НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра телекомунікацій

До захисту допущено:

Завідувач кафедри

_____ Сергій КРАВЧУК

«___»_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія та програмування інфокомунікацій»

спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Розробка веб-застосунку для обчислення підмереж IP»

Виконав:
студент IV курсу, групи ТЗ-01
Кухтій Олександр Сергійович _____

Керівник:
Доцент кафедри ТК НН ІТС, к.т.н., с.н.с.,
Міночкін Дмитро Анатолійович _____

Рецензент:
Доцент кафедри ІТТ, к.т.н., доцент,
Новогрудська Ріна Леонідівна _____

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

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут телекомунікаційних систем
Кафедра телекомунікацій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інженерія та програмування інфокомунікацій»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій КРАВЧУК
«___»_____2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Кухтію Олександру Сергійовичу

1. Тема роботи «Розробка веб-застосунку для обчислення підмереж IP», керівник роботи Міночкін Дмитро Анатолійович, к.т.н., с.н.с., затверджені наказом по університету від «22» травня 2024 р. № 2064-с.

2. Термін подання студентом роботи «10» червня 2024 р.

3. Вихідні дані до роботи: Інформаційні матеріали про мережі та створення веб-застосунків. Структурований план розробки матеріалів дипломної роботи.

4. Зміст роботи:
-Розглянути математичні принципи розрахунку підмереж.
-Здійснити огляд існуючих інструментів для обчислення підмереж IP.
-Проаналізувати використання технологій HTML, CSS, JavaScript, React, Redux, TypeScript та Tailwind для розробки веб-застосунку.
-Вивчити можливості Firebase для забезпечення бекенду: авторизація користувачів та збереження даних.
-Розробити та реалізувати інтерфейс користувача для введення даних та відображення результатів розрахунків.
-Провести тестування застосунку, включаючи модульне тестування, інтеграційне тестування та валідацію даних.
-Оцінити продуктивність та зручність використання веб-застосунку.

-Підготувати технічну документацію та інструкції для користувачів.
-Підготуватися до захисту дипломної роботи, включаючи створення презентації та доповіді.

5. Перелік ілюстративного матеріалу:

Слайд №1 Тема роботи

Слайд №2 Вступ

…

Слайд №10 Висновок

6. Дата видачі завдання «10» жовтня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Створення, оформлення, погодження та затвердження технічного завдання для роботи.	10.10.2023-25.12.2023	Виконано
2	Аналіз інформаційних ресурсів, пошук і вивчення відповідної науково-технічної літератури.	05.01.2024 - 25.01.2024	Виконано
3	Вибір відповідних технологій для розробки веб-застосунку: HTML, CSS, JavaScript, React, Redux, TypeScript, Tailwind, Firebase.	26.01.2024 – 31.01.2024	Виконано
4	Проектування архітектури системи та розробка прототипів користувацького інтерфейсу.	01.02.2024 - 20.02.2024	Виконано
5	Реалізація серверної частини застосунку, налаштування Firebase для зберігання даних та авторизації користувачів.	21.02.2024 - 24.03.2024	Виконано
6	Розробка клієнтської частини застосунку: створення сторінок сайту, реалізація логіки обчислення підмереж, інтеграція фронтенду з бекендом.	26.03.2024 - 28.04.2024	Виконано
7	Проведення тестування та налагодження веб-застосунку: модульне тестування, інтеграційне тестування, валідація форм, оптимізація продуктивності.	29.04.2024 - 11.05.2024	Виконано
8	Оформлення технічної документації, створення інструкцій для користувачів та підготовка звіту про виконану роботу.	12.05.2024 - 25.05.2024	Виконано
9	Підготовка до захисту дипломної роботи: створення презентації, підготовка доповіді.	26.05.2024 - 05.06.2024	Виконано

Студент Олександр КУХТІЙ

Керівник Дмитро МІНОЧКІН

РЕФЕРАТ

Пояснювальна записка: 61 сторінок, 5 рисунків, 25 джерел.

Мета роботи полягає у створенні веб-застосунку для спрощення процесу розбиття IP-мереж на підмережі відповідно до заданих вимог щодо кількості хостів.

У роботі розглянуто основні поняття комп'ютерних мереж, процес розбиття мережі на підмережі (subnetting) та концепцію VLSM. Проаналізовано існуючі рішення калькуляторів підмереж та визначено їх переваги і недоліки.

Для розробки веб-застосунку використано бібліотеки React, Redux, Firebase та сучасні технології веб-розробки. Реалізовано інтерфейс користувача, алгоритм розрахунку підмереж, авторизацію користувачів та збереження історії розрахунків.

Розроблений застосунок дозволяє автоматизувати процес обчислення підмереж IP відповідно до заданих вимог, візуалізувати результати та зберігати історію обчислень для кожного користувача.

Ключові слова: IP-МЕРЕЖА, ПІДМЕРЕЖА, SUBNETTING, VLSM, ВЕБ-ЗАСТОСУНОК, REACT, REDUX, FIREBASE.

ABSTRACT

Explanatory note: 61 pages, 5 figures, 25 sources.

The aim of the work is to create a web application to simplify the process of dividing IP networks into subnets according to the specified requirements for the number of hosts.

The work covers the basic concepts of computer networks, the process of dividing a network into subnets (subnetting), and the VLSM concept. Existing subnet calculator solutions are analyzed, and their advantages and disadvantages are identified.

For the development of the web application, React, Redux, Firebase libraries and modern web development technologies are used. The user interface, subnet calculation algorithm, user authentication, and calculation history saving are implemented.

The developed application allows automating the process of calculating IP subnets according to specified requirements, visualizing the results, and saving the calculation history for each user.

Keywords: IP NETWORK, SUBNET, SUBNETTING, VLSM, WEB APPLICATION, REACT, REDUX, FIREBASE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	Application Programming Interface	прикладний програмний інтерфейс
CRUD	Create, Read, Update, Delete	створення, читання, оновлення, видалення
CSS	Cascading Style Sheets	каскадні таблиці стилів
HTML	HyperText Markup Language	мова гіпертекстової розмітки
IP	Internet Protocol	інтернет-протокол
JS	JavaScript	мова програмування
SPA	Single Page Application	одностраничне веб-застосування
VLSM	Variable Length Subnet Mask	змінна довжина маски підмережі

ВСТУП

Комп'ютерні мережі відіграють важливу роль у сучасному світі, забезпечуючи ефективний обмін даними та зв'язок між різними пристроями та користувачами. Одним з ключових аспектів функціонування мереж є належне розподілення та використання IP-адрес. IP-адреса (Internet Protocol) – це унікальний ідентифікатор, який присвоюється кожному пристрою, під'єднаному до мережі, і дозволяє йому взаємодіяти з іншими вузлами.

Проте, з постійним зростанням кількості підключених пристроїв, виникає потреба в ефективному управлінні обмеженими ресурсами IP-адрес. Для цього використовується процес, відомий як розбиття мережі на підмережі (subnetting). Цей процес дозволяє розділити одну велику мережу на менші логічні підмережі, кожна з яких має власний діапазон IP-адрес. Це забезпечує більш ефективне використання доступних адрес та полегшує управління мережею.

Концепція змінної довжини маски підмережі (VLSM) додатково покращує цей процес, дозволяючи використовувати маски підмереж різної довжини для оптимального розподілу адрес відповідно до потреб кожної окремої підмережі. Це дозволяє уникнути марнотратного використання адресного простору та забезпечити належну масштабованість мережі.

Процес розбиття мережі на підмережі та визначення відповідних масок підмереж може бути досить складним і трудомістким, особливо для великих мереж з різноманітними вимогами до кількості хостів у кожній підмережі. Це часто вимагає ретельних розрахунків та аналізу, що може призвести до помилок та неефективного використання ресурсів.

Саме тому виникає потреба в зручному та ефективному інструменті, який би автоматизував процес обчислення підмереж та допоміг би спростити цю задачу. Веб-застосунок для обчислення підмереж IP є одним з таких рішень, що дозволяє користувачам легко та зручно виконувати необхідні розрахунки та отримувати детальну інформацію про розподіл мережі на підмережі.

Розроблений веб-застосунок пропонує інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам вводити необхідні параметри, такі як початкова IP-адреса, маска мережі та вимоги до кількості хостів у кожній підмережі. На основі цих даних, застосунок виконує необхідні обчислення та візуалізує результати у зручному форматі, включаючи таблиці, діаграми та детальну інформацію про кожну отриману підмережу.

Одним з ключових аспектів розробленого веб-застосунку є використання сучасних технологій веб-розробки, таких як React, Redux та Firebase. React – це популярна бібліотека JavaScript для створення користувацьких інтерфейсів, яка забезпечує ефективну роботу та гарну продуктивність. Redux – це патерн управління станом застосунку, який допомагає організувати та керувати даними в додатку. Firebase – це хмарна платформа від Google, яка забезпечує зручну інтеграцію бекенд-сервісів, таких як база даних, авторизація користувачів та хостинг. Використання цих технологій дозволило створити сучасний, масштабований та ефективний веб-застосунок, який відповідає високим стандартам якості та забезпечує зручний користувацький досвід.

Крім основної функціональності обчислення підмереж, веб-застосунок також пропонує додаткові можливості, такі як авторизація користувачів та збереження історії розрахунків. Завдяки інтеграції з Firebase, користувачі можуть створювати власні облікові записи та зберігати свої попередні розрахунки у хмарній базі даних. Це дозволяє легко відстежувати та переглядати історію обчислень з будь-якого пристрою, що є зручним для подальшого аналізу та планування мережевої інфраструктури.

Розроблений веб-застосунок для обчислення підмереж IP має широкий спектр застосувань та може бути корисним для різних категорій користувачів. Мережеві адміністратори та фахівці з IT-інфраструктури можуть використовувати його для ефективного планування та розподілу IP-адрес у великих корпоративних мережах. Студенти та викладачі у сфері комп'ютерних мереж можуть використовувати застосунок як навчальний інструмент для кращого розуміння процесу розбиття мереж та практичного застосування

набутих знань. Крім того, веб-застосунок може бути корисним для малих та середніх підприємств, які потребують налаштування власних локальних мереж, а також для домашніх користувачів, які бажають оптимізувати використання IP-адрес у своїх домашніх мережах.

Загалом, розробка веб-застосунку для обчислення підмереж IP є актуальною та важливою задачею, яка може полегшити роботу фахівців у сфері комп'ютерних мереж, підвищити ефективність використання ресурсів IP-адрес та сприяти кращому розумінню концепцій розбиття мереж на підмережі. Використання сучасних технологій веб-розробки та хмарних сервісів забезпечує високу якість, зручність та масштабованість розробленого застосунку, що робить його цінним інструментом у сфері мережевих технологій.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ

1.1. Основні поняття комп'ютерних мереж

Комп’ютерна мережа — сукупність пристроїв, з’єднаних каналами передавання даних, для спільного користування апаратними, програмними та інформаційними ресурсами під керуванням спеціального програмного забезпечення.

Комп’ютерні мережі призначені для:

- швидкого обміну даними між окремими комп’ютерами даних;
- віддаленого керування комп’ютерами;
- спільного доступу до периферійних пристроїв.

У комп’ютерній мережі комп’ютери можуть виконувати різні функції. Комп’ютер, який керує розподілом ресурсів мережі, називають сервером (від англ. server — той, хто подає); комп’ютери, які користуються ресурсами мережі, називають клієнтами, або робочими станціями.

Рис. 1.1 Класифікація комп’ютерних мереж

У загальному випадку під інформаційною мережею будемо розуміти сукупність територіально розосереджених кінцевих систем і об'єднуючої їх телекомунікаційної мережі, що забезпечує доступ прикладних процесів будь-якої з цих систем до всіх ресурсів інформаційної мережі і їхнє спільне використання. Поняття «інформаційна мережа», на відміну від поняття «телекомунікаційна мережа», є більш узагальненим і відображає множину інформаційних процесів, які протікають в мережі [6]. Ці процеси виникають у

результаті взаємодії кінцевих систем, під'єднаних до телекомунікаційної мережі. Інформаційні мережі призначені для надання користувачам послуг, пов'язаних з обміном інформацією, її споживанням, а також обробкою, зберіганням і накопиченням.

