КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка системи батьківського контролю для веб сервісів»

Хоменко Влад Романович

(прізвище, ім’я та по батькові студента повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
к.т.н., доц. Гончаренко Т. А.

_____
„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка системи батьківського контролю для веб сервісів»

Виконав: студент 4-го курсу, групи КН-20-1

Спеціальності: 122 «Комп`ютерні науки

Освітня програма: «Інформаційні управляючі системи і технології»

(шифр і назва напряму підготовки, спеціальності)

Хоменко В. Р.
(прізвище та ініціали)

Керівник д.т.н., проф. Бородавка Є. В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Кафедра: інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
к.т.н., доц. Гончаренко Т. А.
_____
„___" _____2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Хоменко Влад Романович
Тема роботи: Розробка системи батьківського контролю для веб сервісів
затверджена наказом ректора КНУБА № 433/2 від « 29 » лютого 2024 р.
2. Керівник роботи: Бородавка Євгеній Володимирович, д.т.н, професор кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту: червень 2024
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Архітектура системи
Р.3. Проектування бази даних
Р.4. Практична реалізація та тестовий приклад
5. Інформаційні слайди:
С.1.Архітектура системи
С.2.Концептуальна схема системи
С.3. Функціональна схема системи
С.4. Проектування бази даних
С.5. Практична реалізація

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Лютий 2024 р.
Р. 2. Архітектура системи	Березень 2024 р.
Р. 3. Проектування бази даних	Квітень 2024 р.
Р. 4. Практична реалізація та тестовий приклад	Травень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Прийом програмного продукту			

8. Дата видачі завдання: 11 лютого 2024 р.

Завідувач _____ Гончаренко Т.А.

(підпис) (прізвище та ініціали)

Керівник _____ Бородавка Є.В.

(підпис) (прізвище та ініціали)

Бакалавр _____ Хоменко В. Р.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Хоменко В. Р. Автоматизація взаємодії інформаційної системи інтернет-магазину з потенційними та існуючими користувачами.

Кваліфікаційна випускна робота бакалавра за спеціальністю: 122 «Комп’ютерні науки», спеціалізація: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2024.

Робота присвячена системі батьківського контролю для веб сервісів у вигляді розширення для браузера. Інструментом реалізації розроблюваної системи є бібліотеки та фреймворки мови програмування JavaScript.

Ключові слова: Хром розширення, безпека, API, TypeScript, React, NestJS.

SUMMARY

Khomenko V. R. Development of a system of parental control of web services.

Bachelor's thesis in the specialty: 122 "Computer Science", specialization: "Information managing systems and technologies". - Kyiv National University of Construction and Architecture - Kyiv, 2024.

The work is devoted to the system of parental control of web services in the form of a browser extension. The tools for implementing the developed system are libraries and frameworks of the JavaScript programming language.

Keywords: Chrome extension, security, API, TypeScript, React, NestJS.

Вступ

У сучасному цифровому віці, де доступ до Інтернету став необхідністю, важливо забезпечити безпеку та контроль за веб-діяльністю дітей. Розвиток інтернет-технологій та поширення веб-сервісів створює потребу в надійних і ефективних інструментах батьківського контролю.

Ця робота присвячена розробці системи батьківського контролю для веб-сервісів, яка дозволить батькам забезпечувати безпеку та контроль за онлайн-активністю своїх дітей. Система буде включати в себе хром-розширення, сервер, авторизацію, базу даних, клієнтську частину та саме браузерне розширення, яке буде взаємодіяти з веб-сервісами.

Розробка системи буде відбуватись за допомогою мови програмування JavaScript (TypeScript) у середовищі Visual Studio Code.

Перелік умовних позначень

API – application programming interface

ІС – інформаційна система

БД – база даних

IDE – інтегроване середовище розробки (Integrated Development Environment)

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Дана дипломна робота спрямована на розробку системи батьківського контролю для веб-сервісів, що має на меті забезпечити батькам можливість ефективно контролювати та обмежувати доступ своїх дітей до інтернет-ресурсів.

Предметною областю даного проекту є сфера онлайн-безпеки, батьківського контролю та керування доступом до веб-ресурсів для дітей. Система батьківського контролю для веб-сервісів покликана надати батькам ефективні інструменти для моніторингу та обмеження доступу своїх дітей до певних веб-сайтів, контенту або онлайн-сервісів.

Основними аспектами предметної області є:

1. Контроль доступу: Система повинна забезпечувати можливість блокувати або дозволяти доступ до певних веб-ресурсів на основі встановлених батьками правил та налаштувань.
2. Фільтрація контенту: Система має надавати можливість фільтрувати небажаний або шкідливий контент, такий як порнографічний, насильницький або екстремістський матеріал.
3. Моніторинг активності: Система повинна забезпечувати моніторинг онлайн-активності дітей, включаючи відвідувані веб-сайти, час, проведений в Інтернеті, та інші відповідні дані.
4. Управління часом: Система має надавати можливість встановлювати обмеження на тривалість використання Інтернету або певних сервісів для забезпечення збалансованого розподілу часу.
5. Конфіденційність та безпека: Система повинна забезпечувати захист особистої інформації дітей та батьків, а також безпечне використання веб-сервісів.

Перевагами веб-контролю є можливість підвищення рівня безпеки та захисту дітей від небажаного або шкідливого контенту, а також можливість контролювати час, проведений дітьми в мережі.

Однак, слід зазначити, що предметна область також має певні виклики та недоліки. По-перше, система повинна постійно оновлюватися та адаптуватися до швидких змін у веб-технологіях та появи нових онлайн-сервісів. По-друге, необхідно забезпечити належний баланс між контролем та свободою дітей, уникаючи надмірних обмежень.

Таким чином, система батьківського контролю для веб-сервісів виступає як інноваційний інструмент для забезпечення безпеки та контролю доступу дітей до інтернет-ресурсів, відповідаючи на актуальні потреби сучасного суспільства.

Технологічні аспекти реалізації системи

Розробка системи батьківського контролю вимагає використання сучасних технологій та підходів. В основі системи можуть лежати різноманітні методи ідентифікації та фільтрації контенту, такі як:

1. Методи фільтрації контенту: Використання алгоритмів машинного навчання для аналізу та класифікації контенту. Це можуть бути нейронні мережі для розпізнавання образів та текстів, що допоможе виявляти небажаний або шкідливий контент. Методи фільтрації можуть базуватися також на ключових словах, URL-адресах, категоріях сайтів і навіть на аналізі поведінки користувачів в Інтернеті.
2. Аутентифікація та авторизація користувачів: Важливою складовою є забезпечення надійного контролю доступу до системи. Використання двофакторної аутентифікації та інших методів підвищення безпеки дозволить запобігти несанкціонованому доступу. Крім того, система повинна підтримувати функції розділення прав доступу, що дозволить батькам налаштовувати різні рівні доступу для дітей різного віку.

3. Моніторинг та звітність: Система повинна мати функції для моніторингу активності дітей в реальному часі та генерації звітів для батьків. Це можуть бути графіки часу користування, відвідані веб-ресурси та інші статистичні дані. Батьки повинні мати можливість отримувати повідомлення про підозрілу активність або спроби доступу до заборонених ресурсів.
4. Інтерфейс користувача: Для забезпечення зручності використання, система повинна мати інтуїтивно зрозумілий інтерфейс для батьківю Інтерфейс повинен бути адаптивним та підтримувати роботу на різних пристроях – комп’ютерах, планшетах, смартфонах.

Виклики та можливі проблеми

Розробка та впровадження системи батьківського контролю супроводжується рядом викликів:

1. Постійні оновлення та підтримка: Веб-сервіси швидко розвиваються, з'являються нові платформи та функції, що потребує постійного оновлення системи для забезпечення її актуальності та ефективності. Це може вимагати значних ресурсів та тісної співпраці з розробниками веб-сервісів.
2. Етичні питання: Баланс між контролем та приватністю дітей є важливим аспектом. Система повинна бути налаштована таким чином, щоб не порушувати особистий простір дітей, водночас забезпечуючи їхню безпеку.
3. Сумісність з різними пристроями: Сучасні діти використовують різні пристрої для доступу до інтернету. Система повинна бути сумісною з широким спектром пристроїв та операційних систем.
4. Обхід системи та контролю: Деякі діти можуть спробувати обійти систему батьківського контролю, використовуючи різні методи, такі як проксі-сервери, віртуальні приватні мережі (VPN) або інші способи приховування своєї активності. Система повинна бути спроектована таким

чином, щоб мінімізувати можливості для обходу та забезпечити надійний контроль.

5. Вплив на продуктивність та безпеку пристроїв: Система батьківського контролю може спричинити додаткове навантаження на пристрої та мережу, що може вплинути на їхню продуктивність та безпеку. Необхідно забезпечити оптимізацію системи для мінімізації такого впливу та запобігання потенційним ризикам для безпеки пристроїв.

Загалом, розробка ефективної системи батьківського контролю вимагає ретельного планування, використання передових технологій та врахування етичних, технічних і безпекових аспектів. Постійний моніторинг змін у галузі веб-технологій та тісна співпраця з батьками та дітьми є ключовими факторами для забезпечення безпечного та збалансованого використання Інтернету.

Перспективи розвитку

У майбутньому розвиток системи батьківського контролю передбачає інтеграцію з більш комплексними рішеннями для забезпечення кібербезпеки сімей. Ці перспективи розвитку можна розділити на кілька ключових напрямків:

1. Інтеграція з іншими системами безпеки: система батьківського контролю може бути інтегрована з антивірусними програмами, фаєрволами та іншими засобами кібербезпеки для створення єдиного захисного середовища. Це дозволить забезпечити не тільки фільтрацію контенту, але й захист від вірусів, шкідливого ПЗ та інших кіберзагроз.
2. Розширення функціональних можливостей: система може бути розширена для контролю доступу не лише до веб-сервісів, а й до ігор, мобільних додатків, месенджерів та соціальних мереж. Це дозволить батькам контролювати активність дітей у різноманітних цифрових середовищах. Наприклад, можна передбачити функції обмеження часу гри, використання певних додатків або обмеження доступу до деяких функцій соціальних мереж.

3. Використання технологій штучного інтелекту: впровадження алгоритмів машинного навчання та штучного інтелекту для підвищення точності фільтрації контенту та автоматичного виявлення нових загроз. ШІ може аналізувати поведінку дітей в Інтернеті та пропонувати батькам оптимальні налаштування для підвищення безпеки.
4. Стандартизація та сертифікація**:** розробка стандартів та сертифікацій для систем батьківського контролю може сприяти підвищенню якості та ефективності таких рішень. Стандартизація допоможе встановити чіткі критерії безпеки та функціональності, що підвищить довіру користувачів та полегшить вибір оптимальної системи.
5. Вдосконалення інтерфейсу користувача: постійне вдосконалення інтерфейсу користувача системи батьківського контролю для забезпечення зручності, інтуїтивності та простоти використання. Це може включати розробку мобільних додатків, веб-інтерфейсів та інтеграцію з голосовими асистентами для полегшення керування системою.

Загалом, перспективи розвитку системи батьківського контролю для веб-сервісів є досить широкими та багатогранними. Впровадження новітніх технологій, розширення функціональних можливостей та інтеграція з іншими системами безпеки дозволять створити більш комплексне та ефективне рішення для забезпечення безпечного та контрольованого доступу дітей до цифрового світу.

1.2 Аналіз об’єкта дослідження

Система батьківського контролю для веб-сервісів є складною інформаційною системою, спрямованою на контроль та обмеження доступу дітей до різноманітного веб-контенту. Предметна область даної системи охоплює сферу онлайн-безпеки та родинних відносин, де батьки можуть ефективно контролювати та управляти веб-активністю своїх дітей..

Основні аспекти предметної області включають в себе:

1. Безпека в Інтернеті: Зростаюча кількість веб-ресурсів та вмісту ставить під загрозу безпеку дітей в мережі. Веб-сайти можуть містити шкідливий контент, віруси, шахрайські схеми та інші небезпеки. Для захисту дітей від цих загроз необхідно впроваджувати системи фільтрації та моніторингу.
2. Обмеження доступу: Система повинна надавати можливість батькам налаштовувати обмеження доступу до певних веб-сайтів, додатків чи сервісів. Це дозволяє контролювати, які ресурси доступні дітям у певний час. Наприклад, можна дозволити доступ до освітніх ресурсів під час навчання і обмежити доступ до соціальних мереж чи ігрових сайтів у цей час.
3. Надання доступу до вибраних сайтів: Система надає можливість батькам встановлювати список веб-сайтів, які є безпечними та допустимими для використання дітьми. Усі інші сайти будуть автоматично блокуватись, забезпечуючи високий рівень безпеки та контролю. Такий підхід гарантує, що дитина зможе отримати доступ тільки до перевірених та схвалених ресурсів. Аналогічно, батьки можуть створювати списки заборонених сайтів, доступ до яких буде заблоковано незалежно від інших налаштувань. Це зручно, коли потрібно заборонити доступ до конкретних ресурсів, які не є загрозливими загалом, але можуть мати небажаний вплив на дитину.
4. Моніторинг активності: Система записує всі відвідані веб-сайти та активність користувачів, згідно з налаштуваннями батьків. Це дозволяє аналізувати поведінку дітей в Інтернеті та приймати обґрунтовані рішення щодо подальших обмежень. Батьки можуть отримувати детальні звіти про активність дітей в мережі, що допоможе їм краще розуміти їхні інтереси та потреби.
5. Інтеграція з іншими пристроями та сервісами: система батьківського контролю повинна бути сумісною з мобільними пристроями, що дозволить контролювати активність дітей на смартфонах та планшетах. Оскільки діти часто використовують мобільні пристрої для доступу до Інтернету,

важливо забезпечити ефективний контроль і на цих платформах. Інтеграція з ігровими консолями, смарт-телевізорами та іншими підключеними до Інтернету пристроями дозволить забезпечити комплексний контроль. Це забезпечить повний контроль над всіма пристроями, які можуть використовувати діти, та створить єдину систему безпеки.

6. Технічна підтримка та оновлення: Постійне оновлення баз даних небажаних сайтів та контенту, щоб система могла ефективно фільтрувати нові загрози. Інтернет постійно змінюється, і нові загрози з'являються щодня, тому важливо мати актуальні дані для забезпечення безпеки. Наявність технічної підтримки, яка може допомогти батькам з налаштуваннями системи та вирішенням проблем. Це особливо важливо для користувачів, які можуть не мати глибоких технічних знань, але бажають забезпечити безпеку своїх дітей.
7. Адаптивність та масштабованість: Система має бути гнучкою та масштабованою, щоб підлаштовуватися під потреби різних сімей та їхніх конкретних вимог. Кількість дітей у сім'ї, їхній вік, рівень цифрової грамотності та індивідуальні інтереси можуть значно відрізнятися. Система повинна надавати можливість створювати індивідуальні профілі для кожної дитини та налаштовувати рівень контролю відповідно до їхніх потреб.
8. Механізми обходу та протидії: Важливо передбачити можливі спроби дітей обійти систему контролю та заходи для запобігання таким ситуаціям. Це може включати виявлення використання проксі-серверів, VPN або інших способів приховування активності в Інтернеті. Система повинна бути здатна виявляти такі спроби та повідомляти про них батьків.

Система батьківського контролю є важливим інструментом для забезпечення онлайн-безпеки дітей, дозволяючи батькам ефективно контролювати доступ до веб-контенту, моніторити активність та встановлювати необхідні обмеження.

Сучасні технології дозволяють створити потужні та гнучкі рішення, що відповідають потребам кожної родини, забезпечуючи захист дітей в цифровому світі.

Таблиця 1.1. Методи контролю та принципи їх роботи

Метод	Опис	Принцип роботи
Моніторинг	Відслідковування відвіданих сайтів	Система записує всі відвідані веб-сайти та активність користувачів, згідно з налаштуваннями батьків
Фільтрація контенту	Фільтрація доступу до небажаного контенту на основі заданих параметрів	Система блокує доступ до визначеного контенту на основі попередньо встановлених правил та налаштувань
Фільтрація за принципом виключень	Батьки визначають конкретні веб-сайти, які є безпечними та дозволеними для використання дитиною	Система блокує всі веб-сайти за замовчуванням, але ті, що знаходяться в списку білих сайтів, будуть доступні

Важливим аспектом є також врахування психологічних та етичних аспектів використання таких систем. Надмірний контроль може викликати у дітей почуття недовіри та обмеження свободи, що може негативно вплинути на їх розвиток та відносини в родині. Тому система повинна бути налаштована таким чином, щоб забезпечити баланс між захистом та довірою. Освітні заходи для батьків допоможуть їм краще розуміти, як правильно використовувати інструменти батьківського контролю, щоб забезпечити максимальну ефективність без негативних наслідків.

