЗМІСТ

1 ВСТУП

1.1 Огляд продукту

TravelMate - це комплексна система для організації та супроводу групових подорожей. Продукт спрямований на забезпечення підтримки мандрівників, які подорожують групами: від планування маршруту та узгодження деталей до моніторингу стану здоров’я учасників і зручного обміну повідомленнями.

Основна ідея полягає в інтеграції функціоналу для координації групових дій, безпечного обміну даними та спільного планування, що дозволяє уникнути використання багатьох розрізнених сервісів і підвищити ефективність комунікації в процесі подорожі.

Ключові функції TravelMate включають:

– Планування маршруту подорожі – створення детального плану пересування з можливістю його редагування.

– Організація групових комунікацій – вбудований чат для оперативної взаємодії.

– Збір і відображення статистики – інформація про стан здоров’я (наприклад, пульс), маршрут та інші показники подорожі.

– Керування подорожами та профілями користувачів, а саме – створення, редагування, видалення подорожей та управління власним профілем.

– Швидка відправка тривожного сигналу.

1.2 Мета

Метою цього документу є надання детального опису розроблюваної програмної системи для організації та супроводу групових подорожей. Специфікація призначена для опису функціональних і нефункціональних вимог першої версії продукту, що включає серверну частину, мобільний застосунок і вебклієнт.

1.3 Межі

Розроблена система призначена для використання в умовах групових подорожей. Функції включають відстеження локації та пульсу користувачів, обмін повідомленнями і виклик допомоги. Нижче наведено обмеження за технічними, функціональними та організаційними аспектами.

Технічні межі:

– Підтримувані платформи: Android, iOS, сучасні веббраузери (Chrome, Firefox, Safari, Edge). Сумісність із застарілими ОС і браузерами не гарантується.

– Частота оновлення: трекінгові дані (геопозиція та пульс) надходять не частіше одного запису на 20 секунд від одного користувача. Вища частота або обробка великих обсягів медичних даних не передбачені.

– Масштабованість: система розрахована на велику кількість одночасних користувачів; у разі перевищення навантаження застосовуються обмеження частоти запитів та балансування. Обробка Big Data і аналітика в режимі реального часу не підтримується

– Залежність від Інтернет-з’єднання: всі функції (відстеження, обмін повідомленнями, оновлення станів) потребують стабільного з’єднання. Офлайн-режим не підтримується.

Функціональні межі:

– Подорожі: реалізовано створення, приєднання та відстеження групових подорожей. Логістичні функції не підтримуються.

– SOS: система дозволяє подати сигнал допомоги (SOS); інтеграція з офіційними службами (102, 112) відсутня; обробку сигналу забезпечують організатори або інші учасники подорожі.

– Інтеграція зі смартгодинниками: мобільний застосунок отримує дані від смартгодинників, сумісних з обраною платформою. Синхронізація може мати затримки через нестабільне Bluetooth-з’єднання або переповнений буфер пристрою.

Організаційні межі:

– Технічне обслуговування: можливі періоди оновлень або профілактики, під час яких окремі функції можуть бути тимчасово недоступні.

– Юридичні та етичні обмеження: система не призначена для критичних медичних застосувань і не замінює служби екстреної допомоги. Рішення щодо дій у надзвичайних ситуаціях приймаються користувачами за їх власною відповідальністю.

Система не призначена для використання як медичний пристрій або засіб екстреного реагування. Критичні рішення під час її використання мають ухвалюватися під контролем людини. Призначення системи — координація групових подорожей, моніторинг місцезнаходження та обмін інформацією між учасниками.

1.4 Посилання

– ISO/IEC 25010:2011 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

– Docker Documentation. Доступно за посиланням: https://docs.docker.com/.

– PostgreSQL Documentation. Доступно за посиланням: https://www.postgresql.org/docs/.

1.5 Означення та абревіатури

У цьому документі застосовуються такі основні терміни, означення та абревіатури:

– API (Application Programming Interface) – інтерфейс прикладного програмування, набір функцій і протоколів для взаємодії між різними компонентами програмного забезпечення.

– HTTP (HyperText Transfer Protocol) – протокол передавання гіпертексту, який використовується для обміну даними між веббраузерами і вебсерверами.

– HTTPS (HyperText Transfer Protocol Secure) – захищена версія протоколу HTTP, що забезпечує шифрування переданих даних за допомогою SSL або TLS.

– JSON (JavaScript Object Notation) – формат обміну даними, який використовується для зручного зберігання та передавання структурованої інформації.

– REST (Representational State Transfer) – архітектурний стиль розробки вебсервісів, що передбачає використання стандартних HTTP-методів (GET, POST, PUT, DELETE тощо).

– JWT (JSON Web Token) – формат токенів, який застосовується для безпечної передачі інформації між учасниками системи.

– UI (User Interface) – користувацький інтерфейс, призначений для взаємодії користувача з програмною системою.

– UX (User Experience) – користувацький досвід, що описує загальне сприйняття та взаємодію користувача із системою.

– GPS (Global Positioning System) – глобальна навігаційна супутникова система для визначення місця розташування.

– CI/CD (Continuous Integration / Continuous Deployment) – безперервна інтеграція та безперервне розгортання, що забезпечують автоматизацію процесів тестування та доставки нових версій програмного забезпечення.

– Push-сповіщення – повідомлення, які система надсилає на пристрій користувача для інформування про важливі події.

– SOS-сповіщення – спеціальне повідомлення, що використовується у системі для виклику екстреної допомоги.