Рис 1.2. Інформаційна мережа

Споживач інформації, що одержав доступ до інформаційної мережі, стає її користувачем (User). Користувачами можуть бути як фізичні, такі юридичні особи (фірми, організації, підприємства). Телекомунікаційна мережа, якій і належить, у складі інформаційної мережі виконує функції транспортувальної системи. Інформаційна мережа (Information Network) – системоутворювальна сукупність територіально розосереджених кінцевих систем, об'єднаних телекомунікаційною мережею, за допомогою якої забезпечується взаємодія прикладних процесів, що активізуються в кінцевих системах, і колективний доступ до їх інформаційних і обчислювальних ресурсів. Базовим компонентом, ядром інформаційної мережі, є телекомунікаційна мережа [9]. Інформаційні технології (Information Technologies) – методи і способи накопичення, обробки, зберігання, відображення, пошуку і забезпечення цілісності інформації.

1.1.1. IP-адреси та маски підмереж

IP-адреса (Internet Protocol Address) є унікальним ідентифікатором, який присвоюється кожному пристрою в мережі. Вона складається з 32-бітового числа, яке зазвичай записується у вигляді чотирьох десяткових чисел, розділених крапками (наприклад, 192.168.0.1). IP-адреси використовуються для ідентифікації та маршрутизації даних у мережі. IP-адреса складається з двох частин – ідентифікатор мережі (префікс мережі, Network ID) та ідентифікатор вузла (номер пристрою, Host ID). Така схема призводить до дворівневої адресної ієрархії. Структура IP-адреси зображено на рис. 1.3

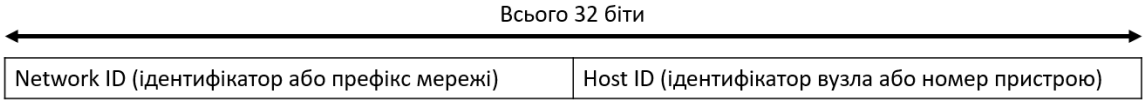

Рис. 1.3. Структура IP-адреси

Маска підмережі є додатковим параметром, який визначає, яка частина IP-адреси відповідає за ідентифікацію мережі, а яка – за ідентифікацію вузла (хоста) у цій мережі. Маска підмережі також представлена у вигляді 32-бітового числа, яке зазвичай записується у десятковому форматі з крапками (наприклад, 255.255.255.0).

Для кращого розуміння, розглянемо приклад IP-адреси 192.168.1.10 з маскою підмережі 255.255.255.0. У цьому випадку перші 24 біти (192.168.1) відповідають за ідентифікацію мережі, а решта 8 бітів (10) – за ідентифікацію вузла в цій мережі. Таким чином, усі пристрої з IP-адресами, що починаються з 192.168.1, належать до однієї мережі.

Маски підмереж використовуються для ефективного розподілу IP-адрес та створення логічних підмереж у великих мережах. Це дозволяє краще організувати трафік, забезпечити безпеку та полегшити управління мережевими ресурсами. Розподіл IP-адреси на мережі та підмережі здійснюється за допомогою побітової операції логічного "І" між IP-адресою та маскою підмережі. Ця операція залишає тільки ті біти, де в обох значеннях є одиниці, в інших бітах встановлюються нулі. Результат цієї операції - це IP-адреса мережі або підмережі.

Наприклад, якщо маємо IP-адресу 192.168.1.8 та маску підмережі 255.255.255.0, то виконуючи операцію "I" отримуємо:

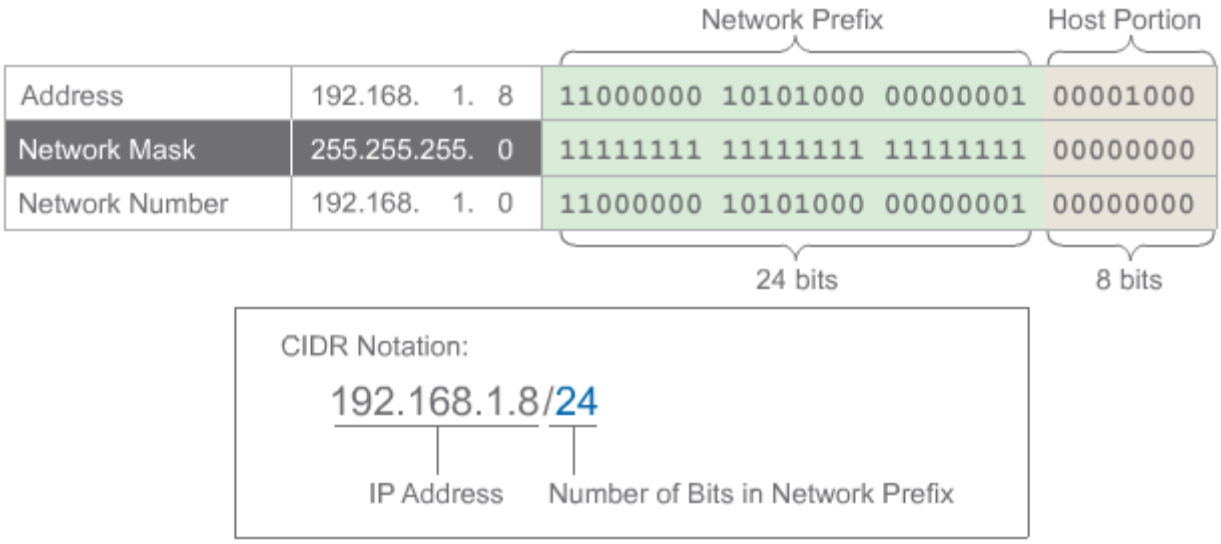

Рис. 1.4. Визначення адреси мережі даної адреси та маски.

Таким чином, IP-адреса 192.168.1.8 належить до мережі 192.168.1.0.

1.1.2. Класи IP-адрес

IP-адреси можна класифікувати на різні класи залежно від їх діапазону та призначення. Існують п'ять основних класів IP-адрес: A, B, C, D та E.

Клас A: Перший біт встановлений у 0. Діапазон адрес від 1.0.0.0 до 126.255.255.255. Ці адреси призначені для дуже великих мереж з великою кількістю вузлів.

Клас B: Перші два біти встановлені у 10. Діапазон адрес від 128.0.0.0 до 191.255.255.255. Ці адреси призначені для середніх мереж.

Клас C: Перші три біти встановлені у 110. Діапазон адрес від 192.0.0.0 до 223.255.255.255. Ці адреси найчастіше використовуються для невеликих мереж та локальних мереж.

Клас D: Перші чотири біти встановлені у 1110. Діапазон адрес від 224.0.0.0 до 239.255.255.255. Ці адреси зарезервовані для багатоадресної передачі даних.

Клас E: Перші чотири біти встановлені у 1111. Діапазон адрес від 240.0.0.0 до 255.255.255.255. Ці адреси зарезервовані для експериментальних та спеціальних цілей [12].

Розподіл IP-адрес на класи допомагає ефективно організувати та керувати мережевими ресурсами. Однак з розвитком Інтернету та збільшенням кількості підключених пристроїв, виникла потреба в більш гнучких та масштабованих рішеннях, що призвело до впровадження концепції безкласової міжмережевої маршрутизації (CIDR) та використання масок підмереж змінної довжини (VLSM).

Розуміння основних понять, таких як IP-адреси, маски підмереж та класи IP-адрес, є важливим для ефективного планування, розгортання та управління комп'ютерними мережами. Ці концепції лежать в основі процесу розбиття мереж на підмережі (subnetting), який розглядатиметься в наступних розділах.

Розбиття мережі на підмережі (subnetting) є важливим процесом, який дозволяє ефективно використовувати обмежені ресурси IP-адрес та полегшувати управління великими мережами. Цей процес включає в себе логічний поділ однієї великої мережі на менші підмережі, кожна з яких має власний діапазон IP-адрес.

1.2.1. Концепція VLSM (Variable Length Subnet Mask)

Традиційно, розбиття мережі на підмережі здійснювалося з використанням фіксованих масок підмереж, що означало, що всі підмережі мали однакову кількість IP-адрес. Проте, такий підхід часто призводив до неефективного використання адресного простору, оскільки деякі підмережі могли мати надлишок адрес, тоді як інші страждали від дефіциту.

Концепція змінної довжини маски підмережі (Variable Length Subnet Mask, VLSM) була розроблена для вирішення цієї проблеми. VLSM дозволяє використовувати маски підмереж різної довжини для різних підмереж, що забезпечує гнучкість та оптимізацію розподілу IP-адрес відповідно до фактичних потреб кожної підмережі [19]. Використання VLSM дозволяє ефективно розподіляти IP-адреси, уникаючи марнотратного використання адресного простору. Підмережі з більшою кількістю вузлів можуть отримати більше IP-адрес, тоді як підмережі з меншою кількістю вузлів – менше адрес. Це забезпечує

більш ефективне використання обмежених ресурсів та полегшує масштабування мережі в майбутньому.

1.2.2. Алгоритм розрахунку підмереж

Розбиття мережі на підмережі з використанням концепції VLSM (Variable Length Subnet Mask) є процесом, який вимагає ретельного планування та розрахунків. Алгоритм розрахунку підмереж складається з кількох ключових кроків, які забезпечують ефективний розподіл IP-адрес та оптимізацію використання адресного простору. Розглянемо детально кожен крок цього алгоритму.

Крок 1: Визначення вимог та характеристик мережі

Перш ніж розпочати процес розрахунку підмереж, необхідно чітко визначити вимоги та характеристики мережі. Це включає в себе такі дані:

- Початкова IP-адреса та маска мережі: Це базова IP-адреса та маска мережі, які будуть використовуватися для розбиття на підмережі.
- Кількість необхідних підмереж: Скільки окремих підмереж потрібно створити для задоволення потреб мережевої інфраструктури.
- Кількість вузлів (хостів) у кожній підмережі: Для кожної підмережі необхідно визначити, скільки вузлів (хостів) має бути підключено до неї.

Ці вихідні дані є критично важливими для забезпечення правильного та ефективного розподілу IP-адрес.

Крок 2: Визначення необхідної кількості бітів для ідентифікації підмереж

Після визначення вимог та характеристик мережі, наступним кроком є обчислення кількості бітів, необхідних для ідентифікації підмереж. Це здійснюється за допомогою формули:

$$2^n \geq k$$

Де n - кількість бітів для ідентифікації підмереж, а k - кількість необхідних підмереж.

Наприклад, якщо потрібно створити 8 підмереж, то мінімальна кількість бітів для ідентифікації підмереж буде 3, оскільки $2^3 = 8$.

Важливо зазначити, що ця формула дає мінімальну кількість бітів, необхідних для ідентифікації підмереж. Іноді може бути доцільним використовувати більшу кількість бітів, щоб забезпечити додаткову гнучкість та масштабованість у майбутньому.

Крок 3: Визначення нових масок підмереж

Після визначення кількості бітів для ідентифікації підмереж, наступним кроком є обчислення нових масок підмереж. Нова маска підмережі формується шляхом об'єднання маски початкової мережі та визначеної кількості бітів для ідентифікації підмереж.

Наприклад, якщо початкова маска мережі є 255.255.255.0 (або /24 у нотації CIDR), а для ідентифікації підмереж визначено 3 біти, то нова маска підмережі буде 255.255.255.224 (або /27 у нотації CIDR).

Нова маска підмережі буде використовуватися для обчислення діапазонів IP-адрес для кожної підмережі.

Крок 4: Розрахунок діапазонів IP-адрес для кожної підмережі

Використовуючи нову маску підмережі, можна обчислити діапазони IP-адрес для кожної підмережі. Цей крок включає в себе визначення першої та останньої IP-адрес у кожному діапазоні.

Перша IP-адреса в діапазоні зарезервована для ідентифікації самої мережі (network address), а остання IP-адреса зарезервована для широкомовлення (broadcast address). Ці адреси не можуть бути присвоєні вузлам (хостам) у мережі.

Діапазон доступних IP-адрес для кожної підмережі визначається шляхом додавання одиниці до першої IP-адреси та віднімання одиниці від останньої IP-адреси.

Наприклад, якщо діапазон IP-адрес для підмережі є 192.168.1.0 - 192.168.1.31, то перша доступна IP-адреса буде 192.168.1.1, а остання - 192.168.1.30.

Цей крок повторюється для кожної підмережі, використовуючи відповідні діапазони IP-адрес.

Крок 5: Розподіл вузлів (хостів) між підмережами

Після обчислення діапазонів IP-адрес для кожної підмережі, наступним кроком є розподіл вузлів (хостів) між цими підмережами. Це здійснюється на основі вимог до кількості вузлів у кожній підмережі, визначених на першому кроці.

Підмережі з більшою кількістю вузлів отримують більше IP-адрес, тоді як підмережі з меншою кількістю вузлів – менше IP-адрес. Така оптимізація забезпечує ефективне використання адресного простору та уникнення марнотратного розподілу IP-адрес [20] .

