Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Автоматизована система моніторингу веб-середовища в IT компанії»

Студента 2 курсу, 2з групи,
спеціальності 121 «Інженерія
програмного забезпечення»
спеціалізації «Інженерія
програмного забезпечення» Демченко Вадима
Андрійовича

_____
підпис студента

Науковий керівник
кандидат технічних наук,
доцент кафедри інженерії
програмного забезпечення та
кібербезпеки Савченко Тетяна
Віталіївна

_____
підпис керівника

Гарант освітньої програми
доктор економічних наук,
професор кафедри інженерії
програмного забезпечення та
кібербезпеки Токар Володимир
Володимирович

_____
підпис гаранта

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет *інформаційних технологій*
Кафедра *інженерії програмного забезпечення та кібербезпеки*
Освітній ступінь *магістр*
Спеціальність *121 «Інженерія програмного забезпечення»*
Спеціалізація / освітня програма *121 «Інженерія програмного забезпечення»*

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
"29" грудня 2020 р.

Завдання
на випускний кваліфікаційний проєкт студентові

Демченко Вадиму Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту: *«Автоматизована система моніторингу веб-середовища в IT компанії»*

Затверджена наказом ректора від "28" грудня 2020 р. № 3923

2. Строк здачі студентом закінченої проєкту 25 листопада 2021 р.

3. Цільова установка та вихідні дані до проєкту

***Мета проекту*:** *створення автоматизованої системи моніторингу для IT компанії, застосування мов програмування, середовищ розробки для веб-середовищ та їх автоматизації*

Об'єкт дослідження: *автоматизований моніторинг*

Предмет дослідження: *розробка автоматизованої системи моніторингу веб-середовища в IT компанії з використанням сучасних технологій*

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальні положення та аналіз предметної області

1.2. Аналіз існуючих програмних рішень

1.3. Актуальність розробки моніторингової системи

1.4. Вимоги до безпеки системи

1.5. Висновки до розділу

РОЗДІЛ 2 ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

2.2. Вибір фреймворку для написання серверної частини

2.3. Вибір бази даних

2.4. Вибір засобу для відображення графіків

2.5. Висновки до розділу

РОЗДІЛ 3 РОЗРОБЛЕННЯ ТА АНАЛІЗ МОНІТОРИНГОВОЇ СИСТЕМИ

3.1. Загальний опис архітектури системи

3.2. Опис серверної частини

3.3. Опис клієнтської частини

3.4. Опис структур даних для зберігання і підрахунку статистики

3.5. Опис структури бази даних

3.6. Аналіз реалізованої моніторингової системи

3.7. Дизайн та вміст веб-сторінок

3.8. Тестування системи

3.9. Рекомендації щодо вдосконалення

3.10. Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2020*	*21.09.2020*
2.	*Розробка та затвердження завдання на проєкт магістра (заоч)*	*29.12.2020*	*22.12.2020*
3.	*Вступ та перелік літературних джерел*	*27.02.2021*	*27.02.2021*
4.	*Розробка технічного завдання*	*20.03.2021*	*20.03.2021*
5.	*Розділ 1. «Аналіз існуючих рішень та обгрунтування теми дипломного проєкту»*	*16.04.2021*	*16.04.2021*
6.	*Розділ 2. «Обгрунтування вибору засобів реалізації»*	*24.05.2021*	*24.05.2021*
7.	*Розділ 3. «Розроблення та аналіз моніторингової системи»*	*21.06.2021*	*21.06.2021*
8.	*Розділ 4.* (*Об'єднання розділів 3 та 4*)	*20.09.2021*	*20.09.2021*
9.	*Розробка програми та методики тестування*	*18.10.2021*	*18.10.2021*
10.	*Написання наукової статті*	*22.05.2021*	*22.05.2021*
11.	*Керівництво користувача*	*21.10.2021*	*21.10.2021*
12.	*Висновки та пропозиції*	*01.11.2021*	*01.11.2021*
13.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
14.	*Підготовка автореферату та презентації доповіді*	*03.11.2021*	*03.11.2021*
15.	*Попередній захист випускного кваліфікаційного проєкту*	*22.11.2021 – 25.11.2021*	*22.11.2021*
16.	*Здача зброшурованої випускного кваліфікаційного проєкту*	*25.11.2021*	*25.11.2021*
17.	*Зовнішнє рецензуваннявипускного кваліфікаційного проєкту*	*26.11.2021*	*26.11.2021*
18.	*Підготовка до публічного захисту випускного кваліфікаційного проєкту*		

7. Дата видачі завдання «29» грудня 2020 р.

8. Науковий керівник твипускного кваліфікаційного проєкту Савченко Т.В.

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Демченко В.А.

(прізвище, ініціали, підпис)

11. Відгук керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційного проєкту_____

(підпис, дата)

Відмітка про попередній захист

_____

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускний кваліфікаційний проєкт студента _____*Демченка В.А.*_____

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми_____*Токар В.В.*_____

(прізвище, ініціали, підпис)

Завідувач кафедри _____*Криворучко О. В.*_____

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Проведено автоматизацію системи моніторингу веб-середовища за рахунок впровадження функцій для ефективного моніторингу. Розкрито поняття системи моніторингу та визначено концепцію з використанням інформаційних технологій для аналізу та збору даних з веб-середовища. Обґрунтована важливість системи моніторингу та ефективність інструментів для перетворення та автоматизації веб-даних, необхідних для зручного та вигідного варіанту для аналізу та обробки даних. Автоматизоване вилучення даних може отримати необхідні дані навіть із джерел, які не мають структури. За допомогою цього можна завантажувати файли та заповнювати форми, якщо потрібно. Основою технічного обслуговування є забезпечення якості даних - процес тестування укарбованого вмісту на якість кожного разу, коли система завантажує його з цільових веб-сайтів.

Ключові слова: Система моніторингу, веб-середовище, ефективність, автоматизація, моніторинг, веб-данні.

ABSTRACT

An automated web environment monitoring system has been implemented to calculate the implementation of functions for effective monitoring. The concept of monitoring system is revealed and the concept of using information technologies for analysis and data collection from the web environment is defined. An important system of monitoring and efficiency tools for converting and automating web data, necessary for convenient and profitable options for data analysis and processing. Automatic data retrieval can obtain the necessary information even from sources that do not have a structure. This allows you to upload files and fill out forms if required. The basis of maintenance is data quality assurance - the process of testing scraped content for quality every time the system downloads it from the target websites.

Keywords: monitoring system, web environment, efficiency, automation, monitoring, web data.

ЗМІСТ

ВСТУП

З кожнем днем кількість веб-середовищ росте і проблема їх моніторингу стає нагальною. Виникає досить значне навантаження на адміністраторів веб-середовищ, які не встигають контролювати їх роботу. Через це виникають збої в роботі середовища, що може призвести до їх недоступності. Однією з головних проблем, яка виникає при роботі з веб-середовищем є їх неправильне адміністрування і не своєчасне прийняття рішень, щодо управління середовищем. Неправильне адміністрування середовища може призвести до перебоїв в його роботі, що може зменшити зацікавленість і кількість користувачів. Щоб запобігти цим проблемам необхідно використовувати моніторингову систему, яка буде інформувати адміністраторів при виникненні проблем з веб-середовищем. Для кращого аналізу система повина забезпечувати обрахунок статистичних показників, таких як:

- мінімальне/максимальне значення для метрики;
- середнього значення;
- відхилення значень.

Ефективна моніторингова система є важливим елементом для хорошого управління середовищем. Вона підтримує поінформованість та своєчасність прийняття рішень системними адміністраторами, керівниками проєктів та іншими зацікавленими сторонами. Така система є ключовою частиною процесу управління проєктом.

Існуючі аналоги моніторингових систем не є досконалими і не мають системи сповіщень для популярних месенджерів і підрахунку статистичних показників по користувацьким метрикам.

Основною метою даного дипломного проєкту є створення автоматизованої моніторингової системи для IT компанії, яка буде автоматично відображати статистику по показникам в реальному часі і виявляти збої в веб-середовищі.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальні положення та аналіз предметної області

З розвитком технологій і збільшенням кількості веб-середовищ виникла проблема їх моніторингу для подальшої підтримки і розвитку. Моніторинг широко використовується у сфері веб-технологій для оцінки продуктивності і збору інформації про використання веб-додатків, що дозволяє слідкувати за діяльністю та визначати ефективність додатку протягом всього циклу його життя.

Моніторингова система надає можливість збирати і обробляти різні метрики (системні і користувацькі) для їх аналізу, що дозволить прийняти рішення стосовно подальшого управління додатком. Споживачами програмної моніторингової системи є адміністратори, які відповідають за підтримку і правильне функціонування ресурсу. На основі зібраних данних вони формують вимоги до додатку над яким ведеться спостереження, а також встановлюють причини, які перешкоджають нормальному функціонуванню веб-ресурсу.

Найпоширеніші причини проблемності роботи веб-додатків для вирішення, яких застосовують моніторингові системи:

- неефективно написаний код. Неефектно написаний написаний код може призвести до безлічі проблем із веб-додатками, включаючи неефективні алгоритми, витоки пам’яті та недоліки програми. Старі версії програмного забезпечення або застарілі інтегровані системи також можуть знижувати продуктивність;

- неоптимізовані бази даних. Оптимізована база даних забезпечує найвищий рівень безпеки та продуктивності. Відсутність індексів сповільнює продуктивність запитів до бази даних, що може сповільнити роботу веб-додатку загалом;
- обробка значної кількості даних. Веб-додатки мають тенденцію до збільшення кількості даних, які потрібно обробляти та надавати користувачам. Розробка плану управління та моніторингу такими даними (з урахуванням їх збільшення) необхідна для коректної роботи будь-якого веб-додатку;
- різке збільшення кількості користувачів. Збільшення користувачів, як правило, вважається показником популярності веб-додатку. Однак потрібно правильно розраховувати ресурси серверної частини, щоб вона могла витримати необхідне навантаження. Планування ресурсів заздалегідь є ключовим моментом для роботи веб-додатку. Таким чином, можна запобігти проблемі, перш ніж кількість користувачів додатку стане критичною;
- неправильний розподіл навантаження на серверах. Неправильний розподіл навантаження може зменшити час відповіді на запити до веб-додатку, неправильно розподіливши використання серверів для нових користувачів веб-додатку. Для запобігання цієї проблеми потрібно налаштувати балансир, який буде розподіляти використання серверів на яких знаходиться вебдодаток;
- мережеве підключення та доступність веб-додатку. Відсутність мережевого підключення на сервері не дозволить відвідувачам отримувати доступ до додатку та призведе до різних помилок. Крім того, підключення до мережі та ефективність брандмауера мають вирішальне значення для доступу та продуктивності. Потрібно

завжди перевіряти доступність додатку і в разі якихось помилок сигналізувати про них.

Щоб відсліткувати і діагностувати проблеми моніторингова система повина відслітковувати такі основні метрики:

- кількість активних користувачів системи;
- використання системних ресурсів (CPU, RAM, ROM);
- час відклику веб-середовища.

1.2. Аналіз існуючих програмних рішень

Моніторингова система SmartBear

SmartBear – моніторингова система, яка допомагає відслідковувати основні показника вашого веб-додатку, такі як:

- час завантаження DOM (час який потрібний для розбору документа сторінки);
- час на завантаження самої сторінки (завантаження всіх скриптів, стилів, картинок);
- швидкість виконання запитів до API (зберігання, редагування чи видалення даних);
- валідація даних, які повертає API (перевірка даних);
- перевірка на доступність веб-додатку.

На рис. 1.1 зображено вигляд користувацького інтерфейсу моніторингової системи SmartBear.

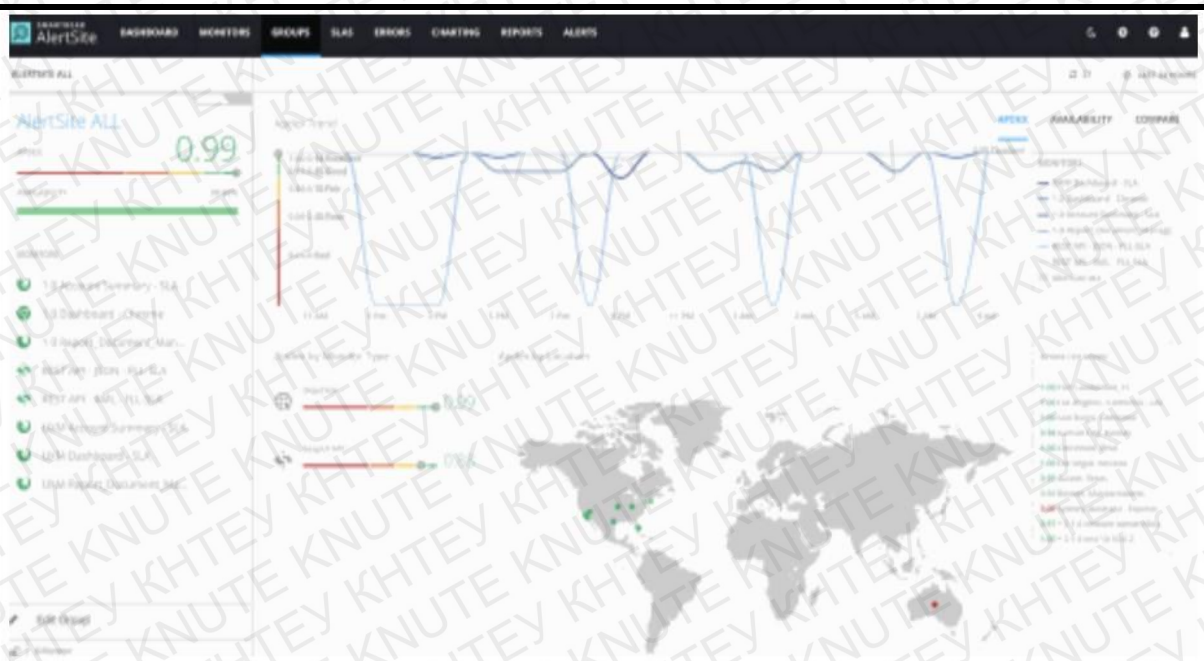

Рис. 1.1. Приклад інтерфейсу моніторингової системи SmartBear

Переваги ПЗ:

- зручний інтерфейс користувача;
- проста в налаштуванні.
- наявність функціоналу для аналізу даних;

Недоліки ПЗ:

- невеликий список даних, які можна відслідковувати;
- немає повної інтеграції з веб-середовищем;
- відсутня система інформування;
- не можливо відслідкувати системні метрики;
- відсутні статистичні показники.

Моніторингова система Dotcom-monitor

Dotcom-monitor – моніторингова система, яка надає можливість відслідковувати роботу веб-додатку. Основною задачею цієї системи є

перевірка доступності додатку. На рис. 1.2. зображено вигляд користувацького інтерфейсу моніторингової системи Dotcom-monitor.