Крім того, важливо враховувати юридичні аспекти використання систем батьківського контролю. В різних країнах можуть існувати різні закони та регулювання щодо збирання та обробки персональних даних, тому система повинна відповідати вимогам законодавства і забезпечувати конфіденційність інформації. Впровадження технологій шифрування та захисту даних допоможе забезпечити високий рівень безпеки та захисту персональних даних користувачів.

В кінцевому рахунку, система батьківського контролю повинна стати невід'ємною частиною сучасного цифрового середовища, спрямованою на забезпечення безпеки дітей в Інтернеті, розвиток відповідального ставлення до використання цифрвих технологій.

1.3 Опис предмету дослідження

Chrome – веб-браузер, розроблений компанією Google, що здобув популярність завдяки своїй простоті, зручності та швидкості роботи. Наразі він є провідним інтернет-браузером у світі з часткою світового ринку 65,31%.

Розширення для браузера – це програмне забезпечення, яке розширює можливості веб-браузера, додаючи до нього нові функції або змінюючи існуючий функціонал. Розширення можуть додавати кнопки на панелі інструментів, модифікувати вміст веб-сторінок або надавати доступ до різних онлайн-сервісів.

Розширення саме в бразері Chrome було обрано через його велику аудиторію, надійність та доступність, оскільки він підтримується на всіх пристроях (телефон,

планшет, комп’ютер), і безкоштовність (безкоштовне створення та використання розширення в Chrome). Chrome Web Store, офіційний магазин розширень для Chrome, пропонує тисячі розширень, що покривають широкий спектр функціональних можливостей, від продуктивності та безпеки до розваг та навчання.

Основні переваги розширень для браузера Chrome

1. Велика аудиторія: завдяки своїй популярності, розширення для Chrome можуть досягти значної аудиторії користувачів по всьому світу. Це робить платформу привабливою для розробників, які бажають створювати корисні та цікаві інструменти для широкого кола користувачів. Chrome доступний на різних платформах, включаючи Windows, macOS, Linux, Android та iOS, що дозволяє користувачам синхронізувати свої розширення та налаштування між різними пристроями.

2. Надійність та доступність: Chrome відомий своєю стабільністю та швидкістю, що забезпечує надійну роботу розширень без значних збоїв або затримок. Встановлення розширень у Chrome здійснюється через Chrome Web Store, що забезпечує простий та зручний процес для користувачів. Розширення легко активуються та налаштовуються відповідно до потреб користувача.

3. Безкоштовність: Створення та використання розширень для Chrome є безкоштовним, що знижує бар'єр для входу для розробників і користувачів. Розробники можуть використовувати потужні інструменти та API для створення розширень, не турбуючись про додаткові витрати. Широка спільнота розробників та користувачів сприяє активному розвитку розширень, обміну досвідом та підтримці нових ідей.

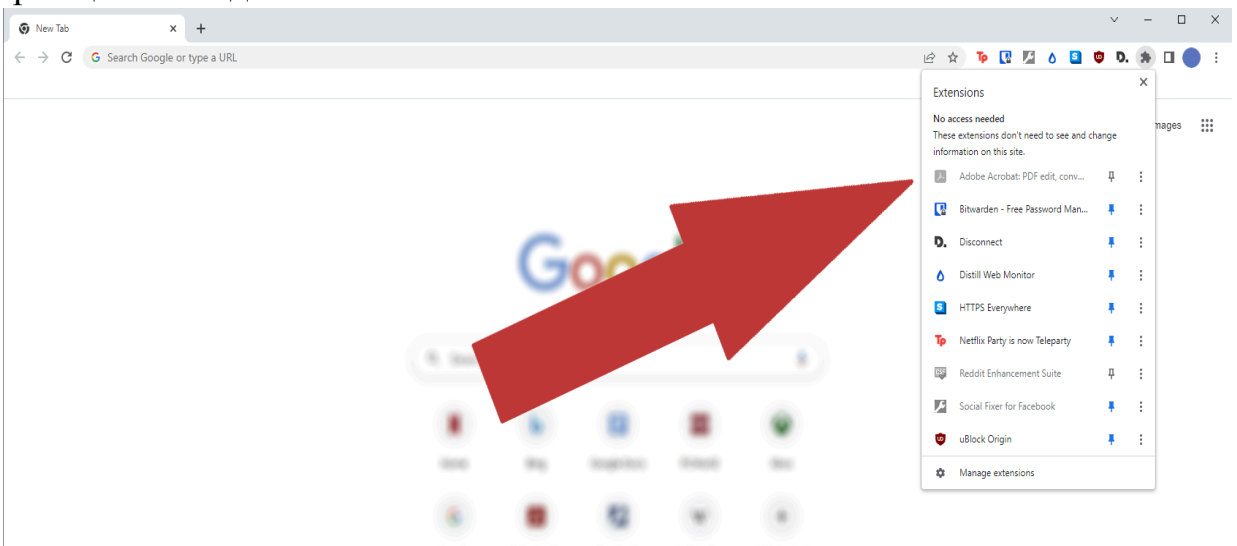


Рисунок 1.2. Хром розширення

API – це набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) для використання в зовнішніх програмних продуктах.

База даних (БД) – це організована структура, призначена для зберігання, зміни й обробки взаємопов’язаної інформації, переважно великих обсягів. Бази даних активно використовують для динамічних сайтів зі значними обсягами даних – часто це інтернет-магазини, портали, корпоративні сайти.

БД виконує функцію зберігання інформації користувача та його аккаунт, список сайтів, доданих до списку заблокованих або безпечних веб-сервісів. В даній роботі використовуватиметься БД PostgreSQL. PostgreSQL є потужною системою управління базами даних, яка підтримує різні типи даних, складні запити та транзакції, забезпечуючи високу продуктивність та надійність.

Використання розширень для браузера Chrome у поєднанні з базою даних PostgreSQL забезпечує потужний інструмент для реалізації системи батьківського контролю, яка дозволить ефективно контролювати та обмежувати доступ до веб-контенту, забезпечуючи безпеку дітей в Інтернеті.

1.4 Аналіз актуальності

У зв'язку зі зростанням використання інтернету серед дітей та підлітків, батькам стає все важливішою необхідність контролювати їхню онлайн активність. Саме тому системи батьківського контролю для веб-сервісів стають надзвичайно актуальними. Основними факторами, що підсилюють актуальність таких систем, є:

- зростання кількості веб-сервісів та додатків, доступних для дітей та підлітків, які можуть містити несприятливий контент або відволікати дітей.
- підвищена обізнаність батьків щодо впливу інтернету на розвиток та поведінку дітей, що спонукає їх до активного контролю та моніторингу онлайн активності;

- розвиток технологій, що дозволяють батькам здійснювати більш точний та ефективний контроль, таких як фільтрація контенту, обмеження часу використання;
- Разом з ростом кількості веб-сервісів для дітей і підлітків збільшується і різноманітність доступного контенту. Від освітніх ресурсів та інтерактивних ігор до стрімінгових платформ та соціальних мереж - діти мають доступ до широкого спектру онлайн-ресурсів. Однак не весь цей контент може бути відповідним для їхнього віку або розвитку, тому важливо мати засоби контролю, щоб забезпечити, що діти використовують Інтернет безпечно та продуктивно.
- Багато веб-сервісів та додатків використовують психологічні та поведінкові техніки, щоб залучити користувачів та збільшити їхню активність. Для дітей, які часто мають меншу самоконтроль та розуміння наслідків, це може стати особливо проблематичним. Системи батьківського контролю можуть допомогти обмежити час, проведений дітьми в Інтернеті, та регулювати їхню взаємодію з різними сервісами.
- Інтернет може бути потужним інструментом для навчання та розвитку, але лише за умови правильного його використання. Системи батьківського контролю можуть допомогти батькам направляти онлайн активність своїх дітей в напрямку освіти та розвитку, обмежуючи доступ до відволікаючих або несанкціонованих ресурсів.

За останні десятиліття значно зросла не лише кількість доступних додатків та веб-сервісів для дітей, але й їхня різноманітність та складність. Це може ускладнити завдання батьків у контролі за онлайн активністю своїх дітей.

Одним з ключових факторів, що робить системи батьківського контролю актуальними, є зростаюча поширеність технологій та доступність пристроїв з доступом до Інтернету для дітей. Смартфони, планшети та ноутбуки стали невід'ємною частиною повсякденного життя, і діти все частіше отримують доступ до цих пристроїв у молодшому віці. Це збільшує ризики їхньої взаємодії з

небажаним або шкідливим контентом в Інтернеті, що підкреслює необхідність ефективних систем батьківського контролю.

Крім того, актуальність систем батьківського контролю посилюється змінами у способах споживання контенту та взаємодії з Інтернетом. Замість традиційного веб-серфінгу, діти все частіше використовують потокові сервіси, соціальні медіа та додатки для розваг і спілкування. Ці платформи можуть містити контент, який не підходить для певних вікових груп, або мати функції, які можуть спричинити ризики для безпеки та конфіденційності дітей.

Ще одним важливим фактором є зміна поведінки та звичок використання Інтернету серед дітей та підлітків. Покоління, що виросло з доступом до цифрових технологій, може мати інші підходи та очікування щодо використання Інтернету, що створює нові виклики для батьків у питаннях контролю та нагляду.

Нарешті, слід враховувати зростаючу стурбованість суспільства щодо впливу Інтернету та технологій на розвиток дітей. Дослідження показують, що надмірне використання цифрових пристроїв та соціальних мереж може мати негативний вплив на здоров'я, концентрацію уваги, соціальні навички та емоційний стан дітей. Системи батьківського контролю можуть допомогти батькам керувати часом, який їхні діти проводять онлайн, та забезпечити збалансоване використання технологій.

Крім того, важливо врахувати ризики, пов'язані зі зловживанням Інтернету та соціальними мережами, такими як кібербулінг, контакт зі шкідливими особами або доступ до непридатного контенту. Системи батьківського контролю можуть допомогти вчасно виявляти та запобігати подібним ситуаціям, забезпечуючи батькам більший контроль та спокій у справах, пов'язаних з безпекою їхніх дітей в Інтернеті.

Отже, аналіз актуальності підтверджує важливість розробки та впровадження систем батьківського контролю для веб-сервісів, щоб забезпечити безпеку та здоровий розвиток дітей у цифровому світі.

1.5 Стан вже існуючих рішень

На сьогоднішній день існують різноманітні браузерні розширення та програми для батьківського контролю, призначені для керування та моніторингу онлайн активності дітей. Деякі з них надають такі можливості, як.

- блокування небажаних веб-сайтів та контенту з несанкціонованим вмістом;
- Встановлення обмежень часу використання інтернету або певних веб-сайтів;
- Моніторинг активності дитини в мережі, включаючи відвідувані сайти та використані додатки;

Дані рішення є функціонуючими та зручними, але з постійним покращенням функціоналу самої платформи, вдосконаленням можливостей мов програмування для створення програми та обробки взаємодії користувача з нею, можливі оновлення та модернізації.

При розробці браузерного розширення слід врахувати такі фактори:

- якість розширення: Розширення повинно бути зручним у використанні, мати зрозумілий інтерфейс;
- функціональність розширення: Рішення повинно мати достатній функціонал для задоволення потреб;

Крім того, батьківські програми контролю можуть надавати можливість встановлення обмежень часу використання Інтернету або певних веб-сайтів. Це дозволяє батькам контролювати, скільки часу їхні діти проводять в Інтернеті, а також визначати конкретні періоди часу, коли доступ до Інтернету дозволений або заборонений.

Моніторинг активності дитини в мережі - ще одна важлива можливість, яку надають батьківські програми контролю. Вони дозволяють батькам відстежувати відвідувані сайти та використані додатки своїми дітьми, щоб мати належний контроль над їхньою онлайн активністю.

При розробці браузерних розширень та програм для батьківського контролю слід враховувати ряд факторів, щоб забезпечити їх ефективність та зручність використання. Це включає якість розширення, функціональність, а також вартість та доступність для користувачів.

Наприклад, деякі з найпопулярніших розширень, такі як BlockSite та Web Blocker, мають різні функції та характеристики, які варто врахувати при виборі програми для батьківського контролю. Таблиця 1.3 наводить порівняльний аналіз таких рішень, включаючи можливості блокування сайтів, фільтрації за ключовими словами, наявність моніторингу використання та інші параметри.

Таблиця 1.3. Порівняння існуючих рішень

Рішення	BlockSite	Web Blocker
Блокування сайтів	Так	Так
Білий список сайтів	Так	Ні
Фільтрація за ключовими словами	Так	Ні
Моніторинг використання	Ні	Ні
Розклад блокування	Так	Так
Вартість	Платний (безкоштовно під час пробного періоду)	Безкоштовний

Незважаючи на наявність численних рішень для батьківського контролю в інтернеті, існує низка недоліків та обмежень, які ще потрібно вирішити. Одним з основних недоліків є відсутність гнучкості та персоналізації для різних вікових груп та потреб користувачів. Більшість рішень пропонують фіксовані набори налаштувань і фільтрів, які можуть бути або занадто обмежувальними, або недостатньо ефективними залежно від конкретної ситуації.

Крім того, багато існуючих рішень не враховують динамічний характер інтернет-контенту та постійно змінюються веб-технології. Фільтри та списки блокованих сайтів можуть швидко застарівати, оскільки нові веб-ресурси з'являються щодня, а існуючі змінюють свій вміст. Це вимагає від батьків постійного оновлення та налаштування системи батьківського контролю, що може бути трудомістким і непрактичним.

Ще одним недоліком є недостатня інтеграція та узгодженість між різними пристроями та платформами. Оскільки діти все частіше використовують кілька пристроїв (смартфони, планшети, комп'ютери) для доступу до інтернету, важливо, щоб система батьківського контролю була сумісною та синхронізувалася між ними.

Безпека та конфіденційність також є важливими питаннями, які деякі існуючі рішення не враховують належним чином. Системи батьківського контролю повинні забезпечувати належний захист персональних даних та приватності користувачів, одночасно забезпечуючи ефективний контроль та моніторинг.

Нарешті, існує потреба в більш зручному та інтуїтивно зрозумілому інтерфейсі для батьків, який би полегшував налаштування та керування системою батьківського контролю. Складні та громіздкі інтерфейси можуть відлякувати батьків від використання таких рішень або призводити до помилок у налаштуваннях.

Ці недоліки та обмеження підкреслюють необхідність постійного вдосконалення та розробки нових рішень для батьківського контролю в інтернеті.

Системи повинні бути гнучкими, адаптивними, безпечними та зручними у використанні, щоб ефективно задовольняти потреби батьків та забезпечувати безпечне та продуктивне використання інтернету для дітей.

Система батьківського контролю для веб-сервісів і браузерні розширення є важливим інструментом для батьків, які прагнуть забезпечити безпеку та відповідальне використання Інтернету своїми дітьми. Шляхом використання таких інструментів батьки можуть контролювати та моніторити онлайн активність своїх дітей, забезпечуючи їм безпечне та продуктивне використання цифрових технологій.

Розробка та впровадження програмного забезпечення для батьківського контролю в браузерах є важливим кроком у забезпеченні безпеки та захисту дітей у цифровому світі.

Крім того, ця система відкриває можливості для покращення комунікації та взаємодії між батьками та дітьми щодо безпечного використання Інтернету. Вона створює сприятливі умови для вироблення в дітей цифрової грамотності та відповідальності, що є критичними в умовах сучасного цифрового середовища.

Отже, розробка та впровадження програмного забезпечення для батьківського контролю в браузерах є важливим кроком у захисті та підтримці дітей у цифровому світі.

1.6 Визначення цілей дослідження та постановка задачі

Метою досліджень в роботі є розробка та реалізація програмного забезпечення для батьківського контролю для браузера.

Реалізація мети здійснюється вирішенням наступних основних завдань:

1. Створення браузерного розширення: Проектування інтерфейсу користувача для зручного та ефективного використання батьками. Реалізація функціоналу,

який дозволить батькам контролювати та обмежувати доступ до веб-сайтів та контенту.

2. Розробка клієнтської частини програми: Проектування та розробка інтерфейсу користувача для батьків. Встановлення можливостей налаштувань та контролю через клієнтську частину програми.

3. Розробка серверної частини програми для збереження та обробки даних: Вибір та налаштування серверної інфраструктури для збереження та обробки даних. Розробка серверних модулів, які забезпечать комунікацію клієнтської частини з базою даних та здійснення операцій з даними.