Під час розподілу вузлів необхідно враховувати, що в кожній підмережі дві IP-адреси зарезервовані для ідентифікації мережі та широкомовлення. Тому фактична кількість доступних IP-адрес для вузлів буде на два менше, ніж загальна кількість IP-адрес у діапазоні.

Крок 6: Візуалізація та представлення результатів

Після виконання всіх необхідних розрахунків, результати розбиття мережі на підмережі потрібно представити у зручному та зрозумілому форматі. Це може включати в себе створення таблиць, діаграм або інших візуальних елементів, які допоможуть користувачам легко інтерпретувати та використовувати отриману інформацію.

Типова таблиця результатів може містити такі стовпці:

- Номер підмережі
- Діапазон IP-адрес підмережі
- Маска підмережі
- Кількість доступних IP-адрес для вузлів
- Кількість вузлів, призначених для цієї підмережі

Крім табличного представлення результатів, можна також використовувати діаграми та графіки для більш наочної візуалізації розподілу IP-адрес між підмережами. Наприклад, кругова діаграма може відображати відсоткове співвідношення розподілу IP-адрес між підмережами, а стовпчикова діаграма – порівняння кількості виділених IP-адрес для кожної підмережі.

Важливо забезпечити зрозумілість та доступність представлення результатів, оскільки ця інформація буде використовуватися мережевими адміністраторами та фахівцями з IT-інфраструктури для планування та налаштування мережевої архітектури. Чітке та детальне представлення допоможе уникнути помилок та неефективного використання ресурсів IP-адрес.

Під час візуалізації результатів також важливо враховувати можливість збереження та експорту цих даних у різних форматах, таких як CSV, PDF або Excel. Це дозволить користувачам зручно зберігати, переглядати та обмінюватися результатами розрахунку підмереж з колегами або використовувати їх у інших системах та інструментах [23] . Крім основних результатів розрахунку підмереж, веб-застосунок може також надавати додаткову корисну інформацію та рекомендації. Наприклад, можна відображати поради щодо оптимізації використання IP-адрес, рекомендації щодо безпеки мережі або посилання на відповідні ресурси та документацію для поглибленого вивчення теми.

Також важливо передбачити можливість зворотного зв'язку від користувачів, що дозволить вдосконалювати веб-застосунок та додавати нові функції на основі їхніх потреб та запитів. Це може включати форму зворотного зв'язку, систему звітування про помилки або канали зв'язку з розробниками.

Загалом, алгоритм розрахунку підмереж є складним процесом, який вимагає ретельного планування, точних розрахунків та ефективної візуалізації результатів. Веб-застосунок, розроблений з використанням сучасних технологій веб-розробки та хмарних сервісів, може значно полегшити та автоматизувати цей процес, забезпечуючи зручний інтерфейс, точність розрахунків та зрозуміле представлення результатів. Це допоможе мережевим адміністраторам та фахівцям з IT-інфраструктури ефективно планувати та керувати мережевими ресурсами, уникаючи помилок та марнотратного використання IP-адрес [25].

1.3. Огляд технологій веб-розробки

Веб-розробка - це процес створення веб-додатків та веб-сайтів, що включає в себе використання різноманітних технологій та інструментів. Для реалізації функціональності веб-застосунку для обчислення підмереж IP були використані сучасні технології веб-розробки, такі як HTML, CSS, JavaScript, React, Redux та Firebase. Ці технології забезпечують ефективність, продуктивність, зручність використання та масштабованість розробленого веб-додатку.

1.3.1. HTML, CSS та JavaScript

HTML (HyperText Markup Language), CSS (Cascading Style Sheets) та JavaScript є основними технологіями, на яких базується веб-розробка. Вони утворюють трикутник, що забезпечує структуру, стилі та інтерактивність веб-сторінок і додатків.

HTML відповідає за структуру та семантику веб-сторінок. За допомогою HTML створюються розмітки, що визначають елементи сторінки, такі як заголовки, абзаци, списки, таблиці та посилання. HTML забезпечує основу для представлення контенту в Інтернеті.

CSS використовується для стилізації та оформлення веб-сторінок. За допомогою CSS можна визначати кольори, шрифти, розміри, відступи, розташування елементів та багато іншого. CSS дозволяє створювати привабливий та зручний для користувача інтерфейс.

JavaScript - це мова програмування, яка додає інтерактивність та динамічність веб-сторінкам. За допомогою JavaScript можна створювати анімації, ефекти, валідацію форм, взаємодію з користувачем, а також маніпулювати елементами DOM (Document Object Model) на сторінці. JavaScript також використовується для розробки веб-додатків, що працюють на стороні клієнта [2]. Ці три технології тісно пов'язані між собою та утворюють основу для веб-розробки. HTML відповідає за структуру, CSS - за стилі, а JavaScript - за поведінку та інтерактивність. Сучасні веб-сайти та додатки зазвичай використовують комбінацію цих технологій для забезпечення привабливого та функціонального користувацького досвіду.

1.3.2. React та Redux

React - це популярна бібліотека JavaScript для створення користувацьких інтерфейсів. Вона була розроблена компанією Facebook і широко використовується для розробки веб-додатків, а також мобільних додатків з використанням React Native.

Ключовими особливостями React є:

- Компонентна архітектура: React дозволяє розбивати користувацький інтерфейс на окремі компоненти, що полегшує повторне використання коду та підтримку.
- Віртуальний DOM: React використовує віртуальне представлення DOM, що значно підвищує продуктивність за рахунок ефективного оновлення лише необхідних частин інтерфейсу.
- JSX: React використовує JSX, синтаксис, схожий на XML, для опису структури компонентів, що полегшує читання та написання коду.
- Односпрямований потік даних: React дотримується односпрямованого потоку даних, що робить додаток більш передбачуваним та полегшує відстеження змін.

Redux - це бібліотека для управління станом додатку, яка часто використовується у поєднанні з React. Redux забезпечує централізоване сховище даних (store) та визначає правила для оновлення стану додатку.

Ключовими принципами Redux є:

- Єдине джерело істини: Стан додатку зберігається в єдиному immutable об'єкті, званому store.
- Односпрямований потік даних: Єдиним способом змінити стан є відправлення дії (action) через функцію, яка називається редюсер (reducer).
- Чисті функції редюсерів: Редюсери є чистими функціями, що приймають поточний стан і дію та повертають новий стан додатку.

Використання React у поєднанні з Redux забезпечує ефективне управління станом додатку, полегшує відстеження змін та сприяє модульності та

повторному використанню коду. Ця архітектура також сприяє кращій масштабованості та підтримці додатків, особливо для великих проектів [4].

1.3.3. Firebase

Firebase - це хмарна платформа, розроблена компанією Google, яка пропонує різноманітні інструменти та сервіси для розробки веб-додатків та мобільних додатків. Використання Firebase у веб-застосунку для обчислення підмереж IP забезпечує зручну інтеграцію з бекенд-сервісами, такими як база даних, авторизація користувачів та хостинг.

Ключові можливості Firebase:

- Хмарна база даних NoSQL: Firebase пропонує хмарну базу даних NoSQL, що дозволяє зберігати та синхронізувати дані в реальному часі. Це забезпечує високу продуктивність та масштабованість.
- Авторизація: Firebase надає готові рішення для авторизації користувачів, включаючи авторизацію через електронну пошту, соціальні мережі та інші провайдери.
- Хостинг: Firebase пропонує зручний хостинг для статичних веб-сайтів та веб-додатків, забезпечуючи високу продуктивність та безпеку.
- Аналітика: Firebase включає інструменти для збору аналітичних даних про використання додатку, що дозволяє відстежувати поведінку користувачів та оптимізувати користувацький досвід.
- Хмарні функції: Firebase надає можливість створювати та розгортати хмарні функції, які можна використовувати для різноманітних задач, таких як обробка подій, обчислення в фоновому режимі та інтеграція з іншими сервісами.

Інтеграція з Firebase у веб-застосунку для обчислення підмереж IP забезпечує зручне зберігання даних користувачів та їх розрахунків у хмарній базі даних, а також можливість авторизації користувачів. Це дозволяє користувачам зберігати свої попередні розрахунки та переглядати історію обчислень з будь-якого пристрою, що є зручним для подальшого аналізу та планування мережевої інфраструктури. Крім зручної інтеграції з базою даних та авторизацією користувачів, використання Firebase для хостингу веб-застосунку для

обчислення підмереж IP також має ряд переваг. Хостинг Firebase забезпечує високу продуктивність, безпеку та масштабованість для статичних веб-додатків. Одна з головних переваг хостингу Firebase - це можливість глобального розповсюдження контенту через глобальну мережу доставки контенту (CDN). Це означає, що веб-застосунок буде завантажуватися швидко та ефективно для користувачів з усього світу, оскільки контент доставлятиметься з найближчого до користувача сервера.

Крім того, Firebase забезпечує автоматичне встановлення SSL-сертифікатів для захищеного з'єднання HTTPS, що є важливим для безпеки та конфіденційності даних користувачів. Це усуває необхідність самостійного налаштування та підтримки сертифікатів SSL. Хостинг Firebase також пропонує зручні інструменти для безперервного розгортання (Continuous Deployment) веб-застосунку. Це дозволяє швидко та легко розгортати нові версії додатку, забезпечуючи автоматичне оновлення змін на хостингу.

Ще однією перевагою є вбудована інтеграція з іншими сервісами Firebase, такими як базою даних, авторизацією та аналітикою. Це спрощує розробку та забезпечує безшовну взаємодію між різними компонентами веб-застосунку.

Окрім основних можливостей Firebase, під час розробки веб-застосунку для обчислення підмереж IP також можуть використовуватися додаткові інструменти та бібліотеки. Наприклад, для створення складних візуалізацій та діаграм, що представляють результати розрахунку підмереж, можна використовувати бібліотеки для візуалізації даних, такі як D3.js або Chart.js. Також може знадобитися використання додаткових бібліотек для обробки форм, валідації введених даних, маршрутизації та інших функцій, необхідних для забезпечення зручного та ефективного користувацького досвіду [16].

Вибір відповідних технологій та інструментів є важливим аспектом розробки веб-застосунку, оскільки це впливає на продуктивність, масштабованість, безпеку та загальну якість додатку. Використання сучасних технологій веб-розробки, таких як React, Redux та Firebase, у поєднанні з додатковими бібліотеками та інструментами, дозволяє створити високоякісний,

ефективний та зручний для користувача веб-застосунок для обчислення підмереж IP.

Висновки

Розгляд теоретичних основ комп'ютерних мереж, зокрема IP-адрес та масок підмереж, створив міцну основу для подальшої розробки калькулятора мереж. Детальне вивчення концепції VLSM та алгоритмів розрахунку підмереж забезпечило розуміння ключових принципів, які дозволяють оптимально використовувати мережеві ресурси. Огляд технологій веб-розробки, включаючи HTML, CSS, JavaScript, React, Redux та Firebase, дозволив визначити найбільш відповідні інструменти для реалізації проекту.

РОЗДІЛ 2

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1. Огляд існуючих калькуляторів підмереж

На ринку існує безліч калькуляторів підмереж IP, доступних як веб-додатки, десктопні програми та мобільні додатки. Вони пропонують різні функції та інтерфейси для обчислення підмереж на основі заданих параметрів. Розглянемо деякі з найбільш популярних та функціональних рішень.

Одним із провідних веб-калькуляторів підмереж є Subnet Calculator від Site24x7. Цей додаток пропонує зручний і інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко вводити необхідні дані та отримувати детальні результати обчислення підмереж. Користувачі можуть вибрати між різними методами введення, такими як введення IP-адреси та маски підмережі, введення IP-адреси та кількості підмереж, або введення IP-адреси та необхідної кількості вузлів у підмережі. Додаток також підтримує обидві версії IP-протоколу: IPv4 та IPv6.

Після введення необхідних даних, Site24x7 Subnet Calculator надає детальну інформацію про обчислені підмережі, включаючи діапазони IP-адрес, маски підмереж, широкомовні адреси та кількість доступних вузлів у кожній підмережі. Крім того, результати відображаються у зрозумілому табличному форматі, що полегшує їх аналіз та порівняння. Однією з переваг цього калькулятора є наявність опції для збереження результатів у файл або друку звіту, що може бути корисним для документування мережевої конфігурації. [24] Іншим популярним веб-калькулятором підмереж є Subnet Mask Calculator від ipcalculators.com. Цей додаток пропонує кілька різних режимів введення даних, включаючи введення IP-адреси та кількості підмереж, IP-адреси та необхідної кількості вузлів, а також введення IP-адреси та маски підмережі. Після введення даних, Subnet Mask Calculator відображає детальну інформацію про обчислені підмережі, включаючи діапазони IP-адрес, маски підмереж, широкомовні адреси та кількість доступних вузлів.