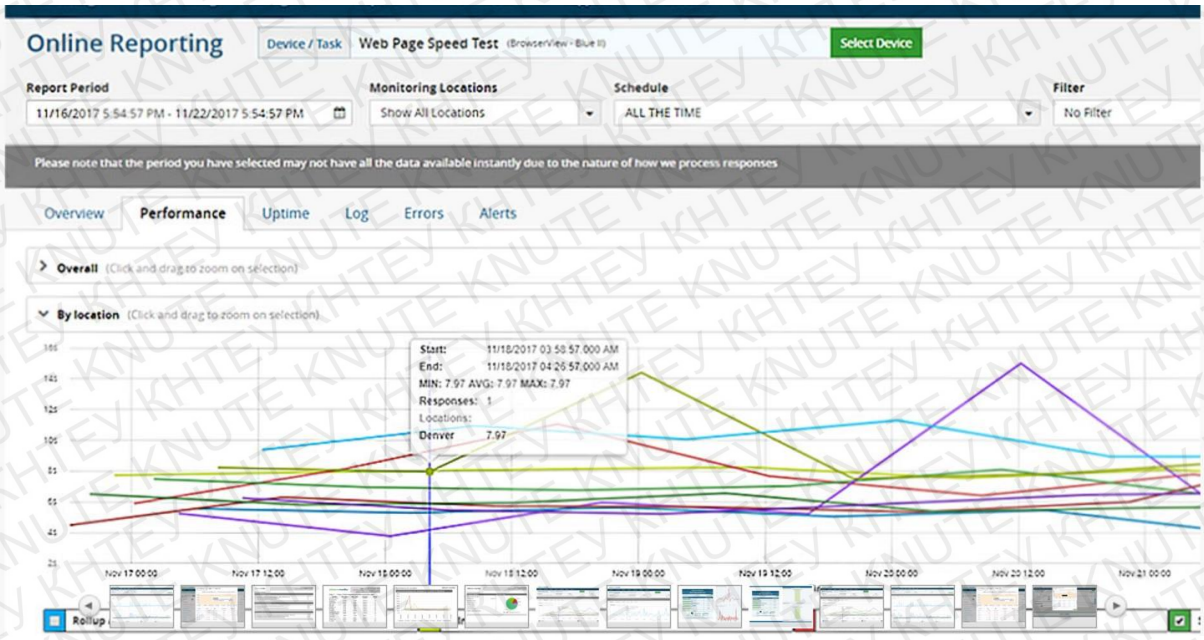

Рис. 1.2. Приклад інтерфейсу моніторингової системи Dotcom-monitor

Переваги ПЗ**:**

- перевіряє доступність всіх ресурсів веб-сайту;
- існує система сповіщень, яка сигналізує про помилки;
- дані відображаються на графіках;
- має можливість інтеграції через API;
- зберігається відео при перевірці доступності ресурсу.

Недоліки ПЗ:

- незручний інтерфейс користувача;
- відсутні підрахунок статистичних показників;
- велика плата за використання системи.

Моніторигова система vRealize Hyperic

Моніторингова система VRealize Hyperic контролює інфраструктуру, операційну систему та різні ПО з єдиною зручною панеллю моніторингу. Дана система сумісна з більш ніж 70 типами програм, вона може відстежувати до 50 000 метрик, в той же час, має систему споавіщень. Адміністратори можуть настроювати та персоналізувати параметри оповіщень, гарантуючи, що всі вони будуть відправлені. На рис. 1.3. зображено вигляд користувацького інтерфейсу моніторингової системи VRealize Hyperic.

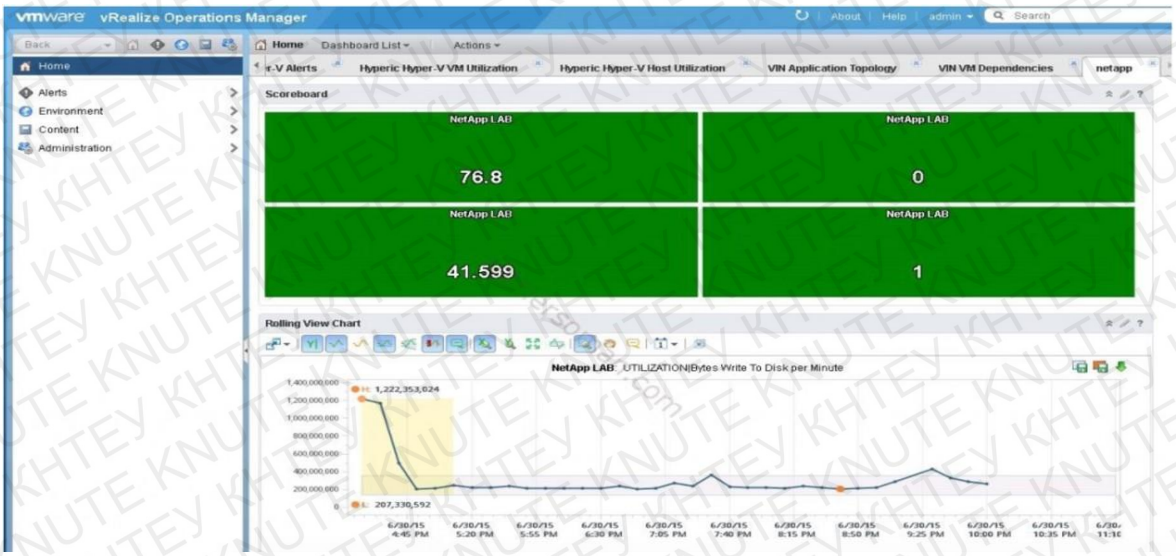

Рис. 1.3. Приклад інтерфейсу моніторингової системи VRealize Hyperic

Дана моніторингова система підходить для адмінастраторів і розробників веб-додатків, оскільки її API дозволяє користувачам створювати і підключати модулі до своїх веб-додатків.

Переваги ПЗ:

- перевіряє доступність всіх ресурсів веб-сайту;
- існує система сповіщень, яка сигналізує про помилки;
- має можливість інтеграції через API;
- можна відслідковувати велику кількість метрик.

Недоліки ПЗ:

- незручний інтерфейс користувача;
- не має можливості персоналізувати відображення графіків;
- відсутні підрахунок статистичних показників.

Аналіз характеристика існуючих рішень зведено до порівняльної таблиці 1.1.

Таблиця 1.1.

Порівняльна характеристика існуючих рішень

Основні функції / Система	Підрахунок статистичних даних	Перевірка доступності веб-додатку	Сповіщення користувачів	Візуалізація даних	Аналіз та обробка моніторингових даних
SmartBear	–	+	–	+	+
Dotcom-monitor	–	+	+	+	+
vRealize Hyperic	–	+	+	+	–

В результаті аналізу існуючих рішень було виявлено те, що вони не надають весь необхідні функціональні можливості, які необхідно користувачеві для детального аналізу веб-додатків. Зокрема кожна з них має один чи більше наступних недоліків:

- відсутній підрахунок статистичних даних;
- збір системних метрик;

- система сповіщень.

1.3. Актуальність розробки моніторингової системи

Розроблювана моніторингова система надає користувачам весь необхідний функціонал для відслідковування процесу роботи веб-середовища. Окрім того користувачам не потрібно буде запускати моніторингову систему в себе на комп'ютері чи шукати сервер на якому її запустити, для кожного користувача буде створюватись окремий екземпляр програми до якого буде мати доступ тільки він. Таке рішення спростить користування системою і надасть безперебійний доступ до неї. Також розроблювальна моніторингова система має систему інформування, яка буде сигналізувати про системні збої.

Розроблюване програмне забезпечення повине відповідати таким вимогам:

- забезпечення обрахунку та відображення необхідних статистичних даних:
  - мінімальне/максимальне значення;
  - середнього значення;
  - середнє квадратичне відхилення.
- збереження, фільтрування і сортування даних за часом;
- інформація про некоректну роботу:
  - обробка і аналіз метрик, для визначення системних збоїв і перевищення заданих значень;
  - відправка сповіщень про збої в мессенджери.
- система має відображати показники використання системних ресурсів фізичної машини на якій запущено веб-додаток;
- система має надавати можливість переглядати графіки за певний проміжок часу;

- захист від збоїв:
    - основні статистичні дані зберігаються у файл через певний проміжок часу;
    - відновлення системи на випадок збою.
- система має надавати змогу користувачам налаштовувати графіки і сповіщення;
- можливість вибирати тип графіку і вид і кількість графіків на сторінці;
- можливість налаштовувати метрики для, яких потрібно відправляти сповіщення;
- система повинна перевіряти доступність веб-додатку;
- система повинна надавати захист персональних даних користувача і використовувати алгоритми хешування для надійного захисту системи.

Нефункціональні вимоги:

- швидкий доступ до системи;
- зручний і інтуїтивний користувацький інтерфейс;
- легкість в налаштуванні.

1.4. Вимоги до безпеки системи

Безпека користувацьких даних є дуже важливим питанням. Щоб забезпечити їх цілісність і конфіденційність потрібно проаналізувати всі ризики і прийняти заходи для їх запобігання.

Таблиця 1.2.

Загрози безпеці

№ з/с	Уразливість	Загроза	Вимоги до безпеки
1	Поганий захист бази даних	Несанкціонований доступ до бази даних	Надійні паролі для бази даних
2	Відсутність шифрування	Викрадення даних	Використання актуальних алгоритмів шифрування
3	Невірно введені дані	Збої в роботі системи, загроза несанкціонованого доступу в систему	Перевірка і валідація введених користувацьких даних

Дані загрози є критичними для коректної роботи системи, тому потрібно вжити заходів для запобігання цих загроз, таких як:

- коректна валідація даних;
- використання актуальних алгоритмів шифрування;
- надійні паролі, для бази даних.

1.5. Висновки до розділу 1

В наш час, коли розробляється безліч веб-середовищ існує проблема моніторингу їх роботи, для відслідковування основних метрик і статистичних показників, таких як: кількість користувачів веб-додатку, кількість переходів, працездатність, безпека та інших.

Розроблена моніторингова система призначена для вирішення даних проблем і надає зручний користувацький інтерфейс і має систему сповіщень.

В цьому розділі були проаналізовані функціональні і нефункціональні вимоги до даного програмного забезпечення. Окрім цього було проведено аналіз ризиків до безпеки даної системи, було створено таблицю ймовірних загроз і дій для їх запобігання. Також було проводено порівняння з аналогами, яке показало, що наявні аналоги моніторингових систем не є досконалими і не мають системи сповіщень для популярних месенджерів чи можливості підрахунку статистичних показників по користувацьким метрикам.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Автоматизована моніторингова система реалізується у вигляді веб-середовища, тому потрібно проаналізувати найпопулярніші мови програмування.

Для аналізу взяті найпопулярніші мови програмування, які відповідають таким вимогам:

- мати постійну підтримку і оновлення;
- велику кількість готових бібліотек;
- мати зручний фреймворк для написання веб-додатків.

Мова програмування Java

Java – С-подібна об’єктно-орієнтована мова програмування, яка підтримується компанією Oracle. Синтаксис мови схожий на С, С++, але управлінням пам’ятю займається віртуальна машина, що спрощує роботу програміста. Основною перевагою мови програмування Java є те, що її код може виконуватись на різних платформах. Весь написаний код інтерпретується в байт-код, який може виконуватись на будь-якій платформі за допомогою віртуальної машини. Свою об’єктноорієнтованість Java запозичила в мови програмування С++, але її було модифіковано і покращенно, для того, щоб спростити роботу розробників. Мова є строго типізованою, це дозволяє зменшити кількість помилок і збільшити надійність програми.

Переваги мови програмування Java:

- програмний код може виконуватись на різних платформах і не потребує модифікацій;
- використовується для написання складних систем, через свою надіжність і підтримку;
- С – подібний синтаксис, легка у вивчені і розуміні.

Недоліки:

- низька продуктивність, використання віртуальної машини знижує продуктивність написаного коду, а також Java сама керує використанням пам'яті, що призводить до використання системних ресурсів.

Мова програмування Python

Python – високопродуктивна мова програмування, яка має зручний мінімалістичний синтаксис, а також є інтерпретованою, об’єктноорієнтованою і має строгу типізацію. Основною перевагою мови програмування Python є низький рівень входження, також вона надає великий набір стандартних функцій. Мова програмування Python не має строгої типізації, що підвищує швидкість написання програм, але знижує безпеку і надійність розробленого програмного забезпечення. Основною перевагою цієї мови програмування є її модульність, розробник може розбивати програму на різні модулі, що є дуже ефективним. Також Python містить багато стандартних модулів, які дозволяють працювати з файлами, створювати графічні інтерфейси і управляти системними викликами.

Переваги:

- простий і зрозумілий синтаксис, та велика кількість стандартних модулів дозволяє швидко розроблювати програмне забезпечення;
- має автоматичне керування пам'яттю;

- має безліч фреймворків і модулів, які використовуються для написання складних систем з використанням машиного навчання.

Недоліки:

- через автоматичне використання пам'яті і нестрогої типізації використання оперативної пам'яті не є мінімальними.

Мова програмування Javascript

JavaScript – прототипна і скриптова мова програмування, яка немає строгої типізації. В основному застосовується на клієнтській частині вебдодатку для керування вмістом сторінки у браузері, але також може застосовуватись для програмування на стороні сервера за допомогою фреймворка NodeJs. На даний момент є однією з найпопулярніших мов програмування, через свою гнучкість і використання у браузерах. Має стандартизацію – EcmaScript, кожного року виходять нові версії, які містять нові функції і покращують використання мови.

Javascript має такі властивості:

- немає строгої типізації;
- керування пам'яттю здійснюється автоматично;
- підтримує прототипну і функціональну парадигми програмування;
- має вбудовану підтримку асинхроності.

Переваги:

- підтримується усіма браузерами і є основною мовою для написання сценаріїв для веб-сторінок;
- універсальність – можна використовувати, як для створення клієнтської так і серверної частин.

Недоліки:

- немає строгої типізації.

Мова програмування C#

C# – C-подібна об’єктно-орієнтована мова програмування, яку створила компанія Microsoft для платформи .Net. C# є дуже схожою до мови програмування Java своїми принципами і синтаксисом, так як їніми похідними є такі мови програмування, як:

- C ++;
- Object Pascal;
- C;
- SmallTalk.

Мова програмування C# є строго типізованою і використовується в основному для створення веб-додатків, в основному для передачі і збереження даних використовує мову розмітки XML.

Переваги:

- C# використовує об’єктно-орієнтований підхід для програмування. Це означає, що потрібно описувати всі сутності предметної області, для якої ведеться розробка програмного забезпечення і це знижує кількість помилок;
- має досить потужний засіб для розробки – Microsoft Visual Studio. Використовується для розробки як консольних програм так і більш складних програм з графічним інтерфейсом, а також вебзастосунків;
- має багато бібліотек, які допоможуть в розробці.

Недоліки:

- дуже сильно прив'язана до платформи .Net і операційної системи Windows, тому вона є не зовсім кросплатформеною і розробка на інших операційних системах є досить проблематичною.

Проаналізувавши різні мови програмування, для розробки було вибрано мову програмування JavaScript оскільки її можна використовувати одночасно для написання клієнтської і серверної частин. Для розроблення серверної частини використовується подієво-орієнтована платформа NodeJs, який має велику кількість готових бібліотек і фреймворків для розробки веб-застосунків. Дана мова має добру підтримку і входить у список найпопулярніших мов програмування. А також в неї є доступна документація з багатьма прикладами.

2.2. Вибір фреймворку для написання серверної частини

Для розробки серверної частини використовується подієвоорієнтована платформа NodeJs, яка є дуже ефективною для написання вебзастосунків, тому що вона використовує асинхрону модель обробки подій, тобто запити не блокують основний потік виконання програми, а відкладаються в окремий стек і виконуються там. Для розробки вебзавстосунків інує безліч фреймворків розглянемо і проаналізуємо їх.

Фреймворк Express.Js

Express.Js – є одним з найдавніших і найпопулярніших фреймворків для платформи Node.Js. Завдяки його стабільності і популярності існує багато прикладів і інструкцій з програмним кодом, які створили інші розробники і протестували на практичних задачах. Основною особливістю є те, що Express.Js володіє необхіднимм мінімальнимм функціональними можливостями для створення веб-додатку, через це являється надійним і добре протестованим. Для створення веб-серверу розробнику потрібно написати всього декілька рядків коду, це означає що розробник отримує легкий і надійний інструмент для вирішення різних задач. Для розширення функціональних можливостей цього фреймворку потрібно використовувати модулі які можна завантажити через пакетний менеджер - npm. Npm містить багато готових пакетів, які допоможуть в розробці.

Переваги:

- простота у використані і налаштуванні;
- гнучкість;
- має добру докуменацію з прикладами програмного коду;
- може поєднуватись з іншими модулями;
- легко масштабується.

Недоліки:

- необхідно шукати необхідні модулі для роботи;
- використовує застарілий підхід callback функцій.

Фреймворк Koa.Js

Koa – фреймворк, який був створений тією ж командою розробників, яка створила Express.Js, як удосконалений його аналог. Він розроблявся, щоб збільшити продуктивність веб-додатку. Він підтримує і використовує стандарт EcmaScript 6, що дозволяє писати більш зрозумілий код. Замість застарілих callback функцій він використовує генератори, які дозволяють підвищити продуктивність роботи. Але даний фреймворк є достатньо новим і не так широко використовується, як Express.js.

Переваги:

- продуктивність і швидкість роботи;
- гнучкість;
- використання генераторів замість callback функцій.

Недоліки:

- невелика популярність зі сторони розробників;
- менше прикладів програмного коду ніж у Express.Js.