4. Створення бази даних: Проектування структури бази даних для зберігання інформації про користувачів та їхню онлайн активність. Реалізація механізмів для збереження та організації даних в базі.

5. Підключення сервера до бази даних: Налаштування підключення серверної частини програми до бази даних. Тестування та впровадження функціоналу, який залежить від роботи з базою даних.

6. Розгортання на середовищі: Розгортання програмного забезпечення для публічного доступу.

Для досягнення поставлених цілей необхідно враховувати певні вимоги та обмеження, які впливатимуть на процес розробки та реалізації програмного забезпечення. Ці вимоги можна розділити на функціональні та нефункціональні.

Функціональні вимоги визначають основні функції та можливості, які повинна забезпечувати система батьківського контролю:

1. Система повинна надавати можливість батькам створювати облікові записи для своїх дітей та керувати їхніми налаштуваннями доступу.
2. Батьки повинні мати змогу блокувати або дозволяти доступ до певних веб-сайтів або категорій контенту.

3. Система повинна забезпечувати моніторинг та журналювання онлайн-активності дітей, включаючи відвідувані веб-сайти та час, проведений на них.
4. Батьки повинні мати можливість встановлювати обмеження часу перебування в Інтернеті для своїх дітей.
5. Система повинна підтримувати різні рівні доступу та налаштування для різних вікових груп або потреб користувачів.
6. Інтерфейс системи повинен бути зручним та інтуїтивно зрозумілим для користувачів.

Нефункціональні вимоги стосуються властивостей та характеристик системи, які не пов'язані безпосередньо з її функціональністю, але є критично важливими для успішної реалізації та експлуатації:

1. Система повинна забезпечувати високий рівень безпеки та конфіденційності даних користувачів.
2. Система повинна бути масштабованою та здатною обробляти зростаючу кількість користувачів та даних.
3. Система повинна бути сумісною з різними веб-браузерами та операційними системами.
4. Система повинна забезпечувати належну продуктивність та швидкодію, щоб не сповільнювати роботу браузера або пристрою користувача.
5. Система повинна бути стійкою до збоїв та мати механізми резервного копіювання та відновлення даних.
6. Система повинна відповідати чинним нормативним вимогам та стандартам у сфері захисту даних та конфіденційності.

Визначення чітких цілей, завдань та вимог є важливим етапом у процесі розробки програмного забезпечення, оскільки це допомагає забезпечити відповідність кінцевого продукту потребам користувачів та забезпечити його якість та ефективність.

Об’єктом дослідження роботи є процес контролю веб сервісів батьками.

Предметом дослідження є розробка та впровадження програмного забезпечення для батьківського контролю в браузерах за допомогою мови програмування JavaScript та його фреймворків і бібліотек.

2. АРХІТЕКТУРА СИСТЕМИ

Система була спроектована та реалізована відповідно до шаблону проектування MVC (Model-View-Controller), який є одним з найпоширеніших та ефективних підходів до організації структури програмного забезпечення. (Рисунок 2.1). У цій архітектурі:

1. Модель (Model) складається з класів BlockList, Auth, Account, які містять в собі моделі та логіку обробки даних. Вони відповідають за доступ до даних, їхню обробку та збереження.
2. Вид (View) відсутній у цій архітектурі, оскільки це система батьківського контролю, інтерфейс якої зазвичай надається через веб-інтерфейс або мобільний додаток.
3. Контролер (Controller) виступає у вигляді ApiController, відповідального за обробку усіх HTTP-запитів. Цей контролер містить методи, які обробляють HTTP-запити та викликають відповідні функції для роботи з даними.

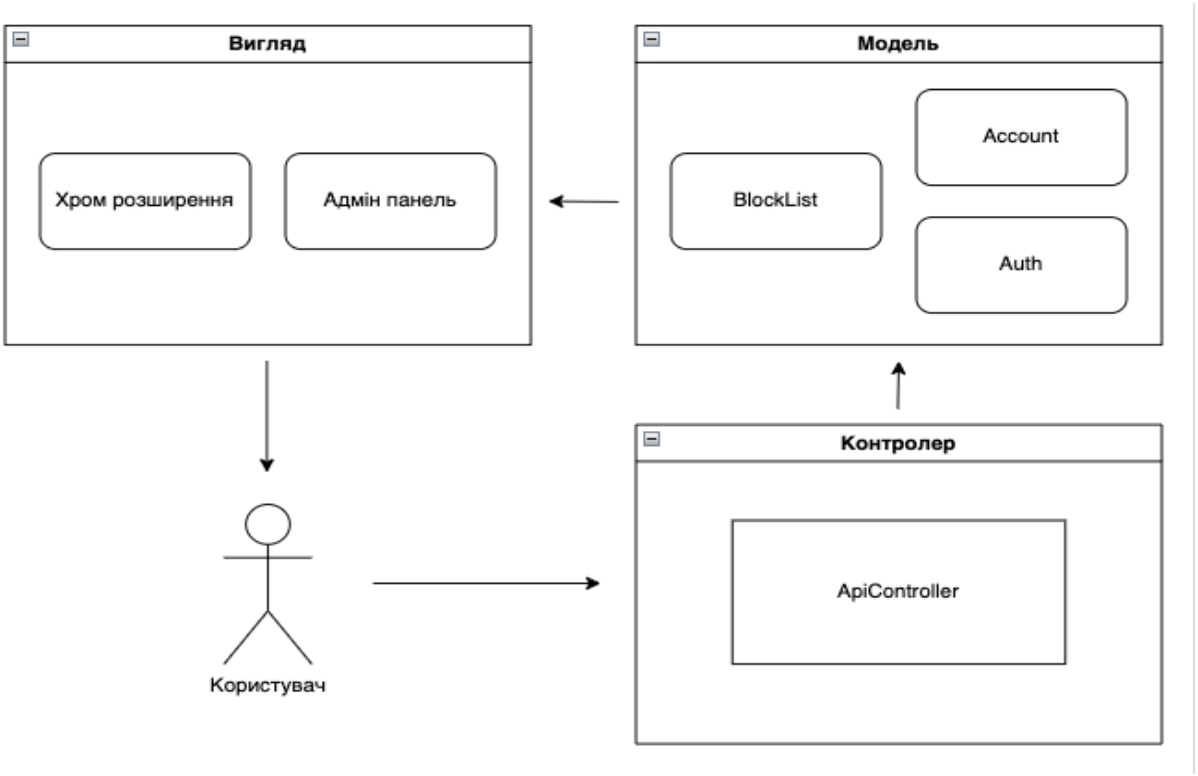


Рисунок 2.1 Шаблон проектування MVC

Всі можливі запити подано у Таблиці 2.1.

Таблиця 2.1 – Обробка HTTP-запитів у контролері системи

Шлях	HTTP-метод доступу	Аргументи методу контролера
/auth/sign-up	POST	SignUpBodyDto
/auth/sign-in	POST	SignInBodyDto
/auth/sign-out	POST	-

/account	GET	-
/account	PATCH	PatchAccountDto
/block-list	GET	-
/block-list/:item	POST	BlockItemDto
/block-list/:item	DELETE	id

Такий поділ відповідальностей між моделлю, контролером та можливим відображенням (через веб-інтерфейс або мобільний додаток) забезпечує гнучкість, масштабованість та підвищену підтримуваність системи. Модель відповідає за управління даними та бізнес-логікою, контролер керує потоком запитів та відповідей, а відображення (якщо воно присутнє) надає інтерфейс для взаємодії користувача з системою.

Архітектура MVC дозволяє розробникам легко модифікувати або замінювати окремі компоненти без впливу на інші частини системи, полегшуючи процес розробки, тестування та обслуговування додатку [1].

2.1 Концептуальна схема системи

Концептуальна схема системи – це високорівневий опис основних компонентів і їх взаємодії в системі. Вона допомагає зрозуміти загальну структуру

та призначення системи без деталей реалізації. Концептуальна схема зазвичай включає наступні елементи:

1. Визначення основних частин системи: У концептуальній схемі виокремлюються основні модулі та компоненти системи. Це можуть бути, наприклад, користувацький інтерфейс, серверна частина, база даних та інші.
2. Взаємодія між компонентами: Опис того, як ці компоненти взаємодіють між собою, обмінюються даними та виконують свої функції.
3. Інформаційні потоки: Вказівка на те, які дані переміщуються між компонентами, яким чином і в якому форматі.
4. Загальна архітектура: Загальний вигляд архітектури системи, який показує, як всі компоненти об'єднані в єдине ціле.

У концептуальній схемі виокремлюються основні модулі та компоненти системи, такі як користувацький інтерфейс, серверна частина, база даних та інші. Це допомагає зрозуміти загальну структуру системи та її функціональність.

Зокрема, в концептуальній схемі також описується взаємодія між компонентами системи. Це включає обмін даними та виконання різних функцій системи. Наприклад, модуль користувацького інтерфейсу може отримувати дані від модуля бізнес-логіки та відображати їх користувачам, а модуль доступу до даних може забезпечувати доступ до бази даних для збереження та отримання інформації.

Іншим важливим аспектом концептуальної схеми є інформаційні потоки. Це вказує на те, які дані переміщуються між компонентами системи та в якому форматі вони передаються. Наприклад, це може бути передача структурованих даних через API або використання спеціалізованих протоколів обміну даними.

Концептуальна схема також вказує, які дані переміщуються між компонентами системи та в якому форматі. Це допомагає зрозуміти, які дані необхідні для роботи системи та як вони обмінюються між різними частинами програми.

На Рисунку 2.2 можна ознайомитися з концептуальною схемою системи:

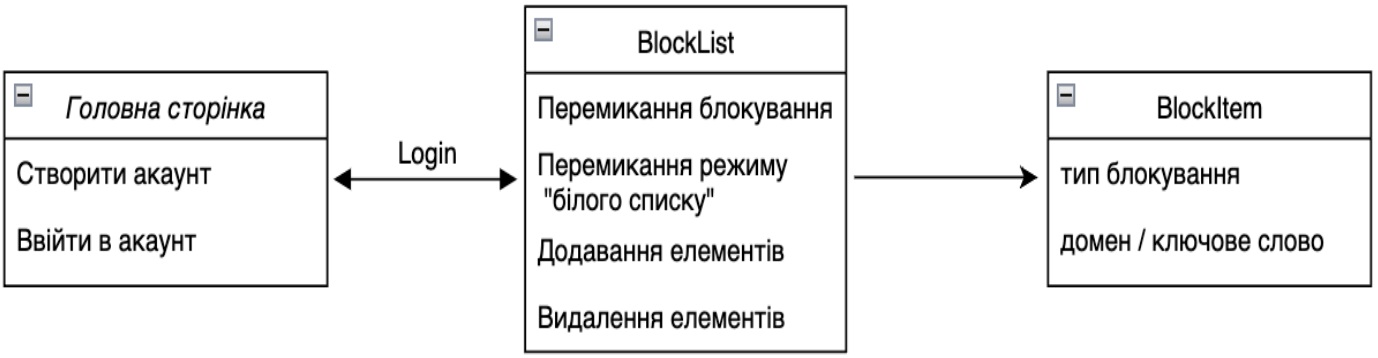


Рисунок 2.2 Концептуальна схема системи

Як видно з концептуальної схеми, клієнтський додаток (веб-додаток або мобільний додаток) взаємодіє з серверною частиною системи через HTTP-запити та отримує відповіді у форматі JSON. Серверна частина, у свою чергу, обробляє ці запити, взаємодіє з базою даних для збереження та отримання необхідних даних (облікових записів користувачів, списків блокування тощо) та повертає відповідну інформацію клієнтському додатку.

Така концептуальна схема забезпечує загальне розуміння структури системи батьківського контролю, її основних компонентів та потоків даних, не заглиблюючись у деталі реалізації. Вона є корисною для візуалізації та документування архітектури системи на високому рівні.

2.2 Функціональна схема системи

Функціональна схема (Functional Flow Block Diagram - FFBD) - це багаторівнева, послідовна в часі, покрокова блок-схема функціонального потоку системи. FFBD відображають завдання, визначені за допомогою функціональної декомпозиції, і показують їх у їх логічному, послідовному взаємозв'язку.

У функціональній схемі FFBD використовується блок-схема для відображення різних функцій та їх взаємозв'язку. Кожен блок представляє окрему

функцію або завдання, яке виконується системою. Завдяки багаторівневій структурі, FFBD дозволяє детально розкрити ієрархію функцій та їх взаємозв'язок на різних рівнях абстракції.

FFBD є багаторівневою та послідовною в часі, що дозволяє деталізувати функції на різних рівнях абстракції. На верхньому рівні зазвичай зображені основні функціональні блоки системи, а на нижчих рівнях ці блоки розкриваються більш детально, показуючи внутрішні функції та їх взаємозв'язки.

Основними елементами функціональної схеми є:

1. Функціональні блоки: Кожен блок представляє окрему функцію або завдання, що виконується системою. Блоки можуть бути декомпозовані на більш дрібні підфункції на нижчих рівнях схеми. Кожен блок має визначену функцію та може мати зв'язки з іншими блоками, що вказує на взаємозв'язок між різними функціями.
2. Потоки даних: Стрілки між блоками вказують на потоки даних або управління, що зв'язують функції між собою. Вони показують, як вихідні дані однієї функції стають вхідними для іншої.
3. Логічна послідовність: Розташування блоків та напрямок потоків даних відображають логічну послідовність виконання функцій у системі.
4. Ієрархічна структура: FFBD має ієрархічну структуру, де верхній рівень представляє загальні функціональні блоки, а нижчі рівні деталізують їх, розкриваючи внутрішні функції та взаємозв'язки.

Функціональна схема дозволяє візуалізувати структуру системи та послідовність виконання різних функцій. Це допомагає розкрити логіку роботи системи та ідентифікувати можливі проблеми або недоліки в дизайні.

На рисунку 2.3 можна ознайомитись з функціональною схемою системи:

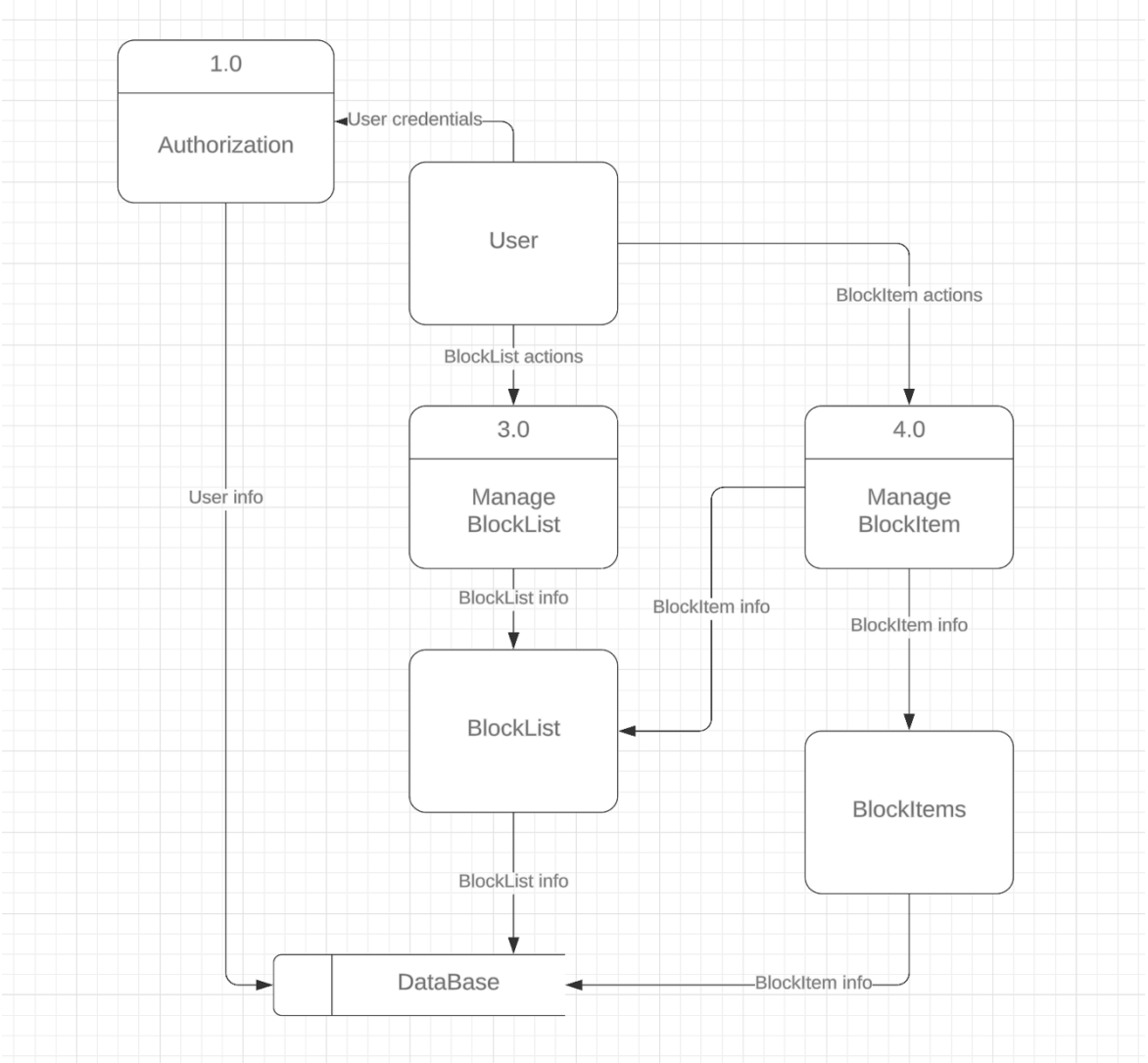


Рисунок 2.3 Функціональна схема системи

Усі ці аспекти роблять функціональну схему системи важливим інструментом для розуміння, аналізу та вдосконалення роботи технічних систем. Її використання допомагає розкрити логічну структуру та послідовність виконання функцій, що є ключовим для успішного розробки та експлуатації складних технічних систем.