На відміну від деяких інших калькуляторів, Subnet Mask Calculator від ipcalculators.com також надає візуальне представлення бітової маски в двійковому вигляді, що може бути корисним для кращого розуміння структури підмереж. Додаток також підтримує обидві версії IP-протоколу: IPv4 та IPv6.

Для користувачів, які надають перевагу десктопним програмам, можна розглянути Subnet Calculator від SolarWinds. Це потужний і багатофункціональний додаток, який пропонує широкий спектр можливостей для обчислення та аналізу підмереж. Користувачі можуть вводити дані різними способами, включаючи введення IP-адреси та маски підмережі, IP-адреси та кількості підмереж, або IP-адреси та необхідної кількості вузлів. SolarWinds Subnet Calculator надає детальні результати обчислення підмереж, включаючи діапазони IP-адрес, маски підмереж, широкомовні адреси, кількість доступних вузлів та інформацію про зарезервовані адреси. Крім того, додаток пропонує функцію візуалізації мережевої топології, що допомагає краще зрозуміти структуру та взаємозв'язки між підмережами.

Одним із зручних інструментів SolarWinds Subnet Calculator є можливість виконувати пошук і фільтрування результатів за певними критеріями, наприклад, за розміром підмережі або кількістю доступних вузлів. Це може значно полегшити аналіз і планування мережевої інфраструктури для великих і складних мереж. Якщо Вам потрібен мобільний додаток для обчислення підмереж, варто розглянути Subnet Masker від Nettitude. Доступний для платформ iOS та Android, цей додаток пропонує зручний та інтуїтивно зрозумілий інтерфейс для роботи з підмережами. Користувачі можуть вводити IP-адресу та маску підмережі, IP-адресу та кількість підмереж, або IP-адресу та необхідну кількість вузлів.

Після введення необхідних даних, Subnet Masker відображає діапазони IP-адрес, маски підмереж, широкомовні адреси та кількість доступних вузлів для обчислених підмереж. Додаток також надає візуальне представлення бітової маски у двійковому вигляді, що допомагає краще зрозуміти структуру підмереж.

Однією з корисних функцій Subnet Masker є можливість зберігати та експортувати результати обчислення у форматі PDF або CSV. Це дозволяє зручно документувати мережеву конфігурацію та обмінюватися результатами з колегами або клієнтами. Окрім комерційних додатків, існують також безкоштовні та відкриті рішення для обчислення підмереж. Одним із таких рішень є Subnet Masker від SubnetMasker.com. Це веб-калькулятор, який пропонує простий і зрозумілий інтерфейс для роботи з підмережами. Користувачі можуть вводити IP-адресу та кількість підмереж, або IP-адресу та необхідну кількість вузлів.(див. рис 2.1.)

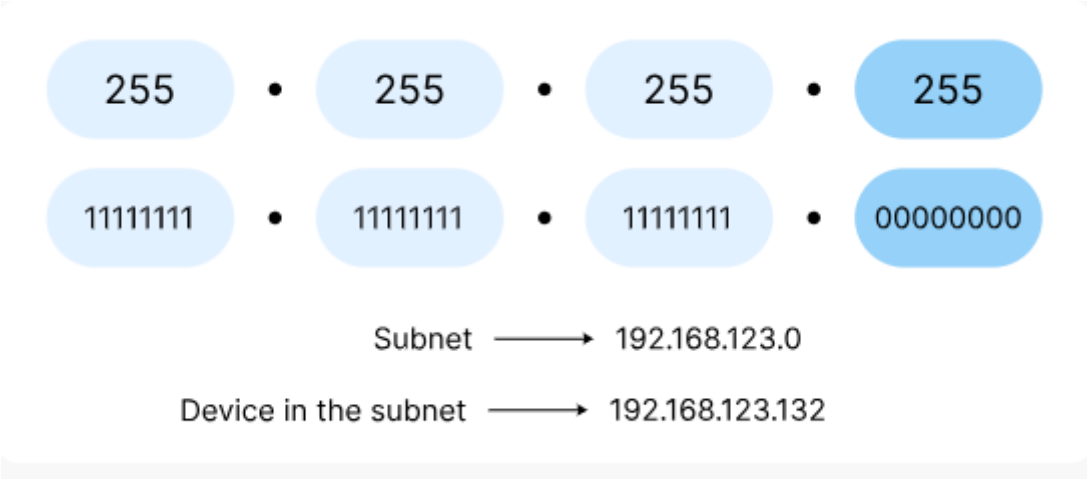

Рис 2.1. Обчислення підмереж

Після введення даних, Subnet Masker відображає діапазони IP-адрес, маски підмереж, широкомовні адреси та кількість доступних вузлів для обчислених підмереж. Додаток також надає візуальне представлення бітової маски у двійковому та шістнадцятковому вигляді.

Хоча Subnet Masker від SubnetMasker.com пропонує більш обмежений набір функцій порівняно з деякими комерційними додатками, він може бути корисним для швидкого та простого обчислення підмереж в основних сценаріях використання. Незважаючи на наявність безлічі калькуляторів підмереж, важливо відзначити, що вони можуть значно відрізнятися за своїми можливостями, інтерфейсом та зручністю використання. Деякі додатки можуть пропонувати більш розширені функції, такі як підтримка кількох підмереж, експорт результатів у різні формати або інтеграцію з іншими інструментами мережевого адміністрування [14] .

Наприклад, Network Calculator від SolarWinds, окрім обчислення підмереж, також пропонує функції для розрахунку діапазонів IP-адрес, конвертації між різними форматами IP-адрес (десятковий, двійковий, шістнадцятковий) та перевірки доступності IP-адрес. Деякі додатки також можуть інтегруватися з системами моніторингу мережі або інструментами керування конфігураціями для автоматизації завдань.

З іншого боку, деякі калькулятори підмереж можуть бути більш спрощеними та зосередженими виключно на обчисленні підмереж, пропонуючи мінімалістичний інтерфейс та обмежений набір функцій. Такі рішення можуть бути зручними для швидкого та простого обчислення підмереж, але не забезпечувати додаткових можливостей для більш складного аналізу та планування мережевої інфраструктури. Крім стандартних функцій обчислення підмереж, деякі додатки можуть включати додаткові інструменти або функції, що полегшують роботу з мережевими конфігураціями. Наприклад, деякі калькулятори підмереж можуть пропонувати функції для перевірки конфліктів IP-адрес, генерування випадкових IP-адрес або підмереж, а також інструменти для визначення типу IP-адреси (приватна, публічна, зарезервована).

Важливим аспектом при виборі калькулятора підмереж є також підтримка різних версій IP-протоколу. Більшість сучасних додатків підтримують як IPv4, так і IPv6, що забезпечує сумісність з існуючими та майбутніми мережевими інфраструктурами. Однак деякі більш старі або спеціалізовані рішення можуть підтримувати лише одну версію IP-протоколу. Окрім функціональних можливостей, при виборі калькулятора підмереж також важливо враховувати зручність використання, інтуїтивність інтерфейсу та ступінь деталізації результатів. Деякі додатки можуть пропонувати більш детальні та зрозумілі звіти, тоді як інші можуть надавати результати в більш стислому вигляді [15].

Загалом, вибір відповідного калькулятора підмереж залежить від конкретних потреб користувача, рівня складності мережевої інфраструктури та вимог до функціональності. Для простих задач може бути достатнім використання базового веб-калькулятора, тоді як для більш складних мереж

може знадобитися більш потужний і функціональний додаток з розширеними можливостями аналізу та документування.

2.2. Переваги та недоліки

Розглянуті в попередньому розділі калькулятори підмереж мають як переваги, так і недоліки, які варто врахувати при виборі відповідного рішення. Розуміння сильних і слабких сторін різних калькуляторів допоможе зробити більш обґрунтований вибір, який відповідатиме конкретним вимогам та потребам. Однією з головних переваг використання спеціалізованих калькуляторів підмереж є спрощення процесу розрахунку та планування мережевої інфраструктури. Замість ручного обчислення підмереж, що може бути складним і схильним до помилок, ці додатки автоматизують цей процес, забезпечуючи швидкі та точні результати.

Крім того, більшість калькуляторів підмереж пропонують зручні інтерфейси, що полегшують введення вихідних даних та інтерпретацію результатів. Деякі додатки, такі як Site24x7 Subnet Calculator або SolarWinds Subnet Calculator, надають візуальні представлення структури підмереж, що допомагає краще зрозуміти взаємозв'язки між різними компонентами мережі. Іншою перевагою є можливість експорту результатів у різні формати, наприклад, PDF, CSV або Excel. Це дозволяє легко документувати мережеві конфігурації та обмінюватися цією інформацією з колегами або клієнтами [13].

Багато калькуляторів підмереж також підтримують обидві версії IP-протоколу: IPv4 та IPv6. Це забезпечує сумісність з існуючими та майбутніми мережевими інфраструктурами, що є особливо важливим в контексті поступового переходу на IPv6. Деякі додатки, такі як SolarWinds Subnet Calculator, пропонують розширені можливості для аналізу та планування мережевої інфраструктури. Наприклад, функції пошуку та фільтрування результатів за певними критеріями можуть значно полегшити роботу з великими та складними мережами [10].

Що стосується недоліків, то одним з основних є те, що більшість калькуляторів підмереж є спеціалізованими інструментами з обмеженими функціями. Вони зосереджені виключно на обчисленні підмереж і не пропонують інтеграції з іншими інструментами мережевого адміністрування або моніторингу. Деякі калькулятори підмереж можуть мати обмежений набір функцій або не підтримувати певні сценарії використання. Наприклад, деякі додатки можуть не підтримувати обчислення кількох підмереж одночасно або не надавати достатньо деталізованих результатів.

Інтерфейси деяких калькуляторів підмереж можуть бути недостатньо інтуїтивними або зручними для користувача. Це може ускладнити введення даних або інтерпретацію результатів, особливо для користувачів з обмеженими знаннями в галузі мережевих технологій. Окремі недоліки можуть бути пов'язані з платформою або способом розповсюдження додатків. Наприклад, веб-калькулятори можуть мати обмежену функціональність порівняно з десктопними додатками, а десктопні додатки, у свою чергу, можуть бути менш зручними для використання на мобільних пристроях. Ще одним недоліком може бути відсутність оновлень або підтримки з боку розробників для деяких калькуляторів підмереж. Це може призвести до проблем сумісності з новими версіями операційних систем або браузерів, а також до відсутності виправлення помилок або додавання нових функцій. У випадку комерційних додатків, таких як SolarWinds Subnet Calculator або Site24x7 Subnet Calculator, недоліком може бути їхня вартість. Хоча ці додатки зазвичай пропонують більше функцій та можливостей, їхня ціна може бути перешкодою для окремих користувачів або невеликих організацій з обмеженим бюджетом [11] .

Важливо також враховувати, що більшість калькуляторів підмереж є автономними інструментами і не інтегруються з існуючими системами управління мережами або інвентаризації активів. Це може ускладнити процес синхронізації даних про мережеву інфраструктуру та вимагати додаткових зусиль для підтримки актуальності інформації. Деякі калькулятори підмереж можуть бути обмеженими у своїх можливостях щодо обробки великих обсягів

даних або складних мережевих конфігурацій. Це може бути проблемою для великих організацій або підприємств з розгалуженими мережами. Нарешті, варто зазначити, що хоча калькулятори підмереж є корисними інструментами, вони не замінюють необхідність глибокого розуміння принципів роботи мереж та IP-адресації. Користувачі все одно повинні мати базові знання в цій галузі, щоб правильно інтерпретувати результати та ефективно планувати мережеву інфраструктуру [7] .

Загалом, при виборі калькулятора підмереж важливо зважити переваги та недоліки різних рішень та обрати той варіант, який найкраще відповідає Вашим конкретним вимогам та обмеженням. Для деяких користувачів простий і безкоштовний веб-калькулятор може бути достатнім, тоді як для інших може знадобитися більш потужний і функціональний додаток, здатний впоратися зі складними мережевими середовищами.

Висновки

Аналіз існуючих рішень у сфері калькуляторів підмереж дозволив визначити їхні сильні та слабкі сторони, що є важливим для покращення власного проекту. Вивчення переваг та недоліків існуючих інструментів допомогло зрозуміти, які функціональні можливості є найбільш затребуваними користувачами та які аспекти потребують удосконалення. Це дозволило сформулювати конкретні вимоги до власного веб-застосунку, спрямовані на створення конкурентоспроможного продукту, що максимально відповідає потребам користувачів.