– Модуль – частина програмного забезпечення, яка виконує певну функцію та є незалежною одиницею у складі системи.

– Система – вебзастосунок TravelMate разом із мобільними клієнтами та серверною частиною.

– Користувач – особа, яка взаємодіє із системою через мобільний застосунок або вебінтерфейс.

– Адміністратор – користувач із розширеними правами для управління параметрами системи та її користувачами.

2.1 Перспективи продукту

Подальший розвиток системи TravelMate передбачає такі можливі напрями:

– Розширення функцій – додавання модулів для бронювання квитків, оплати послуг, взаємодії з картографічними сервісами (Google Maps, OpenStreetMap).

– Інтеграція сторонніх сервісів – підключення зовнішніх API: платіжних систем, платформ для розміщення, гідів тощо.

– Масштабування – забезпечення стабільної роботи при зростанні кількості активних користувачів і запитів.

– Розвиток соціальних функцій – реалізація механізмів публікації відгуків, формування рейтингів і рекомендацій.

– Підтримка багатомовності – локалізація інтерфейсу та контенту для використання в різних регіонах.

– Аналітика та алгоритми рекомендацій – використання статистичних моделей і методів машинного навчання для аналізу активності користувачів і формування персоналізованих пропозицій.

2.2 Функції продукту

Система TravelMate реалізує низку ключових функцій, спрямованих на забезпечення ефективної організації подорожей, взаємодії учасників та контролю за безпекою мандрівників. Основні функції продукту:

— Реєстрація та автентифікація користувачів. Система забезпечує створення облікових записів для користувачів та їхню безпечну автентифікацію з використанням сучасних методів захисту (паролі, токени тощо).

— Створення та управління подорожами. Користувачі можуть створювати нові подорожі, вказуючи ключову інформацію (маршрут, час, учасників), а також редагувати та видаляти їх за потреби.

— Приєднання до подорожі. Користувачі можуть приєднуватися до вже створених подорожей, отримуючи доступ до їхньої інформації та можливість взаємодії з іншими учасниками.

— Відображення учасників на мапі. Система відображає поточне розташування всіх учасників подорожі на карті в реальному часі для підвищення безпеки та зручності.

— Відстеження телеметрії. Збір і відображення інформації про фізичний стан користувачів (наприклад, пульс), що дозволяє швидко реагувати на можливі надзвичайні ситуації.

— Модуль SOS. Реалізовано функцію надсилання екстрених повідомлень у разі виникнення надзвичайних ситуацій для оперативної допомоги учасникам.

— Чат та обмін повідомленнями. Система надає можливість спілкування між учасниками подорожі у реальному часі, що підвищує ефективність взаємодії та координації.

— Перегляд статистики та історії подорожей. Користувачі можуть переглядати історію своїх подорожей, а також отримувати статистичні дані, що допомагають аналізувати активність та досвід.

Сукупність цих функцій дозволяє TravelMate стати потужним інструментом для організації, координації та підвищення безпеки групових подорожей.

2.3 Характеристики користувачів

Основними користувачами системи TravelMate є мандрівники та організатори групових подорожей. Кожна з цих категорій має власні вимоги та потреби, що враховані при розробці продукту.

Мандрівники шукають зручний та безпечний спосіб подорожувати разом із групою та цінують можливість отримувати актуальну інформацію про маршрут та інших учасників. Ця категорія користувачів потребує швидкої та зрозумілої комунікації під час подорожі, а також вони зацікавлені в функціях моніторингу здоров’я (пульс, фізичний стан) та безпеки (SOS). Організатори подорожей відповідають за координацію групи. Вони потребують простих інструментів для створення та управління подорожами, залучення учасників. Також є технічні користувачі – адміністратори. Вони потребують доступу до інтерфейсів управління правами доступу.

2.4 Загальні обмеження

Під час розробки та використання системи TravelMate слід враховувати низку загальних обмежень:

— Система потребує стабільного підключення до Інтернету для коректної роботи функцій, пов’язаних із обміном даними в реальному часі (наприклад, відстеження локації, чат, SOS).

— Обмеження пам’яті користувацьких пристроїв можуть впливати на швидкість і стабільність роботи застосунку.

— Деякі функції (наприклад, SOS-сповіщення або моніторинг пульсу) можуть бути доступні лише на пристроях, що підтримують відповідне обладнання (датчики пульсу, GPS).

— Доступ до даних про місцеперебування та особистої інформації має бути захищений та обмежений для сторонніх осіб.

— Усі користувачі повинні мати базові цифрові навички для роботи з додатком.

— Інтерфейс має бути достатньо універсальним, проте не перевантаженим функціями, щоб уникнути складнощів при використанні.

2.5 Припущення і залежності

Припущення: Користувачі мають постійний доступ до стабільного підключення до Інтернету.

Залежність: За відсутності підключення до Інтернету користувачі не зможуть використовувати більшість функцій застосунку, зокрема відстеження місцеперебування, чат та SOS-сповіщення.

Припущення: Усі користувачі зареєстровані в системі та пройшли автентифікацію.

Залежність: Незареєстровані користувачі не матимуть доступу до функцій застосунку.

Припущення: Користувацькі пристрої (смартфони, планшети, комп’ютери) відповідають мінімальним системним вимогам та підтримують необхідні браузери й операційні системи.

Залежність: Використання застарілих пристроїв або браузерів може призвести до помилок, зниження продуктивності або неможливості використання окремих функцій.