Блок "Керування списком блокування" є ключовим компонентом системи батьківського контролю. Він надає функціональність для додавання, видалення та перегляду списку заблокованих ресурсів, таких як веб-сайти, додатки або інші цифрові ресурси. Батьки можуть додавати нові елементи до списку блокування, видаляти існуючі елементи та переглядати поточний список заблокованих ресурсів.

Блок "Керування елементом списку" забезпечує додаткову функціональність для роботи з окремими елементами списку заблокованих ресурсів. Він дозволяє батькам переглядати деталі кожного елемента, такі як назва ресурсу, URL-адреса (для веб-сайтів), категорія або інші релевантні дані. Крім того, цей блок надає можливість редагувати та оновлювати інформацію про елемент списку, наприклад, змінювати назву, категорію або додавати коментарі чи нотатки.

Блок "Автентифікація користувача" відповідає за процеси реєстрації, входу та виходу з системи. Він забезпечує перевірку облікових даних користувачів, створення нових облікових записів та безпечний вихід з системи. Цей блок взаємодіє з блоком "Керування обліковим записом" для створення, оновлення та отримання даних облікових записів користувачів.

Кожен з цих блоків може бути деталізований на нижчих рівнях функціональної схеми, розкриваючи внутрішні функції та їх взаємозв'язки. Наприклад, блок "Автентифікація користувача" може бути розкритий на функції "Реєстрація", "Вхід" та "Вихід", які деталізують процеси реєстрації нового користувача, автентифікації існуючого користувача та безпечного виходу з системи відповідно.

3. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

3.1 Опис вибору СУБД

База даних — набір впорядкованої та структурованої інформації, що зберігається в електронному вигляді. Завдяки БД (Базам даних) набагато зручніше виконувати управління, коригування, оновлення, контроль та впорядкування.

Однією з ключових переваг баз даних є оперативність їхньої роботи при додаванні та використанні інформації. Завдяки вдосконаленим алгоритмам, які

використовуються у їхній структурі, знаходження потрібних даних займає лише кілька секунд. Крім того, бази даних забезпечують внутрішню взаємодію даних: зміна у одній частині може вплинути на інші — це сприяє ефективнішій та швидшій обробці інформації [2].

Бази даних для веб-сайтів функціонують як спеціально організовані зв'язані між собою структури. У них зберігається весь необхідний контент для оптимальної роботи сайту: дані користувачів, цінові пропозиції, асортимент товарів.

Створення запитів до баз даних часто виконується за допомогою мови структурованих запитів (Structured Query Language, SQL). SQL надає можливість додавати, змінювати та видаляти інформацію, що зберігається у таблицях.

Системи управління базами даних (СУБД) представляють собою комплекс програмних та мовних інструментів, спрямованих на створення, зберігання, керування та використання баз даних. Вони забезпечують ефективне та надійне зберігання даних і надають користувачам зручний інтерфейс для роботи з ними.

Роль СУБД в управлінні даними полягає в структурованому доступі до інформації, у мінімізації дублювання даних та забезпеченні цілісності та безпеки даних.

Перші системи управління базами даних з'явилися у 1960-х роках та були спрямовані на зберігання даних у ієрархічній або мережевій структурах. У 1970-х роках з'явилися реляційні системи управління базами даних, які залишаються найпоширенішими до сьогодні. У 1980-х роках стали розвиватися об'єктно-орієнтовані системи управління базами даних, що підходять для зберігання даних про складні об'єкти. Важливим етапом у історії СУБД була розробка мови структурованих запитів SQL у 1970-х роках.

Останні роки свідчать про розвиток хмарних систем управління базами даних, які надають користувачам доступ до даних через Інтернет.

Управління даними систем управління базами даних здійснюється шляхом ефективного та систематизованого зберігання інформації. Наприклад, у онлайн-магазині СУБД ефективно управляють інформацією про товари, замовлення та клієнтів, забезпечуючи швидкий доступ до потрібної інформації.

Інструменти вилучення та оновлення дозволяють керувати даними в базі, забезпечуючи ефективну взаємодію з інформацією. Оптимізація запитів гарантує швидкодію системи, особливо у сферах, де необхідний моментальний доступ, наприклад, у медичних організаціях для швидкого отримання медичних записів.

Системи управління базами даних класифікуються за різними критеріями, включаючи модель даних (реляційні, ієрархічні, мережеві), функціональне призначення (оперативні та аналітичні), розподіл (централізовані та розподілені), тип використання (SQL-орієнтовані та NoSQL) і ліцензію (відкриті та пропрієтарні). Знання переваг і недоліків кожного типу допомагає вибрати найкращий варіант залежно від конкретних потреб проєкту [3].

При проектуванні бази даних, крім вибору самої системи управління базами даних (СУБД), також важливо враховувати такі аспекти, як оптимізація структури даних, забезпечення цілісності та безпеки інформації, а також розробка ефективних запитів для отримання необхідної інформації.

Планування індексів на полях, які часто використовуються для пошуку, може значно підвищити швидкодію запитів. Ретельне проектування структури таблиць, використання нормалізації для уникнення дублювання даних та забезпечення цілісності даних, а також відповідна настройка параметрів СУБД, що відповідають за кешування та оптимізацію запитів, також важливі для забезпечення ефективності та надійності бази даних.

Крім того, необхідно приділяти увагу аспектам безпеки даних. Встановлення прав доступу до таблиць та обмежень на виконання операцій з даними для різних користувачів, шифрування конфіденційної інформації та регулярне резервне копіювання даних є важливими заходами для запобігання втраті чи несанкціонованому доступу до інформації.

Коли мова йде про проектування бази даних, важливо враховувати різноманітні аспекти, щоб забезпечити ефективність, надійність і безпеку системи. Наприклад, визначення відповідного типу даних для кожного поля, яке буде зберігатися в базі даних, допомагає економити простір та забезпечує точність збереження інформації.

Крім того, нормалізація бази даних дозволяє уникнути аномалій та забезпечити цілісність даних шляхом розбиття таблиць на менші, оптимізовані структури. Це допомагає уникнути дублювання даних та забезпечує зручність у використанні бази даних.

Під час вибору конкретної СУБД важливо враховувати потреби проекту та його масштабів. Наприклад, для великих обсягів даних та вимог до високої продуктивності може бути вибрана розподілена база даних, що дозволяє розподіляти дані на кілька серверів для оптимізації швидкодії та масштабованості системи.

З урахуванням усіх цих аспектів та відповідно до специфіки проекту планується структура бази даних, її взаємозв'язки та індексація, щоб забезпечити оптимальну продуктивність та надійність системи. Крім того, ретельно розробляються запити та зберігання процедур для ефективного доступу до даних та оптимізації їх обробки.

Таким чином, ефективне проектування бази даних є ключовим етапом у розробці будь-якої інформаційної системи, оскільки від цього залежить продуктивність, масштабованість та безпека обробки даних.

Вибір відповідної СУБД є вирішальним для успіху та стабільності проекту. Важливо враховувати тип даних, вимоги проекту, продуктивність, масштабованість та зручність використання. Слід також оцінювати швидкість виконання запитів, можливості індексування, вертикальну та горизонтальну масштабованість, сумісність та інтеграцію з інструментами розробки. Крім того, необхідно враховувати рівень складності адміністрування системи.

Таблиця 3.1 Порівняння типів СУБД

Тип СУБД	Опис	Переваги	Недоліки

Ієрархічні	Організовують дані у вигляді ієрархії, де кожен запис має батька та нащадків	– Простота в моделюванні зв’язків.	– Обмеження в гнучкості структури даних.
Мережеві	Дозволяють встановлювати складні зв’язки між даними, створюючи структуру, схожу на граф	– Підтримка складних і взаємопов’язаних даних.	– Складність в управлінні структурою і підтримці запитів. – Складнощі в підтримці та обслуговуванні.
Реляційні	Засновані на теорії відносин і таблиць. Кожна таблиця представляє окреме відношення.	– Простота у використанні та моделюванні даних.	– Продуктивність може страждати за великих обсягів даних. – Складність у поданні складних ієрархій даних.
Об’єктно-орієнтовані	Працюють з об’єктами, де дані представлено у вигляді об’єктів із методами та властивостями.	– Зручність у моделюванні складних об’єктних структур. – Підтримка успадкування, поліморфізму та інкапсуляції.	– Складність у міграції з традиційних СУБД. – Обмежена підтримка мови SQL.

Об’єктно-реляційні	Комбінують риси реляційних і об’єктно-орієнтованих СУБД.	– Поєднання переваг реляційних і об’єктно-орієнтованих моделей. – Підтримка більш складних структур даних.	– Додаткова складність у проєктуванні.

Системи управління базами даних (СУБД) пропонують різноманітні засоби безпеки, серед яких контроль доступу, шифрування даних, забезпечення цілісності даних та резервне копіювання. Контроль доступу обмежує можливість доступу до даних лише авторизованим користувачам, шифрування захищає дані від несанкціонованого використання, а підтримка цілісності та резервне копіювання гарантують надійність і збереження даних.

Розглянемо Postgres, або PostgreSQL, як він частіше відомий, як одну з найпопулярніших відкритих систем управління базами даних (СУБД). Однією з його головних переваг є його розширюваність та гнучкість. Postgres підтримує широкий спектр функцій, що робить його відмінним вибором для будь-якого рівня складності проекту

PostgreSQL є потужною об'єктно-реляційною системою управління базами даних з відкритим вихідним кодом, що поєднує в собі переваги реляційних баз даних та об'єктно-орієнтованого програмування. Вона використовує клієнт-серверну архітектуру з окремими процесами для кожного підключення, а також застосовує мультиверсійний контроль паралельності (MVCC) для забезпечення узгодженості даних і Write-Ahead Logging (WAL) для забезпечення стійкості до збоїв і можливості відновлення.

Крім того, PostgreSQL має потужну систему розширень, яка дозволяє розширити його функціональність за допомогою власних розширень або сторонніх модулів. Це відкриває безліч можливостей для використання PostgreSQL у різних

областях, включаючи обробку геоданих, роботу з JSON та XML, реалізацію повнотекстового пошуку та багато іншого.

Однією з ключових переваг PostgreSQL є його відкритість і активна спільнота користувачів та розробників. Це означає, що ви можете легко знайти підтримку, документацію та рішення для своїх проблем, а також активно брати участь у розвитку та покращенні системи.

Загалом, PostgreSQL - це потужна та надійна СУБД, яка підходить для широкого спектру проектів, від малих веб-сайтів до великих корпоративних систем. Його гнучкість, розширюваність та відкритість роблять його одним із найпопулярніших виборів серед розробників та архітекторів.

Таблиця 3.2 порівняння PostgreSQL з іншими СУБД

Критерій	PostgreSQL	MySQL	SQLite
Вартість	Безкоштовно	Безкоштовно	Безкоштовно
Відкритість	Відкритий вихідний код	Відкритий вихідний код	Відкритий вихідний код
Підтримувані мови програмування	SQL, PL/pgSQL, Python, Perl, Java, C, C++, Ruby	SQL, MySQL, PHP, Python, Perl	SQL, Python, C++, Java, PHP, Ruby
Підтримувані операційні системи	Windows, Linux, macOS, FreeBSD	Windows, Linux, macOS	Windows, Linux, macOS, Solaris, HP-UX, AIX, z/OS
Підтримувані архітектури	x86, x86-64, ARM, RISC-V	x86, x86-64, ARM	x86, x86-64, ARM, RISC-V

Продуктивність	Хороша	Хороша	Хороша
Надійність	Висока	Висока	Висока
Безпека	Висока (підтримка SSL, контроль доступу, шифрування)	Середня (підтримка SSL, контроль доступу)	Низька (обмежені можливості безпеки)
Функціональність	Підтримка складних запитів, індексів, тригерів, переглядів, розширень	Підтримка складних запитів, індексів, тригерів, переглядів	Обмежена підтримка індексів, тригерів, переглядів
Масштабованість	Висока	Висока	Низька

Завдяки тому, що PostgreSQL належить до об'єктно-реляційних СУБД і має відкритий вихідний код, ця система надає великі можливості для налаштування та адаптації. Підтримка широкого набору функцій забезпечує гнучкість у реалізації різних сценаріїв використання. Висока продуктивність, надійність та рівень безпеки гарантують ефективну і захищену роботу з даними, що робить PostgreSQL оптимальним вибором для системи батьківського контролю для веб сервісів.

3.2 Опис всіх таблиць та полів бази даних

Діаграми «сутність-зв’язок» (ERD) призначені для розробки моделей даних та забезпечують стандартний засіб визначення даних і відношень між ними. Фактично за допомогою ERD здійснюється деталізація сховищ даних проектованої

системи, а також документуються сутності системи та засоби їхньої взаємодії, включаючи ідентифікацію об'єктів, важливих для предметної області (сутності), властивостей цих об'єктів (атрибутів) і їхніх відношень з іншими об'єктами (зв’язків) [4].

Сутності: Кожна сутність відображає клас або тип об'єкту, який має унікальні атрибути. Наприклад, у нашій базі даних можуть бути такі сутності, як «Користувач», «Сайт», «Блокований_сайт» тощо. Кожна з цих сутностей має свої унікальні атрибути, які визначають характеристики об'єктів цього класу. Сутність являє собою множину екземплярів реальних або абстрактних об'єктів (людей, подій, станів, ідей, предметів і т. ін.), що мають спільні атрибути або характеристики. Будь-який об'єкт системи може бути представлений лише однією сутністю, що повинна бути унікально ідентифікована. При цьому ім'я сутності повинно відображати тип або клас об'єкту, а не його конкретний екземпляр (наприклад, КНИГА, а не назва конкретної книги). Кожна сутність володіє однім або декількома атрибутами, що однозначно ідентифікують кожен примірник (екземпляр) сутності. При цьому будь-який атрибут може бути визначений як ключовий.

Відношення**:** Відношення визначають зв'язки між сутностями. Наприклад, у нашій системі може бути відношення між сутностями «Користувач» і «Сайт», яке показує, які сайти відвідує кожен користувач. Відношення відображаються за допомогою граматичних конструкцій, таких як «має», «володіє», «відвідує» тощо. Відношення в самому загальному вигляді являє собою зв'язок між двома і більшою кількістю сутностей. Найменування відношення здійснюється за допомогою граматичного звороту дієслова (має, визначає, може володіти і т. ін.)

Атрибути**:** Атрибути є характеристиками кожної сутності, що дозволяють ідентифікувати та описувати об'єкти. Наприклад, для сутності «Користувач» атрибутами можуть бути ім'я, прізвище, електронна адреса тощо.

Головною перевагою ERD в контексті реляційних баз даних є їхня тісна відповідність зі структурою самих баз даних. Реляційні бази даних побудовані на

таблицях, у яких зберігаються структуровані дані – окрема таблиця для кожного об'єкта, а взаємозв'язки між об'єктами реалізуються за допомогою первинних та зовнішніх ключів. ERD забезпечують простий та інтуїтивно зрозумілий спосіб візуалізації цих ключових елементів та їхніх взаємозв'язків, що сприяє безперешкодному переходу від проектування до впровадження та обслуговування бази даних.