РОЗДІЛ 3

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1. Вимоги до застосунку

Крім основних функціональних та технічних вимог, веб-застосунок для обчислення підмереж IP повинен відповідати низці додаткових вимог, які забезпечать високу якість, зручність використання та гнучкість для подальшого розвитку та оптимізації. Ці вимоги ґрунтуються на аналізі кращих практик веб-розробки, вивченні потреб користувачів та тенденцій на ринку програмного забезпечення.

Однією з ключових вимог є забезпечення відмовостійкості та високої доступності застосунку. Оскільки веб-застосунок буде доступний через Інтернет, він повинен бути розгорнутий на надійній хмарній платформі з резервним копіюванням даних та механізмами автоматичного відновлення після збоїв. Це допоможе мінімізувати простої та гарантувати безперебійну роботу застосунку для користувачів з різних куточків світу. Важливою вимогою є також забезпечення належної масштабованості та продуктивності застосунку. Оскільки кількість користувачів може зростати з часом, архітектура застосунку повинна бути розроблена таким чином, щоб легко адаптуватися до збільшення навантаження на систему. Це може включати використання хмарних обчислювальних ресурсів з автоматичним масштабуванням, оптимізацію коду та кешування даних для підвищення швидкодії.

Для забезпечення високої якості та стабільності застосунку необхідно впровадити ретельне тестування на всіх етапах розробки. Це включає модульне тестування окремих компонентів, інтеграційне тестування для перевірки взаємодії між різними частинами системи, а також тестування користувацького інтерфейсу та сценаріїв використання. Регресійне тестування також має бути невід'ємною частиною процесу розробки, щоб гарантувати, що нові зміни не призводять до виникнення нових помилок або регресії функціональності.

Для полегшення процесу тестування та впровадження безперервної інтеграції та безперервного розгортання (CI/CD) застосунок повинен бути

розроблений з використанням відповідних практик та інструментів. Це може включати використання систем управління версіями, таких як Git, налаштування автоматизованих конвеєрів збірки та розгортання, а також інтеграцію з системами моніторингу та аналітики для відстеження продуктивності та виявлення потенційних проблем. Ще однією важливою вимогою є забезпечення доступності застосунку для людей з обмеженими можливостями. Веб-застосунок повинен відповідати стандартам доступності, таким як WCAG (Web Content Accessibility Guidelines), забезпечуючи належну підтримку засобів адаптації, таких як програми зчитування екрана, збільшення масштабу та альтернативні способи взаємодії [8] .

Крім того, застосунок повинен бути локалізованим та підтримувати кілька мов інтерфейсу. Це дозволить користувачам з різних країн та культурних середовищ ефективно взаємодіяти з додатком, використовуючи свою рідну мову. Локалізація повинна охоплювати не лише переклад текстових елементів інтерфейсу, а й відповідне форматування дат, чисел та інших даних відповідно до культурних норм.

Для забезпечення належного рівня безпеки та конфіденційності даних застосунок повинен відповідати актуальним нормативним вимогам та стандартам, таким як GDPR (General Data Protection Regulation) та CCPA (California Consumer Privacy Act). Це включає реалізацію надійних механізмів шифрування даних, аутентифікації користувачів, контролю доступу та регулярного оновлення системи для усунення виявлених вразливостей.

Застосунок також повинен мати гнучку та модульну архітектуру, що дозволить легко інтегрувати нові функції та можливості в майбутньому. Це може включати використання мікросервісної архітектури, розробку окремих модулів чи плагінів, а також застосування принципів об'єктно-орієнтованого програмування та шаблонів проєктування для забезпечення високої підтримуваності коду.

Важливою вимогою є також забезпечення належної документації та технічної підтримки для користувачів застосунку. Документація повинна

включати детальні інструкції з використання різних функцій, опис можливих помилок та способів їх вирішення, а також технічну інформацію для розробників або адміністраторів системи. Технічна підтримка може надаватися через різні канали, такі як електронна пошта, форуми, чати або системи відстеження помилок [3] .

Окрім функціональних та технічних вимог, під час розробки веб-застосунку необхідно також враховувати вимоги щодо дизайну та юзабіліті (зручності використання). Дизайн інтерфейсу повинен бути привабливим, інтуїтивно зрозумілим та відповідати сучасним тенденціям у сфері користувацького досвіду (UX). Це включає використання чітких та зрозумілих іконок, правильне розміщення елементів керування, застосування зрозумілих візуальних ієрархій та використання відповідних кольорових схем.

Крім того, веб-застосунок для обчислення підмереж IP повинен відповідати вимогам щодо доступності та зручності використання на різних пристроях, таких як настільні комп'ютери, ноутбуки, планшети та смартфони. Це може включати впровадження адаптивного або responsive дизайну, оптимізацію для сенсорного введення та забезпечення належної швидкодії на пристроях з різною потужністю та характеристиками з'єднання з Інтернетом.

Ще однією важливою вимогою є забезпечення належного рівня продуктивності та швидкодії застосунку, особливо при обробці великих обсягів даних або складних обчислень. Це може включати оптимізацію алгоритмів, використання кешування, асинхронної обробки даних та паралельних обчислень, де це можливо.

3.2. Архітектура застосунку

Архітектура веб-застосунку для обчислення підмереж IP відіграє ключову роль у забезпеченні належної продуктивності, масштабованості, безпеки та можливості подальшого розвитку системи. Тому при проектуванні архітектури необхідно враховувати низку важливих чинників та найкращих практик веб-розробки. Одним з найбільш поширених підходів до розробки сучасних веб-

застосунків є використання архітектури клієнт-сервер. У цій моделі веб-застосунок складається з двох основних компонентів: клієнтської частини (frontend) та серверної частини (backend).

Клієнтська частина веб-застосунку, як правило, реалізується за допомогою HTML, CSS та JavaScript і відповідає за відображення користувацького інтерфейсу, обробку вхідних даних та взаємодію з користувачем. Для розробки клієнтської частини можуть використовуватися сучасні JavaScript-фреймворки, такі як React, Angular або Vue.js, які забезпечують високу продуктивність, реактивність та можливість створення складних користувацьких інтерфейсів. З іншого боку, серверна частина веб-застосунку відповідає за обробку запитів від клієнтської частини, взаємодію з базами даних, виконання складних обчислень та забезпечення безпеки системи. Серверна частина може бути реалізована за допомогою різних технологій та мов програмування, таких як Node.js, Python, Ruby, Java або .NET.

Для забезпечення ефективної комунікації між клієнтською та серверною частинами веб-застосунку використовуються протоколи передачі даних, такі як HTTP або WebSocket. Зазвичай клієнтська частина відправляє запити на сервер за допомогою HTTP-протоколу, а сервер відповідає, повертаючи необхідні дані або виконуючи запитані операції. У контексті веб-застосунку для обчислення підмереж IP, клієнтська частина може відповідати за відображення інтерфейсу для введення вихідних даних (IP-адреси, маски підмереж тощо), відправлення цих даних на сервер для обчислень та візуалізацію отриманих результатів. Натомість серверна частина буде виконувати власне обчислення підмереж на основі отриманих від клієнта даних та повертати результати назад до клієнтської частини.

Для забезпечення належної продуктивності та масштабованості веб-застосунку доцільно розглянути використання архітектури мікросервісів. У цій моделі застосунок розбивається на невеликі незалежні модулі (мікросервіси), кожен з яких відповідає за окрему функціональність або бізнес-процес. Ці мікросервіси можуть бути розгорнуті окремо та взаємодіяти один з одним за

допомогою легковагих протоколів передачі повідомлень, таких як HTTP, gRPC або AMQP. Наприклад, у веб-застосунку для обчислення підмереж IP можна виділити такі окремі мікросервіси: сервіс обчислення IPv4 підмереж, сервіс обчислення IPv6 підмереж, сервіс авторизації користувачів, сервіс збереження та отримання історії розрахунків тощо. Такий підхід забезпечує високу модульність, полегшує масштабування та оновлення окремих компонентів системи, а також підвищує загальну відмовостійкість застосунку [22].

Для зберігання даних, таких як інформація про користувачів, історія розрахунків або інші метадані, веб-застосунок може використовувати різні типи баз даних. Традиційно для веб-застосунків використовуються реляційні бази даних, такі як MySQL, PostgreSQL або Oracle. Однак у контексті сучасних хмарних та масштабованих рішень все більшої популярності набувають NoSQL бази даних, такі як MongoDB, Cassandra або DynamoDB. NoSQL бази даних мають перевагу у тому, що вони зазвичай забезпечують вищу масштабованість, відмовостійкість та гнучкість у порівнянні з традиційними реляційними базами даних. Крім того, вони краще підходять для зберігання неструктурованих або напівструктурованих даних, що може бути актуальним для деяких компонентів веб-застосунку.

Для забезпечення безпеки веб-застосунку необхідно впровадити належні механізми аутентифікації та авторизації користувачів. Одним із поширених рішень є використання JSON Web Tokens (JWT) для безпечної передачі даних про аутентифікацію між клієнтською та серверною частинами застосунку. JWT забезпечують компактний та самодостатній спосіб передачі інформації про аутентифікованого користувача, що дозволяє уникнути необхідності зберігати стан сесії на сервері. Для додаткового захисту веб-застосунку також рекомендується використовувати шифрування даних під час передачі (HTTPS) та впровадити заходи щодо захисту від загроз, таких як SQL-ін'єкції, крос-сайтовий скриптинг (XSS) та крос-сайтові підробки запитів (CSRF).

Важливим аспектом архітектури веб-застосунку є також забезпечення належного рівня відмовостійкості та високої доступності. Це може бути

досягнуто шляхом розгортання застосунку на кластері серверів з механізмами балансування навантаження та резервного копіювання даних. Також варто розглянути можливість використання хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform, які пропонують готові рішення для масштабування та відмовостійкості. Одним із ключових компонентів сучасних веб-застосунків є система безперервної інтеграції та безперервного розгортання (CI/CD). Ця система забезпечує автоматизацію процесів збірки, тестування та розгортання застосунку, що дозволяє швидко та ефективно впроваджувати нові функції та виправлення помилок.

Для забезпечення ефективного моніторингу та відстеження продуктивності веб-застосунку необхідно інтегрувати його з системами збору та аналізу метрик. Це дозволить виявляти потенційні вузькі місця, проблеми з продуктивністю та своєчасно реагувати на них. Популярними інструментами для моніторингу веб-застосунків є Prometheus, Grafana, New Relic та AppDynamics. Важливо також приділити увагу питанням масштабованості та горизонтального масштабування веб-застосунку. Оскільки кількість користувачів може зростати з часом, архітектура застосунку повинна бути спроектована таким чином, щоб легко розподіляти навантаження між декількома серверами або контейнерами. Це може бути досягнуто шляхом використання балансувальників навантаження, кешування даних та розподілу обчислювальних ресурсів [21] .

Для полегшення розгортання та керування веб-застосунком на різних середовищах (розробка, тестування, продакшн) рекомендується використовувати технології контейнеризації, такі як Docker або Kubernetes. Контейнеризація дозволяє упакувати весь застосунок та його залежності в портативні, ізольовані контейнери, які можна легко переміщувати та запускати на будь-якій системі, що підтримує контейнеризацію.

Ще одним важливим аспектом архітектури веб-застосунку є забезпечення належної підтримки та розширюваності. Для цього необхідно дотримуватися принципів модульного дизайну, розділяючи функціональність на чіткі, слабко зв'язані компоненти, які можна легко замінити або розширити в майбутньому.

Також важливо забезпечити належну документацію коду, архітектури та процесів розробки, щоб полегшити підтримку та внесення змін у майбутньому. Ще одним ключовим компонентом архітектури сучасного веб-застосунку є система кешування. Кешування дозволяє зберігати часто запитувані дані в оперативній пам'яті або розподіленому кеш-сховищі, що значно підвищує продуктивність застосунку та зменшує навантаження на бази даних та інші ресурсомісткі компоненти. Популярними рішеннями для кешування є Redis, Memcached та вбудовані механізми кешування в базах даних.

У контексті веб-застосунку для обчислення підмереж IP можна використовувати кешування для зберігання часто використовуваних або обчислювально складних результатів розрахунків підмереж. Це дозволить уникнути необхідності повторних обчислень для однакових вхідних даних, що значно підвищить продуктивність та швидкодію застосунку. Нарешті, важливо звернути увагу на питання оптимізації продуктивності веб-застосунку на рівні коду та алгоритмів. Це може включати використання ефективних алгоритмів обчислення підмереж, оптимізацію запитів до бази даних, мінімізацію передачі даних між клієнтом та сервером, а також застосування загальних оптимізацій JavaScript та HTML/CSS для покращення відгуку та швидкодії користувацького інтерфейсу [19].