Припущення: Для роботи функцій моніторингу пульсу користувачі мають пристрої з необхідними датчиками (наприклад, фітнес-браслети або смартгодинники).

Залежність: Відсутність сумісного обладнання обмежує доступ до функцій моніторингу фізичного стану.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Система передбачає наявність зручного, інтуїтивно зрозумілого та ергономічного інтерфейсу, який забезпечує користувачам легку та ефективну взаємодію з програмним забезпеченням. Інтерфейс має забезпечувати швидкий доступ до основних функцій системи та бути адаптивним для роботи на різних типах пристроїв, включаючи персональні комп’ютери, планшети та мобільні телефони.

Користувацький інтерфейс реалізується у вигляді веб-додатку та мобільного застосунку, які відповідають вимогам користувачів і технічним можливостям відповідних платформ. Структура додатку повинна бути логічною, із чітко організованою навігацією та інтуїтивно зрозумілими елементами управління, що спрощує виконання основних завдань.

Для різних типів користувачів передбачено адаптацію інтерфейсу залежно від їхніх ролей у системі. Це дозволяє забезпечити персоналізований доступ до функціональності, відповідно до прав та обов’язків користувача.

3.1.2 Властивості програмного продукту

Апаратне забезпечення програмної системи TravelMate включає компоненти та засоби, необхідні для стабільної та безперебійної роботи

системи. Оскільки основна частина функціоналу реалізована як онлайн-сервіс із доступом через веб-браузер та мобільний застосунок, апаратна інфраструктура охоплює:

– Сервери для зберігання та обробки даних — фізичні або віртуальні сервери, а також хмарні платформи, що забезпечують достатні обчислювальні потужності та ресурси для надійної роботи бекенду, зберігання бази даних, обробки запитів користувачів та трекінгу;

– Мережеві пристрої, такі як маршрутизатори, комутатори та мережеві кабелі, які забезпечують стабільне та безперервне з'єднання між користувачами (через Інтернет або мобільні мережі) і серверами системи;

– Користувацькі пристрої, включаючи персональні комп’ютери, планшети та смартфони з підтримкою сучасних веб-браузерів та мобільних платформ, що використовуються для доступу до інтерфейсу системи;

– Bluetooth-пристрої, які підключаються до мобільного застосунку для зчитування показників пульсу та іншої інформації.

Забезпечення сумісності та оптимальної роботи на цих пристроях є ключовим фактором успішного функціонування системи TravelMate.

3.1.3 Програмний інтерфейс

Програмний інтерфейс (API) системи TravelMate служить основним каналом взаємодії між клієнтськими застосунками (веб та мобільними) та серверною частиною. API реалізовано у вигляді RESTful сервісу, що забезпечує стандартизований та ефективний обмін даними через HTTP-протокол.

Набір доступних операцій включає аутентифікацію та управління користувачами, створення, редагування та перегляд групових подорожей і маршрутів, обробку трекінгу місцезнаходження та пульсу, надсилання повідомлень і сигналів тривоги, а також доступ до журналу активностей.

Обмін інформацією відбувається у форматі JSON, що гарантує структурованість і простоту обробки даних між клієнтами та сервером.

Використання RESTful API забезпечує гнучкість, масштабованість системи і відкриває можливості для інтеграції з іншими сервісами у майбутньому, а також спрощує розробку та підтримку проєкту.

Для роботи з веб-інтерфейсом користувачам необхідний сучасний веб-браузер з підтримкою стандартів JavaScript, HTML5 та CSS3. Мобільний застосунок підтримується на пристроях з операційними системами Android версії 8 і новіше та iOS версії 13 і новіше.

3.1.4 Комунікаційний протокол

Система TravelMate підтримує сучасні протоколи комунікації, які забезпечують надійний та безпечний обмін даними між клієнтськими застосунками (веб та мобільними) і серверною частиною. Основним протоколом передачі даних є HTTP/HTTPS, що гарантує сумісність із широким спектром мережевих середовищ та забезпечує захищене з’єднання за допомогою SSL/TLS.

Для обміну інформацією між компонентами використовується архітектурний стиль REST, який спрощує інтеграцію з іншими системами та сервісами. Усі запити та відповіді здійснюються у форматі JSON, що дозволяє легко та ефективно обробляти дані.

Крім того, система забезпечує надійний рівень безпеки під час передачі даних: реалізовані механізми аутентифікації, авторизації та шифрування для захисту конфіденційної інформації та цілісності даних.

Веб-застосунок вимагає стабільного підключення до Інтернету для коректної роботи. Мобільний застосунок TravelMate також підтримує офлайн режим, що дозволяє користувачам працювати з базовими функціями без доступу до мережі. У цьому випадку дані локально кешуються і синхронізуються із сервером при відновленні підключення до Інтернету.

3.1.5 Обмеження пам’яті

Обмеження пам’яті є важливим фактором у розробці програмного забезпечення TravelMate, особливо з урахуванням різноманітності платформ — від серверного середовища до мобільних пристроїв із обмеженими ресурсами.

На серверній стороні застосовуються масштабовані ресурси пам’яті, проте для забезпечення високої продуктивності та стабільності системи необхідно оптимально використовувати доступні ресурси, уникаючи надмірного споживання пам’яті.

Мобільний застосунок працює на пристроях з обмеженою оперативною пам’яттю, тому важливо використовувати ефективні алгоритми та структури даних, мінімізувати зайве копіювання інформації, а також кешувати дані з урахуванням обмежень пам’яті.