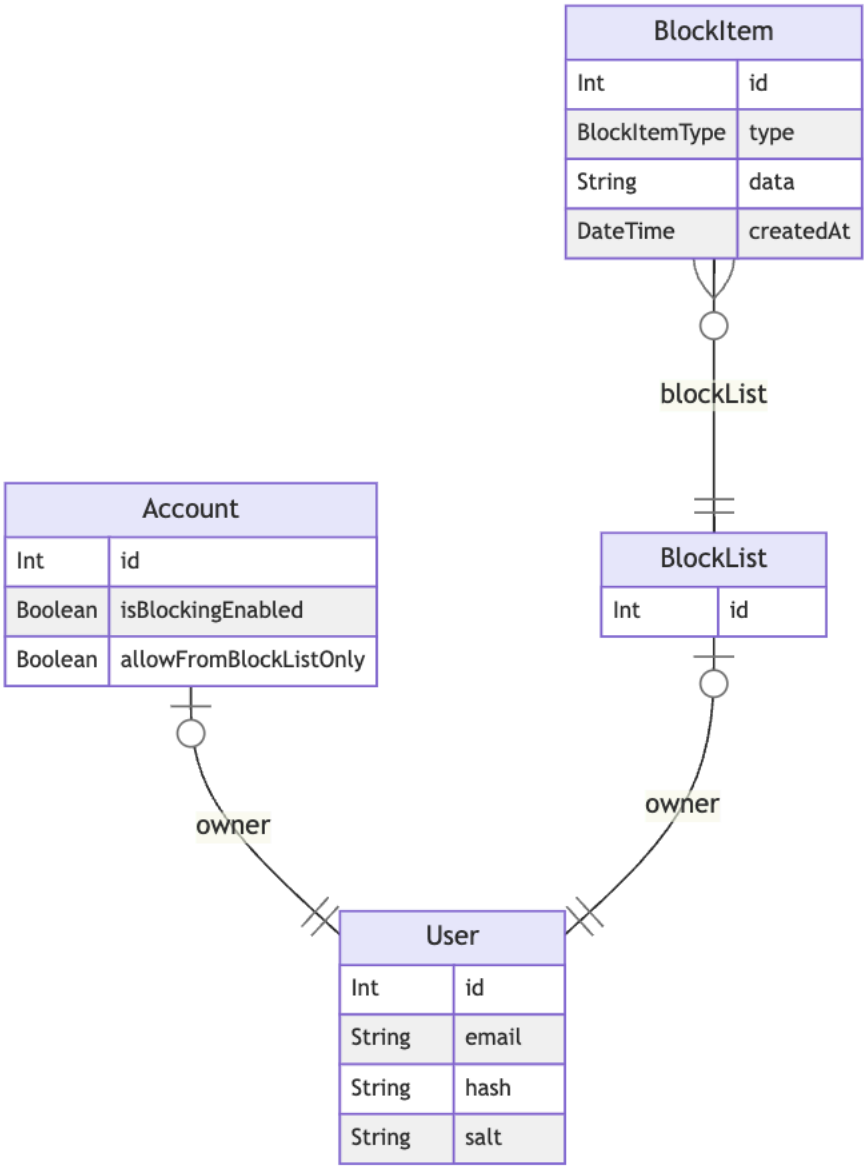


Рисунок 3.1 ERD бази даних

Таблиця 3.3 Опис таблиць та полів БД

Таблиця	Назва поля	Тип	Опис поля
User	id	Int	PK, ідентифікатор
	email	String	Унікальна пошта користувача
	hash	String	Хеш пароля користувача
	salt	String	Сіль для хешування пароля
	account	Account?	Зв’язок з таблицею Account
	blockList	BlockList?	Зв’язок з таблицею BlockList
Account	id	Int	PK, ідентифікатор
	ownerId	Int	Унікальний ідентифікатор власника
	owner	User	Зв’язок з таблицею User
	isBlockingEnabled	Boolean	Флаг активації блокування
	allowFromBlockListOnly	Boolean	Флаг режиму “білого списку”

BlockList	id	Int	PK, ідентифікатор
	ownerId	Int	Унікальний ідентифікатор власника
	owner	User	Зв’язок з таблицею User
	items	BlockItem[]	Масив елементів списку
BlockItem	id	Int	PK, ідентифікатор
	blockListId	Int	Ідентифікатор списку
	blockList	BlockList	Зв’язок з таблицею BlockList
	type	BlockItemType	Тип елемента списку
	data	String	Дані елемента списку

Дана таблиця показує поля та таблиці, які були створені для системи, а саме:

– таблиця User містить всю інформацію про користувача системи, включаючи його ідентифікатор, електронну пошту, хешовані дані для пароля та зв'язки з таблицями Account та BlockList..

– Таблиця Account містить інформацію про обліковий запис користувача, такі як ідентифікатор власника, прапорці для активації блокування та режиму "білого списку", а також зв'язок з таблицею User..

– таблиця BlockList зберігає інформацію про списки блокування, включаючи ідентифікатор власника, зв'язок з таблицею User та масив елементів списку.

– таблиця BlockItem містить інформацію про окремі елементи списку блокування, такі як ідентифікатор, ідентифікатор списку, зв'язок з таблицею BlockList, тип елемента (веб-сайт, додаток тощо) та дані елемента (URL, назва додатка тощо)..

Проектування бази даних за допомогою ERD дозволяє зрозуміти структуру даних та відношення між ними, що в свою чергу спрощує подальшу розробку, впровадження та обслуговування системи. Використання такої методології дозволяє створити ефективну та гнучку базу даних, яка відповідає потребам користувачів.

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТОВИЙ ПРИКЛАД

4.1 Вибір та порівняння середовищ розробки

У сучасному світі розробка веб-додатків є однією з найбільш затребуваних сфер програмування. Важливу роль у цьому процесі відіграє вибір середовища розробки (IDE), яке може суттєво вплинути на продуктивність та зручність роботи розробника. Серед великої кількості доступних інструментів виділяються такі середовища розробки як Visual Studio Code та WebStorm.

Visual Studio Code (VS Code) - це безкоштовний редактор коду з відкритим вихідним кодом, розроблений всесвітньо відомою компанією Microsoft. VS Code швидко набув популярності серед розробників завдяки своїй гнучкості, розширюваності та потужній екосистемі плагінів. Основними перевагами VS Code є:

1. Безкоштовність та відкритий вихідний код: VS Code є вільно доступним для використання та модифікації, що робить його привабливим варіантом для розробників та команд з обмеженим бюджетом.
2. Великий вибір плагінів та розширень: Завдяки активній спільноті розробників, VS Code має величезну бібліотеку плагінів та розширень, які додають нові функції та покращують продуктивність роботи.
3. Підтримка багатьох мов програмування: VS Code підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, Python, C++, Java та багато інших.
4. Зручний інтерфейс та можливість кастомізації: Інтерфейс VS Code є інтуїтивно зрозумілим та легким у налаштуванні відповідно до індивідуальних потреб розробника.

Серед недоліків VS Code можна відзначити:

1. Споживання системних ресурсів: При використанні великої кількості розширень, VS Code може споживати значні ресурси системи, що може вплинути на продуктивність роботи.
2. Менша функціональність "з коробки": Порівняно з комерційними продуктами, VS Code може потребувати додаткових налаштувань та встановлення розширень для отримання необхідної функціональності.

WebStorm - це комерційний IDE від компанії JetBrains, який спеціалізується на мовах JavaScript та TypeScript для фронтенд розробки. Основними перевагами WebStorm є:

1. Професійні інструменти для розробки JavaScript/TypeScript: WebStorm має вбудовані потужні інструменти для розробки JavaScript та TypeScript, включаючи інтелектуальне автодоповнення, рефакторинг коду та інструменти для відладки.
2. Підтримка сучасних фреймворків: WebStorm надає вбудовану підтримку для популярних фреймворків, таких як React, Angular та Vue, спрощуючи розробку та налагодження додатків.
3. Потужні інструменти для відладки та тестування: WebStorm пропонує комплексні інструменти для відладки коду, включаючи налагоджувач для різних середовищ (Node.js, Chrome, Safari тощо), а також інтеграцію з фреймворками для тестування.
4. Інтеграція з системами контролю версій: WebStorm має вбудовану підтримку для популярних систем контролю версій, таких як Git, Mercurial та Subversion, що полегшує роботу в команді.

Серед недоліків WebStorm можна виділити:

1. Платний: WebStorm є комерційним продуктом, хоча доступна безкоштовна пробна версія.

2. Складніший у налаштуванні: Через багатий функціонал та велику кількість налаштувань, WebStorm може бути дещо складнішим у налаштуванні та освоєнні порівняно з більш легкими редакторами коду, такими як VS Code. Розробникам може знадобитися більше часу, щоб розібратися з усіма можливостями та налаштувати середовище розробки відповідно до своїх потреб.

Таблиця 4.1 Порівняльна характеристика середовищ розробки

	Visual Studio Code	WebStorm
Відкритий код	Так	Ні
Плагіни та розширення	Дуже багато	Достатньо
Підтримка мов	Багато (реалізовано через плагіни)	JavaScript, TypeScript
Ціна	Безкоштовний	Платний
Швидкість роботи	Дуже висока	Висока

Visual Studio Code було обрано основним середовищем розробки системи батьківського контролю з огляду на кілька важливих чинників, які сприяють підвищенню ефективності та продуктивності розробки:

1. Широкий вибір плагінів та розширень: Завдяки значній бібліотеці плагінів та розширень, VS Code дозволяє легко налаштувати середовище розробки відповідно до потреб проекту та індивідуальних вподобань розробників.
2. Низьке навантаження на систему в порівнянні з WebStorm: VS Code є більш легким для системних ресурсів, що забезпечує швидку та плавну роботу під час розробки.
3. Безкоштовність: Як безкоштовне рішення, VS Code є привабливим вибором для команд з обмеженим бюджетом або окремих розробників.

Хоча WebStorm пропонує потужні інструменти для розробки JavaScript/TypeScript, його комерційна природа та більш ресурсомісткий характер були вирішальними факторами для вибору більш легкого та гнучкого рішення у вигляді Visual Studio Code.

4.2 Вибір мови програмування

При розробці веб-додатків важливо обрати мову програмування, яка відповідатиме потребам проекту. Однією з найпопулярніших мов програмування для веб-розробки є TypeScript.

TypeScript - це надбудова над JavaScript, яка додає статичну типізацію та низку інших функцій до мови. TypeScript був розроблений компанією Microsoft і є відкритим програмним забезпеченням. Основними перевагами використання TypeScript є:

1. Статична типізація: Однією з найбільших переваг TypeScript є статична типізація, яка дозволяє виявляти помилки типів під час компіляції, а не під час виконання програми. Це спрощує процес розробки, підвищує стабільність коду та полегшує його підтримку в майбутньому.
2. Покращене автодоповнення та інструменти розробки: Багато інтегрованих середовищ розробки (IDE), таких як Visual Studio Code, надають покращене автодоповнення та інші корисні інструменти для роботи з TypeScript, що полегшує написання коду та підвищує продуктивність розробників.

3. Сумісність з JavaScript: TypeScript є надмножиною над JavaScript, що означає, що весь валідний JavaScript-код також є валідним TypeScript-кодом. Це дозволяє поступово переходити від JavaScript до TypeScript без значних змін у вже існуючому коді, а також використовувати величезну кількість існуючих бібліотек та фреймворків, написаних на JavaScript.
4. Кращі практики та підтримка: TypeScript заохочує використання кращих практик програмування, таких як модульність, інкапсуляція та абстракція. Крім того, TypeScript має активну спільноту розробників та отримує постійну підтримку від Microsoft, що забезпечує стабільність та розвиток мови.

Серед недоліків TypeScript можна відзначити:

1. Час на навчання: Для того, щоб повноцінно використовувати TypeScript, розробникам може знадобитися додатковий час на вивчення мови та її особливостей, особливо якщо вони раніше працювали лише з JavaScript.
2. Перевірка типів під час компіляції: Оскільки TypeScript компілюється в JavaScript, перевірка типів відбувається тільки на етапі компіляції, а не під час виконання коду. Це може призвести до пропуску деяких помилок, які можуть виявитися тільки в процесі виконання програми.

Ще одним важливим аспектом TypeScript є його активна спільнота та постійний розвиток. Microsoft активно підтримує TypeScript, регулярно випускаючи нові версії з поліпшеннями та новими функціями. Крім того, величезна кількість сторонніх бібліотек та інструментів розробляється з підтримкою TypeScript, що розширює його можливості та забезпечує більшу гнучкість для розробників [5].

Таблиця 4.2 порівняння мов JavaScript та TypeScript

	TypeScript	JavaScript

Типізація	Статична	Динамічна
Виявлення помилок	Під час компіляції	Під час виконання
Автодоповнення	Покращене завдяки типам	Обмежене
Розширення	Надмножина JavaScript, що забезпечує зворотну сумісність	Базова мова програмування
Сумісність	Можна використовувати JavaScript код	Можливе використання TypeScript

TypeScript може бути чудовим вибором для проектів, які потребують стабільності, швидкості розробки та легкої підтримки коду, особливо у великих командних проектах. Статична типізація, покращені інструменти розробки та сумісність з існуючим JavaScript-кодом роблять TypeScript потужним інструментом для створення сучасних веб-додатків.

У контексті системи батьківського контролю, використання TypeScript забезпечує надійну основу для розробки, дозволяючи створювати стабільний та легко підтримуваний код. Завдяки статичній типізації, команда розробників може швидше виявляти та виправляти потенційні помилки, що підвищує якість кінцевого продукту. Крім того, TypeScript дозволяє ефективно використовувати існуючі JavaScript-бібліотеки та фреймворки, розширюючи функціональність системи.

Вибір TypeScript як мови програмування для реалізації системи батьківського контролю був обґрунтованим рішенням, що відповідає вимогам проекту та забезпечує ефективний процес розробки та підтримки коду.

4.3 Технології клієнтської частини

При виборі технологій для фронтенд-розробки важливо враховувати фреймворк, який забезпечить ефективну та зручну розробку. Серед фреймворків та бібліотек, які використовуються для створення користувацьких інтерфейсів, особливо виділяються React, Angular та Vue.js. Для розробки клієнтської частини системи було обрано бібліотеку React.

React - це бібліотека для побудови користувацьких інтерфейсів, розроблена компанією Facebook (Meta) [6].

Однією з переваг React є його підтримка статичної типізації. Це означає, що ви можете вказати типи даних для ваших компонентів та їхніх властивостей, що дозволяє попередити багато помилок ще до виконання програми. За допомогою інструментів, таких як TypeScript або Flow, ви можете забезпечити більшу надійність та стабільність вашого коду.

React також має перевагу у швидкості розробки, завдяки своєму декларативному підходу та можливості повторного використання компонентів. Компоненти React можуть бути легко перевикористані в різних частинах додатку, що економить час та зусилля розробників. Крім того, використання JSX (синтаксичний цукор для JavaScript) дозволяє писати код, який є зрозумілішим та легшим для читання, особливо для розробників, знайомих з HTML.

Ще однією ключовою особливістю React є використання віртуального DOM (Document Object Model). React використовує віртуальний DOM для ефективного оновлення інтерфейсу. Замість того, щоб оновлювати всі елементи інтерфейсу в реальному часі, React порівнює віртуальний DOM з реальним і вносить зміни тільки там, де вони потрібні. Це дозволяє підтримувати високу продуктивність навіть при роботі з великою кількістю даних та складним інтерфейсом.

Однією з ключових переваг React є активна та велика спільнота розробників. Існує безліч сторонніх бібліотек, компонентів та інструментів, розроблених спільнотою, які значно спрощують процес розробки. Більшість проблем можна вирішити за допомогою пошуку в Інтернеті або звернувшись до спільноти за допомогою форумів, соціальних мереж або спеціалізованих платформ, таких як Stack Overflow або Reddit.

Ще однією важливою перевагою React є його екосистема допоміжних бібліотек та інструментів. React Developer Tools - розширення для браузерів, що надає розробникам детальну інформацію про ієрархію компонентів та їхній стан, що значно полегшує відлагодження та оптимізацію додатків. React Native дозволяє використовувати ті ж самі принципи та концепції, що й React, для створення нативних мобільних додатків для iOS та Android, забезпечуючи спільну кодову базу та підвищуючи продуктивність розробників.

Незважаючи на свої багаті переваги, React має деякі недоліки. Одним з них є необхідність використання додаткових бібліотек для рішення певних завдань. React, як "бібліотека для створення користувацьких інтерфейсів", не включає в себе багато додаткових можливостей, таких як маршрутизація, управління станом тощо. Це може змусити розробників звертатися до сторонніх бібліотек або плагінів, що збільшує складність проекту та збільшує його об'єм.