Фреймворк Sails.js

Sails.js – це фреймворк, який в себе включає достатні функціональні можливості для створення веб-додатків без використання зовнішніх модулів. Завдяки цьому розробнику потрібно менше часу витрачати для пошуку необхідних модулів. Але це і є недоліком, тому що якщо він не містить необхідного модуля, то знайти потрібний і встановити є проблемою. Даний фреймворк має вбудовану ORM для управління різними базами даних – Waterline ORM. Але вона є недосконалою і не підтримує багато нових функцій, таких як: транзакції і нові функції виправлення помилок.

Переваги:

- широкі базові функціональні можливості;
- підтримка Socket.Io.

Недоліки:

- повільний в роботі;
- обмеження вбудованої ORM;
- документація не містить потрібних прикладів коду.

Таблиця 2.1.

Порівняння фреймворків для платформи Node.Js

Фреймворк для платформи Node.Js	Зручність створення веб додатків	Гнучкість (використання зовнішніх модулів)	Швидкість роботи(1- 5)	Наявність прикладів програмного коду
Express.Js	Є досить зручним для створення вебдодатків, містить всі необхідні функціональні можливості	Легко працює і підтримує зовнішні модулі	4	Документація містить всі необхідні приклади програмного коду

Koa.Js	Є досить зручнимдля створення вебдодатків, містить весь необхідний мінімальний функціонал	Легко працює і підтримує зовнішні модулі	5	Недостатня кількість прикладів, пов’язана з тим, що фреймворк досить новий
Sails.Js	Має широкий функціонал, що дозволяє зручно створювати веб-додатки без зовнішніх модулів	Важко знайти необхідні модулі і підключити їх	2	Мало прикладів програмного коду, пов’язано з не велекою популярністю фреймворку. Документація не дуже хороша

Проаналізувавши існуючі фреймворки для свторення веб-додатків для платформи Node.Js для створення моніторингової системи було обрано фреймворк Express.Js. Він має більшу надійність, ніж інші фреймоврки, а також має хорошу документацію з великою кількістю програмного коду.

2.3. Вибір бази даних

Розроблюване програмне забезпечення повинно зберігати велику кількість користувацьких даних з складною структурою. Розглянемо і проаналізуєму існуючі системи екрування базами даних.

База даних Postgre SQL

PostgreSQL – широко розповсюдженна база данних, яка має досить потужний функціонал. Належить до групи реляційних баз даних, її вихідний код є відкритим, що дозволяє розробникам і різним компаніям постійно вдосконалювати його. Реляційну базу даних PostgreSQL використовують для

розробки складних програмних систем, для яких важливим є питання безпеки їхніх даних. Має вбудовану мову прграмування PL/pgSQL, що дозволяє писати складні сценарії і алгоритми.

Переваги:

- велика підтримка зі сторони великих компаній і велика кількість обновлень;
- може масштабуватись, як і вертикально так і горизонтально;
- є можливість оброблювати і зберігати велику кількість даних;
- має інтуїтивно зрозумілий інтерфейс користувача PgAdmin.

Недоліки:

- складність встановлення і налаштування, а також подальшого адміністрування.

База даних MySQL

MySQL – одна з найбільш використовуваних реляційних баз даних. Має відкритий код системи і весь час оновлюється завдяки своїй популярності. Дана система керування базою даних має підтримку майже у всіх мовах програмування і легка в налаштуванні. Має зручний консольний інтерфейс для управління базами даних а також велику кількість додатків з графічним інтерфейсом для управління базами данних.

Переваги:

- легка в налаштуванні і адмініструванні;
- підтримує велику кількість типів даних і таблиць;
- існує багато програмних прикладів використання;
- легко масштабується і має безліч вбудованих функцій безпеки.

Недоліки:

- потребує більшого часу на виконання запитів до бази данних і виконання дій з даними (створення, оновлення, видалення);
- недуже надійна в процесах роботи з даними (транзакції і аудит).

База даних MongoDb

MongoDb – нереляційна, документо-орієнтована база даних. Для зберігання даних використовує текстовий формат даних – JSON. Використовується для зберігання даних, об’єкти яких можуть не мати чіткої структури, що є її перевагою. Має досить ефективну масштабованість за рахунок шардингу. Для використання різними мовами програмування необхідно встановити драйвер, який надасть користувачу доступ до бази данних за допомогою зручної функціональної мови запитів.

Переваги:

- відсутність схем для даних;
- легко масштабується;
- дані зберігаються у форматі Json;
- швидкий доступ до даних.

Недоліки:

- не можна використовувати мову запитів SQL.

Таблиця 2.2.

Порівняння існуючих баз даних

Назва	Масштабованість	Мова запитів	Гнучкість зберігання даних	Складність в налаштуванні(1-5)	Швидкість обробки запитів
PostgreQsl	Підтримує горизонтальне масштабування, але через додаткові засоби	Sql	Дані зберігаються в таблицях із заданою структурою	5	Висока швидкість обробки вхідних запитів
MySql	Важко масштабується	Sql	Дані зберігаються в таблицях із заданою структурою	3	Оброблює запити повільніше, ніж в інших базах даних
MongoDb	Легко масштабується через вбудований шардинг і реплікацію	Функціональна	Дані не мають чіткої структури	1	Швидко оброблює вхідні запити

Проаналізувавши існуючі системи контролю базами даних для розроблюваної монітирингової системи було обрано СКБД MongoDb. Вона має функціональну мову запитів, що є зручним для розробки так, як мова JavaScript є функціональною. Також дана база даних легко масштабується, завдяки вбудованому шардингу і реплікації, на відміну від інших баз даних, які потрібно

налаштовувати за допомогою додаткового ПО і це є досить важким процесом. Її налаштування не потребує складних зусиль і для використання потрібно встановити спеціальний драйвер. Для зручного опису схеми даних існує окрема бібліотека Mongoose, що дозволяє описувати схему даних і покращує операції з даними підвищуючи безпеку.

2.4. Вибір засобу для відображення графіків

ChartJs

ChartJs – графічна бібліотека для створення графіків і діаграм, яка на основі вхідних даних будує графіки і повертає сторінку у форматі HTML. Має багато налаштувань для графіків і діаграм (розміри, колір, кількість точок). Дана бібліотека є простою у використанні, але сторінка з графіками генерується на стороні сервера, що не є дуже ефективним рішенням.

Переваги:

- простота у використанні;
- великий набір графіків;
- генерується Html сторінка з графіками.

Недоліки:

- графіки статичні;
- генерація відбувається на стороні сервера.

Grafana

Grafana – платформа з відкритим кодом, яка використовується для відображення користувацьких метрик на графіках в реальному часі. Дана платформа дозволяє користувачам створювати дашборди на яких можна розмістити різні види графіків. Garafana надає користувачеві такий функціонал:

- зміна масштабу графіків;

- сортування значень в таблицях;
- багато видів графіків;
- виділення окремим коліром графіку;
- можливість розміщувати більше одного графіку на дашборді.

Переваги:

- велика кількість налаштувань; • відображення метрик у реальному часі;
- працює окремо від серверної частини.

Недоліки:

- складність налаштування.

Проаналізувавши дані засоби для візуалізації користувацьких метрик було вибрано платфому Grafana, так як вона надає більш широкі функціональні можливості і надає змогу відображати зміну метрик в реальному часі.

2.5. Висновки до розділу 2

В даному розділі було проаналізовано технології та програмні засоби, які будуть використовуватись в розроблюваному програмному забезпеченні, такі як: мови програмування, бази даних, фреймворки для платформи Node.Js і технології для відображення графіків. Серед мов програмування було проаналізовано такі мови: Java, Python, Javascript, C#; серед фреймворків для платформи Node.Js: Express.js, Koa.js, Sails.js; серед баз даних: PostgreSQL, MongoDb, MySQL; серед технологій для відображення графіків: Chart.js, Grafana. Було описано їхні основні недоліки і переваги і на їх основі вибрано ті, які будуть використовуватись в розроблюваному програмному забезпеченні.

Серед мов програмування було обрано мову програмування Javacsript, через її універсальність, наявність платформи Node.Js для написання серверної частини системи, а також наявність великої кількості зовнішніх модулів.

Серед фреймворків для платформи Node.Js було обрано Express.Js через його надійність, а також хорошу документацію з великою кількістю програмного коду.

Серед баз даних було обрано MongoDb. Вона легко масштабується, завдяки вбудованому шардингу і реплікації, а також має зручну функціональну мову запитів.

Серед засобів для відображення графіків було обрано Grafana, так як вона надає більш широкі функціональні можливості і надає змогу відображати зміну метрик в реальному часі.

РОЗДІЛ 3
РОЗРОБЛЕННЯ ТА АНАЛІЗ МОНІТОРИНГОВОЇ СИСТЕМИ

3.1. Загальний опис архітектури системи

Розроблюввльна моніторингова система представлена у вигляді вебзастосунку, який має клієнт-серверну архітектуру. Дана архітектура є домінуючою серед всіх видів розроблення додатків. На рис. 3.1 зображено архітектуру системи.

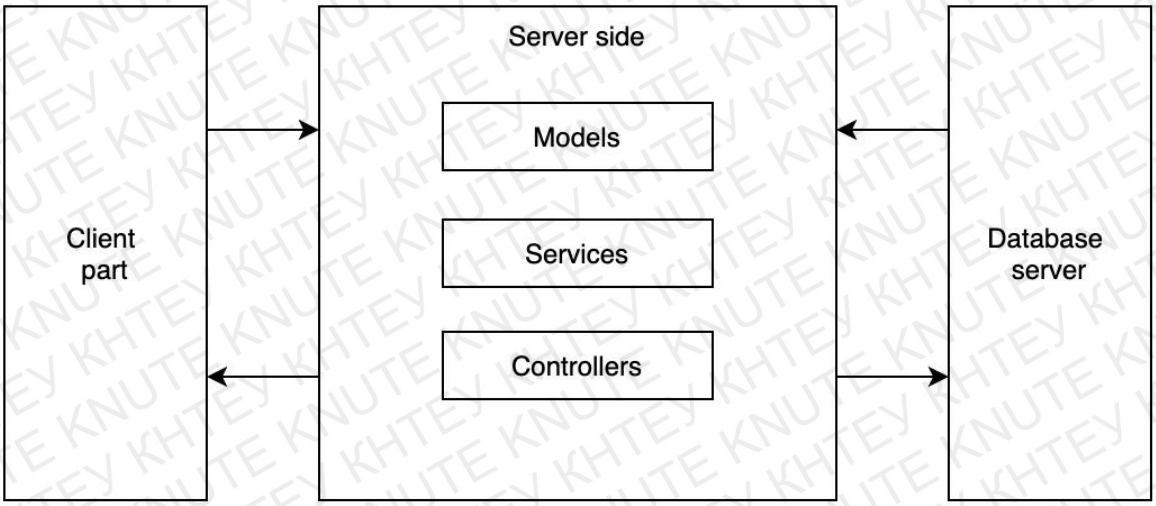

Рис. 3.1. Архітектура систем

Вона має високу гнучкість і дозволяє відділити клієнтську логіку, яка може виконуватись, як в браузері так і в додатку на мобільному пристрої від серверної частини

Клієнт-серверна архітектура складається з таких компонентів:

- набору серверів, які відповідають за обробку данних;
- набору клієнтів, які користуються функціями серверів;
- мережі, яка забезпечує зв'язок між клієнтами і серверами.

Всі частини в клієнт-серверній архітектурі є окремеми одиницями і функціонують незалежно один від одного. Взаємодія компонентів визначається пеаним розподілом їхніх обов'язків. Можна відокремити такі рівні операцій:

- рівень представлення даних – інтерфейс користувача, який відповідає за відображення даних користувачам;
- прикладний рівень – відповідає за логіку обробки інформації;
- рівень управління даними – надає доступ і управління даними.

Оскільки для створення серверної частини використовувався платформа NodeJs, було обрано фреймворк Express. Даний фреймворк використовує архітектурний шаблон MVC (Model-View-Controller):

- Model – представлення даних, зазвичай описується структура бази даних, відповідає за зміну даних при певній команді з контролера;
- View – відображає дані моделі користувачу;
- Controller – реагує на дії користувача і надає команду моделі про зміну даних.

На рис. 3.2 показано схему архітектурного шаблону MVC.

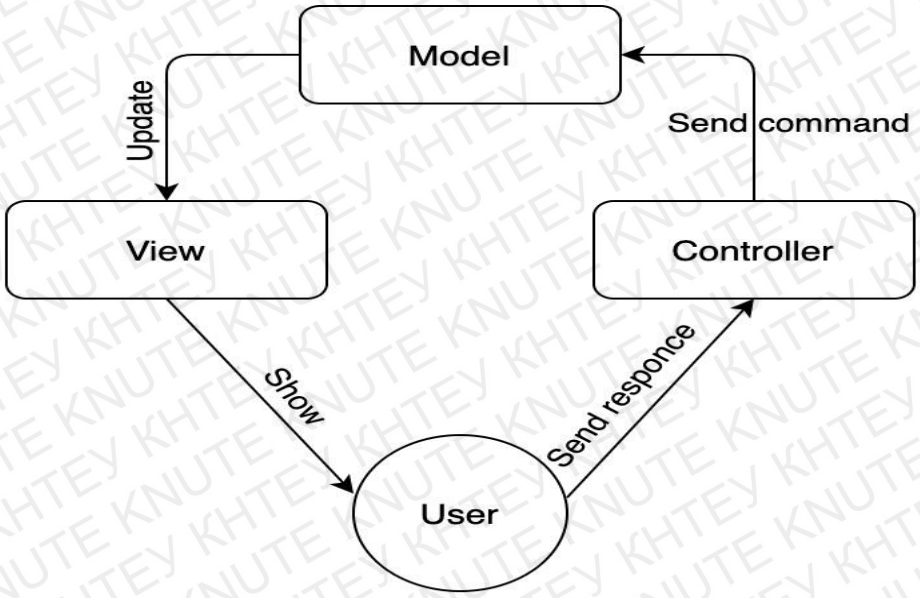

Рис. 3.2. Схема MVC

Серверна частина розроблювальної моніторингової системи виконує такі функції:

- обробка вхідних запитів;
- обрахунок та відображення необхідних статистичних даних:
  - мінімальне/максимальне значення для метрики;
  - середнього значення;
  - середнє квадратичне відхилення.
- збереження, фільтрування і сортування даних за часом;
- сповіщення про некоректну роботу:
  - обробка і аналіз метрик, для визначення системних збоїв і перевищення заданих значень;
  - відправка сповіщень про збої в мессенджери.
- захист від збоїв: основні статистичні дані зберігаються у файл через певний проміжок часу; відновлення системи на випадок збою;
- система надає змогу користувачам налаштовувати графіки і сповіщення; система має захист персональних даних користувача і використовувати алгоритми хешування для надійного захисту системи.

Клієнтська частина надає користувачам інтерфейс для перегляду їх додатків і надає доступ до сторінки перегляду статистики по конкретному додатку. Також клієнтська частина представленна у вигляді модуля, який потрібно підключити до свого додатку і налаштувати, які дані необхідно відображувати на графіках і які статистичні дані необхідно обраховувати.

Клієнтська частина розроблювальної моніторингової системи виконує такі функції:

- надає користувацький інтерфейс для управління сисьемою і відображення користувацьких даних:
  - реєстрація користувача;
  - перегляд додатків, над якими ведеться моніторингу;

- доступ до дашбордів зі статистичними даними;
- внесення змін до інформації про користувача;
- управління додатками користувача.

- надає модуль для підключення до клієнтського додатку, який має свої налаштування:
  - вид метрик;
  - кількість і вид графіків на сторінці Grafana сервісу;
  - збір системних метрик;
  - перевірка доступності веб-ресурсів;
  - відправка сповіщень.

На рис. 3.3. показано схему використання системи на якій показано взаємодію користувача з клієнтською і серверною частинами моніторингової системи.