Загалом, проектування архітектури веб-застосунку для обчислення підмереж IP вимагає ретельного планування та врахування різноманітних факторів, таких як продуктивність, масштабованість, безпека, відмовостійкість та підтримуваність. Дотримання сучасних практик веб-розробки, використання відповідних технологій та інструментів, а також ретельне тестування та моніторинг забезпечать створення якісного, надійного та ефективного веб-застосунку, який зможе задовольнити потреби користувачів та підтримати майбутнє зростання системи.

3.3. Реалізація інтерфейсу користувача

Розробка зручного та інтуїтивно зрозумілого інтерфейсу користувача є критично важливою для успіху будь-якого веб-застосунку, включаючи застосунок для обчислення підмереж IP. Якісний інтерфейс забезпечує позитивний досвід взаємодії з додатком, полегшує виконання необхідних завдань та сприяє залученню та утриманню користувачів.

Одним з ключових аспектів розробки інтерфейсу користувача є дотримання принципів юзабіліті (зручності використання) та доступності. Інтерфейс повинен бути інтуїтивно зрозумілим, логічно структурованим та забезпечувати легку навігацію між різними розділами та функціями додатку. Важливо також враховувати особливості різних пристроїв та розмірів екранів, на яких буде використовуватися застосунок. Це означає, що інтерфейс повинен бути адаптивним та responsive, забезпечуючи оптимальне відображення та зручність використання як на настільних комп'ютерах, так і на мобільних пристроях.

Під час розробки інтерфейсу необхідно приділити особливу увагу візуальній ієрархії та організації елементів. Ключові компоненти та функції повинні бути чітко виділені та легкодоступні для користувачів. Водночас, другорядні елементи не повинні відволікати увагу від основного функціоналу. Використання принципів матеріального дизайну (Material Design) або іншої сучасної системи дизайну може допомогти забезпечити узгоджений та привабливий вигляд інтерфейсу. Це включає ретельний підбір кольорових схем, типографіки, іконок та інших візуальних елементів, які повинні бути гармонійно поєднані та створювати цілісний користувацький досвід.

Під час розробки інтерфейсу веб-застосунку для обчислення підмереж IP необхідно передбачити кілька ключових розділів або сторінок. Головна сторінка, як правило, повинна містити форму для введення вихідних даних, таких як IP-адреса, маска підмережі або кількість бажаних підмереж. Ця форма має бути чіткою, зрозумілою та включати відповідні підказки та валідацію введених даних.

Після введення вихідних даних та ініціювання обчислення підмереж, користувачеві повинна бути представлена сторінка з результатами. Ця сторінка повинна містити детальну інформацію про обчислені підмережі, включаючи діапазони IP-адрес, маски підмереж, широкомовні адреси, кількість доступних вузлів та інші релевантні дані. Результати повинні бути подані у зрозумілому та структурованому вигляді, наприклад, у формі таблиці або списку. Крім того, сторінка результатів може включати додаткові функції, такі як візуалізація структури підмереж у вигляді діаграм або схем, можливість експорту результатів у різні формати (PDF, CSV тощо), а також зручні засоби для фільтрації та сортування даних.

Для покращення користувацького досвіду може бути корисною наявність сторінки з історією попередніх розрахунків. Ця сторінка дозволить користувачам переглядати, порівнювати та за необхідності повторно завантажувати результати минулих обчислень. Це може бути особливо корисним для планування та аналізу складних мережевих конфігурацій. Якщо веб-застосунок передбачає авторизацію користувачів, необхідно забезпечити наявність відповідних сторінок для реєстрації, входу та управління обліковими записами. Ці сторінки повинні бути інтуїтивними, зручними та забезпечувати належний рівень безпеки, включаючи шифрування даних та процедури відновлення паролів. Окрім основних розділів, пов'язаних з обчисленням підмереж, веб-застосунок може містити додаткові сторінки, такі як сторінка "Про додаток", сторінка з інструкціями або довідковими матеріалами, а також сторінка зворотного зв'язку для отримання відгуків та пропозицій від користувачів.

Важливим аспектом розробки інтерфейсу користувача є забезпечення належної доступності для людей з обмеженими можливостями. Це включає підтримку програм зчитування екрана, налаштування контрастності кольорів для покращення читабельності, підтримку різних способів взаємодії (клавіатура, голосове введення тощо), а також дотримання відповідних стандартів доступності, таких як WCAG (Web Content Accessibility Guidelines).

Під час розробки інтерфейсу користувача необхідно також приділити увагу питанням продуктивності та оптимізації. Це може включати мінімізацію кількості HTTP-запитів, ефективне використання кешування, лінивне завантаження ресурсів (наприклад, зображень або скриптів), а також застосування прийомів оптимізації CSS та JavaScript для забезпечення плавної роботи та швидкого відгуку інтерфейсу. Нарешті, важливим кроком у процесі розробки інтерфейсу користувача є проведення ретельного тестування та ітераційного вдосконалення на основі зворотного зв'язку від реальних користувачів. Це може включати юзабіліті-тестування, A/B-тестування різних варіантів дизайну, збір аналітичних даних про поведінку користувачів та опитування або інтерв'ювання цільової аудиторії [17].

Загалом, розробка якісного та зручного інтерфейсу користувача є комплексним процесом, який вимагає ретельного планування, дотримання принципів юзабіліті та доступності, застосування сучасних підходів і технологій, а також постійного вдосконалення на основі зворотного зв'язку від користувачів. Лише поєднання усіх цих аспектів дозволить створити веб-застосунок для обчислення підмереж IP з інтуїтивним, зрозумілим та привабливим інтерфейсом

3.4. Реалізація алгоритму розрахунку підмереж

Серцем веб-застосунку для обчислення підмереж IP є алгоритм, який виконує власне розрахунки на основі введених користувачем вихідних даних. Від ефективності та точності цього алгоритму залежить загальна якість та корисність додатку. Тому при його реалізації необхідно враховувати кілька ключових аспектів.

По-перше, алгоритм повинен підтримувати обидві версії IP-протоколу: IPv4 та IPv6. Це забезпечить сумісність застосунку як з існуючими, так і з майбутніми мережевими інфраструктурами, оскільки перехід на IPv6 стає все більш актуальним у зв'язку з вичерпанням адресного простору IPv4. Для обчислення підмереж IPv4 алгоритм, як правило, використовує бітові операції з IP-адресами та масками підмереж, представленими у двійковому форматі. Це

дозволяє визначати діапазони IP-адрес для кожної підмережі, а також обчислювати мережеві та широкомовні адреси.

Для IPv6, через значно більший адресний простір, алгоритм може використовувати більш складні математичні операції, такі як розширене зсуву бітів та операції з масками префіксів. Це дозволяє правильно визначати межі підмереж та обчислювати необхідні параметри. Незалежно від версії IP-протоколу, алгоритм повинен забезпечувати високу точність обчислень та коректно обробляти різні вхідні параметри, такі як IP-адреса, маска підмережі, кількість бажаних підмереж або необхідна кількість вузлів у підмережі. Це вимагає ретельної валідації та нормалізації вхідних даних, а також обробки можливих помилкових ситуацій.

Під час реалізації алгоритму важливо також приділити увагу його ефективності та продуктивності. Оскільки веб-застосунок може обробляти великі обсяги даних або виконувати складні обчислення, алгоритм повинен бути оптимізований для мінімізації використання ресурсів та забезпечення прийнятного часу відгуку. Одним з підходів до підвищення ефективності алгоритму є використання кешування проміжних результатів або часто використовуваних обчислень. Наприклад, якщо користувач вводить ті самі вихідні дані кілька разів, замість повторного обчислення підмереж можна повернути раніше обчислений та закешований результат.

Крім того, для складних обчислень може бути доцільним використання асинхронної обробки або розпаралелювання обчислень за допомогою технологій, таких як Web Workers або потоки (threads). Це дозволить уникнути блокування основного потоку виконання і забезпечити плавну роботу користувацького інтерфейсу навіть під час виконання ресурсомістких операцій.

Під час реалізації алгоритму розрахунку підмереж необхідно також враховувати можливість обробки різних сценаріїв використання та різних типів вхідних даних. Наприклад, користувач може ввести IP-адресу та маску підмережі, або IP-адресу та кількість бажаних підмереж, або IP-адресу та

необхідну кількість вузлів у підмережі. Алгоритм повинен коректно обробляти всі ці випадки та надавати відповідні результати.

Важливим аспектом є також забезпечення належної обробки помилкових ситуацій та неприпустимих вхідних даних. Алгоритм повинен виявляти такі ситуації та генерувати зрозумілі повідомлення про помилки, які можна відобразити користувачеві в інтерфейсі застосунку. Це допоможе уникнути некоректної роботи або збоїв додатку та забезпечить кращий користувацький досвід. Під час реалізації алгоритму розрахунку підмереж важливо також враховувати питання масштабованості та можливості майбутнього розширення функціональності. Алгоритм повинен бути спроектований модульно, з чіткими інтерфейсами та слабкими зв'язками між його компонентами. Це полегшить додавання нових функцій або модифікацію існуючої логіки в майбутньому.

Наприклад, алгоритм може бути розділений на окремі модулі для обчислення підмереж IPv4 та IPv6, а також на модулі для обробки різних типів вхідних даних. Такий підхід забезпечить кращу підтримуваність коду та полегшить внесення змін або додавання нових функцій у майбутньому. Під час реалізації алгоритму важливо також приділити увагу питанням безпеки та захисту від потенційних загроз. Це може включати обробку можливих атак, таких як ін'єкції коду або спроби переповнення буферів, а також забезпечення належного контролю доступу та обмеження ресурсів для запобігання відмові в обслуговуванні (DoS).

Ще одним важливим аспектом є забезпечення належної документації та тестування алгоритму. Документація повинна містити детальний опис алгоритму, його компонентів, вхідних та вихідних даних, а також можливих помилкових ситуацій. Це полегшить роботу інших розробників з кодом та забезпечить кращу підтримуваність у майбутньому. Тестування алгоритму є критично важливим для забезпечення його коректності та надійності. Необхідно розробити комплексний набір тестових випадків, який охоплює різні сценарії використання, різні типи вхідних даних, граничні випадки та можливі помилкові

ситуації. Це дозволить виявляти та виправляти потенційні помилки на ранніх стадіях розробки та запобігати виникненню проблем у майбутньому.

Під час тестування алгоритму важливо також перевірити його продуктивність та масштабованість, виконуючи навантажувальне тестування з великими обсягами даних або складними обчисленнями.

3.5. Інтеграція з Firebase

Одним із ключових компонентів розробленого веб-застосунку для обчислення підмереж IP є інтеграція з хмарною платформою Firebase від Google. Firebase надає потужний набір інструментів та сервісів, які значно спрощують розробку веб- та мобільних додатків, забезпечуючи зручну інфраструктуру для зберігання даних, автентифікації користувачів, хостингу додатків та багато іншого.

У контексті веб-застосунку для обчислення підмереж IP, Firebase використовується для забезпечення двох основних функцій: зберігання історії обчислень користувачів та автентифікації користувачів. Для інтеграції з Firebase було використано спеціальний SDK (набір інструментів розробника), який дозволяє легко підключитися до відповідних сервісів Firebase та взаємодіяти з ними безпосередньо з коду додатку. Процес інтеграції розпочався з налаштування нового проєкту на консолі Firebase та підключення відповідного SDK до проєкту веб-застосунку. Після цього було створено компоненти React, які відповідали за взаємодію з різними сервісами Firebase, такими як автентифікація користувачів та хмарна база даних.

Для забезпечення зберігання історії обчислень користувачів було використано хмарну базу даних Firebase Realtime Database. Ця база даних являє собою NoSQL-сховище даних у форматі JSON, що забезпечує швидкий та ефективний доступ до даних, а також автоматичну синхронізацію даних між різними клієнтами в режимі реального часу. Під час першого запуску веб-застосунку було створено початкову структуру бази даних, яка складалася з кореневого вузла "users", під яким зберігалися підвузли для кожного зареєстрованого користувача. Ідентифікатор кожного підвузла відповідав

унікальному ідентифікатору користувача, отриманому від сервісу автентифікації Firebase.

Коли користувач виконував обчислення підмереж IP, результати цих обчислень зберігалися у відповідному підвузлі користувача у формі JSON-об'єктів. Кожен об'єкт містив детальну інформацію про обчислення, включаючи вхідні дані (початкову IP-адресу, маску підмережі, вимоги до кількості хостів), а також результати обчислень (список підмереж з їхніми IP-адресами, масками підмереж та кількістю хостів). Для взаємодії з базою даних Firebase було використано спеціальний API, який надається SDK Firebase. Цей API дозволяє виконувати операції читання, запису, оновлення та видалення даних у базі даних за допомогою зручних методів та асинхронних викликів.