Для запобігання витокам пам’яті і забезпечення стабільності роботи застосунків застосовується автоматичне управління пам’яттю (Garbage Collector) платформи .NET, а також проводиться ретельне тестування для виявлення і усунення потенційних проблем.

Загалом, ефективне управління пам’яттю та оптимізація ресурсів є ключовими для забезпечення продуктивності, надійності та стабільності системи TravelMate на всіх підтримуваних платформах.

3.1.6 Операції

Система TravelMate підтримує виконання основних операцій, необхідних для організації групових подорожей і моніторингу учасників у реальному часі. Користувачі можуть реєструватися, авторизуватися, створювати та редагувати групові подорожі, додавати маршрути і точки маршруту, отримувати інформацію про трекінг місцезнаходження та пульс учасників. Система також забезпечує обмін повідомленнями між учасниками та надсилання сигналів тривоги у разі необхідності. Всі операції реалізовані через RESTful API та доступні через веб-інтерфейс і мобільний застосунок.

3.1.7 Функції продукту

Основні функції системи TravelMate включають:

— Реєстрація та автентифікація користувачів із розмежуванням ролей.

— Управління груповими подорожами: створення, редагування, перегляд.

— Планування та редагування маршрутів з можливістю додавання точок маршруту.

— Відстеження місцезнаходження учасників у режимі реального часу.

— Збір та відображення даних пульсу з підключених пристроїв.

— Система повідомлень між користувачами.

— Надсилання сигналів тривоги з миттєвим оповіщенням.

— Логування дій користувачів для аудиту та контролю.

— Підтримка роботи мобільного застосунку в офлайн режимі з подальшою синхронізацією.

— Адаптивний інтерфейс для різних пристроїв.

3.1.8 Припущення та залежності

Припущення:

— Користувачі мають стабільне інтернет-з’єднання для роботи веб-інтерфейсу та синхронізації мобільного застосунку.

— Користувачі використовують сучасні веб-браузери, які підтримують JavaScript, HTML5 і CSS3.

— Мобільні пристрої користувачів підтримують операційні системи Android 8.0+ та iOS 13+.

— Підключені фітнес-пристрої сумісні з Bluetooth API застосунку.

Залежності:

— Серверна інфраструктура має належні ресурси для обробки запитів і збереження даних.

— Всі компоненти системи взаємодіють через RESTful API.

— Для забезпечення безпеки використовується протокол HTTPS та механізми аутентифікації.

3.2 Властивості програмного продукту

Програмний продукт TravelMate характеризується такими ключовими властивостями:

— Надійність: Забезпечує безперебійну роботу сервісів, стабільне збереження та обробку даних користувачів, а також коректну синхронізацію між веб- та мобільними застосунками;

— Масштабованість: Система розроблена з урахуванням можливості розширення функціоналу та збільшення кількості користувачів без втрати продуктивності;

— Безпека: Використання сучасних стандартів шифрування даних (HTTPS), механізмів аутентифікації та авторизації гарантує захист інформації користувачів;

— Продуктивність: Оптимізовані алгоритми обробки даних та швидкий обмін інформацією між клієнтом і сервером забезпечують високу швидкість реакції системи;

— Зручність використання: Інтуїтивно зрозумілий інтерфейс, адаптивність до різних пристроїв (ПК, планшети, мобільні) та підтримка офлайн-режиму для мобільного застосунку підвищують комфорт користування;

— Сумісність: Підтримка основних операційних систем і браузерів, а також інтеграція з Bluetooth-пристроями для збору даних пульсу;

— Модульність і гнучкість: Архітектура системи дозволяє легко додавати нові функції та інтегрувати сторонні сервіси без значних змін у базовому коді.

3.3 Атрибути програмного продукту

3.3.1 Надійність

Система повинна стабільно функціонувати при тривалому навантаженні, включаючи обробку даних, що надходять з високою періодичністю. Передбачається, що трекінг геопозиції та серцебиття користувача надходитиме з інтервалом до 20 секунд. Збереження таких даних має виконуватися без втрат і затримок, незалежно від кількості активних користувачів.

У разі втрати інтернет-з’єднання на стороні клієнта, відмови каналу передачі даних або збоїв з’єднання з сервером, клієнт повинен тимчасово буферизувати дані з подальшою передачею після відновлення зв’язку. Якщо з’єднання з базою даних тимчасово недоступне, система повинна або повторити запит із затримкою, або виконати автоматичне перемикання на резервний канал доступу, якщо він визначений.

Усі критичні операції, пов’язані із збереженням або оновленням даних, мають виконуватися транзакційно. У випадку помилки транзакція повинна бути скасована повністю. Жодна операція не може залишитися в частково виконаному стані.

Усі помилки і винятки, що виникають під час роботи сервісів, мають реєструватися у центральному журналі системних подій. Логування повинно містити час події, ідентифікатор користувача (якщо застосовно), тип модуля, джерело помилки та статус виконання операції.

У разі відмови зовнішніх служб, таких як кеш-сервер або служба надсилання листів, система повинна зберігати основну функціональність та переходити у деградований режим без повного припинення роботи. Після відновлення відповідних служб має бути реалізовано автоматичне повернення до штатного режиму.

Надійність обробки критичних сценаріїв, таких як збереження трекінгу, виклик допомоги, або приєднання до подорожі, має бути пріоритетною. Ці функції повинні залишатися доступними навіть за умов часткової втрати працездатності окремих компонентів системи.