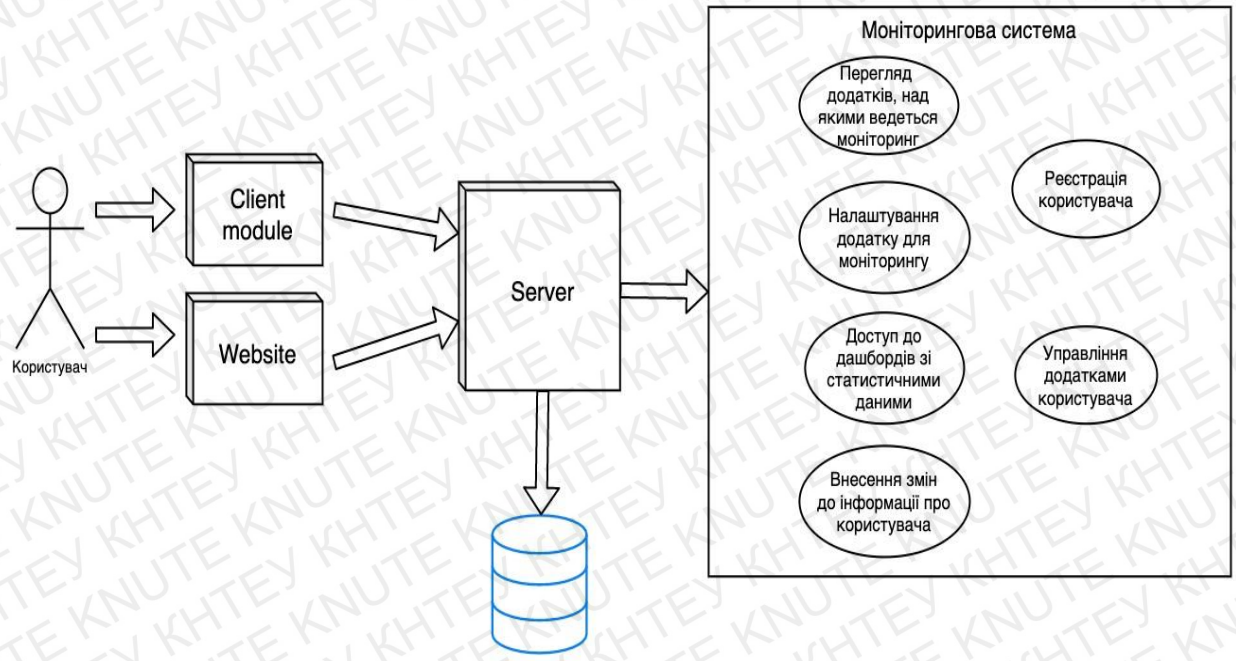

Рис. 3.3. Схема використання системи

3.2. Опис серверної частини

Серверна частина складається з таких сервісів:

- Profile API – мікросервіс, який відповідає за авторизацію і аутентифікацію користувачів. Для авторизації користувачів сервіс використовує JWT токени. Токен створюється на основі даних користувача і містить в собі 3 основні частини: ○ Header – містить алгоритм хешування; ○ Payload – містить основні дані про користувача; ○ Sigature – підпис згенерований на основі двох попередніх частин.

Токен має певний час дії, після якого його потрібно оновити. Це дозволяє збільшити надійність і безпеку, тому що якщо токен буде викрадено, то через деякий час срок його дії закнічиться і він не буде валідним;

- Statsmeter API – мікросеріс, який відповідає за обробку всіх клієнтських метрик, підраховує статистичні показники по користувацьким метрикам, такі як:
  - Мінімальне значення;
  - Максимальне значеня;
  - Середнє значення;
  - Середнє квадратичне відхилення;
  - Кванилі.

Даний мікросервіс отримує дані від користувача за допомогою вебсокетів. Дані передаються у зашифрованому вигляді, сервіс декодує, обробляє і зберігає їх для подальшого відображення на графіках. Statsmeter API містить такі сервіси:

- AlertsService – сервіс, який відповідає за відправку повідомлень користувачам, коли якась метрика перевищила задане значення;
- EventBusService – сервіс, в який записуються нові користувацькі метрики, які потім обробляються і видаляються.

- MetricsStorageService – сервіс, який відповідає за збереження користувацьких метрик;
- ListenerService – сервіс, який відповідає за з'єднання із клієнською частиною і відповідає за відправку даних за допомогою вебсокетів.

- Cluster API – мікросервіс, який відповідає за створення клієнтських контейнерів для сервісів Grafana, Graphite і Statsmeter, які взаємодіють один з одним. Даний сервіс створює конфігурації для інших сервісів і запускає контейнери з ними за допомогою системи Kubernetes. Cluster Api містить такі сервіси:
  - ClusterService – сервіс, який відвідає за створення кластерів, надає базовий CRUD (create, read, update delete операції), а кінцеві точки для розгортання і згортання кластерів додатку;
  - ClusterHealthCheckJob – сервіс, який перевіряє доступність кластерів на платформі Kubernetes, а також звільняє ресурси, які не використовуються;
  - ConfigComposerService – сервіс, який створює файли конфігурацій для сервісів Grafana, Graphite і Statsmeter;
  - ConfigValidator – сервіс, який відповідає за валідацію конфігурацій сервісів;
  - DeploymentService – сервіс, який відповідає за розгортання і запуск користувацьких кластерів на платформі Kubernetes, а також за їх видалення;
  - DiskResizeService – сервіс, який перевіряє кількість зайнятого диску на кластерах користувача і якщо місця недостатньо, то кластеру виділяється більше місця.

- Grafana service – сервіс, який відповідає за відображення користувацьких метрик на графіках в реальному часі. Дана платформа дозволяє користувачам створювати дашборди на яких можна розмістити різні види графіків. Garafana надає користувачеві такий функціонал:
  - Зміна масштабу графіків;
  - Сортування значень в таблицях;
  - Багато видів графіків;
  - Виділення окремим коліром графіку;
  - Можливість розміщувати більше одного графіку на дашборді.
- Graphite service – сервіс, який відповідає за збереження моніторингових даних від різних клієнтів у базу даних і їх відображення;
- Slack-bot Api – сервіс, який відповідає за відправкусповіщень користувачам в месенджер Slack.

На рис. 3.4. зображено сервіси з яких складається серверна частина.

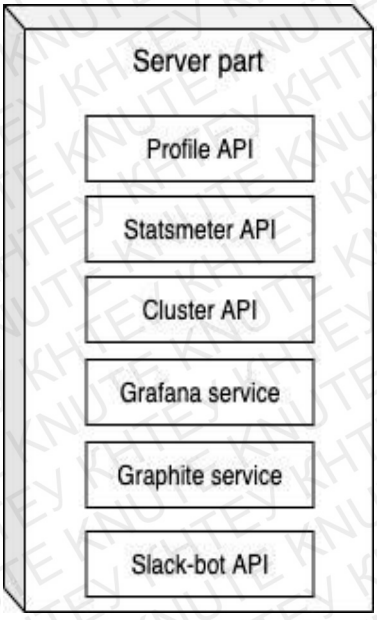

Рис. 3.4. Структура сервісів серверної частина

3.3. Опис клієнтської частини

Клієнтська частина моніторигової системи складається з двох частин: • Веб інтерфейс користувача;

• Модуль (бібліотека) клієнта, який підключається до веб-додатку за яким потрібно вести моніторинг.

Веб-інтерфейс надає можливість переглядати додатки за якими ведеться моніторинг і зупиняти цей процес при необхідності.

Клієнтський модуль надає змогу пористувачам встановити спеціальний пакет до свого веб-додатку і налаштувати відправку метрик і їх тип за допомогою спеціальних налаштувань.

Таблиця 3.1

Налаштування системи

token	Токен авторизації
passwordProtect	Налаштування захису паролем
application	Назва додатку
flushIntervalInSeconds	Інтервал через який потрібно обновлювати значення на графіку
Retention.frequency	Частота оновлення
Retention.keep	Час зберігання метрик
admins	Масив з адміністраторами веб-додатку
plugins	Масив з плагінами. Є два плагіни system-monitor – відповідає за збір системних метрик і health-check – перевіряє доступність ресурсів

Відправка даних здійснюється за допомогою веб сокетів. Це спеціальна технологія, яка дозволяє створювати з'єднання між клієнтом і сервером для обміну повідомленнями в реальному часі. Ця технологія використовує двоспрямований потік даних, щодозволяє не дублювати повідомлення і надає бистрий зв'язок для відправки даних в реальному часі.

На рис. 3.5. зображено приклад відображення користувацьких метрик.

Рис. 3.5. Приклад відображення даних на графіках

3.4. Опис структур даних для зберігання і підрахунку статистики

Для зберігання і підрахунку статистичних даних система використовує структуру даних основану на алгоритмі FIFO (first in first out). Даний алгоритм працює по принципу черги – елементи добавляються в дану чергу і видаляються через певні проміжки часу. Дана структура даних має обмежену кількість елементів в черзі, які видаляються через певний проміжок часу.

Структура даних, яка зображена на рис. 3.6 має фіксований розмір і час протягом якого вона зберігає дані, час розділяється на кожну комірку і протягом певного проміжку часу статистичні дані накопичуються в певній комірці даної

структури даних. Статистичні дані переобраховуються зразу при додаванні нової метрики. Існують такі стистичні показники:

- Мінімальне значення;
- Максимальне значення;
- Середнє значення;
- Кількість елементів;
- Середнє квадратичне відхилення;

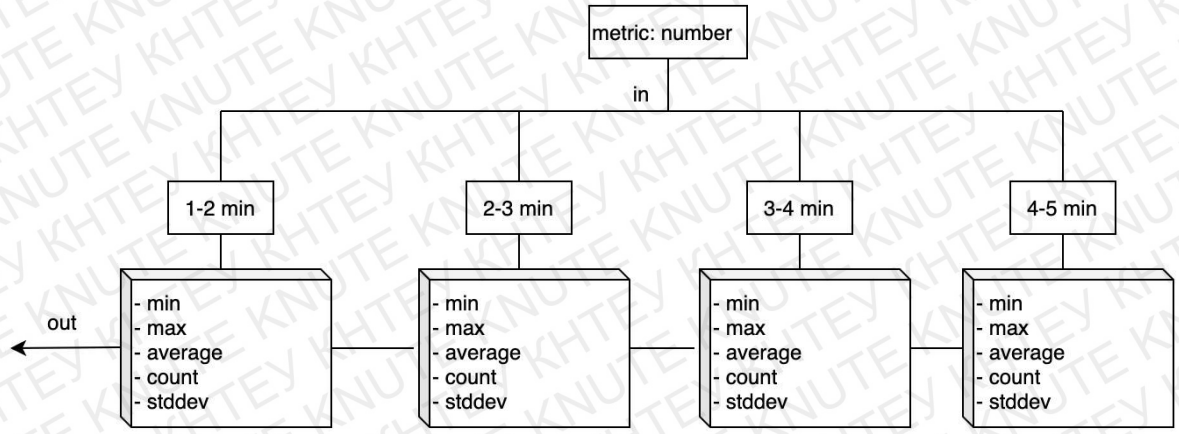

Рис. 3.6. Опис структури даних

Структура дозволяє переобчислюватисереднє квадратичне відхилення на основі попередніх значень, це означає, що потрібно переобчислювати тільки суму квадратів і загальну суму.

Середнє квадратичне відхилення це статистичний показник, який показує на скільки в середньому відхиляються значення від середнього їх значення.

3.5. Опис структури бази даних

Для реалізації моніторингової системи було використано базу даних MongoDb, які використовуються системою. Також для збереження даних використовується жорсткий диск на якому зберігаються статистичні дані.

Система має 5 сутностей:

- Profile
- Cluster
- Task
- Ports
- Tags

На рис. 3.7. зображено схему бази даних системи.

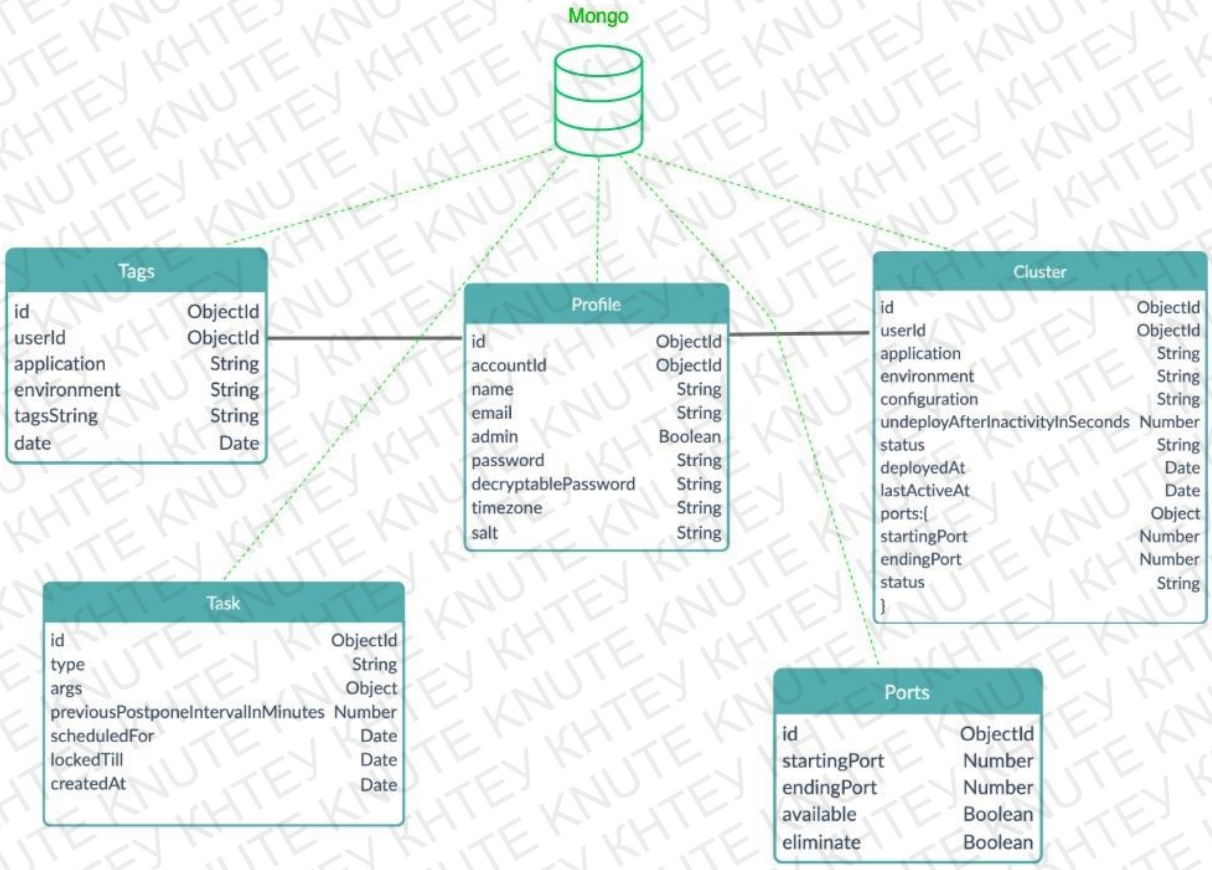

Рис. 3.7. Схема бази даних

Profile

Таблиця Profile зберігає інформацію про всіх користувачів системи.

В таблиці 3.2. показано назви полів і їх типи даних.

Таблиця 3.2.

Поля сутності Profile

Назва поля	Тип даних
id	ObjectId
accountId	ObjectId
name	String
email	String
admin	Boolean
password	String
timezone	String
salt	String

Cluster

Таблиця Cluster зберігає інформацію про запущенні для моніторингу користувацькі додатки. В таблиці 3.3. показано назви полів і їх типи даних.

Таблиця 3.3.

Поля сутності Cluster

Назва поля	Тип даних
id	ObjectId
userId	ObjectId
application	String
environment	String
configuration	String
undeployAfterInactivityInSeconds	Number

status		String
deployedAt		Date
lastActiveAt		Date
ports		Object
startingPort	Number	
endingPort	Number	

Task

Таблиця Task зберігає інформацію про завдання, які використовуються планувальником для розгортання кластерів з користувацькими сервісами (Statsmeter, Grafana, Graphite). В таблиці 3.4. показано назви полів і їх типи даних.

Таблиця 3.4.

Поля сутності Task

Назва поля	Тип даних
id	ObjectId
type	String
args	Object
previousPostponeIntervalInMinutes	Number
scheduledFor	Date
lockedTill	Date
createdAt	Date

Ports

Таблиця Ports зберігає інформацію про використанні порти користувацькими додаткам, для виділення вільних портів. В таблиці 3.5. показано назви полів і їх типи даних.