Під час збереження результатів обчислень, веб-застосунок створював новий JSON-об'єкт з відповідними даними та викликав метод "push" API бази даних Firebase для додавання цього об'єкта до підвузла поточного користувача. Кожен об'єкт отримував унікальний ідентифікатор, який автоматично генерувався Firebase. Для відображення історії обчислень користувача було створено окремий компонент React, який взаємодіяв з базою даних Firebase для отримання даних про попередні обчислення. Цей компонент викликав метод "on" API бази даних Firebase для підписки на зміни даних у підвузлі поточного користувача. Завдяки автоматичній синхронізації Firebase, компонент миттєво отримував оновлені дані про нові обчислення, які зберігалися в базі даних.

Отримані дані про історію обчислень візуалізувалися у вигляді таблиці або списку, дозволяючи користувачам легко переглядати інформацію про попередні обчислення, такі як вхідні дані, отримані підмережі та їх деталі. Крім того, для кожного запису історії було реалізовано можливість видалення, що дозволяло користувачам прибирати непотрібні записи з їхньої історії.

Одним з ключових переваг використання Firebase є автоматична синхронізація даних між різними пристроями та клієнтами. Це означає, що користувач може переглядати та редагувати свою історію обчислень з будь-якого пристрою, на якому він автентифікований у веб-застосунку, без необхідності

додаткової синхронізації або передачі даних. Для забезпечення належної продуктивності та швидкодії при роботі з базою даних Firebase, у веб-застосунку було реалізовано механізми кешування та оптимізації запитів. Зокрема, використано кешування даних на стороні клієнта для уникнення зайвих запитів до бази даних та прискорення відображення історії обчислень. Також було застосовано оптимізацію запитів до бази даних, використовуючи відповідні методи Firebase SDK для отримання лише необхідних даних та уникнення зайвого трафіку.

Крім функцій зберігання історії обчислень, Firebase також надає потужні інструменти для автентифікації користувачів. У веб-застосунку було інтегровано систему автентифікації Firebase, яка дозволяє користувачам реєструватися, входити в систему та керувати своїми обліковими записами. Детальний опис інтеграції автентифікації Firebase наведено в розділі 3.6 "Авторизація та управління користувачами". Для забезпечення безпеки та конфіденційності даних користувачів, Firebase пропонує потужні механізми захисту, такі як шифрування даних під час передачі та зберігання, а також контроль доступу на основі ролей та правил безпеки. У веб-застосунку було налаштовано відповідні правила безпеки, які дозволяють доступ до збережених даних лише автентифікованим користувачам [17].

Одним з ключових переваг використання Firebase є можливість легкого масштабування додатку та його компонентів без необхідності додаткової конфігурації або налаштувань.

3.6. Авторизація та управління користувачами

Одним із ключових аспектів розробленого веб-застосунку для обчислення підмереж IP є можливість авторизації користувачів та збереження їхньої історії розрахунків. Для забезпечення цієї функціональності було інтегровано систему автентифікації та управління користувачами на основі хмарної платформи Firebase від Google. Firebase надає потужний набір інструментів та сервісів для розробки веб- та мобільних додатків, включаючи зручну систему автентифікації

користувачів. Завдяки вбудованій підтримці різних методів автентифікації, таких як електронна пошта та пароль, авторизація через соціальні мережі (Google, Facebook, Twitter тощо), а також анонімна авторизація, Firebase забезпечує гнучкість та масштабованість у питаннях управління користувачами.

Для інтеграції системи автентифікації Firebase у веб-застосунок було використано спеціальний SDK (набір інструментів розробника), який дозволяє легко підключитися до відповідних сервісів Firebase та взаємодіяти з ними безпосередньо з коду додатку. Цей SDK надає зручний програмний інтерфейс (API) для виконання різноманітних операцій, пов'язаних з авторизацією користувачів, таких як реєстрація, вхід, вихід з системи, відновлення паролю тощо. Процес інтеграції авторизації Firebase у веб-застосунок розпочався з налаштування відповідного проєкту на консолі Firebase та підключення SDK до проєкту. Після цього було створено компоненти React для відображення форм реєстрації та входу в систему, які взаємодіяли з API Firebase для виконання відповідних операцій автентифікації.

Під час реєстрації нового користувача, веб-застосунок запитує необхідні дані, такі як електронна пошта та пароль, та відправляє їх до Firebase для створення нового облікового запису. Firebase забезпечує належну перевірку та валідацію введених даних, а також надсилає користувачу електронний лист для підтвердження реєстрації. Після успішної реєстрації користувач отримує можливість входу в систему, вводячи свої облікові дані (електронну пошту та пароль) у відповідну форму. Веб-застосунок взаємодіє з API Firebase для перевірки автентичності введених даних та, у разі успішної автентифікації, надає доступ до функціональності застосунку.

Крім традиційної автентифікації за допомогою електронної пошти та паролю, Firebase також підтримує альтернативні методи автентифікації, такі як авторизація через соціальні мережі (Google, Facebook, Twitter тощо). Ця функціональність була реалізована у веб-застосунку, дозволяючи користувачам зручно входити в систему, використовуючи свої облікові записи в соціальних мережах. Для забезпечення безпеки та конфіденційності даних користувачів,

Firebase пропонує потужні механізми захисту, такі як шифрування даних під час передачі та зберігання, а також контроль доступу на основі ролей та правил безпеки. У веб-застосунку було налаштовано відповідні правила безпеки, які дозволяють доступ до збережених даних лише автентифікованим користувачам.

Окрім функцій автентифікації, Firebase також забезпечує зручну хмарну базу даних для зберігання та синхронізації даних між різними клієнтами (веб-браузерами, мобільними пристроями тощо). У контексті веб-застосунку для обчислення підмереж IP, ця база даних використовується для зберігання історії обчислень кожного користувача. Після успішної автентифікації користувача, веб-застосунок отримує унікальний ідентифікатор користувача від Firebase та використовує його для створення власного вузла в базі даних Firebase, де зберігатиметься історія обчислень цього користувача. Кожного разу, коли користувач виконує нове обчислення підмереж, результати обчислень зберігаються у базі даних Firebase під відповідним вузлом користувача.

Для відображення історії обчислень у веб-застосунку було створено спеціальний компонент React, який взаємодіє з базою даних Firebase та отримує дані про попередні обчислення користувача. Ці дані візуалізуються у вигляді таблиці або списку, дозволяючи користувачам легко переглядати свою історію та, за необхідності, видаляти або редагувати попередні розрахунки. Одним з ключових переваг використання Firebase є автоматична синхронізація даних між різними пристроями та клієнтами. Це означає, що користувач може переглядати та редагувати свою історію обчислень з будь-якого пристрою, на якому він автентифікований у веб-застосунку, без необхідності додаткової синхронізації або передачі даних.

Для забезпечення належної продуктивності та швидкодії при роботі з базою даних Firebase, у веб-застосунку було реалізовано механізми кешування та оптимізації запитів. Зокрема, використано кешування даних на стороні клієнта для уникнення зайвих запитів до бази даних та прискорення відображення історії обчислень. Також було застосовано оптимізацію запитів до бази даних, використовуючи відповідні методи Firebase SDK для отримання лише

необхідних даних та уникнення зайвого трафіку. Крім функцій авторизації та збереження історії обчислень, Firebase також надає додаткові сервіси, такі як хостинг веб-додатків, аналітика користувачів, повідомлення у реальному часі тощо. Хоча ці сервіси не були використані у поточній версії веб-застосунку, їх можна легко інтегрувати в майбутньому для забезпечення додаткових можливостей та покращення користувацького досвіду.

Загалом, інтеграція Firebase у веб-застосунок для обчислення підмереж IP забезпечила зручну та безпечну систему авторизації користувачів, а також надійне зберігання та синхронізацію історії обчислень між різними пристроями та клієнтами.

Висновки

Процес розробки веб-застосунку включав визначення вимог, проектування архітектури та реалізацію основних компонентів, що дозволило створити зручний та функціональний інструмент для розрахунку підмереж. Застосування сучасних технологій веб-розробки, таких як React, Redux та Firebase, забезпечило високу продуктивність та надійність застосунку. Інтеграція механізмів авторизації та управління користувачами підвищила безпеку та зручність використання. В результаті вдалося створити продукт, що відповідає сучасним стандартам якості та зручності.

РОЗДІЛ 4

ТЕСТУВАННЯ ЗАСТОСУНКУ

4.1. Модульне тестування

Для забезпечення належної якості веб-застосунку та гарантування відсутності помилок у ключових функціях, необхідно провести ретельне модульне тестування. Цей вид тестування передбачає перевірку окремих модулів або компонентів програми на предмет коректності їх функціонування у відповідності до встановлених вимог. З метою всебічної перевірки функціональності калькулятора підмереж IP, було створено комплексний набір модульних тестів. Ці тести охоплюють різноманітні сценарії використання, включаючи введення валідних та невалідних даних, обробку граничних випадків, а також перевірку точності обчислень для різних діапазонів IP-адрес та масок підмереж.

Одним з ключових аспектів тестування є перевірка коректності валідації введених даних. Для цього було створено серію тестів, які імітують введення неправильних IP-адрес, негативних значень маски підмережі, некоректної кількості підмереж або хостів у них. Ці тести гарантують, що застосунок належним чином виявляє та інформує користувача про помилки у вхідних даних, запобігаючи некоректним обчисленням. Окремий блок тестів присвячений перевірці алгоритму розрахунку підмереж. Тут передбачено різноманітні випадки, починаючи від простих прикладів з невеликою кількістю підмереж до складних сценаріїв з великою кількістю підмереж різних розмірів. Кожен тест перевіряє правильність обчислення IP-адрес та масок підмереж, а також відповідність розподілу хостів у кожній підмережі заданим вимогам.

Для забезпечення комплексності тестування, було створено тести, що імітують граничні випадки, такі як максимальна та мінімальна кількість підмереж, максимальна та мінімальна кількість хостів у підмережі, а також використання різних класів IP-адрес (A, B, C). Ці тести гарантують стабільність та коректність роботи застосунку в екстремальних умовах.

Крім функціональних тестів, було також створено тести для перевірки продуктивності та ефективності роботи алгоритму розрахунку підмереж. Ці тести включають сценарії з великою кількістю підмереж та хостів, що дозволяє виявити потенційні вузькі місця та оптимізувати алгоритм для забезпечення високої швидкодії. Для автоматизації процесу тестування та полегшення регулярного запуску тестів було використано сучасні фреймворки для модульного тестування, такі як Jest та Enzyme. Ці інструменти дозволяють швидко та ефективно виконувати весь набір тестів, а також легко інтегрувати їх у процес безперервної інтеграції та розгортання застосунку [1].

Загалом, модульне тестування відіграє ключову роль у забезпеченні високої якості веб-застосунку для обчислення підмереж IP. Ретельно розроблені тести гарантують коректність роботи ключових функцій, виявлення та усунення помилок на ранніх стадіях розробки, а також сприяють підвищенню стабільності та надійності застосунку в цілому.

4.2. Інтеграційне тестування

Після успішного проведення модульного тестування, наступним кроком у забезпеченні якості веб-застосунку для обчислення підмереж IP є інтеграційне тестування. Цей вид тестування має на меті перевірку взаємодії різних компонентів, модулів та підсистем застосунку між собою, а також їх коректну інтеграцію в єдине ціле. Для проведення інтеграційного тестування було розроблено комплексну стратегію, яка охоплює різні аспекти взаємодії компонентів застосунку. Одним з ключових напрямків є тестування інтеграції користувацького інтерфейсу з бізнес-логікою обчислення підмереж.

У цьому контексті було створено серію тестів, які імітують різноманітні сценарії взаємодії користувача з інтерфейсом, включаючи введення даних, відправку форм, отримання результатів обчислень та відображення інформації на сторінці. Ці тести перевіряють, що компоненти інтерфейсу коректно передають дані до відповідних модулів бізнес-логіки, а також належним чином відображають результати обчислень, отримані від цих модулів.

Окремий блок тестів присвячений перевірці інтеграції модулів обчислення підмереж з системою збереження даних та авторизацією користувачів. Ці тести імітують різні сценарії авторизації користувачів, збереження та отримання історії обчислень, а також перевіряють коректність взаємодії між відповідними компонентами застосунку. Для забезпечення комплексності тестування було створено тести, які перевіряють інтеграцію компонентів візуалізації даних (таблиці, діаграми тощо) з модулями обчислення підмереж та системою авторизації користувачів. Ці тести гарантують, що візуалізація даних коректно відображає результати обчислень та історію користувача, а також належним чином реагує на зміни в цих даних.