Основні вимоги до надійності системи:

— система повинна зберігати дані трекінгу без втрат при навантаженні;

— при втраті зв’язку на клієнті дані мають буферизуватись та надсилатись після відновлення з’єднання;

— у разі недоступності бази даних система має повторювати запит або перемикатись на резервний канал;

— критичні операції повинні виконуватись транзакційно, без часткового збереження;

— усі помилки повинні реєструватися в журналі подій із деталізацією контексту;

— при відмові зовнішніх служб система має зберігати основну функціональність у деградованому режимі;

— критичні функції повинні бути доступні за будь-яких умов.

3.3.2 Доступність

Система повинна бути доступною для авторизованих користувачів протягом більшої частини календарної доби з урахуванням можливого технічного обслуговування. Компоненти системи мають бути готові до обробки запитів у багатокористувацькому режимі без зниження швидкості реакції інтерфейсів або порушення логіки роботи.

Передбачається, що обробка запитів від клієнтів має відбуватись без суттєвих затримок, включаючи обробку трекінгових даних, надсилання повідомлень, перегляд маршрутів і зміну станів участі. Усі основні функції мають бути доступні з мобільних пристроїв, а також через веб-інтерфейс без обмежень щодо типу операційної системи чи браузера.

У разі перевищення навантаження система повинна підтримувати обмеження частоти запитів або інші механізми балансування навантаження для забезпечення рівного доступу всім активним користувачам. Якщо певна функціональність тимчасово недоступна, користувач має отримати відповідне повідомлення з можливістю повторити дію після відновлення доступу.

Система повинна також підтримувати повторні підключення без втрати сесії користувача та мати стабільну реакцію на несподівані мережеві перебої. Аутентифікація та авторизація мають бути доступні з будь-якої локації, якщо підключення відповідає вимогам безпеки.

Основні вимоги до доступності системи:

— система повинна бути доступною для користувачів у режимі 24/7, за винятком періоду технічного обслуговування;

— основний функціонал має оброблятися без затримок у багатокористувацькому режимі;

— трекінг, повідомлення, перегляд маршрутів і участь мають залишатися доступними у пікові періоди;

— у разі перевантаження має діяти механізм контролю частоти запитів;

— при тимчасовій недоступності функції користувач повинен отримувати інформативне повідомлення;

— підтримується повторне підключення без втрати авторизації або сесії;

— мобільна та веб-версії повинні мати повний функціональний доступ незалежно від пристрою.

3.3.3 Безпека

Система повинна гарантувати конфіденційність, цілісність і доступність персональних та трекінгових даних користувачів. Усі з’єднання між клієнтом і сервером мають бути захищені протоколом передачі даних з шифруванням, що унеможливлює перехоплення або модифікацію інформації під час транспортування.

Механізм автентифікації повинен ґрунтуватися на токенах доступу. Кожен запит до захищених ресурсів має супроводжуватись перевіркою достовірності та ролі користувача. Права доступу до функцій системи повинні розмежовуватись залежно від ролі (звичайний користувач, адміністратор) та контексту дії (учасник подорожі або її організатор).

Чутливі дані, зокрема координати переміщення та фізіологічні показники, повинні зберігатися у зашифрованому вигляді із застосуванням стійкого криптографічного алгоритму. Дані для автентифікації (логіни, паролі) мають зберігатися у вигляді хешів із використанням криптографічного сольового алгоритму.

Усі дії користувачів та адміністраторів, пов’язані зі зміною даних, надсиланням повідомлень, викликом допомоги, підтвердженням участі або блокуванням, повинні фіксуватися у журналі активності. Записи мають включати тип дії, ідентифікатор користувача, час події та результат виконання. Повинна бути забезпечена захищеність від типових атак на рівні веб-інтерфейсу (XSS, CSRF, SQL injection).

Основні вимоги до безпеки системи:

— усі з’єднання мають бути захищені з використанням HTTPS;

— автентифікація користувачів повинна виконуватись на основі токенів доступу з перевіркою ролей;

— персональні та трекінгові дані мають зберігатися в зашифрованому вигляді;

— облікові дані повинні зберігатися у вигляді хешів із сольовим захистом;

— повинна бути реалізована перевірка прав доступу до функціоналу;

— усі критичні дії мають реєструватися у журналі подій;

— система повинна бути захищена від основних типів атак на рівні API та веб-інтерфейсу.

3.3.4 Супроводжуваність

Архітектура програмного забезпечення повинна забезпечувати можливість легкого оновлення, модифікації, тестування та усунення помилок без впливу на стабільність всієї системи. Усі компоненти мають бути реалізовані як незалежні модулі з чітко визначеними інтерфейсами взаємодії. Логіка програми повинна бути відокремлена від інтерфейсів користувача та від доступу до бази даних.

У серверній частині має використовуватись розподіл за логічними шарами: контролери, сервіси, сховище даних. У клієнтських додатках повинна дотримуватись компонентна структура з чітким поділом станів, подій та візуального представлення. Усі конфігураційні параметри мають бути винесені у зовнішні конфігураційні файли або змінні середовища.

Код програми повинен бути читабельним і зрозумілим для сторонніх розробників. Усі публічні методи, сервіси, компоненти мають супроводжуватись технічною документацією або коментарями. Під час

внесення змін у функціонал має бути забезпечена можливість перевірки за допомогою модульних та інтеграційних тестів.

Оновлення окремих частин системи повинно бути можливим без повного перезапуску всіх компонентів. Для цього має бути забезпечена мінімальна залежність між модулями та підтримка стандартів взаємодії через API.