Таблиця 3.5.

Поля сутності Ports

Назва поля	Тип даних
id	ObjectId
startingPort	Number
endingPort	Number
available	Boolean
eliminate	Boolean

Tags

Таблиця Tags зберігає інформацію про використанні теги користувацьких додатків. В таблиці 3.6. показано назви полів і їх типи даних.

Таблиця 3.6.

Поля сутності Tags

Назва поля	Тип даних
id	ObjectId
userId	ObjectId
application	String
environment	String
tagsString	String
date	Date

3.6. Аналіз реалізованої моніторингової системи

Розроблена моніторингова система призначена для відслідковування користувацьких і системих метрик і надання зручного інтерфейсу системним адміністраторам для їх відслідковування. Вона надає такий функціонал своїм користувачам:

- можливість зареєструватись в системі. Реєстрація нового користувача відбувається на серверній частині:
  - Перевіряється, чи такий користувач ще не зареєстрований в системі;
  - Використовується алгоритм SHA512 для хешування паролю;
  - Всі дані користувача із захешованим паролем зберігаються в базу даних;
  - За допомогою модуля jsonwebtoken створюється JWT токен, який відправляється користувачу і він його використовує в запитах до системи.
- налаштування типу графіків, які будуть відображатись користувачеві. Налаштування кількості і типу графіків в системі реалізовано за допомогою створення файлів конфігурацій в яких описано, які саме графіки і метрики повині бути на сторінці і їх кількість.
- налаштування метрик, які будуть відслідковуватись. Існує 5 видів метрик:
  - Gauge – вид метрики, який показує поточне значення метрики;
  - Histogram – вид метрики, який показує скільки значень потрапляє у конкретні інтервали значень протягом конкретного часу;
  - Counter – вид метрики, який показує кількість певних метрик;
  - Statistic – вид метрики, який показує на графіку зміну статистичних показники таких, як:

- Min – мінмальне значення;
- Max – максимальне значення;
- Median – статистична медіана, характеристика розподілу випадкових велечин;
- Stddev – стандартне відхилення;
- Avg – середнє значення;
- P90, p70 – квантилі;

Таблиця 3.7.

Пояснення полів метрики

Назва	Пояснення
metric	Назва метрики
tags	Масив тегів для яких налаштовувати сповіщення
threshold	Значення метрики при якому потрібно відправляти сповіщення про її перевищення
condition	“gt” чи “lt” – значення має вижче чи нижче зазначеної метрики
unhealthyDelay	Час протягом якого значення метрики повино бути в зоні перевищення
healthyDelay	Час протягом якого значення метрики повино бути в допустимих межах для видалення сповіщення
admins	Адміністратори, яким потрібно надсилати сповіщення
channels	Канали в які потрібно відправляти сповіщення
message	Текст повідомлення
repeatInterval	Інтервал протягом якого потрібно повторно надіслати сповіщення

• налаштування плагінів для збору системних метрик і перевірки доступності ресурсів.

Система складається з 2 частин клієнтської і серверної, які є взаємозв'язаними. На рис. 3.8. зображено структурну схему системи.

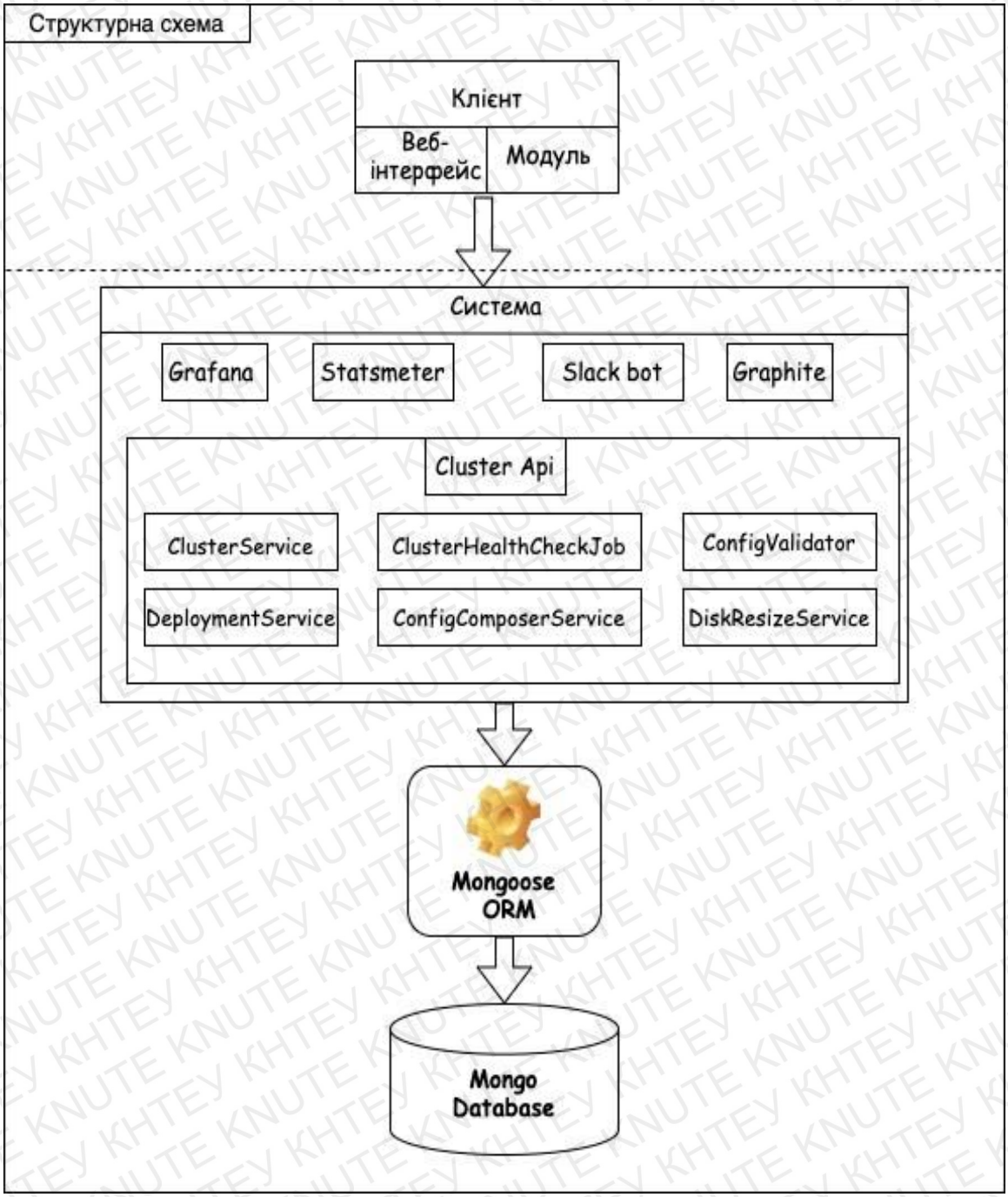

Рис. 3.8. Структурна схема

3.7. Дизайн та вміст веб-сторінок

Щоб створити дизайн і вміст сторінок системи було використано такі технології:

- HTML;
- CSS;
- Javascript.

Для створення клієнтської части було застосовано фреймворк Angular, який надає весь необхідний функціонал для створення веб-сайту. Також для створення адапнивного дизайну сайту було використано Angular Material і Bootstrap 4, що дозволяє адаптувати веб-сторінки до різних пристроїв.

Головною сторінкою в системі є сторінка перегляду веб-додатків над якими ведеться моніторинг. На цій сторінці користувач може бачити список своїх додатків і перейти на сторінку з графіками на яких відображається зміна користувацьких метрик. Також в користувача є можливість скопіювати токен доступу, який потрібно розмістити в файлі конфігурацій додатку над яким ведеться моніторинг, для подальшої автентифікації користувача. На рис. 3.9. зображено головну сторінку системи.

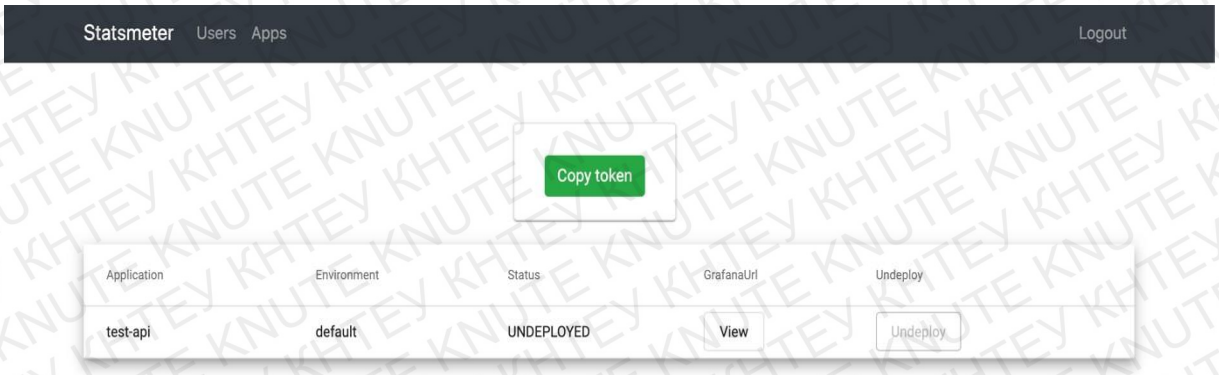

Рис. 3.9. Вигляд сторінки з користувацькими додатками

На рис. 3.10. зображено форму для входу в систему. Користувач повинен ввести свій логін і пароль для входу в систему. Якщо дані не вірні на сторінці з'явиться сповіщення з цією інформацією. Якщо введені дані вірні, то користувача перенаправить на сторінку з додатками над якими ведеться моніторинг.

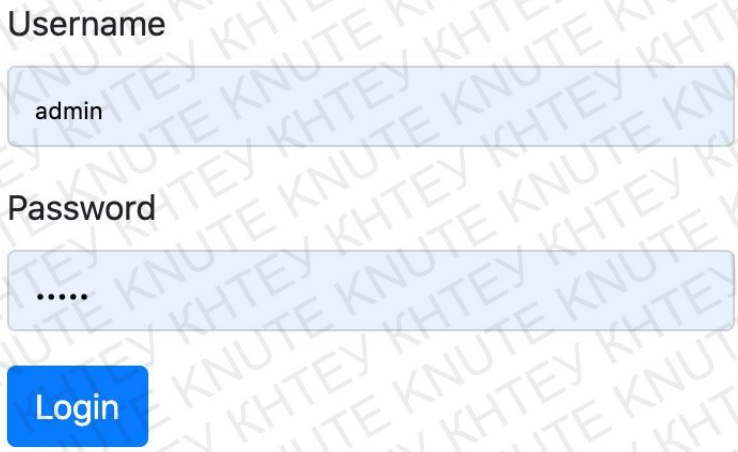

Рис. 3.10. Форма входу в систему

Коли користувач увійшов в систему у нього є можливість переглянути свої додатки над якими ведеться моніторинг, перейти на сторінку з графіками, на якій будуть показанні користувацькі метрики на графіках, зупинити моніторинг і внести зміни до своїх даних на персональній сторінці.

При переході на сторінку з графіками відкриється сторінка в сервісі Grafana, яка містить всі налаштовані графіки користувача. На графіках відображаються всі метрики в реальному часі, на графіку можна виділити проміжок часу на якому потрібно більш детально розглянути зміну метрик чи змінити його розмір в налаштуваннях. Також в користувачів системи є можливість добавити потрібні графіки на свій сайт чи на сторінку адміністратора для зручнішого моніторингу і доступу до моніторингових даних. Для цього потрібно скопіювати необхідний елемент в налаштуваннях певного графіку.

На рис. 3.11. і рис. 3.12. зображенно вигляд сторінок із відображенням графіків із системними і користувацькими метриками.

Рис. 3.11. Сторінка відображення системних метрик

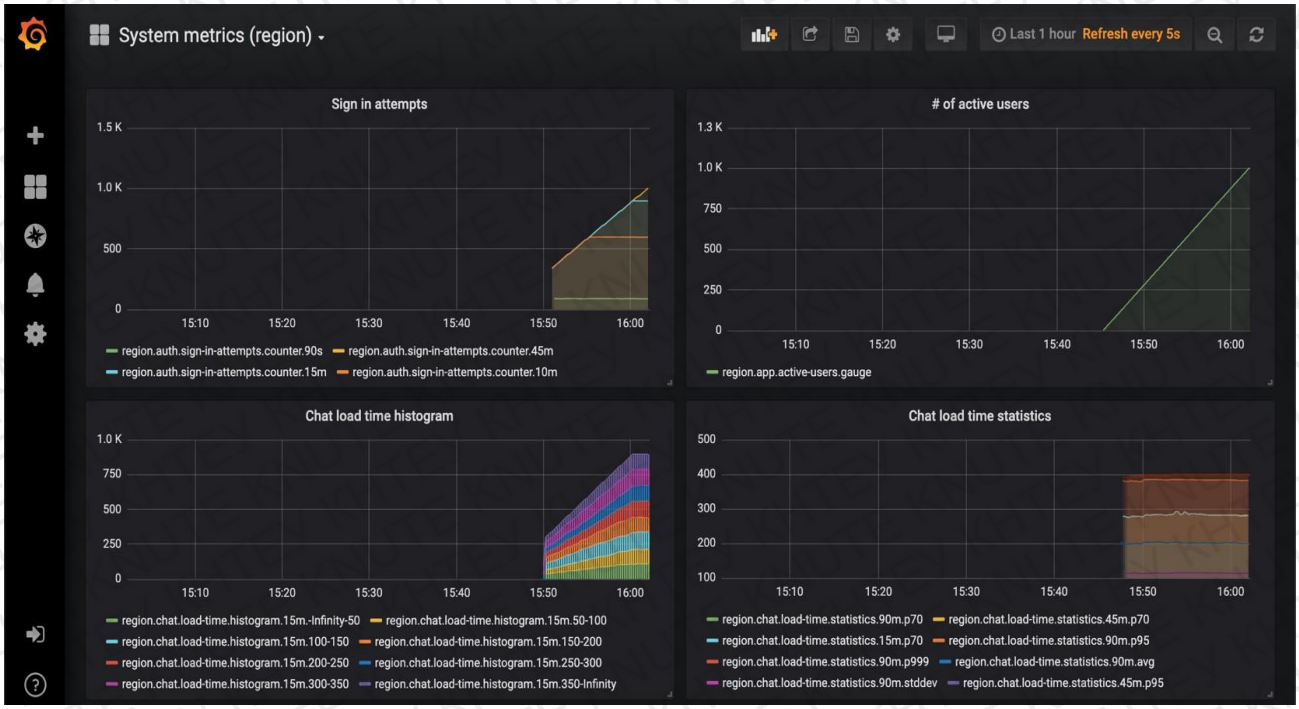

Рис. 3.12. Сторінка відображення користувацьких метрик

3.8. Тестування системи

Одним з факторів ефективної моніторингової системи є показник кількості оброблених користувацьких метрик. Протестуємо систему і створену структуру даних тестом на продуктивнысть (performance testing), щоб дізнатись кількість метрик, яку може оброблювати система. Данний вид тестування дозволяє протистувати продуктивність системи і визначити кількість даних, які може оброблюватись системою.