Angular - це повнофункціональний фреймворк для розробки веб-застосунків, розроблений компанією Google. На відміну від React, Angular охоплює більшу частину функціоналу, необхідного для побудови веб-додатків, включаючи маршрутизацію, управління станом, роботу з формами та ін. Angular має строгу структуру та правила, що забезпечують високу продуктивність та масштабованість, але водночас зменшують гнучкість.

Vue.js - ще один популярний фреймворк для створення інтерфейсів, який поєднує в собі деякі підходи React та Angular. Vue.js має відносно невеликий розмір, легкий у вивченні та пропонує гнучку та зрозумілу структуру для створення веб-додатків. Проте, як і React, Vue.js вимагає додаткових бібліотек для вирішення певних завдань [7].

В таблиці 4.3 наведено порівняльну таблицю найпопулярніших бібліотек та фреймворків для розробки клієнтської частини:

Таблиця 4.3 порівняння найпопулярніших бібліотек та фреймворків

	React	Angular	Vue.js
Підхід	Бібліотека	Фреймворк	Фреймворк
Структура	Гнучка, без жорстких правил	Строга	Гнучка, але з деякими правилами
Продуктивність	Висока завдяки віртуальному DOM	Висока	Висока завдяки віртуальному DOM
Популярність	Висока	Середня	Висока

Таким чином, вибір технології для фронтенд-розробки — це завдання, яке потребує уважного аналізу вимог проекту, рівня кваліфікації команди та інших факторів. Однак React, завдяки своїм перевагам та підтримці спільноти,

залишається однією з найпопулярніших та найефективніших технологій для розробки сучасних веб-додатків.

4.4 Технології серверної частини

На серверній стороні одними з найпопулярніших рішень для розробки на Node.js є Express та NestJS. Для розробки системи було обрано NestJS.

NestJS - це прогресивний Node.js фреймворк для створення ефективних, надійних та масштабованих серверних додатків. Він використовує TypeScript та натхненний архітектурними концепціями Angular [9].

На серверній стороні важливо забезпечити належну безпеку та автентифікацію. Для цього в NestJS можна використовувати різні бібліотеки та модулі, такі як Passport.js, JWT (JSON Web Tokens) або Auth0. Вони дозволяють легко впровадити механізми автентифікації, авторизації та керування доступом на основі ролей (RBAC).

Крім того, NestJS дозволяє легко створювати RESTful API, що є стандартом для сучасних веб-додатків. Він забезпечує зручний шлях для визначення маршрутів, контролерів та обробників, а також підтримує вбудовані можливості для валідації, фільтрації, перехоплення винятків та обробки помилок.

Одною з ключових переваг NestJS є його модульність та ін'єкція залежностей. Це сприяє написанню більш модульного, тестованого та підтримуваного коду. Крім того, NestJS тісно інтегрований з екосистемою Node.js, що дозволяє легко використовувати сторонні бібліотеки та модулі.

Хоча NestJS має багато переваг, він також має деякі недоліки. Наприклад, він має вищий поріг входження порівняно з більш простими фреймворками, такими як Express. Це пов'язано з використанням TypeScript та додатковими абстракціями, які потрібно вивчити. Крім того, NestJS може бути надмірно складним для невеликих проектів або прототипів, де проста та мінімалістична структура Express може бути більш підходящою.

Таблиця 4.4 порівняння NestJS та Express

	NestJS	Express
Підхід	Повноцінний фреймворк	Мінімалістичний фреймворк
Структура	Модульна, організована	Мінімальна
Типізація	Статична (TypeScript)	Динамічна (JavaScript)

Вибір між NestJS та Express залежить від конкретних вимог проекту, розміру команди, досвіду розробників та очікуваної складності застосунку. Для великих та складних проектів, де потрібна модульність, типізація та організована структура, NestJS може бути найкращим вибором [9].

Для роботи з базами даних використовуються ORM, такі як Prisma та TypeORM, що полегшують взаємодію з базами даних та керування міграціями. Для розробки системи було обрано Prisma ORM.

ORM (Object-Relational Mapping) – це технологія, яка дозволяє розробникам взаємодіяти з базами даних через об'єктно-орієнтоване програмування, замість написання SQL-запитів. Використання ORM дозволяє автоматично перетворювати дані з таблиць бази даних в об'єкти програмного коду, що спрощує процес розробки та зменшує кількість необхідного коду [10].

Основні переваги ORM:

1. Абстрагування бази даних: ORM абстрагує низькорівневі операції з базою даних, дозволяючи розробникам працювати з даними на рівні об'єктів.
2. Автоматична генерація SQL: ORM автоматично генерує SQL-запити для взаємодії з базою даних.
3. Зменшення коду: Завдяки ORM зменшується обсяг коду, оскільки більшість CRUD (Create, Read, Update, Delete) операцій автоматизуються.
4. Підвищення продуктивності: ORM може оптимізувати запити та управління кешем, що покращує продуктивність додатків.

Недоліки ORM:

1. Виконання складних запитів: Деякі складні SQL-запити важко реалізувати за допомогою ORM, що може призвести до необхідності використання чистого SQL.

Prisma ORM - це сучасний інструмент ORM для Node.js та TypeScript, який дозволяє легко взаємодіяти з базами даних.

Prisma ORM забезпечує абстрактний рівень для взаємодії з базами даних, дозволяючи працювати з даними на рівні об'єктів замість низькорівневих SQL-запитів. Це спрощує розробку, підвищує продуктивність та полегшує міграцію між різними базами даних.

Для забезпечення високої доступності та масштабованості серверної частини можна використовувати контейнеризацію за допомогою Docker та оркестрацію контейнерів за допомогою Kubernetes. Це дозволить легко розгортати та масштабувати додаток у хмарному середовищі або на власній інфраструктурі.

Переваги Prisma:

1. Схематична типізація: автоматично генерує типи для баз даних.
2. Інтуїтивний API: легкий у використанні та зрозумілий.
3. Міграції баз даних: вбудовані інструменти для керування міграціями.

Окрім основних фреймворків та ORM, під час розробки серверної частини також використовуються різноманітні допоміжні бібліотеки та інструменти. Наприклад, для логування та моніторингу застосовуються бібліотеки, такі як Pino. Вони дозволяють налаштовувати рівні логування, форматувати логи та зберігати їх у файлах або відправляти до зовнішніх сервісів.

Для обробки асинхронних операцій та управління потоком виконання можна використовувати бібліотеки, такі як RxJS. Вони забезпечують зручний спосіб роботи з асинхронним кодом та уникнення проблем, пов'язаних із callback hell.

Для кешування даних та підвищення продуктивності можна використовувати системи кешування, такі як Redis. Вони дозволяють зберігати часто використовувані дані в оперативній пам'яті, прискорюючи доступ до них та зменшуючи навантаження на базу даних.

Для автоматизації процесів розгортання та безперервної інтеграції/безперервного розгортання (CI/CD) використовуються інструменти, такі як Jenkins. Вони дозволяють автоматизувати процеси збірки, тестування та розгортання додатків, забезпечуючи швидший цикл розробки та зменшуючи ризик людських помилок.

Окрім того, для забезпечення безпеки та конфіденційності даних використовуються різноманітні механізми, такі як шифрування, захист від атак типу CSRF та XSS, обмеження швидкості запитів та інші заходи безпеки.

Вибір технологій для клієнтської та серверної частини проекту обумовлений їхніми перевагами, продуктивністю та відповідністю вимогам проекту. Використання TypeScript та React на клієнтській стороні забезпечує надійність, гнучкість та продуктивність розробки інтерфейсу користувача. На серверній частині використання NestJS та Prisma ORM дозволяє створювати структуровані, типізовані та ефективні серверні додатки

4.5 Головний алгоритм програми

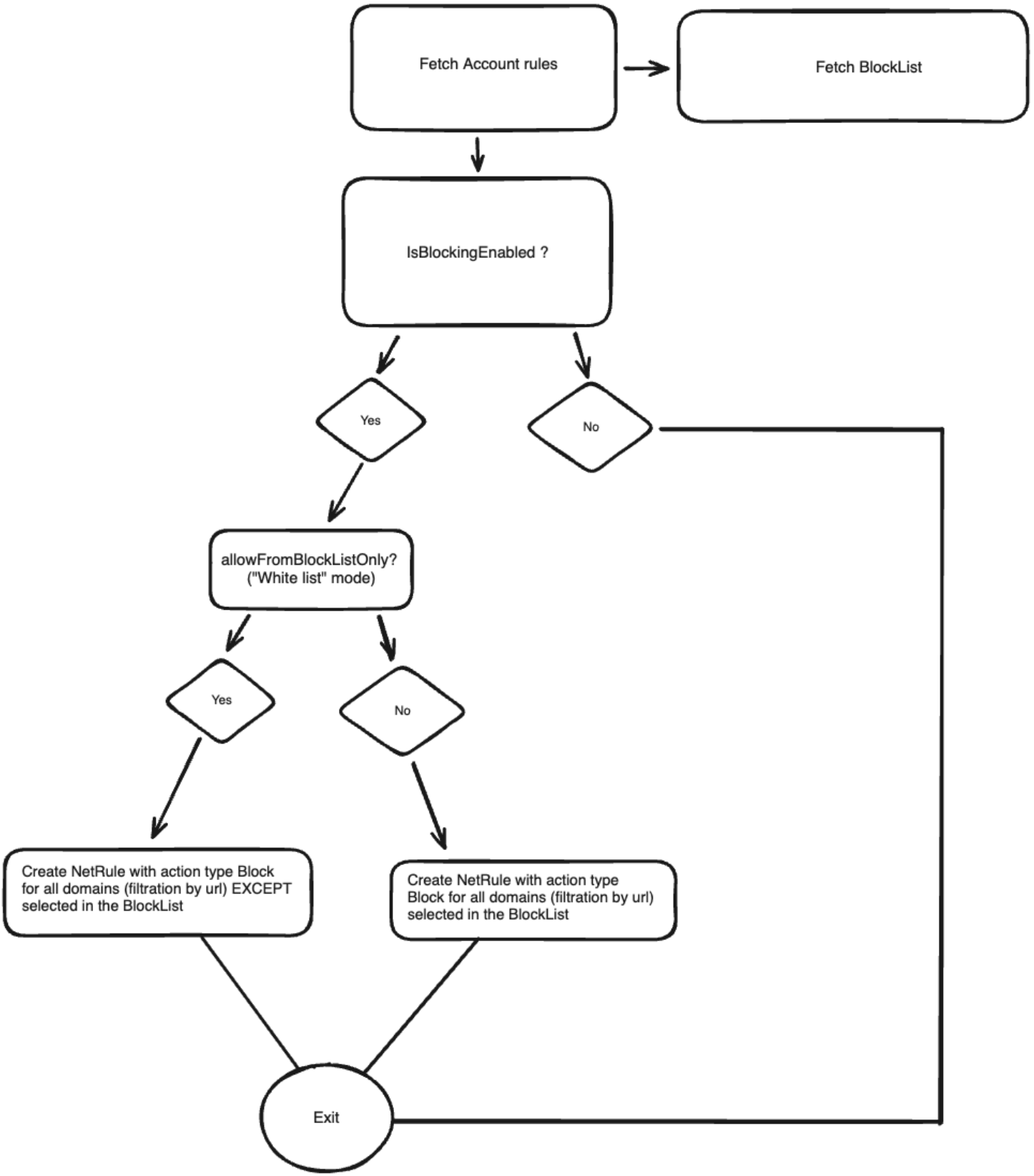


Рисунок 4.1 – Головний алгоритм програми

Ознайомитися з реалізацією можна в Додатку В.

4.6 Користувацький інтерфейс програмної системи

Користувацький інтерфейс є критично важливим компонентом будь-якої програмної системи, оскільки він забезпечує зв'язок між користувачем та функціональністю додатку. У розробленій системі користувацький інтерфейс створений з використанням бібліотек React та Shadcn UI, які забезпечують сучасний, інтуїтивний та привабливий досвід для користувачів.

React - це популярна бібліотека JavaScript для створення користувацьких інтерфейсів, яка забезпечує ефективну роботу та високу продуктивність завдяки своїй віртуальній DOM-моделі та компонентній архітектурі. Shadcn UI - це бібліотека компонентів інтерфейсу користувача, яка надає стильні та легко налаштовувані елементи інтерфейсу, такі як кнопки, форми, модальні вікна та інші.

Користувацький інтерфейс системи складається з декількох ключових сторінок та компонентів, які забезпечують доступ до основних функцій додатку. На головній сторінці користувачі можуть переглядати загальну інформацію про систему та отримувати доступ до інших розділів.

Сторінка авторизації дозволяє користувачам увійти в систему, використовуючи свої облікові дані. Після успішної автентифікації користувачі отримують доступ до захищених розділів додатку, таких як особистий кабінет та функціональність, відповідно до їхніх прав доступу.

Особистий кабінет є центральною точкою для користувачів, де вони можуть переглядати та керувати своїми персональними даними, налаштуваннями та історією взаємодії з системою.

Крім того, система може містити додаткові сторінки та компоненти, такі як сторінки пошуку, перегляду деталей, редагування та створення нових записів, залежно від конкретних вимог та функціоналу додатку.

Під час розробки користувацького інтерфейсу особлива увага приділялася забезпеченню зручності використання, доступності та привабливого дизайну. Компоненти інтерфейсу організовані логічно та інтуїтивно, з використанням стандартних патернів дизайну та найкращих практик.

На Рисунку 4.2 можна ознайомитись з усіма діями користувача в системі.

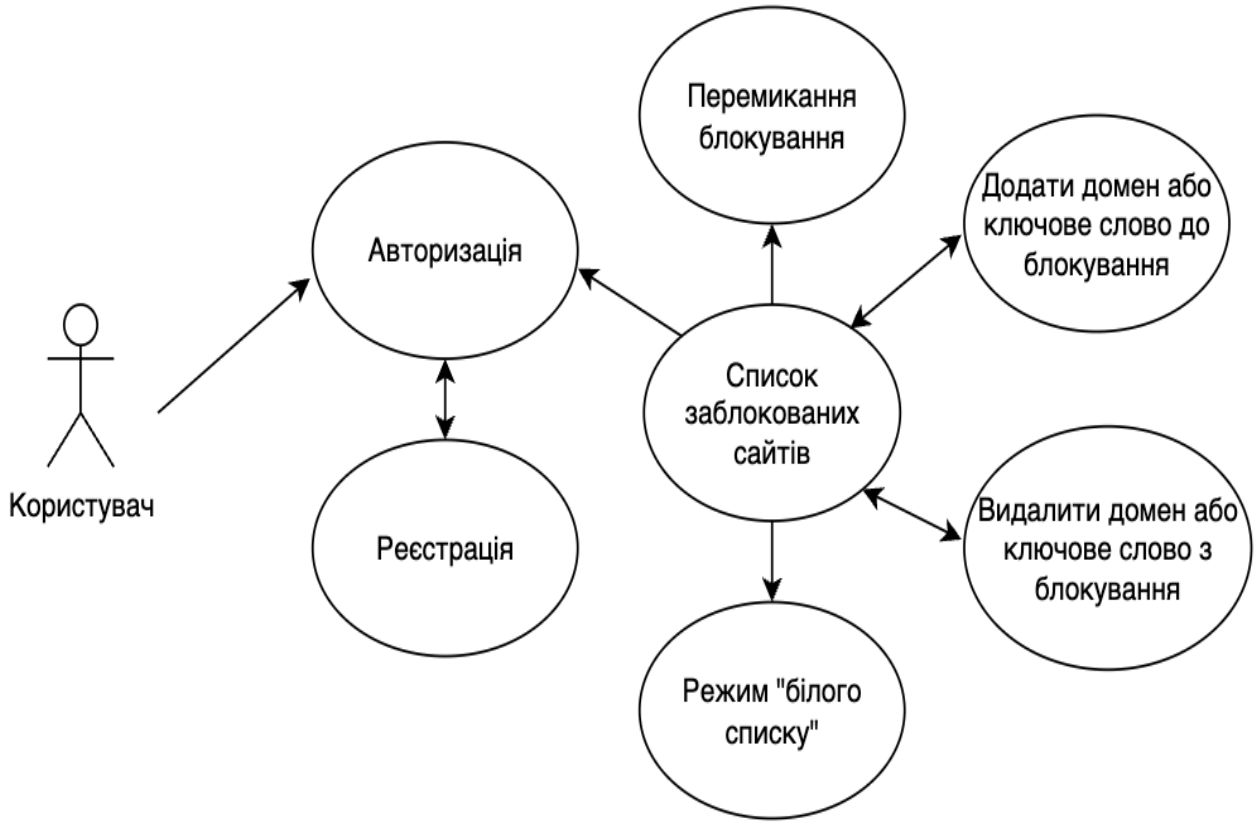


Рисунок 4.2 – Діаграма прецедентів

Ознайомитись з реалізацією користувацького інтерфейсу можна в розділі “Сценарій роботи користувача з системою”.