Основні вимоги до супроводжуваності системи:

— код має бути структурованим, модульним і зрозумілим для підтримки;

— архітектура повинна забезпечувати відокремлення логіки, інтерфейсу та даних;

— усі параметри конфігурації мають зберігатися окремо від основного коду;

— повинна бути підтримка розгортання оновлень без зупинки всієї системи;

— система повинна бути покрита модульними та інтеграційними тестами;

— документація до публічних компонентів має бути наявною і зрозумілою;

— усі частини системи мають бути незалежними та взаємодіяти через чітко визначені інтерфейси.

3.3.5 Переносимість

Програмне забезпечення повинно бути розроблене з урахуванням кросплатформенності всіх компонентів. Серверна частина має підтримувати розгортання в різних середовищах, включаючи хмарну інфраструктуру, віртуальні машини та локальні сервери під управлінням операційних систем

Windows. Для забезпечення незалежності від платформи передбачається використання контейнеризації на основі Docker.

Клієнтські додатки повинні бути сумісні з популярними платформами мобільних пристроїв і веб-браузерів. Мобільна частина має забезпечувати повноцінне функціонування на операційних системах Android та iOS. Веб-інтерфейс повинен коректно відображатися та працювати в актуальних версіях основних браузерів, незалежно від операційної системи користувача.

Передбачено можливість перенесення серверного коду на інші хости без потреби змін у програмній логіці або структурі збирання. Усі залежності системи мають бути описані явно та підтримувати автоматизоване встановлення через стандартні засоби управління пакетами. Дані користувачів, що зберігаються у базі, повинні бути експортовані у форматах, придатних для перенесення між середовищами.

Основні вимоги до переносимості системи:

— серверна частина повинна працювати у Windows-середовищах;

— мобільний застосунок має підтримувати Android та iOS без втрати функціональності;

— веб-інтерфейс повинен працювати в основних браузерах незалежно від платформи;

— усі сервіси мають підтримувати контейнеризацію з використанням Docker;

— перенесення системи на інші хостинг-середовища не повинно потребувати зміни коду;

— усі залежності повинні бути визначені у стандартних конфігураційних файлах;

— дані користувачів повинні мати формат для безпечного експорту та імпорту.

3.3.6 Продуктивність

Система повинна забезпечувати стабільну продуктивність у багатокористувацькому середовищі з одночасним обслуговуванням великої кількості активних сеансів. Усі компоненти мають реагувати на запити користувачів у межах прийнятного часу, незалежно від інтенсивності навантаження. Особливу увагу слід приділити модулям, що обробляють дані в реальному часі, зокрема трекінг координат і пульсу.

Інтервал обробки трекінгових даних не повинен перевищувати визначеного порогу, щоб гарантувати актуальність інформації. Серверна частина має ефективно масштабуватись при зростанні навантаження. Повинні застосовуватись механізми кешування для оптимізації доступу до часто запитуваних даних, включаючи останні трекінгові точки та результати кластеризації маршрутів.

Запити до бази даних повинні бути оптимізованими, з використанням індексів на критичних полях. Система має обробляти об’єми трекінгових записів, що надходять із частотою до декількох записів щосекунди від багатьох користувачів, без зниження швидкості реакції інтерфейсу чи API.

Клієнтські інтерфейси повинні завантажуватись і реагувати швидко, навіть при великій кількості доступних подорожей, повідомлень або записів статистики. Завантаження сторінок або мобільних екранів не повинно перевищувати визначеного граничного часу.

Основні вимоги до продуктивності системи:

— система повинна підтримувати одночасну роботу багатьох користувачів без погіршення продуктивності;

— час відповіді API не повинен перевищувати 1 секунди при стандартному навантаженні;

— система має обробляти трекінгові дані з частотою до 1 запису кожні 20 секунд на користувача без затримок;

— для часто запитуваних даних має бути реалізоване кешування;

— запити до бази даних повинні бути оптимізованими за рахунок індексування;

— веб- і мобільні клієнти повинні забезпечувати швидке завантаження інтерфейсу при великому обсязі даних.

3.4 Вимоги бази даних

У системі повинна бути використана реляційна база даних, що забезпечує підтримку транзакцій, референтної цілісності, масштабованості та високої доступності. Структура даних має бути повністю нормалізованою, з розмежуванням основних сутностей, таких як користувачі, подорожі, участь, повідомлення, записи трекінгу, екстрені виклики тощо. Між сутностями повинні бути чітко визначені зв’язки, з підтримкою зовнішніх ключів та обмежень цілісності.

Доступ до бази даних повинен здійснюватися через проміжний шар логіки, з використанням об’єктно-реляційного мапінгу. Прямий доступ до таблиць з боку клієнтського коду має бути заборонений.

З метою забезпечення продуктивності система повинна підтримувати індексацію полів, які найчастіше використовуються для пошуку, сортування та фільтрації. Для зберігання великих обсягів трекінгових даних бажано реалізувати партиціювання за часовими або подієвими ознаками.

Збереження чутливих даних, таких як геолокаційна інформація або біометричні показники, повинно здійснюватися у зашифрованому вигляді з

використанням сучасного симетричного алгоритму. Усі запити до бази даних повинні логуватися з метою забезпечення можливості аудиту.

Архітектура бази має передбачати можливість масштабування та подальшого розширення функціоналу без порушення цілісності або втрати даних. У разі потреби повинна бути можливість архівування історичних записів у додаткові таблиці або окреме сховище.