Для даного тесту було створено файл із 1000000 числових записів. Дані з цього файлу були прочитанні і поступово добавлялись в створену структуру даних, яка обраховує користувацькі метрики і виводились проміжні логи при оброкі кожних 100000 метрик. На рис. 3.13. показано результати тесту на продуктивність обробки даних.

```
[Mon May 04 2020 21:54:49 GMT+0300 (Eastern European Summer Time)] Started performance test
[Mon May 04 2020 21:54:49 GMT+0300 (Eastern European Summer Time)] processed 0 records in 1ms
[Mon May 04 2020 21:54:49 GMT+0300 (Eastern European Summer Time)] processed 100000 records in 299ms
[Mon May 04 2020 21:54:49 GMT+0300 (Eastern European Summer Time)] processed 200000 records in 441ms
[Mon May 04 2020 21:54:49 GMT+0300 (Eastern European Summer Time)] processed 300000 records in 578ms
[Mon May 04 2020 21:54:49 GMT+0300 (Eastern European Summer Time)] processed 400000 records in 697ms
[Mon May 04 2020 21:54:50 GMT+0300 (Eastern European Summer Time)] processed 500000 records in 813ms
[Mon May 04 2020 21:54:50 GMT+0300 (Eastern European Summer Time)] processed 600000 records in 926ms
[Mon May 04 2020 21:54:50 GMT+0300 (Eastern European Summer Time)] processed 700000 records in 1040ms
[Mon May 04 2020 21:54:50 GMT+0300 (Eastern European Summer Time)] processed 800000 records in 1150ms
[Mon May 04 2020 21:54:50 GMT+0300 (Eastern European Summer Time)] processed 900000 records in 1259ms
[Mon May 04 2020 21:54:50 GMT+0300 (Eastern European Summer Time)] processed 1000000 records took 1367ms
MetricsStorage processed 731528 records per second
```

Рис. 3.13. Результати тесту на продуктивність обробки даних

З результатів тестування на продуктивність структури даних бачимо, що для обробки 1000000 записів було витрачено 1367 мілісекунд, тобто в секунду структура даних може обробити 731528 записів. Даний показник показує, що система може справитись з значними навантаженнями. Щоб дізнатись скільки метрик може обробити система, якщо дані метрикик надсилає користувач. Було створено інтеграційний тест в якому створюється клієнт, який надсилає користувацькі метрики за допомогою веб-сокетів. На рис. 3.14. показано результати тесту на продуктивність системи.

Рис. 3.14. Результати теста на продуктивність системи

З результатів тестування на продуктивність структури даних бачимо, що для відпраки і обробки 1000000 записів було витрачено 5583 мілісекунд, тобто в секунду система може обробити 179083 записів. Даний показник показує, що система може справитись з значними навантаженнями при отримані і відправці метрик.

Також для основних модулів системи написані юніт тести. Юніт тестування дозволяє перевірити передбачену поведінку окремого модуля і впевнетись, що всі його функції вірно працюють з різними наборами даних. На рис. 3.15. показано результати юніт тестів модуля, який оброблює користувацькі метрики.

```
  MetricsStorage
[2020-05-05T13:22:03.498] [INFO] Application - Express server listening on port 3030
[2020-05-05T13:22:03.502] [INFO] MetricsManagement - Tags ["region","server"] are successfully initialized.
    ✓ should return stats by metric name
    ✓ should return stats by metric name and tags
  Counters
    ✓ should return initial zero values
    ✓ should update reservoir and return correct value
    ✓ should update reservoir after interval pass when getting stats
    ✓ should update reservoir after interval pass when event fired
    ✓ should evict old metric from reservoir after interval passed
  Gauges
    ✓ should update gauge value
    ✓ should return metric stats
  Histogram
    ✓ should return initial zero values
    ✓ should update reservoir for left interval on getting stats if time interval passed
    ✓ should update reservoir for middle interval on getting stats if time interval passed
    ✓ should update reservoir for right interval on getting stats if time interval passed
    ✓ should update reservoir on event fire if interval passed
    ✓ should update reservoir if time passed
    ✓ should evict old metric from reservoir after interval passed
  Statistics
    ✓ should calculate statistics (41ms)
    Percentile
      ✓ should not return percentiles data before any data arrived
      ✓ should calculate percentiles just after new data arrived
      ✓ should calculate percentile of single signed of integer part
      ✓ should calculate percentile of percentile value with trailing zeros
      ✓ should calculate percentiles of two-sined fractional part percentile
      ✓ should calculate percentiles of three-sined fractional part percentile
      ✓ should calculate percentiles correctly
      ✓ should return percentiles for actual records only
    Median
      ✓ should return no median value before any data arrived
      ✓ should calculate medians just after new data arrived
      ✓ should return median for actual records only
      ✓ should return center reservoir element value as median if reservoir length is odd
      ✓ should return average of two central reservoir elements if reservoir length is even
    Min
      ✓ should return no min value before any data arrived
      ✓ should calculate min just after new data arrived
      ✓ should return min for actual records only
      ✓ should calculate min of negative elements
      ✓ should return min value for reservoir of one element that eqls to that element
    Max
      ✓ should return no max value before any data arrived
      ✓ should calculate max just after new data arrived
      ✓ should return max for actual records only
      ✓ should calculate max element
      ✓ should calculate max element of negative digits correctly
      ✓ should return max value for reservoir of one element that eqls to that element
    Standard deviation
      ✓ should return no max value before any data arrived
      ✓ should return no standard deviation result for reservoir of 1 element
      ✓ should return standard deviation for actual records only
      ✓ should calculate standard deviation just after new data arrived
    Average
      ✓ should return no average value before any data arrived
      ✓ should calculate average just after new data arrived
      ✓ should return average for actual records only
      ✓ should calculate average value
  Store reservoir
[2020-05-05T13:22:27.655] [WARN] MetricsStorage - Reservoirs directory does not exist. Creating...
[2020-05-05T13:22:27.655] [DEBUG] MetricsStorage - Recorded reservoirs for 104 metrics to persistent storage in 0 ms
    ✓ should save gauge metric (105ms)
[2020-05-05T13:22:03.763] [WARN] MetricsStorage - Reservoirs directory does not exist. Creating...
[2020-05-05T13:22:03.763] [DEBUG] MetricsStorage - Recorded reservoirs for 104 metrics to persistent storage in 0 ms
    ✓ should create reservoirs directory if it doesn't exist
[2020-05-05T13:23:15.791] [WARN] MetricsStorage - Reservoirs directory does not exist. Creating...
[2020-05-05T13:23:15.791] [DEBUG] MetricsStorage - Recorded reservoirs for 104 metrics to persistent storage in 0 ms
    ✓ should save counter metrics (138ms)
[2020-05-05T13:23:03.925] [WARN] MetricsStorage - Reservoirs directory does not exist. Creating...
[2020-05-05T13:23:03.925] [DEBUG] MetricsStorage - Recorded reservoirs for 104 metrics to persistent storage in 0 ms
    ✓ should save histogram metrics (132ms)
[2020-05-05T13:22:28.060] [WARN] MetricsStorage - Reservoirs directory does not exist. Creating...
[2020-05-05T13:22:28.060] [DEBUG] MetricsStorage - Recorded reservoirs for 104 metrics to persistent storage in 0 ms
    ✓ should save statistic metrics (65ms)
  Restore reservoir
[2020-05-05T13:22:04.129] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should restore gauge metric
[2020-05-05T13:22:04.132] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should restore counter metrics
[2020-05-05T13:22:04.137] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should not restore counter metric which is not present in config
[2020-05-05T13:22:04.140] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should restore histogram metrics reservoir
[2020-05-05T13:22:04.152] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should not restore histogram metric which is not present in config
[2020-05-05T13:22:04.154] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should restore statistic metrics reservoir
[2020-05-05T13:22:04.158] [INFO] MetricsManagement - Metrics are restored from persistent storage
    ✓ should not restore statistic metric which is not present in config
```

Рис. 3.15. Приклад результатів модульних тестів

3.9. Рекомендації щодо вдосконалення

Щодо подальшого вдосконалення моніторингової системи можна виділити наступні пункти:

- створити модуль які буде оброблювати статистичні дані за допомогою AI (Artetificial intelligence) і робити передбачення на основі статистичних даних;
- добавити можливість інтегрувати всі графіки автоматично на сайт користувача системи;
- створити cron job, який буде періодично перевіряти чи запущена моніторингова система і якщо виникла помилка, то буде її перезапускати;
- добавити comandline interface для зручного налаштування системи і створення всіх необхідних файлів конфігурації.

3.10. Висновки до розділу 3

В цьому розділі було проаналізовано розроблену моніторингову систему. Було проаналізовано весь функціонал створеної моніторингової системи і показано налаштування за допомогою яких користувач може налаштувати систему для свого користування. Також було описано, які функціональні можливості потрібно буде добавити до системи.

Було розглянуто архітектуру розроблюваної моніторингової системи і створено схему використання і структурну схему системи. При описі архітектури системи було розглянуто архітектурну модель MVC, яка використовується в системі.

Перевагами даної моделі є:

- єдина концепція системи;
- незалежність елементів один від одного;
- легко протестувати окремі компоненти.

Також було показано структуру даних, яка використовується в системі для обрахунку статистичних показників і базується на принципі FIFO (first in first out). Перевагою даної структури даних є те, що вона використовує формули для обрахунку статистичних показників на основі попередніх обрахунків. Було показано схему бази даних на якій зображено всі сутності, які наявні в ситемі і описано тип даних для кожного поля сутності.

Важливим пунктом в даному розділі є пункт “Тестування системи”. В даному пункті були описані юніт тести і тести на продуктивність системи. За результатами даних тестів видно, що система добре протестована і може витримати значні навантаження. Велика кількість користувацьких метрик, які потрібно обробити не буде проблемою для створеної моніторингової системи.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Основною метою даного дипломного проєкту було створення моніторингової системи, яка буде відображати і обчислювала статистику по користувацьким метрикам в реальному часі і відправляти сповіщення при винекнення збоїв в додатках.

Було проведено аналіз проблем в існуючих системах моніторингу вебдодатків і було створенно функціональні та нефункціональні вимоги до розроблюваної моніторингової системи. На основі вимог до даної системи та аналізу існуючих програмних рішень для розробки було обрано основні засоби реалізації, а саме:

- платформа Node.js для створення серверної частини системи;
- база даних MongoDb;
- Grafana – сервіс для візуалізації графіків.

Користувачі розробленої моніторингової системи мають можливість переглядати всю необхідну статистичну інформацію для тестування та визначення найпопулярніших ресурсів, а також переглядати навантаження на сервер, на якому запускається їх додаток. За допомогою системних показників можна визначати і прогнозувати, які системні ресурси потрібні для коректної роботи веб-додатку.

Розроблена моніторингова система надає такі можливості:

- зберігання та відображення користувацьких метрик;
- обрахунок та відображення необхідних статистичних даних (мінімальних/максимальних значень, середнього значення і середнього квадратичного відхилення);

- відправка сповіщень в месенджер, якщо якась метрика перевищила задане значення;
- відображення системних метрик
- перегляд графіків за певний проміжок часу.

Автоматизована моніторингова система веб-середовища в IT компанії була протестована на продуктивність. За результатами даних тестів видно, що система може витримати значні навантаження. Велика кількість користувацьких метрик, які потрібно обробити не буде проблемою для створеної моніторингової системи.

Щоб вдосконалити дану моніторингову систему, необхідно створити модуль які буде оброблювати статистичні дані за допомогою AI і робити передбачення на основі статистичних даних;

Моніторингова система була створена відповідно до сформованих вимог і є універсальним рішенням для підтримки і адміністрування вебдодатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Automatic application monitoring. Ensure high availability of your applications and reduce downtime: Why and how to implement automated monitoring.» by Joe Loewengruber, IBM 2019. [Електронний ресурс]. - Режим доступу до ресурсу: https://www.ibm.com/garage/method/practices/manage/practice_automated_monitoring/
2. «Study on Monitoring of the Integrated Automation System with Web Technology» Bing Xin Sun, September 13 “Applied Mechanics and Materials” [Електронний ресурс]. - Режим доступу до ресурсу: https://www.researchgate.net/publication/272771106_Study_on_Monitoring_of_the_Integrated_Automation_System_with_Web_Technology
3. «Journal of Industrial Information Integration» Volume 9, March 2018, Pages 24-34 «Design and implementation of an automated network monitoring and reporting back system» Rafiullah Khan, Sarmad Ullah Khan. [Електронний ресурс]. - Режим доступу до ресурсу: https://pureadmin.qub.ac.uk/ws/portalfiles/portal/147406623/2017_Design_and_Implementation_of_an_Automated_Network_Monitoring_and_Reporting_Back_System.pdf
4. «Real Time Fault Monitoring of Industrial Processes» by Anastasios D. Pouliezos, G. Stavrakakis, February 1994. - Режим доступу до ресурсу: https://www.researchgate.net/publication/234791864_Real_Time_Fault_Monitoring_of_Industrial_Processes

5. «Global Information Architecture for Industrial Automation» by Birgit Vogel-Heuser, Gunther Kegel, Klaus Bender, Klaus Wucherer, January 2009. . - Режим доступу до ресурсу: https://www.researchgate.net/publication/263849223_Global_Information_Architecture_for_Industrial_Automation
6. Scott G. Chaplowe & J. Bradley Cousins. (2016). Monitoring and Evaluation Training: A Systematic Approach.Thousand Oaks, CA: Sage. 464
7. Effectiveness of Marine Protected Areas: A Guidebook for Monitoring and Evaluation. Marine Laboratory, Institute of Tropical Ecology, Leyte State University, Visca, Baybay, Leyte 6521-A, Philippines
8. «Проблема обробки великої кількості даних» [Електронний ресурс]. Режим доступу до ресурсу: https://www.jetinfo.ru/bolshie-dannye-bolshayaproblema/
9. Проблема швидкості загрузки веб-ресурсу [Електронний ресурс]. Режим доступу до ресурсу: https://netpeak.net/ru/blog/sayt-zagruzhayetsyamedlenno-kogda-ne-nuzhno-panikovat/
10. Балансування навантаження на сервері [Електронний ресурс]. Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BD%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B8
11. Проблеми налаштування серверу [Електронний ресурс]. Режим доступу до ресурсу: https://www.faq.in.ua/articles/26-problemy-z-serverom-shchorobyty.html
12. Мова програмування Java - [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java
13. Мова програмування Javascript - [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JavaScript

14. Мова програмування C# - [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp
15. Порівняння фреймворків для платформи Node.JS - [Електронний ресурс]. Режим доступу до ресурсу: https://umbrellait.com/ru/blog/choosing-the-best-nodejs-framework/
16. Бобик О. І., Берегова Г. І., Копитко Б. І. Теорія ймовірностей і математична статистика: Навч. підручник. 2006. – 440 с
17. Чернова Н. И. Математическая статистика: Учеб. пособие / Новосиб. гос. ун-т. Новосибирск, 2007. 148 с
18. Grafana - Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Grafana
19. Graphite - Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Graphite_(software)
20. John Bullinaria, Data Structures and Algorithms, UK Version of 27 March 2019, CA: 656 pages
21. AI [Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Artificial_intelligence
22. Cron - [Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cron

ДОДАТКИ

Додаток А

App

```
import
React, {
useState
} from
'react';
import Splash from './components/Splash';
import DashboardContainer from './containers/DashboardContainer';


const App: React.FC = React.memo(() => {

 // const [firstVisit, setFirstVisit] = useState(false);


 // splash demo
 const [firstVisit, setFirstVisit] = useState(true);


 // Splash image
 return firstVisit ? <Splash setFirstVisit={setFirstVisit} /> : <DashboardContainer />;
});


export default App;
```