В результаті, інтерфейс створеної системи має достатньо просту структуру, яка відображає функціонал, доступний кінцевому користувачу.

4.7 Сценарій роботи користувача з системою

Сценарій роботи користувача з системою розпочинається з встановлення розширення для веб-переглядача. Після завантаження розширення користувачеві відображається модальне вікно з повідомленням про необхідність авторизації (Рисунок 4.3). На цьому етапі користувач може або увійти в існуючий обліковий запис, або зареєструвати новий.

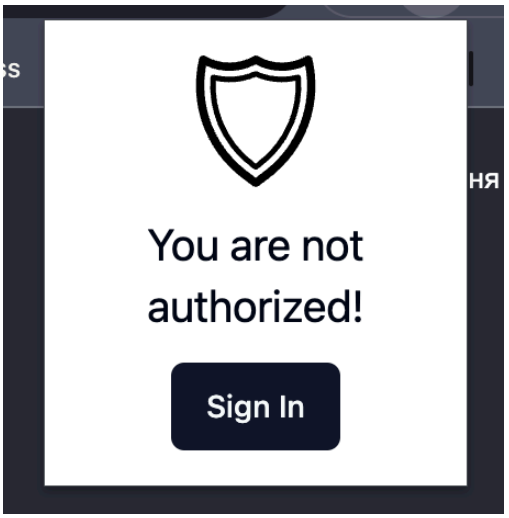


Рисунок 4.3 Модальне вікно з повідомленням про авторизацію

Натискаючи на кнопку “Увійти”, користувач потрапляє на сторінку авторизації (Рисунок 4.4)

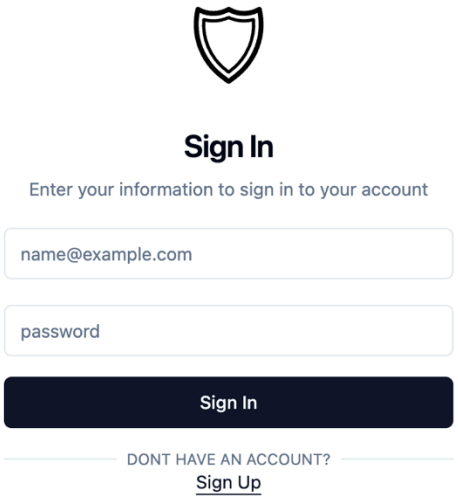


Рисунок 4.4 Сторінка авторизації

Якщо користувач ще не має облікового запису, він може натиснути на кнопку "Зареєструватись" і перейти до процесу реєстрації нового облікового запису.

Після успішної авторизації або реєстрації користувача буде перенаправлено на головну сторінку системи (Рисунок 4.5). Головна сторінка є центральним пунктом управління для користувача, де він може виконувати різні дії та налаштування.

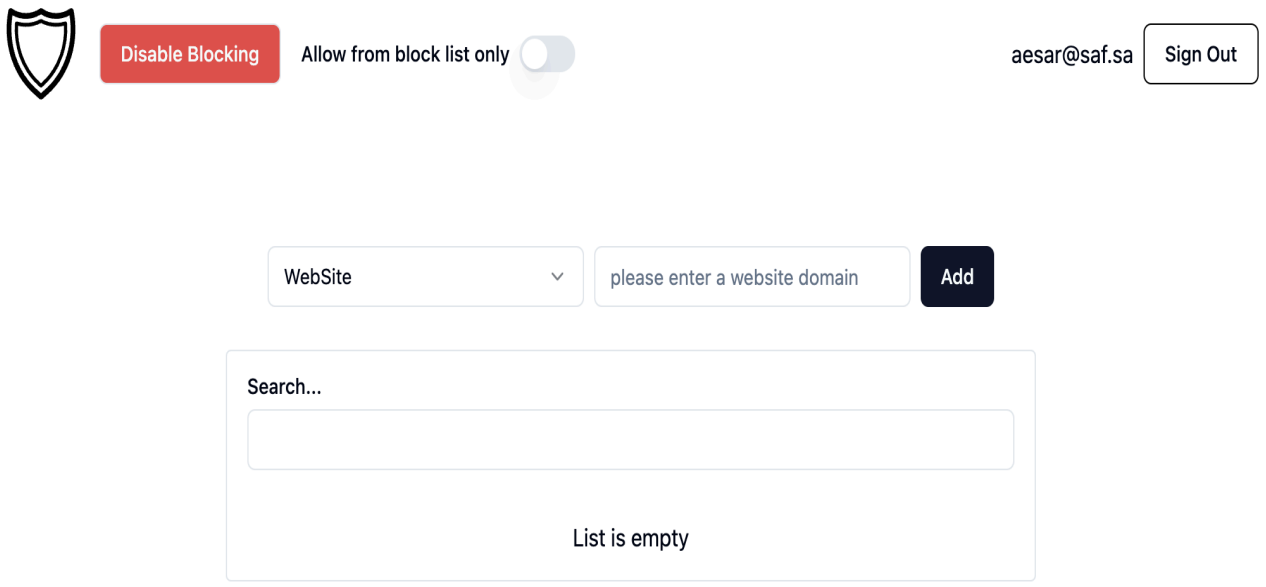


Рисунок 4.5 Головна сторінка

На головній сторінці користувач має доступ до наступних функцій:

1. Перемикання блокування: Користувач може ввімкнути або вимкнути функцію блокування сайтів за допомогою відповідного перемикача.

2. Перемикання режиму "білого списку": У цьому режимі користувачеві будуть доступні лише ті сайти, які він сам додасть до списку дозволених, а всі інші будуть заблоковані за замовчуванням. Цей режим дозволяє забезпечити більш жорсткий контроль доступу до веб-ресурсів.

3. Додавання доменів сайтів або ключових слів до списку заблокованих: Користувач може додавати конкретні домени сайтів або ключові слова до списку заблокованих. Це дозволяє блокувати доступ до певних веб-ресурсів або контенту, який містить визначені ключові слова.

4. Вихід з облікового запису: Користувач може здійснити вихід з облікового запису за допомогою відповідної кнопки.

Щоб продемонструвати процес блокування сайту, розглянемо наступний приклад (Рис. 4.6-4.7):

Користувач вводить домен сайту або ключове слово, яке потрібно заблокувати, в поле введення на головній сторінці (Рисунок 4.6). Після натискання кнопки "Додати" введений елемент буде доданий до списку заблокованих.

Коли користувач спробує відвідати заблокований сайт або сторінку, яка містить заблоковане ключове слово, система відобразить повідомлення про блокування з відповідним поясненням (Рисунок 4.7).

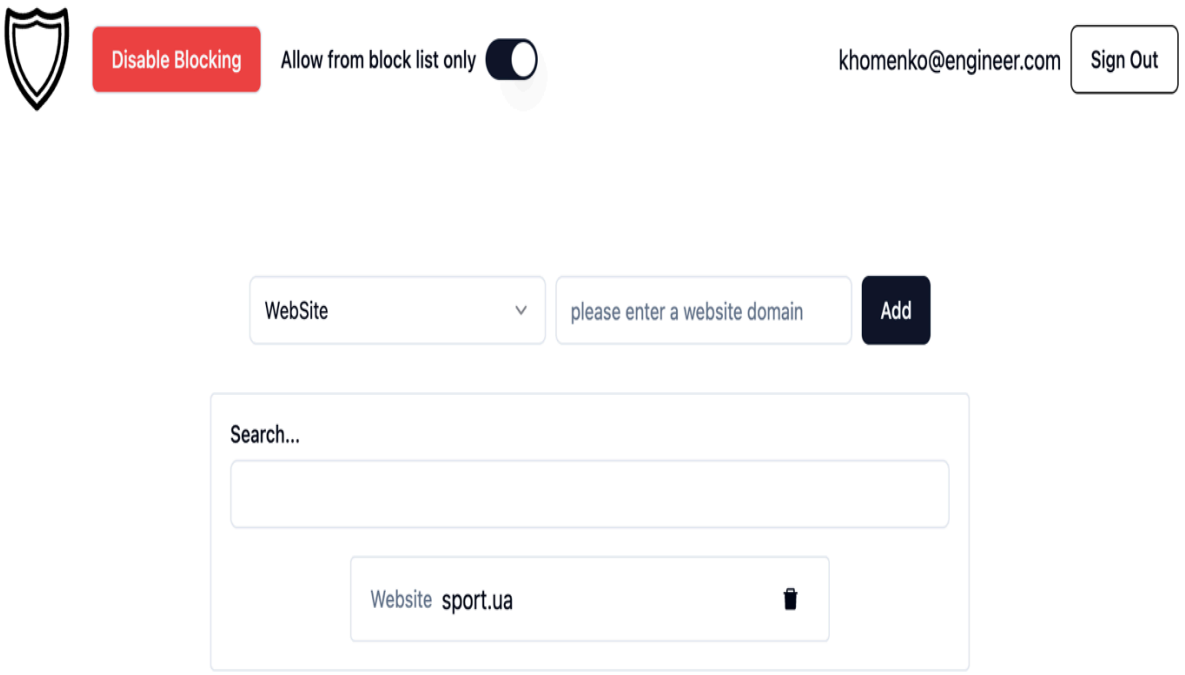


Рисунок 4.6 Додавання сайту до списку

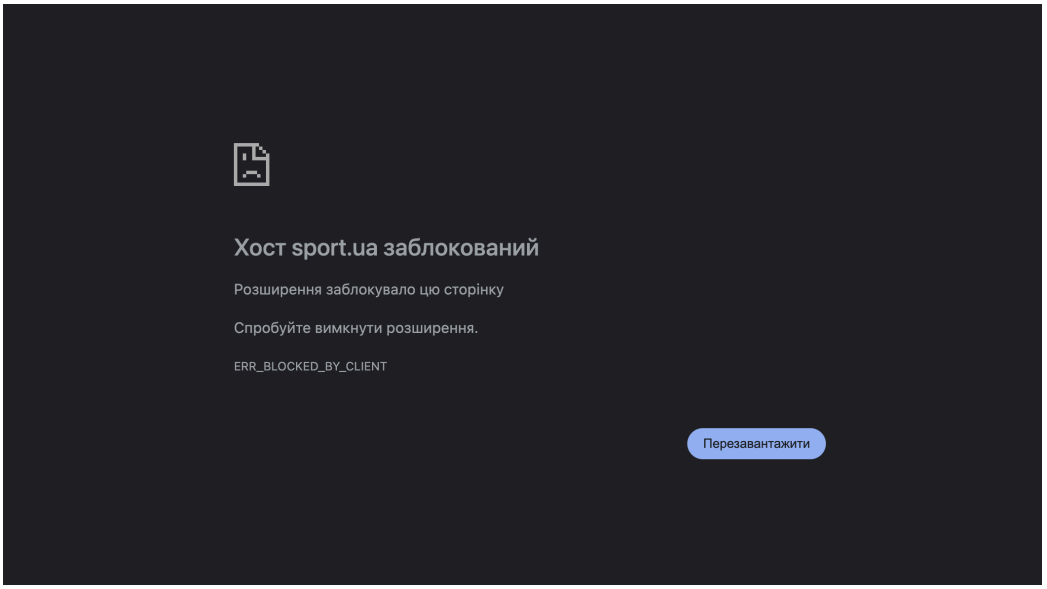


Рисунок 4.7 Сайт заблоковано розширенням

ВИСНОВКИ

В результаті виконання атестаційної роботи було розроблено систему, яка забезпечує батьківський контроль для веб-сервісів. Основними компонентами системи є хром-розширення, сервер, система авторизації, база даних, клієнтська частина та браузерне розширення, яке взаємодіє з веб-сервісами. Для розробки використовувались сучасні технології розробки, такі як JavaScript (TypeScript), NestJS, PostgreSQL, Prisma ORM у середовищі Visual Studio Code.

Отже, в результаті виконання атестаційної випускної роботи було проведено:

- аналіз предметної області;
- комплексне проектування системи;
- комплексне проектування бази даних.

На основі розглянутого дослідження зроблено висновок, що задачу забезпечення безпеки та батьківського контролю для веб-сервісів можна значно оптимізувати, а також покращити ефективність та зменшити кількість витраченого часу батьками завдяки впровадженню сучасної системи батьківського контролю, яка дозволяє централізовано керувати доступом до веб-ресурсів та забезпечує прозорість і оперативність контролю.

Список використаних джерел

1. Архітектурний шаблон MVC [Електронний ресурс] // Режим доступу: https://www.geeksforgeeks.org/mvc-design-pattern/\
2. Основні поняття баз даних [Електронний ресурс] // Режим доступу: https://sites.google.com/view/ddkbmta-info/лекції/системи-керування-базами-даних-microsoft-access/основні-поняття-баз-даних
3. Системи управління БД [Електронний ресурс] // Режим доступу: https://wphost.me/tips/what-is-db/
4. Моделювання даних за допомогою ERD діаграм [Електронний ресурс] // Режим доступ https://kursoviks.com.ua/bd_kompyuternyye/article_post/1580-laboratorna-robota-no3-modelyuvannya-danikh-za-dopomogoyu-diagram-sutnist-zvyazok-erd-entity-relationship-diagrams-v-informatsiynikh-sistemakh-i-tekhnologiyakh
5. Порівняння JavaScript та TypeScript [Електронний ресурс] // Режим доступу: https://refine.dev/blog/javascript-vs-typescript/
6. Документація React [Електронний ресурс] // Режим доступу: https://react.dev/
7. Порівняння React, Angular і Vue [Електронний ресурс] // Режим доступу: https://dou.ua/forums/topic/39933/
8. Документація NestJS [Електронний ресурс] // Режим доступу: https://nestjs.com/
9. Порівняння NestJS та Express [Електронний ресурс] // Режим доступу: https://flatirons.com/blog/nestjs-vs-express/
10. ORM [Електронний ресурс] // Режим доступу: https://www.hwlibre.com/uk/реляційне-відображення-об’єкта-orm/
11. Parental controls on children’s computer and Internet use [Електронний ресурс]

// Режим доступу: https://www.researchgate.net/publication/240448141_Parental_controls_on_children's_computer_and_Internet_use

12. Do parental control tools fulfil family expectations for child protection? [Електронний ресурс] // Режим доступу: https://www.tandfonline.com/doi/full/10.1080/17482798.2023.2265512
13. Parental control and adolescent Internet addiction: the moderating effect of parent-child relationships [Електронний ресурс] // Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10248257/
14. Parental controls & online child protection: a survey of tools & methods [Електронний ресурс] // Режим доступу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1268433
15. What are extensions? [Електронний ресурс] // Режим доступу: https://developer.chrome.com/docs/extensions/mv2/overview
16. Chrome Extensions [Електронний ресурс] // Режим доступу: https://developer.chrome.com/docs/extensions/get-started
17. Advanced security for NestJS Applications [Електронний ресурс] // Режим доступу: https://javascript.plainenglish.io/advanced-security-for-nestjs-applications-92c8ec1b90ee
18. PostgreSQL database security best practices [Електронний ресурс] // Режим доступу: https://www.percona.com/blog/postgresql-database-security-best-practices/
19. What is System Design [Електронний ресурс] // Режим доступу: https://www.geeksforgeeks.org/what-is-system-design-learn-system-design/
20. Lucidchart diagrams [Електронний ресурс] // Режим доступу: https://www.lucidchart.com/pages

ДОДАТКИ

Додаток А. Об'єкти передачі даних програми

```
class SignUpBodyDto {
  @ApiProperty({
    example: 'test@gmail.com',
  })
  @IsEmail()
  email: string;

  @ApiProperty({
    example: '1234',
  })
  @IsNotEmpty()
  password: string;
}
```

```
class SignInBodyDto {
  @ApiProperty({
    example: 'test@gmail.com',
  })
  @IsEmail()
  email: string;

  @ApiProperty({
    example: '1234',
  })
  @IsNotEmpty()
  password: string;
}
```

```
class PatchAccountDto {
  @ApiProperty({ required: false })
  @IsBoolean()
  @IsOptional()
  isBlockingEnabled: boolean;

  @ApiProperty({ required: false })
  @IsBoolean()
  @IsOptional()
  allowFromBlockListOnly: boolean;
}
```

```
class BlockItemDto {
```

```
  @ApiProperty()
  id: number;

  @ApiProperty()
  blockListId: number;

  @ApiProperty({
    enum: [$Enums.BlockItemType.KeyWord, $Enums.BlockItemType.Website],
  })
  type: $Enums.BlockItemType;

  @ApiProperty()
  data: string;

  @ApiProperty()
  createdAt: Date;
}
```