3.5 Інші вимоги

Програмне забезпечення має підтримувати функціонування у кросплатформеному середовищі та складатися з трьох взаємопов’язаних частин: серверної, мобільної та веб-клієнтської. Архітектура системи повинна бути розподіленою та підтримувати модульність, із чітким поділом відповідальності між компонентами.

Серверна частина повинна бути реалізована як набір REST-сервісів, що працюють на платформі з відкритим вихідним кодом із підтримкою контейнеризації через Docker. Передбачається можливість розгортання у віртуальному середовищі або хмарній інфраструктурі з використанням технологій автоматизованого керування контейнерами. Сервер повинен забезпечувати збереження, обробку та надання даних у форматі, сумісному з мобільними та веб-клієнтами. Для кешування та підвищення продуктивності доцільно використовувати систему оперативного зберігання даних з низькою латентністю.

Мобільний додаток має забезпечувати прийом даних зі смарт-годинника користувача, включаючи координати місцезнаходження та частоту серцебиття. Дані мають передаватися на сервер із заданою періодичністю в реальному часі або з мінімальною затримкою. Для забезпечення захищеного обміну

інформацією передбачається використання HTTPS-протоколу, а також авторизація через токени доступу.

Веб-клієнт повинен забезпечувати зручний інтерфейс для доступу до функціоналу системи: перегляд та керування подорожами, участь у групах, спілкування, перегляд статистики трекінгу та маршрутів. Інтерфейс має бути реалізований із застосуванням сучасних веб-технологій з адаптивною верстkою для підтримки різних типів пристроїв.

Кожен із компонентів повинен підтримувати інтеграцію з єдиною централізованою базою даних і мати уніфікований механізм автентифікації. Комунікація між частинами повинна здійснюватися у стандартизованому форматі, з використанням JSON як платформонезалежного способу обміну даними.

Усі складові системи мають підтримувати логування, моніторинг, автоматизоване розгортання та бути підготовленими до горизонтального масштабування. Система повинна залишатися стабільною при підключенні великої кількості одночасних користувачів та надходженні інтенсивного потоку трекінгових даних.