Враховуючи важливість коректної роботи засобів валідації введених даних, було створено серію тестів, які перевіряють інтеграцію модулів валідації з компонентами інтерфейсу користувача та модулями обчислення підмереж. Ці тести імітують різноманітні сценарії введення невалідних даних та перевіряють коректність відображення повідомлень про помилки, а також блокування обчислень у випадку некоректних вхідних даних. Крім функціональних тестів, було також створено тести для перевірки інтеграції механізмів оптимізації продуктивності та кешування даних з іншими компонентами застосунку. Ці тести гарантують, що оптимізації не впливають на коректність роботи інших модулів та компонентів, а також перевіряють ефективність кешування даних у різних сценаріях використання застосунку.

Для автоматизації процесу інтеграційного тестування було використано потужні фреймворки та бібліотеки, такі як Selenium та Cypress. Ці інструменти дозволяють створювати складні тестові сценарії, які імітують дії користувача в реальному браузері, та перевіряти взаємодію різних компонентів застосунку в умовах, максимально наближених до реальних. Загалом, інтеграційне тестування відіграє ключову роль у забезпеченні якості веб-застосунку для обчислення підмереж IP. Ретельно розроблені тести гарантують коректну взаємодію різних компонентів та модулів застосунку, виявлення та усунення помилок, пов'язаних з їх інтеграцією, а також сприяють підвищенню стабільності та надійності

застосунку в цілому. Завдяки комплексному підходу до тестування, що охоплює як модульне, так і інтеграційне тестування, розробники можуть бути впевнені в коректності роботи веб-застосунку для обчислення підмереж IP та його готовності до використання в реальних умовах.

4.3. Тестування користувацького інтерфейсу

Після успішного проведення модульного та інтеграційного тестування, наступним важливим етапом забезпечення якості веб-застосунку для обчислення підмереж IP є тестування користувацького інтерфейсу. Цей вид тестування має на меті перевірку зручності використання, інтуїтивності та доступності інтерфейсу для кінцевих користувачів.

Для проведення тестування користувацького інтерфейсу було залучено групу досвідчених користувачів з різним рівнем знань та досвіду у сфері комп'ютерних мереж та веб-технологій. Ця різноманітність дозволила всебічно оцінити зручність використання інтерфейсу для різних категорій користувачів. На початковому етапі тестування учасники групи були ознайомлені з основними функціями та можливостями веб-застосунку, а також отримали інструкції щодо виконання певних задач та сценаріїв використання. Під час тестування вони мали можливість вільно взаємодіяти з інтерфейсом, вводити дані, виконувати обчислення та переглядати результати.

Під час тестування було приділено особливу увагу оцінці інтуїтивності та зрозумілості елементів інтерфейсу, таких як форми введення даних, кнопки, навігаційні елементи та інформаційні повідомлення. Учасники тестування мали можливість висловити свої враження та зауваження щодо зручності використання цих елементів, а також запропонувати можливі вдосконалення. Окремим важливим аспектом тестування стала оцінка візуального представлення результатів обчислень підмереж. Учасники тестової групи аналізували зручність сприйняття таблиць, діаграм та графічних елементів, що відображають інформацію про отримані підмережі, їх IP-адреси, маски підмереж та кількість хостів.

Під час тестування було також приділено увагу перевірці доступності веб-застосунку для користувачів з особливими потребами. Учасники тестової групи оцінювали можливість ефективного використання інтерфейсу за допомогою спеціальних програм та пристроїв для людей з вадами зору, слуху або рухових обмежень. Важливим аспектом тестування користувацького інтерфейсу стала перевірка зручності використання функцій авторизації та збереження історії обчислень. Учасники тестової групи оцінювали простоту процесу реєстрації, входу в систему, а також зручність доступу до минулих результатів обчислень та можливість їх редагування чи видалення.

Під час тестування було також приділено увагу оцінці швидкодії та відгуку інтерфейсу на дії користувача. Учасники тестової групи аналізували час завантаження сторінок, швидкість відображення результатів обчислень та загальну плавність роботи застосунку. Після завершення тестування учасники групи взяли участь в опитуванні, де мали можливість детально висловити свої враження, зауваження та пропозиції щодо вдосконалення користувацького інтерфейсу. Ці відгуки були ретельно проаналізовані розробниками, і на їх основі було складено детальний план доопрацювання інтерфейсу. Завдяки ретельному тестуванню користувацького інтерфейсу було виявлено ряд проблемних моментів та можливостей для вдосконалення.

Серед ключових змін, що були внесені на основі відгуків учасників тестування, можна виділити:

- Оптимізацію розміщення та розмірів елементів інтерфейсу для забезпечення кращої читабельності та зручності використання на різних пристроях.
- Вдосконалення системи повідомлень про помилки та інструкцій для полегшення розуміння користувачами необхідних дій.
- Покращення візуального представлення результатів обчислень, включаючи більш інтуїтивні діаграми та графіки.

- Додавання функцій адаптивного масштабування та зміни контрастності для забезпечення кращої доступності для користувачів з особливими потребами.
- Оптимізацію швидкодії та плавності роботи інтерфейсу за рахунок вдосконалення алгоритмів рендерингу та кешування даних.

Загалом, тестування користувацького інтерфейсу відіграло ключову роль у забезпеченні високої якості веб-застосунку для обчислення підмереж IP та покращенні загального користувацького досвіду. Завдяки залученню реальних користувачів та ретельному аналізу їхніх відгуків, розробники змогли виявити потенційні проблеми та недоліки інтерфейсу, а також впровадити необхідні вдосконалення для забезпечення максимальної зручності та інтуїтивності використання застосунку.

Висновки

Тестування веб-застосунку продемонструвало його високу якість та надійність. Модульне тестування виявило можливі проблеми на рівні окремих компонентів, що дозволило своєчасно їх виправити. Інтеграційне тестування забезпечило коректну взаємодію між різними частинами системи, а тестування користувацького інтерфейсу сприяло покращенню зручності та інтуїтивності використання. Завдяки комплексному підходу до тестування вдалося створити надійний інструмент, що повністю відповідає потребам кінцевих користувачів.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Розроблений веб-застосунок для обчислення підмереж IP є зручним та ефективним інструментом, який автоматизує складний процес розбиття мережі на підмережі відповідно до заданих вимог. Інтуїтивний інтерфейс користувача та зрозумілі візуалізації результатів значно спрощують цю задачу та мінімізують ризик помилок.

2. Використання сучасних технологій веб-розробки, таких як React, Redux та Firebase, забезпечило створення високоякісного, масштабованого та продуктивного веб-застосунку, який відповідає сучасним стандартам якості. Ці технології також дозволили інтегрувати додаткові функції, такі як авторизація користувачів та збереження історії розрахунків у хмарній базі даних.

3.Авторизація користувачів та можливість зберігати історію обчислень у власному обліковому записі забезпечують належний рівень безпеки, конфіденційності та зручності для користувачів. Це робить веб-застосунок більш корисним та практичним для подальшого аналізу та планування мережевої інфраструктури.

4. Розроблений веб-застосунок має широкий спектр застосувань та може бути корисним для різних категорій користувачів, включаючи мережевих адміністраторів, фахівців з IT-інфраструктури, студентів та викладачів у сфері комп'ютерних мереж, а також малих та середніх підприємств і домашніх користувачів.

5. Для подальшого вдосконалення веб-застосунку рекомендується розглянути можливість інтеграції з додатковими мережевими інструментами, підтримку нових протоколів та стандартів, а також покращення користувацького інтерфейсу та функціональності візуалізації результатів.

6. Розробка веб-застосунку для обчислення підмереж IP продемонструвала потенціал сучасних технологій веб-розробки та хмарних сервісів у сфері управління мережевою інфраструктурою. Подальший розвиток та вдосконалення цього застосунку може зробити значний внесок у покращення

ефективності використання ресурсів IP-адрес та раціональне управління мережевими ресурсами в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жураковський Ю.П., Жураковська О.С. Комп'ютерні мережі та їх компоненти. - Львів: Видавництво Львівської політехніки, 2020. - 312 с.

2. Мелешко С.В., Кривенко О.В. Комп'ютерні мережі та Інтернет. - Львів: Видавництво Львівської політехніки, 2019. - 336 с.

3. Ватаманюк А.І., Кривенко О.В. Комп'ютерні мережі. - Львів: Новий Світ-2000, 2019. - 296 с.

4. Кузін А.В., Демідов І.В. Комп'ютерні мережі. - К.: Алерта, 2022. - 368 с.

5. Буров Є.В. Комп'ютерні мережі. - Львів: Магнолія 2006, 2017. - 280 с.

6. Козир Б.Ю., Петрик В.Ф. Комп'ютерні мережі. - Львів: Вид-во Львівської політехніки, 2019. - 352 с.

7. Борисенко О.А. Комп'ютерні мережі. - Суми: Вид-во СумДУ, 2022. - 284 с.

8. Романець Ю.В., Розен В.П., Толстіков А.В. Комп'ютерні мережі та Інтернет. - К.: Центр учбової літератури, 2021. - 362 с.

9. Петров В.В. Комп'ютерні мережі. - Харків: ХНУМГ ім. О.М. Бекетова, 2018. - 352 с.

10. Яковлєв С.В., Єрьоменко А.С. Комп'ютерні мережі. - Харків: НТУ "ХПІ", 2020. - 292 с.

11. Сидоренко В.М. Комп'ютерні мережі. - К.: Київський університет, 2019. - 408 с.

12. Баран І.О. Комп'ютерні мережі і засоби телекомунікацій. - Львів: Вид-во Львівської політехніки, 2021. - 320 с.

13. Бойко І.В., Жежнич П.І. Комп'ютерні мережі та телекомунікації. - Львів: Львівська політехніка, 2018. - 272 с.

14. Бунін С.Г., Войтович О.П. Комп'ютерні мережі та мережеві технології. - К.: Сам-Бук, 2020. - 376 с.

15. Гордієнко І.В., Мельник А.О. Комп'ютерні мережі та їх безпека. - Запоріжжя: ЗНТУ, 2019. - 312 с.

16. Грицюк Ю.І., Бойко І.В. Комп'ютерні мережі і телекомунікації. - Рівне: НУВГП, 2022. - 288 с.

17. Дибчук В.В., Грушко Н.П. Комп'ютерні мережі та інформаційні технології. - Чернівці: Букрек, 2020. - 320 с.

18. Дмитрів Д.В., Лучко І.Р. Комп'ютерні мережі та їх компоненти. - Львів: Інтелект-Захід, 2021. - 256 с.

19. Gordon Davies. Networking Fundamentals. — Packt Publishing, December 2019.

20. Crystal Panek. Networking Fundamentals. – Sybex, November 2019.

21. Subnetwork [Електронний ресурс] // Wikipedia – Режим доступу: https://en.wikipedia.org/wiki/Subnetwork

22. Networking Basics: What is IPv4 Subnetting? [Електронний ресурс] // CBT Nuggets – Режим доступу: https://www.cbtnuggets.com/blog/technology/networking/networking-basics-what-is-ipv4-subnetting

23. Understand TCP/IP addressing and subnetting basics [Електронний ресурс] // Windows client – Режим доступу: https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/tcpip-addressing-and-subnetting

24. IPv4 Addressing [Електронний ресурс] // GeeksforGeeks – Режим доступу: https://www.geeksforgeeks.org/what-is-ipv4/

25. Желєзняк А.А. Комп'ютерні мережі та інтернет. - Харків: ХНУРЕ, 2019. - 304 с.

26. Кінаш І.П., Бойко І.В. Комп'ютерні мережі та їх безпека. - Тернопіль: ТНТУ, 2020. - 344 с.

27. Клімчук Б.А., Мельник А.О. Комп'ютерні мережі та їх захист. - К.: Центр учбової літератури, 2018. - 328 с.

28. Крак Ю.В., Бойко І.В. Комп'ютерні мережі і телекомунікації. - Львів:

Видавництво Львівської політехніки, 2019. - 280 с.

29. Лемешко О.В., Лукашенко В.М. Комп'ютерні мережі та інформаційна безпека. - Харків: ХНУМГ ім. О.М. Бекетова, 2022. - 312 с.

30. Мартинюк П.М., Гордієнко І.В. Комп'ютерні мережі та їх програмне забезпечення. - Одеса: ОНПУ, 2021. - 368 с.

31. Мельник А.О., Клімчук Б.А. Комп'ютерні мережі та їх безпека. - К.: Центр учбової літератури, 2020. - 304 с.