Applications

```
import React, { useContext, useEffect, useState, useRef } from 'react';
import {
  IconButton,
  Grid,
  Modal,
  Card,
  CardHeader,
  CardContent,
  Typography,
} from '@material-ui/core';
import DeleteForeverOutlinedIcon from '@material-ui/icons/DeleteForeverOutlined';
import { makeStyles } from '@material-ui/core/styles';

import { DashboardContext } from '../context/DashboardContext';
import ServicesModal from '../modals/ServicesModal';
import '../stylesheets/Applications.scss';

type ClickEvent = React.MouseEvent<HTMLElement>;

const Applications: React.FC = React.memo((props) => {
  const { applications, getApplications, deleteApp } = useContext(DashboardContext);
  const [open, setOpen] = useState<boolean>(false);
  const [index, setIndex] = useState<number>(0);
  const [app, setApp] = useState<string>('');

  // Dynamic refs
  const delRef = useRef<any>([]);

  useEffect(() => {
    getApplications();
  }, []);

  // Ask user for deletetion confirmation
```

```
  const confirmDelete = (event: ClickEvent, app: string, i: number) => {
    const message = `The application '${app}' will be permanently deleted. Continue?`;
    if (confirm(message)) deleteApp(i);
  };

  // Handle clicks on Application cards
  const handleClick = (event: ClickEvent, selectedApp: string, i: number) => {
    if (delRef.current[i] && !delRef.current[i].contains(event.target)) {
      setIndex(i);
      setApp(selectedApp);
      setOpen(true);
    }
  };

  const useStyles = makeStyles(theme => ({
    // cards: myPostgres, myMongo, ToddDB
    paper: {
      height: 280,
      width: 280,
      textAlign: 'center',
      whiteSpace: 'nowrap',
      backgroundColor: '#ffffff',
      borderRadius: 3,
      border: '0',
      boxShadow: '0 6px 6px 0 rgba(153, 153, 153, 0.14), 0 6px 6px -2px rgba(153, 153, 153, 0.2), 0 6px 8px
0 rgba(153, 153, 153, 0.12)',
      '&:hover, &.Mui-focusVisible': {
        backgroundColor: `#3788fc`,
      },
      '&:active': {
        backgroundColor: `#3788fc`,
      },
    },
    iconbutton: {
      position: 'relative',
      bottom: 20,
```

```
    right: 47,
    boxShadow: 'none',
    '&:hover, &.Mui-focusVisible': { color: '#ffffff' },
    backgroundColor: 'transparent',
   },
   fontStyles: {
    color: '#444d56',
    fontSize: '22px',
    [theme.breakpoints.up('lg')]: {
     fontSize: '22px',
     fontFamily: 'Roboto',
     fontWeight: 100,
    },
   },
}));

const classes = useStyles();

return (
 <>
  {applications.map((app: string[], i: number | any | string | undefined) => (
   <Grid item lg={4} md={6} sm={12} key={i}>
    <div id="card-hover">
     <Card
      // key={`card-${i}`}
      className={classes.paper}
      variant="outlined"
      onClick={event => handleClick(event, app[0], i)}
     >
      <CardHeader
       avatar={
        <IconButton
         ref={element => (delRef.current[i] = element)}
         className={classes.iconbutton}
         aria-label="Delete"
         onClick={event => confirmDelete(event, app[0], i)}
```

```
            >
              <DeleteForeverOutlinedIcon />
            </IconButton>
            }
          ></CardHeader>
          <CardContent>
            <Typography className={classes.fontStyles}>{app[0]}</Typography>
          </CardContent>
        </Card>
      </div>
    </Grid>
  ))}
  <Modal open={open} onClose={() => setOpen(false)}>
    <ServicesModal i={index} app={app} />
  </Modal>
 </>
 );
});

export default Applications;
```

Setting

```
import React, {useContext, useEffect, useState} from 'react';
import '../stylesheets/Settings.scss';

// DASHBOARD CONTEXT
import { DashboardContext } from '../context/DashboardContext';

// Need to add flag to turn off the splash at start
// Need to add flag to turn off getting started page
// Need to add flag to turn on/off live data (ideally persist on restart)

//Typescript
type ClickEvent = React.MouseEvent<HTMLElement>;

const Settings: React.SFC = React.memo((props) => {
 //use context from Dash board regarding currentMode
 let { changeMode } = useContext(DashboardContext);
 const handleClick = (mode: string) => {
  changeMode(mode);
 }

 return (
  <div className="settings">
   <button className="mode" id="lightMode" onClick={() => handleClick("light mode")}>
    Light
   </button>

   <button className="mode" id="darkMode" onClick={() => handleClick("dark mode")}>
    Dark
   </button>
  </div>
 );
});

export default Settings;
```

Splash

```
import React, {useContext, useEffect, useState} from 'react';
import '../stylesheets/Settings.scss';

// DASHBOARD CONTEXT
import { DashboardContext } from '../context/DashboardContext';

// Need to add flag to turn off the splash at start
// Need to add flag to turn off getting started page
// Need to add flag to turn on/off live data (ideally persist on restart)

//Typescript
type ClickEvent = React.MouseEvent<HTMLElement>;

const Settings: React.SFC = React.memo((props) => {
  //use context from Dash board regarding currentMode
  let { changeMode } = useContext(DashboardContext);
  const handleClick = (mode: string) => {
    changeMode(mode);
  }

  return (
    <div className="settings">
      <button className="mode" id="lightMode" onClick={() => handleClick("light mode")}>
        Light
      </button>

      <button className="mode" id="darkMode" onClick={() => handleClick("dark mode")}>
        Dark
      </button>
    </div>
  );
});

export default Settings;
```

Header

```
import React, { useContext } from 'react';
import { useHistory, Link } from 'react-router-dom';
import ListIcon from '@material-ui/icons/List';
import { ApplicationContext } from '../context/ApplicationContext';
import { DashboardContext } from '../context/DashboardContext';
import '../stylesheets/Header.scss';
import {lightAndDark} from '../components/Styling';
export interface HeaderProps {
  app: string[];
  service: string;
  live: boolean;
  setLive: React.Dispatch<React.SetStateAction<boolean>>;
}

const Header: React.FC<HeaderProps> = React.memo(function Header({ app, service, setLive, live }) {
  const history = useHistory();

  const { servicesData } = useContext(ApplicationContext);
  const { mode } = useContext(DashboardContext);

  let currentModeCSS = (mode === 'light mode')? lightAndDark.lightModeHeader : lightAndDark.darkModeHeader;
  return (
    <div className="microservice-header" style={currentModeCSS}>
      <h1 className="microserviceTitle">{app}</h1>
      <select name="microservice" value={service} onChange={e => history.replace(e.target.value)}>
        {servicesData.map(({ _id, microservice }: any) => (
          // <option key={_id} value={`${microservice}`} selected={service === microservice}>
          <option key={_id} value={`${microservice}`}>
            {microservice}
          </option>
        ))}
        <option defaultValue='Select service'>
          communications
        </option>
        {/* <option value="communications" selected={service === 'communications'}>
          communications
        </option> */}
      </select>
      <div className="header">
        <Link className="link" id="return" to="/applications">
          <span>
            <ListIcon className="icon" id="returnIcon" />
          </span>
          <p id="returnToDash">Dashboard</p>
        </Link>
        {/* <button id="returnButton" onClick={() => history.goBack()}><ListIcon className="icon" id="returnIcon" /></button> */}
```

```
        <button onClick={() => setLive(!live)}>
          {live ? (
            <div>
              <span id="live">Live</span>
            </div>
          ) : (
            <div id="gatherLiveData">Gather Live Data</div>
          )}
        </button>
      </div>
    </div>
  );
});

export default Header;
```

Contact

```
import React, {useContext, useEffect, useState} from 'react';
import '../stylesheets/Contact.scss';
import { DashboardContext } from '../context/DashboardContext';
import { lightAndDark } from './Styling';

const Contact = React.memo((props) => {
 let { mode } = useContext(DashboardContext);

 let currentMode = (mode === 'light mode')? lightAndDark.lightModeText : lightAndDark.darkModeText;

 return (
  <div className="contact">
   <div className="contact-border">
    <div className="contact-container">
     <div className="contact-blurb">
      <h1 style={currentMode}>Contact Us</h1>
      <br />
      <p style={currentMode}>Please feel free to provide any feedback, concerns, or comments.</p>
      <p style={currentMode}>
       You can find issues the team is currently working on 
       <a    style={currentMode}  id="issueLink"  href="https://github.com/open-source-labs/Chronos/issues"
target="_blank">
        here
       </a>
       .
      </p>
     </div>
    </div>
    <div className="email-container">
     <form>
      <label style={currentMode} htmlFor="fname">First Name:  </label>
      <input type="text" id="fname" name="firstname" placeholder="Your name.." />
      <br />
      <label style={currentMode} htmlFor="lname">Last Name:  </label>
      <input type="text" id="lname" name="lastname" placeholder="Your last name.." />
      <br />
      <label style={currentMode} htmlFor="email">E-mail:  </label>
      <input type="text" id="email" name="email" placeholder="Your e-mail address.." />
      <br />
      <label style={currentMode} htmlFor="subject">Subject:  </label>
      <input type="text" id="subject" name="subject" placeholder="Subject" />
      <br />
      <label style={currentMode} id="messageLabel" htmlFor="message">Message: <span>
        <textarea id="message" name="message" placeholder="Write something.."></textarea>
       </span>
      </label>

      <br />
      <label style={currentMode} htmlFor="myfile">Select a file: </label>
```

```
          <input style={currentMode} type="file" id="myfile" name="myfile" accept="image/*"></input>
          <br />
          <input style={currentMode} id="contact-submit" type="submit" value="Submit" />
        </form>
      </div>
    </div>
  </div>
 );
});

export default Contact;
```

Home

```
import { faGlassMartiniAlt } from '@fortawesome/free-solid-svg-icons';
// import axios from 'axios';
import React, { useState } from 'react';
import { Link } from 'react-router-dom';

import '../stylesheets/Home.scss';
//
const { ipcRenderer } = window.require('electron');

interface PersonProps {
  email: string;
  password: string;
}

type ClickEvent = React.MouseEvent<HTMLElement>;

// const [email] = React.useState<string>('');
// const [password] = React.useState<string>('');
// function handleChange(e) {

// }
const adminUser = {
  email:  "admin@admin.com",
  password: "admin123"
}

const Home = React.memo(function Home(props) {
const [open, setOpen] = useState<boolean>(false);
const [loginInfo, setLoginInfo] = React.useState<PersonProps>({
  email: '',
  password: ''
});

function myFunction() {
  location.replace("/applications")
}
function pageRedirect(){
  setTimeout(function(){
    window.location.href = '/applications';
 }, 5000);
 }

function submitLogin(e: any) {
  e.preventDefault();
// console.log(e);
  console.log(loginInfo);
  //check local state if the username is there
   if(loginInfo.email === adminUser.email && loginInfo.password === adminUser.password){
```

```
    console.log("Logged In");
    myFunction();
   // alert('Welcome Back!\n\n (\\__/) \n (=\'.\'=) \n(\'\')__(\'\')')
   // //  pageRedirect();
   // //  "window.location='/applications'"
   } else {
    alert('Log In credentials are wrong!\n\n (\\__/) \n (=\'.\'=) \n(\'\')__(\'\')')
   }
  }
//database connection/fetch
function checkLoginStatus() {
  // axios('http://localhost:3000/loggedIn', {withCredentials: true})
}



  return (
    <div className="home">
     <p className="welcomeMessage">Welcome Back To Chronos! Your all-in-one application monitoring tool</p>

     <form className="form" onSubmit={submitLogin}>
          <label className="login">
            <input type="email" name="email" id="email" placeholder="your@email.here" onChange={e => setLoginInfo({...loginInfo, email: e.target.value})} ></input>
            <br></br>
            <input type="password" name= "password" id="password" placeholder="enter password" onChange={e => setLoginInfo({...loginInfo, password:e.target.value})} ></input>
            <hr />
          </label>
          <br></br>
          <br></br>
          <br></br>
         <button className="link"  id="submitBtn" type="submit" onClick={() => (true)}>
           Log In
          </button>
         </form>

     <br></br>
     {/* <Link className="link" to="/applications">
      Get Started
     </Link> */}
    </div>
  );
});

export default Home;
```

Package.js

```
import { faGlassMartiniAlt } from '@fortawesome/free-solid-svg-icons';
// import axios from 'axios';
import React, { useState } from 'react';
import { Link } from 'react-router-dom';

import '../stylesheets/Home.scss';
//
const { ipcRenderer } = window.require('electron');

interface PersonProps {
  email: string;
  password: string;
}

type ClickEvent = React.MouseEvent<HTMLElement>;

// const [email] = React.useState<string>('');
// const [password] = React.useState<string>('');
// function handleChange(e) {

// }
const adminUser = {
  email:  "admin@admin.com",
  password: "admin123"
}

const Home = React.memo(function Home(props) {
const [open, setOpen] = useState<boolean>(false);
const [loginInfo, setLoginInfo] = React.useState<PersonProps>({
  email: '',
  password: ''
});

function myFunction() {
  location.replace("/applications")
}
function pageRedirect(){
  setTimeout(function(){
    window.location.href = '/applications';
 }, 5000);
 }

function submitLogin(e: any) {
  e.preventDefault();
// console.log(e);
  console.log(loginInfo);
  //check local state if the username is there
   if(loginInfo.email === adminUser.email && loginInfo.password === adminUser.password){
```

```
    console.log("Logged In");
    myFunction();
   // alert('Welcome Back!\n\n (\\__/) \n (=\'.\'=) \n(\'\')__(\'\')')
   // //  pageRedirect();
   // //  "window.location='/applications'"
   } else {
    alert('Log In credentials are wrong!\n\n (\\__/) \n (=\'.\'=) \n(\'\')__(\'\')')
   }
  }
//database connection/fetch
function checkLoginStatus() {
  // axios('http://localhost:3000/loggedIn', {withCredentials: true})
}



  return (
    <div className="home">
     <p className="welcomeMessage">Welcome Back To Chronos! Your all-in-one application monitoring tool</p>

     <form className="form" onSubmit={submitLogin}>
          <label className="login">
            <input type="email" name="email" id="email" placeholder="your@email.here" onChange={e => setLoginInfo({...loginInfo, email: e.target.value})} ></input>
            <br></br>
            <input type="password" name= "password" id="password" placeholder="enter password" onChange={e => setLoginInfo({...loginInfo, password:e.target.value})} ></input>
            <hr />
          </label>
          <br></br>
          <br></br>
          <br></br>
         <button className="link"  id="submitBtn" type="submit" onClick={() => (true)}>
           Log In
          </button>
         </form>

     <br></br>
     {/* <Link className="link" to="/applications">
      Get Started
     </Link> */}
    </div>
  );
});

export default Home;
```

BookServer

```
const express = require('express');
const path = require('path');
const cors = require('cors');
const chronos = require('chronos-tracker');
require('./chronos-config');
const controller = require('./BookController.js');
require('dotenv').config();

// req
chronos.propagate();

const app = express();

app.use(express.json());
app.use('/', chronos.track());

app.use(cors());
app.use('/', express.static(path.resolve(__dirname, '../frontend')));

// TEST EXPRESS SERVER
app.use((req, res, next) => {
 console.log(
  `**********************************************************************************************
   FLOW TEST --- METHOD:${req.method},
   PATH: ${req.url},
   BODY: ${JSON.stringify(req.body)},
   ID: ${req.query.id}
   **********************************************************************************************`
 );
 next();
});

//  This route will create a new book!
app.post('/books/createbook', controller.createBook, (req, res) => {
 res.status(200).json(res.locals.createBook);
});

// This route will delete a book
app.delete('/books/deletebook:id?', controller.deleteBook, (req, res) => {
 res.status(200).json(res.locals.deleteBook);
});

// This route will get all the books in the database
app.get('/books/getbooks', controller.getBooks, (req, res) => {
 res.status(200).json(res.locals.getBooks);
});

// This route gets orders from the Orders application
```

```
app.get('/books/getordersinfo', controller.getorderinfo, (req, res) => {
  res.status(200).json(res.locals.getorderinfo);
});

// Global error handler
app.use((error, req, res, next) => {
  const defaultErr = {
    log: 'Express error handler caught unknown middleware error',
    status: 400,
    message: { err: 'An error occurred' },
  };
  const errorObj = Object.assign(defaultErr, error);
  console.log(`Here is the error object's response: ${errorObj.log}`);

  res.status(errorObj.status).json(errorObj.message);
});

app.listen(8888, () => {
  console.log(`Book server running on port 8888...`);
});
```

BookModel

```
const mongoose = require('mongoose');
require('dotenv').config();

//  pull schema from the mongoose object
const { Schema } = mongoose;

// DB link for books data.
// Search dotenv documentation for details
const book_db_URI = `${process.env.BOOKS_DB}`;

// const URI = process.env.MONGO_URI || myURI;

// connect the database, if error, log will be sent to the terminal
mongoose
  .connect(book_db_URI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('Connected!!!********* Books Database is live!!!'))
  .catch((err) => console.log('Connection Error ', err));

//  Schema for the database
const BooksSchema = new Schema({
  title: {
    type: String,
    required: true,
  },
  author: {
    type: String,
    required: true,
  },
  numberOfPages: {
    type: Number,
    required: false,
  },
  publisher: {
    type: String,
    required: false,
  },
});

// Database Model creation to be exported
const BookModel = mongoose.model('BookModel', BooksSchema);

module.exports = BookModel;
```