Додаток Б. Скрипти створення бази даних

```
-- CreateEnum
CREATE TYPE "BlockItemType" AS ENUM ('Website', 'KeyWord');

-- CreateTable
CREATE TABLE "User" (
    "id" SERIAL NOT NULL,
    "email" TEXT NOT NULL,
    "hash" TEXT NOT NULL,
    "salt" TEXT NOT NULL,

    CONSTRAINT "User_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "Account" (
    "id" SERIAL NOT NULL,
    "ownerId" INTEGER NOT NULL,
    "isBlockingEnabled" BOOLEAN NOT NULL,

    CONSTRAINT "Account_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "BlockList" (
    "id" SERIAL NOT NULL,
    "ownerId" INTEGER NOT NULL,

    CONSTRAINT "BlockList_pkey" PRIMARY KEY ("id")
);

-- CreateTable
CREATE TABLE "BlockItem" (
    "id" SERIAL NOT NULL,
    "blockListId" INTEGER NOT NULL,
    "type" "BlockItemType" NOT NULL,
    "data" TEXT NOT NULL,
    "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT "BlockItem_pkey" PRIMARY KEY ("id")
);

-- CreateIndex
CREATE UNIQUE INDEX "User_email_key" ON "User"("email");

-- CreateIndex
CREATE UNIQUE INDEX "Account_ownerId_key" ON "Account"("ownerId");

-- CreateIndex
```

```
CREATE UNIQUE INDEX "BlockList_ownerId_key" ON "BlockList"("ownerId");

-- AddForeignKey
ALTER TABLE "Account" ADD CONSTRAINT "Account_ownerId_fkey" FOREIGN KEY ("ownerId")
REFERENCES "User"("id") ON DELETE RESTRICT ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "BlockList" ADD CONSTRAINT "BlockList_ownerId_fkey" FOREIGN KEY ("ownerId")
REFERENCES "User"("id") ON DELETE RESTRICT ON UPDATE CASCADE;

-- AddForeignKey
ALTER TABLE "BlockItem" ADD CONSTRAINT "BlockItem_blockListId_fkey" FOREIGN KEY
("blockListId") REFERENCES "BlockList"("id") ON DELETE RESTRICT ON UPDATE CASCADE;
```

Додаток В. Головний алгоритм програми

```
export async function getBlockListNetRules() {
  const blockList = await blockListControllerGetList();

  const allowFromBlockListOnlyш = await accountControllerGetAccount().then(
    (r) => r.allowFromBlockListOnly,
  );

  const domainRules = blockList.items
    .filter((item) => item.type === BlockItemDtoType.Website)
    .map(
      (item): NetRule => ({
        id: item.id,
        action: {
          type: allowFromBlockListOnly ? NetRuleActionType.ALLOW : NetRuleActionType.BLOCK,
        },
        condition: {
          urlFilter: item.data,
          resourceTypes: [NetRuleResourceType.MAIN_FRAME],
        },
      }),
    );

  const defaultBlockRule = {
    id: domainRules.length + 1,
    action: {
      type: NetRuleActionType.BLOCK,
    },
    condition: {
      urlFilter: '*',
      resourceTypes: [NetRuleResourceType.MAIN_FRAME],
    },
  };

  const allowAdminPanelRule = {
    id: domainRules.length + 2,
    action: {
      type: NetRuleActionType.ALLOW,
    },
    condition: {
      urlFilter: ADMIN_URL,
      resourceTypes: [NetRuleResourceType.MAIN_FRAME],
    },
  };

  const allRules = [allowAdminPanelRule, ...domainRules];

  if (allowFromBlockListOnly) {
    allRules.push(defaultBlockRule as NetRule);
  }
```

```
  return allRules;
}
```

Додаток Г. Презентація

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА ТА АРХІТЕКТУРИ

Київ 2024

Кваліфікаційна робота на здобуття освітнього рівня “бакалавр” за темою:

Розробка системи батьківського контролю для веб сервісів

Керівник КР: д.т.н., проф. Бородавка Є.В.
Розробив: студент спеціальності
122 “Комп’ютерні науки” ОС “бакалавр”
Хоменко В.Р.

Аналіз предметної області та постановка задачі.

- **Предметною областю** даного проекту є сфера онлайн-безпеки, батьківського контролю та керування доступом до веб-ресурсів для дітей. Система батьківського контролю для веб-сервісів покликана надати батькам ефективні інструменти для моніторингу та обмеження доступу своїх дітей до певних веб-сайтів, контенту або онлайн-сервісів.

- **Об’єктом дослідження** роботи є процес контролю веб сервісів батьками.

- **Предметом дослідження** є розробка та впровадження програмного забезпечення для батьківського контролю в браузерах за допомогою мови програмування JavaScript та його фреймворків і бібліотек

Актуальність.

Основними факторами, що підсилюють актуальність системи, є:

01

Зростання кількості веб-сервісів та додатків, доступних для дітей та підлітків, які можуть містити несприятливий контент або відволікати дітей

02

Однак, не весь цей контент може бути відповідним для їхнього віку, тому важливо мати засоби контролю, щоб забезпечити, що діти використовують Інтернет безпечно та продуктивно.

03

Системи контролю можуть допомогти направляти активність дітей в напрямку освіти, обмежуючи доступ до загрозливих ресурсів.

Стан вже існуючих рішень.

- На сьогоднішній день існують різноманітні браузерні розширення, призначені для керування та моніторингу онлайн активності дітей. Деякі з них надають такі можливості, як:
  - Блокування небажаних веб-сайтів та контенту з несанкціонованим вмістом;
  - Встановлення обмежень часу використання інтернету або певних веб-сайтів;
  - Моніторинг активності;

- Дані рішення є функціонуючими та зручними, але з постійним покращенням функціоналу самої платформи, вдосконаленням можливостей мов програмування для створення програми та обробки взаємодії користувача з нею, можливі оновлення та модернізації.

Порівняння існуючих рішень.

Найпопулярніші розширення* – BlockSite та Web Blocker

* згідно статистики Chrome Web Store.

Рішення	BlockSite	Web Blocker
Блокування сайтів	Так	Так
Білий список сайтів	Так	Ні
Фільтрація за ключовими словами	Так	Ні
Моніторинг використання	Ні	Ні
Розклад блокування	Так	Так
Вартість	Платний (безкоштовно під час пробного періоду)	Безкоштовний

Архітектура системи.

- Система була спроектована та реалізована відповідно до шаблону проектування MVC (Model-View-Controller), який є одним з найпоширеніших та ефективних підходів до організації структури програмного забезпечення.

Модель (Model) складається з класів BlockList, Auth, Account.

Вид (View) у цій архітектурі є веб-інтерфейс.

Контролер (Controller) виступає у вигляді ApiController.

Шаблон проектування MVC

Концептуальна схема системи.

- Концептуальна схема системи – це високорівневий опис основних компонентів і їх взаємодії в системі. Вона допомагає зрозуміти загальну структуру та призначення системи без деталей реалізації.

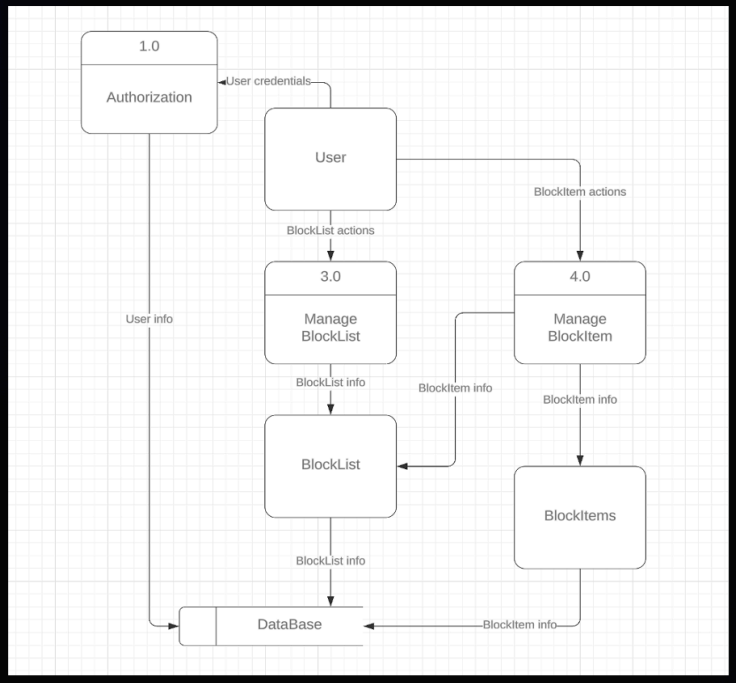
- У концептуальній схемі виокремлюються основні модулі та компоненти системи, такі як користувацький інтерфейс, серверна частина, база даних та інші. Це допомагає зрозуміти загальну структуру системи та її функціональність.

Концептуальна схема системи

Функціональна схема системи.

- Функціональна схема (Functional Flow Block Diagram - FFBD) - це багаторівнева, послідовна в часі, покрокова блок-схема функціонального потоку системи. FFBD відображають завдання, визначені за допомогою функціональної декомпозиції, і показують їх у їх логічному, послідовному взаємозв'язку.

- FFBD є багаторівневою та послідовною в часі, що дозволяє деталізувати функції на різних рівнях абстракції. На верхньому рівні зазвичай зображені основні функціональні блоки системи, а на нижчих рівнях ці блоки розкриваються більш детально.

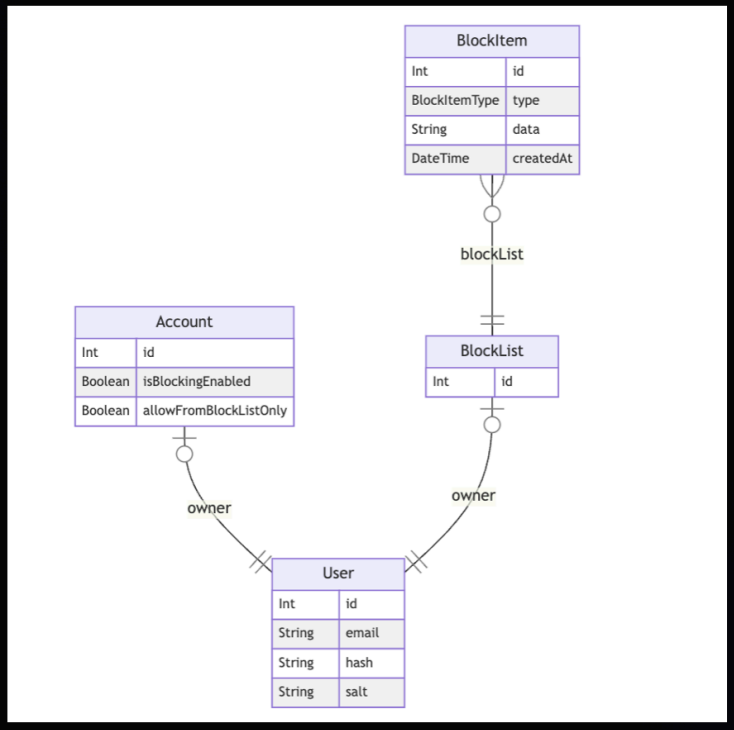


Функціональна схема системи

Проектування бази даних.

- База даних — набір впорядкованої та структурованої інформації, що зберігається в електронному вигляді. Завдяки БД (Базам даних) набагато зручніше виконувати управління, коригування, оновлення, контроль та впорядкування.

- Для системи було обрано PosgreSQL, яка належить до об'єктно-реляційних СУБД і має відкритий вихідний код, ця СУБД надає великі можливості для налаштування та адаптації. Підтримка широкого набору функцій забезпечує гнучкість у реалізації різних сценаріїв використання. Висока продуктивність, надійність та рівень безпеки гарантують ефективну і захищену роботу з даними.



Діаграма“сутність-зв’язок”.

Практична реалізація.

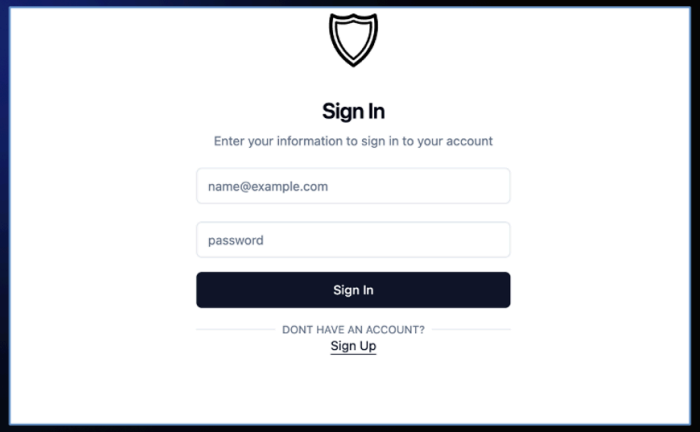
- Visual Studio Code було обрано основним середовищем розробки системи.
- Мовою програмування було обрано надбудову над JavaScript, що додає статичну типізацію, а саме TypeScript.

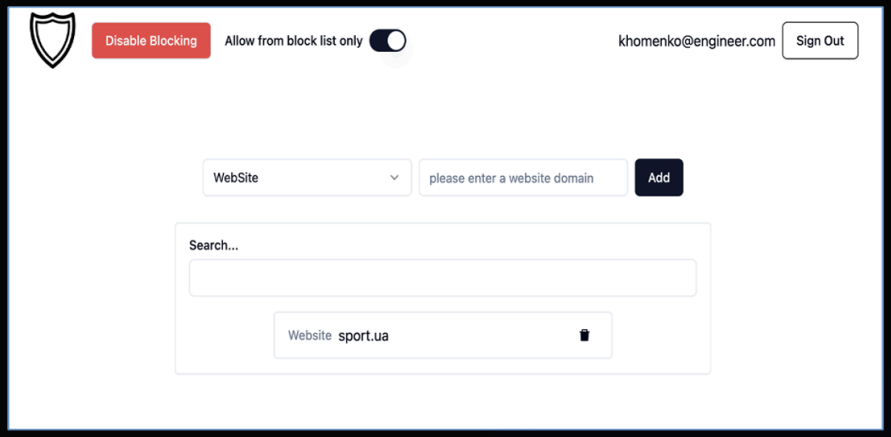
Технології клієнтської частини:

- React (бібліотека для побудови користувацьких інтерфейсів)
- ShadcnUI (бібліотека стилів)
- Tanstack Query (бібліотека для запитів)

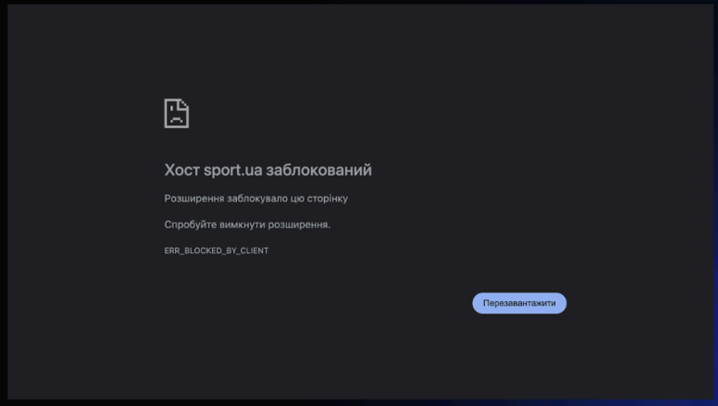
Технології серверної частини:

- NestJS (фреймворк для серверних додатків)
- Prisma (ORM для бази даних)





Результати розробки.



Висновки.

- В результаті виконання атестаційної роботи було розроблено систему, яка забезпечує батьківський контроль для веб-сервісів. Основними компонентами системи є хром-розширення, сервер, система авторизації, база даних, клієнтська частина та браузерне розширення, яке взаємодіє з веб-сервісами. Для розробки використовувались сучасні технології розробки, такі як JavaScript (TypeScript), NestJS, PostgreSQL, Prisma ORM у середовищі Visual Studio Code.

- На основі розглянутого дослідження зроблено висновок, що задачу забезпечення безпеки та батьківського контролю для веб-сервісів можна значно оптимізувати, а також покращити ефективність та зменшити кількість витраченого часу батьками завдяки впровадженню сучасної системи батьківського контролю, яка дозволяє централізовано керувати доступом до веб-ресурсів та забезпечує прозорість і оперативність контролю.