ДОДАТОК Ґ

Фрагменти коду сервісу «TrackingService»

```
using System.Globalization;
using BL.Abstractions.Tracking;
using Domain.DataTypes;
using Domain.DataTypes.Enums;
using Domain.Entities.DTOs.Tracking;
using Domain.Entities.Models.ParticipationModels;
using Domain.Entities.Models.TrackingModels;
using Infrastructure.Abstractions;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace BL.Services.Tracking;

public class TrackingService(
    IGenericRepository<Participation> participationRepo,
    IGenericRepository<HealthAndGpsTracking> trackingRepo,
    IGenericRepository<EmergencyCall> emergencyCallRepo,
    IDistanceCalculatorService haversineDistanceCalculator,
    IRedisCacheService redisCacheService,
    IPulseSmoothingBuffer buffer,
    IAesEncryptionService aesEncryptionService,
    IEmergencyCallService emergencyCallService,
    IConfiguration configuration,
```

```
  25      ILogger<TrackingService> logger)
  26      : ITrackingService
  27 {
  28      private readonly string? _aesKey =
configuration["Encryption:AesKey"];
  29      private DateTime _lastSosSent = DateTime.MinValue;
  30
  31      public async Task<Result<bool>> SaveTrackingDataAsync(Guid
userId, TrackingDataCreateDto dto)
  32      {
  33          var participation = await
GetValidatedParticipation(userId, dto.TravelGroupId);
  34          if (participation == null)
  35              return Result<bool>.Failure("Користувач не має права
надсилати трекінг для цієї подорожі");
  36
  37          var smoothed = buffer.AddAndSmooth(userId,
dto.HeartRate);
  38
  39          var encryptedLat =
aesEncryptionService.Encrypt(dto.Latitude.ToString(CultureInfo.Invaria
ntCulture), _aesKey);
  40          var encryptedLng =
aesEncryptionService.Encrypt(dto.Longitude.ToString(CultureInfo.Invari
antCulture), _aesKey);
  41          var encryptedRaw =
aesEncryptionService.Encrypt(dto.HeartRate.ToString(), _aesKey);
  42          var encryptedSmoothed =
aesEncryptionService.Encrypt(smoothed.ToString(), _aesKey);
  43
```

```
44          var lastEntry = await
GetLastTrackingPoint(participation.Id, dto.Timestamp);
45
46          double? distanceKm = null;
47          if (lastEntry != null)
48          {
49              distanceKm =
haversineDistanceCalculator.CalculateDistanceKm(
50                  lastEntry.Latitude, lastEntry.Longitude,
51                  dto.Latitude, dto.Longitude);
52
53              participation.TotalDistanceKm += distanceKm.Value;
54              var updateResult = await
participationRepo.UpdateAsync(participation);
55              if (!updateResult.IsSuccess)
56                  return Result<bool>.Failure("Не вдалося оновити
інформацію про участь");
57          }
58
59          var newEntry = CreateTrackingEntry(participation.Id,
encryptedLat, encryptedLng, encryptedRaw, encryptedSmoothed,
dto.Timestamp, distanceKm);
60
61          var cacheResult = await CacheLastPoint(participation.Id,
dto, smoothed, newEntry.Timestamp);
62          if (!cacheResult.IsSuccess)
63              return
Result<bool>.Failure(cacheResult.ErrorMessage);
64
```

```
  65          await CheckAndSendSosIfNeeded(userId, dto.TravelGroupId,
participation.Id, dto.Latitude, dto.Longitude);
  66
  67          var addResult = await trackingRepo.AddAsync(newEntry);
  68          return addResult.IsSuccess
  69              ? Result<bool>.Success(true)
  70              : Result<bool>.Failure("Не вдалося зберегти дані
трекінгу");
  71      }
  72
  73      private async Task<Result<bool>> CacheLastPoint(Guid
participationId, TrackingDataCreateDto dto, int smoothed, DateTime
timestamp)
  74      {
  75          var pointsResult = await
redisCacheService.GetLastPointsAsync(participationId, 5);
  76          if (!pointsResult.IsSuccess)
  77              return
Result<bool>.Failure(pointsResult.ErrorMessage);
  78
  79          var points = pointsResult.Data;
  80          if (points.Count == 5)
  81              points.RemoveAt(0);
  82
  83          points.Add(new LastTrackingPointDto
  84          {
  85              Latitude = dto.Latitude,
  86              Longitude = dto.Longitude,
  87              SmoothedHeartRate = smoothed,
```

```
  88              Timestamp = timestamp
  89          });
  90
  91          var setResult = await
redisCacheService.SetLastPointsAsync(participationId, points);
  92          return setResult.IsSuccess
  93              ? Result<bool>.Success(true)
  94              : Result<bool>.Failure("Не вдалося зберегти дані в
кеш");
  95      }
  96
  97      private async Task CheckAndSendSosIfNeeded(Guid userId, Guid
travelGroupId, Guid participationId, double latitude, double
longitude)
  98      {
  99          const int criticalPulseLow = 35;
 100          const int criticalPulseHigh = 190;
 101          const int criticalConsecutiveCount = 5;
 102          var sosCooldown = TimeSpan.FromMinutes(5);
 103
 104          var pointsResult = await
redisCacheService.GetLastPointsAsync(participationId,
criticalConsecutiveCount);
 105          if (!pointsResult.IsSuccess || pointsResult.Data == null
|| pointsResult.Data.Count < criticalConsecutiveCount)
 106          {
 107              logger.LogInformation("[SOS] Not enough points for
ParticipationId {ParticipationId}", participationId);
 108              return;
```

```
 109         }
 110
 111         var isCritical = pointsResult.Data.All(p =>
 112             p.SmoothedHeartRate < criticalPulseLow ||
p.SmoothedHeartRate > criticalPulseHigh);
 113
 114         if (!isCritical)
 115         {
 116             logger.LogInformation("[SOS] Pulse values not
critical for ParticipationId {ParticipationId}", participationId);
 117             return;
 118         }
 119
 120         if ((DateTime.UtcNow - _lastSosSent) < sosCooldown)
 121         {
 122             logger.LogInformation("[SOS] Cooldown not passed for
ParticipationId {ParticipationId}", participationId);
 123             return;
 124         }
 125
 126         var existing = await
emergencyCallRepo.GetByConditionAsync(e =>
 127             e.ParticipationId == participationId && e.Status ==
EmergencyCallStatus.Active);
 128
 129         if (existing.IsSuccess && existing.Data.Any())
 130         {
 131             logger.LogWarning("[SOS] Already active SOS exists
for ParticipationId {ParticipationId}", participationId);
```

```
132             return;
133         }
134
135         _lastSosSent = DateTime.UtcNow;
136         const int maxAttempts = 2;
137         int attempt = 0;
138         Result<bool>? sosResult = null;
139
140         while (attempt < maxAttempts)
141         {
142             try
143             {
144                 sosResult = await
emergencyCallService.CreateEmergencyCallAsync(
145                     userId,
146                     new EmergencyCallCreateDto
147                     {
148                         TravelGroupId = travelGroupId,
149                         Latitude = latitude,
150                         Longitude = longitude,
151                         EmergencyType = "AutomaticPulseSOS"
152                     });
153
154                 if (sosResult.IsSuccess)
155                 {
156                     logger.LogInformation("[SOS] Successfully
created for ParticipationId {ParticipationId}", participationId);
```

```
157                     break;
158                 }
159
160                 logger.LogWarning("[SOS] Attempt {Attempt} failed: {Error}", attempt + 1, sosResult.ErrorMessage);
161             }
162             catch (Exception ex)
163             {
164                 logger.LogError(ex, "[SOS] Exception on attempt {Attempt} for ParticipationId {ParticipationId}", attempt + 1, participationId);
165             }
166
167             attempt++;
168             await Task.Delay(1000);
169         }
170
171         if (sosResult == null || !sosResult.IsSuccess)
172         {
173             logger.LogError("[SOS] Failed after {MaxAttempts} attempts for ParticipationId {ParticipationId}", maxAttempts, participationId);
174         }
175     }
176
177     private async Task<Participation?> GetValidatedParticipation(Guid userId, Guid groupId)
178     {
179         var query = participationRepo.Query()
```

```
 180             .Where(p => p.UserId == userId && p.TravelGroupId ==
groupId && p.Status == ParticipationStatus.Accepted)
 181             .Include(p => p.TravelGroup);
 182
 183         var participation = await query.FirstOrDefaultAsync();
 184         if (participation == null || !participation.HasJoined ||
participation.LeftAt != null || participation.TravelGroup.Status !=
TravelGroupStatus.Active)
 185             return null;
 186
 187         return participation;
 188     }
```