BookController

```
require('dotenv').config();
const fetch = require('node-fetch');
const BookModel = require('./BookModel');

const BookController = {};

// This controller creates a book in the book db
BookController.createBook = (req, res, next) => {
  const { title, author, numberOfPages, publisher } = req.body;
  BookModel.create(
    {
      title,
      author,
      numberOfPages,
      publisher,
    },
    (err, result) => {
      if (err) {
        console.log(`This is the error I am getting back ${err}`);
        return res.send(404).json(err);
      }

      res.locals.createBook = result;
      return next();
    }
  );
};

// This controller creates a book in the book db

BookController.getBooks = (req, res, next) => {
  BookModel.find({}, (err, result) => {
    if (err) {
      console.log('Book retrieval was not successful', err);
      return res.status(404).json(err);
    }
    res.locals.getBooks = result;
    // console.log('Book retrieval was successful', res.locals.getBooks);
    return next();
  });
};

// This controller deletes books
BookController.deleteBook = (req, res, next) => {
  const { id } = req.query;
  BookModel.findOneAndDelete({ _id: id }, (error, result) => {
    if (error) {
      console.log(`Deletion was not successful ${error}`);
```

```
    return res.status(404).json(error);
   }
   res.locals.deleteBook = result;
   return next();
  });
};

//  This controller gets order info from the order application
BookController.getorderinfo = (req, res, next) => {
  // console.log('req.headers (in bookController.js):', req.headers);
  fetch(`http://orders:7777/orders/getorders`, {
   method: 'GET',
   headers: {
    'Content-Type': 'Application/JSON',
    Accept: 'application/json',
   },
  })
   .then((response) => response.json())
   .then((results) => {
    res.locals.getorderinfo = results;
    return next();
   })
   .catch((error) => {
    console.log(`There was an error in getting orders data ${error}`);
   });
};
module.exports = BookController;
```

Style.css

```
.header
{
background:#333;
color:gray;
}

.footer
{
background:#333;
color:gray;
text-align:center;
}

.table{
        overflow:auto;
}

.navbar-inverse {
  background-color: #333;
  border-color: transparent;
}

.navbar {
  border: 0px solid #fff;
  min-height: 40px;
  margin-bottom: 0px;
}

.navbar-inverse .navbar-nav > li > a {
  color: #FFF;
```

```
}

.navStyle {
        float: center;
}

.navStyle ul {
    list-style: none;
    padding: 0px;
}

.navStyle ul li {
    display: inline-block;
    margin: 0 0px;
}

.navStyle ul li:first-child {
        /*margin:0px;*/;
}

.navStyle ul li:last-child {
    margin: 0 0 0 0px;
}

.navStyle ul li a {
    display: block;
    font-size: 14px;
    color: #222222;
    font-family: 'Raleway', sans-serif;
    text-decoration: none;
    text-transform: uppercase;
    font-weight: 600;
    transition: all 0.2s ease-in-out;
```

```
    -moz-transition: all 0.2s ease-in-out;
    -webkit-transition: all 0.2s ease-in-out;
}

.navStyle ul li a:hover {
    color: #005490;
}

.navStyle > li > a:hover, .nav > li > a:focus {
    text-decoration: none;
    color: #005490;
    background-color: #111;
}

.navbar-inverse .navbar-nav > li > a:hover, .navbar-inverse .navbar-nav > li > a:focus {
    color: #005490;
    background-color: transparent;
}

.navbar-inverse .navbar-nav > .active > a, .navbar-inverse .navbar-nav > .active > a:hover, .navbar-inverse
.navbar-nav > .active > a:focus {
    color: #fff;
        background-color: #005490;
}

.navStyle > li.active > a,
.navStyle > li.active > a:hover {
    text-decoration: none;
    color: #005490;
    background-color: transparent;
}

.borderLeft {
    border-left: 1px solid #DADADA;
```

```
}

.borderTop {
   margin-top: 30px;
   border-top: 1px solid #DADADA;
}

.mrgTop {
   margin-top: 30px;
}




h1,h2,h3,h4,h5
{
        color:#005490;

}

.box {
  width: 20%;
  margin: 0 auto;
  background: rgba(255,255,255,0.2);
  padding: 35px;
  border: 2px solid #fff;
  border-radius: 20px/50px;
  background-clip: padding-box;
  text-align: center;
}

.overlay {
```

```
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  background: rgba(0, 0, 0, 0.7);
  transition: opacity 500ms;
  visibility: hidden;
  opacity: 0;
 }
 .overlay:target {
  visibility: visible;
  opacity: 1;
 }

 .popup {
  margin: 70px auto;
  padding: 20px;
  background: #fff;
  border-radius: 5px;
  width: 30%;
  position: relative;
  transition: all 5s ease-in-out;
 }

 .popup h2 {
  margin-top: 0;
  color: #333;
  font-family: Tahoma, Arial, sans-serif;
 }
 .popup .close {
  position: absolute;
  top: 20px;
```

```
  right: 30px;
  transition: all 200ms;
  font-size: 30px;
  font-weight: bold;
  text-decoration: none;
  color: #333;
}
.popup .close:hover {
  color:blue;
}
.popup .content {
  max-height: 30%;
  overflow: auto;
}

@media only screen and (max-width:1222px)
{
.box
{
width: 70%;
}
.popup
{
width: 70%;
}
#nav
{
width:100%;
margin:0px auto;
background-color:gray;
}

#nav ul
```

```
{
list-style:none;
}

#nav li
{
clear:both;'
padding:10px;
border-left:none;
border-top:1px solid gray;
}

#nav li:first-child
{
border-top:none;
}

#nav li a
{
text-decoration:none;
color:white;
display:block;
}

#nav a:hover
{
background:black;
color:white;
display:block;
}
}

@media only screen and (max-width:700px)
```

```
{
.box
{
width: 70%;
}
.popup
{
width: 70%;
}
#nav
{
width:100%;
margin:0px auto;
background-color:gray;
}

#nav ul
{
list-style:none;
}

#nav li
{
clear:both;'
padding:10px;
border-left:none;
border-top:1px solid gray;
}

#nav li:first-child
{
border-top:none;
}
```

```
#nav li a
{
text-decoration:none;
color:white;
display:block;
}

#nav a:hover
{
background:black;
color:white;
display:block;
}

.normal
{
width:100%;
margin:0px auto;
border-collapse:collapse;
margin-bottom:12px;
}
.normal tr, td
{
padding-top:6px;
}

.normal tr
{

}

.normal td, th
```

```
{
float:left;
vertical-align:top;
}

.normal td:before
{
top:6px;
}
}
```

Додаток П

AdminHeader

```
<div class="container-fluid header">
<div id="nav">
<nav class="navbar navbar-inverse" role="navigation">
<div class="navbar-header">
<label><a href="home.php">LATCH STORES</a></label>
<button type="button" id="nav-toggle" class="navbar-toggle" data-toggle="collapse" data-target="#main-nav"> <span class="sr-only">Toggle navigation</span> <span class="icon-bar"></span> <span class="icon-bar"></span> <span class="icon-bar"></span> </button>
</div>
<div id="main-nav" class="collapse navbar-collapse navStyle">
<ul>
        <li><a href="homepage.php">HOME</a></li>
        <li><a href="managesettings.php">MANAGE SETTINGS</a></li>
        <li><a href="manageitems.php">MANAGE ITEMS</a></li>
        <li><a href="manageorders.php">MANAGE ORDERS</a></li>
        <li><a href="managepayments.php">MANAGE PAYMENTS</a></li>
        <li><a href="managemessages.php">MANAGE MESSAGES</a></li>
        <li><a href="manageusers.php">MANAGE USERS</a></li>
</ul>
</div>
</nav>
</div>
</div>
```

Додаток Р

SideBar

```
<section id="reservation-form" class="mt50 clearfix">
 <div class="col-sm-12 col-md-4">
  <form class="reservation-vertical clearfix" role="form" method="post" name="reservationform" id="reservationform">
   <h2 class="lined-heading"><span>Dashboard</span></h2>
   <div class="price">

   <?php if($credit_permit == 1){ ?>

    <h4>Balance:</h4>
    <?php
        require_once('database/topfile.php');
        echo $bal;
    ?>
    <br>

   <span> <a href="#" data-toggle="modal" data-target="#topup"><i class="fa fa-plus"></i> top up</a></span></div>

   <?php } ?>

   <?php if($open_msg == 1){ ?>

   <a href="#" data-toggle="modal" data-target="#myModal" type="submit" class="btn btn-primary btn-block"><i class="fa fa-paper-plane"></i> Open Message</a><br>

   <?php } ?>

   <div style="border:1px dotted #dcdcdc; text-align:center; padding:10px; border-radius:1px;">
    <h4>Sender Id :<?php if($as_sender_id == ""){echo "SMSLEOPARD";}else{echo $as_sender_id;} ?></h4>
```

```
    </div>


  </form>


  <!-- Modal -->
  <div class="modal fade mt100" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
          <h4 class="modal-title" id="myModalLabel">Open Message</h4>
        </div>
        <div class="modal-body">
          <div class="form-group">
            <label for="openphonenumber" accesskey="E">Phone Number</label>
            <input name="openphonenumber" type="text" id="openphonenumber" value="" class="form-control" placeholder="Please enter phone number" />
          </div>
          <div class="form-group">
            <label for="phonenumber" accesskey="E">Message</label>
            <textarea name="message" id="openmessage" class="form-control" placeholder="Please enter message"></textarea>
          </div>
          <div id="openmessagetext"></div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
          <button type="button" id="sendopenmessage" class="btn btn-primary">Send Message</button>
        </div>
      </div>
    </div>
  </div>
```

```
 <!-- Modal -->
<div class="modal fade mt100" id="topup" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>
        <h4 class="modal-title" id="myModalLabel">How to top up</h4>
      </div>
      <div class="modal-body">
        1. Go to MPESA Menu<br>
        2. Select Lipa Na MPESA Paybill<br>
        3. Enter Business Number <b style="color:#80D7E1;">525900</b><br>
        4. Enter Account Number <b style="color:#80D7E1;"><?php echo $as_username; ?>.api</b><br>
        5. Enter Amount<br>
        6. Enter MPESA PIN<br>
        7. Send<br><br>
        Incase of any problems, contact Africas Talking support team at <b>info@africastalking.com</b>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>


  </div>
 </section>
```

ProcessUpload

```
<?php
############ Configuration ##############
$thumb_square_size        = 200; //Thumbnails will be cropped to 200x200 pixels
$max_image_size           = 500; //Maximum image size (height and width)
$thumb_prefix             = "thumb_"; //Normal thumb Prefix
$destination_folder       =  'C:/wamp/www/seedlingscode/seedlingscode/admin/uploads/';  //upload directory ends with / (slash)
$jpeg_quality             = 90; //jpeg quality
$seedlings = $_POST['seedlings'];
##########################################


//continue only if $_POST is set and it is a Ajax request
if(isset($_POST) && isset($_SERVER['HTTP_X_REQUESTED_WITH']) && strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest'){

    // check $_FILES['ImageFile'] not empty
    if(!isset($_FILES['image_file']) || !is_uploaded_file($_FILES['image_file']['tmp_name'])){
                die('Image file is Missing!'); // output error when above checks fail.
    }

    //uploaded file info we need to proceed
    $image_name = $_FILES['image_file']['name']; //file name
    $image_size = $_FILES['image_file']['size']; //file size
    $image_temp = $_FILES['image_file']['tmp_name']; //file temp

    $image_size_info    = getimagesize($image_temp); //get image size

    if($image_size_info){
        $image_width        = $image_size_info[0]; //image width
        $image_height       = $image_size_info[1]; //image height
        $image_type         = $image_size_info['mime']; //image type
```

```
}else{
        die("Make sure image file is valid!");
}

//switch statement below checks allowed image type
//as well as creates new image from given file
switch($image_type){
        case 'image/png':
                $image_res =  imagecreatefrompng($image_temp); break;
        case 'image/gif':
                $image_res =  imagecreatefromgif($image_temp); break;
        case 'image/jpeg': case 'image/pjpeg':
                $image_res = imagecreatefromjpeg($image_temp); break;
        default:
                $image_res = false;
}

if($image_res){
        //Get file extension and name to construct new file name
        $image_info = pathinfo($image_name);
        $image_extension = strtolower($image_info["extension"]); //image extension
        $image_name_only = strtolower($image_info["filename"]);//file name only, no extension

        //create a random name for new image (Eg: fileName_293749.jpg) ;
        $new_file_name = $image_name_only. '_' .  rand(0, 9999999999) . '.' . $image_extension;

        //folder path to save resized images and thumbnails
        $thumb_save_folder      = $destination_folder . $thumb_prefix . $new_file_name;
        $image_save_folder      = $destination_folder . $new_file_name;

        //call normal_resize_image() function to proportionally resize image
        if(normal_resize_image($image_res, $image_save_folder, $image_type, $max_image_size,
$image_width, $image_height, $jpeg_quality))
        {
```

```
                    //call crop_image_square() function to create square thumbnails
                    if(!crop_image_square($image_res, $thumb_save_folder, $image_type, $thumb_square_size, $image_width, $image_height, $jpeg_quality))
                    {
                        die('Error Creating thumbnail');
                    }

                    /* We have succesfully resized and created thumbnail image
                    We can now output image to user's browser or store information in the database*/
                    echo '<div align="center">';
                    echo '<img src="uploads/'.$thumb_prefix . $new_file_name.'" alt="Thumbnail">';
                    //echo '<br />';
                    //echo '<img src="uploads/folder/'. $new_file_name.'" alt="Resized Image">';
                    echo '</div>';


                    mysql_connect("localhost","root","") or die('cannot connect to the server');
                    mysql_select_db("besttreeseedlings") or die('database selection problem');
                    $sql="INSERT INTO picturedetails(seedid,pname,pnameold,type,size) VALUES('$seedlings','$thumb_prefix$new_file_name','$new_file_name','$image_type','$image_size')";
                    mysql_query($sql) or die(mysql_error());
                }

                imagedestroy($image_res); //freeup memory
        }
}

#####  This function will proportionally resize image #####
function normal_resize_image($source, $destination, $image_type, $max_size, $image_width, $image_height, $quality){

        if($image_width <= 0 || $image_height <= 0){return false;} //return false if nothing to resize

        //do not resize if image is smaller than max size
```

```
	if($image_width <= $max_size && $image_height <= $max_size){
		if(save_image($source, $destination, $image_type, $quality)){
			return true;
		}
	}

	//Construct a proportional size of new image
	$image_scale	= min($max_size/$image_width, $max_size/$image_height);
	$new_width		= ceil($image_scale * $image_width);
	$new_height		= ceil($image_scale * $image_height);

	$new_canvas		= imagecreatetruecolor( $new_width, $new_height ); //Create a new true color image

	//Copy and resize part of an image with resampling
	if(imagecopyresampled($new_canvas, $source, 0, 0, 0, 0, $new_width, $new_height, $image_width, $image_height)){
		save_image($new_canvas, $destination, $image_type, $quality); //save resized image
	}

	return true;
}

##### This function corps image to create exact square, no matter what its original size! ######
function crop_image_square($source, $destination, $image_type, $square_size, $image_width, $image_height, $quality){
	if($image_width <= 0 || $image_height <= 0){return false;} //return false if nothing to resize

	if( $image_width > $image_height )
	{
		$y_offset = 0;
		$x_offset = ($image_width - $image_height) / 2;
		$s_size = $image_width - ($x_offset * 2);
	}else{
```

```
            $x_offset = 0;
            $y_offset = ($image_height - $image_width) / 2;
            $s_size = $image_height - ($y_offset * 2);
        }
        $new_canvas    = imagecreatetruecolor( $square_size, $square_size); //Create a new true color image

        //Copy and resize part of an image with resampling
        if(imagecopyresampled($new_canvas, $source, 0, 0, $x_offset, $y_offset, $square_size, $square_size, $s_size, $s_size)){
            save_image($new_canvas, $destination, $image_type, $quality);
        }

        return true;
}

##### Saves image resource to file #####
function save_image($source, $destination, $image_type, $quality){
        switch(strtolower($image_type)){//determine mime type
            case 'image/png':
                imagepng($source, $destination); return true; //save png file
                break;
            case 'image/gif':
                imagegif($source, $destination); return true; //save gif file
                break;
            case 'image/jpeg': case 'image/pjpeg':
                imagejpeg($source, $destination, $quality); return true; //save jpeg file
                break;
            default: return false;
        }
}
?>
```