ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: Проектування та розробка інтерактивного вебзастосунка для управління файловими архівами на сервері

Виконав: здобувач вищої освіти 4 курсу,
групи ІСтаТ 2022-1 спеціальності
126 Інформаційні системи та технології
(шифр і назва спеціальності)

Кіріл ЄЛІСТРАТОВ
(ім'я та прізвище)

Керівник: Марина НОВОЖИЛОВА
(ім'я та прізвище)

Рецензент: к.т.н., доц. Юрій ПАХОМОВ
(ім'я та прізвище)

м. Харків – 2026 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра Комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ

Марина Новожилова

«22» _червня_ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Єлістратов Кіріл Вячеславович
(прізвище, ім'я, по батькові)

1. Тема роботи Проектування та розробка інтерактивного вебзастосунка для управління файловими архівами на сервері
керівник роботи проф. доктор наук Новожилова М. В
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від ««22» травня 2026 р. № 440-03
2. Термін подання роботи здобувачем вищої освіти 16.06.2026
3. Вихідні дані до роботи Проектування та розробка інтерактивного вебдодатка (вебзастосунка) для управління файловими архівами на сервері

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження існуючих серверних рішень для обробки файлів та визначення цільової аудиторії, проектування інформаційного і математичного забезпечення програмного модуля, опис етапів практичної реалізації та функціонального тестування створеного вебзастосунка

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація – 15 аркушів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Марина НОВОЖИЛОВА	11.05.2026	19.05.2026
Розділ II	Марина НОВОЖИЛОВА	18.05.2026	06.06.2026
Розділ III	Марина НОВОЖИЛОВА	30.05.2026	06.06.2026
Розділ IV	Вікторія МАЛИШЕВА	03.06.2026	11.06.2026

7. Дата видачі завдання 11.05.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми дипломної роботи	03.05.2026	Викон.
2	Затвердження теми, наукового керівника, завдань, та календарного плану, підготовка до дипломної роботи	05.06.2026	Викон.
3	Написання I розділу	11.05.2026	Викон.
4	Подання розділу на перевірку керівнику, внесення правок	19.05.2026	Викон.
5	Написання II розділу	18.05.2026	Викон.
6	Подання розділу на перевірку керівнику, внесення правок	06.06.2026	Викон.
7	Написання III розділу	30.05.2026	Викон.
8	Подання розділу на перевірку керівнику, внесення правок	06.06.2026	Викон.
9	Написання IV розділу	03.06.2026	Викон.
10	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	11.06.2026	Викон.
11	Подання доопрацьованого варіанту роботи керівнику	15.06.2026	Викон.
12	Відпралення роботи на перевірку плагіату	16.06.2026	Викон

Здобувач вищої освіти _____ Кіріл ЄЛІСТРАТОВ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марина НОВОЖИЛОВА
(підпис) (прізвище та ініціали)"

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи здобувача вищої освіти групи ІСтаТ 2022-1 спеціальності 126 «Інформаційні системи та технології» Єлістратова Кіріла Вячеславовича за темою Проектування та розробка інтерактивного вебдодатка (вебзастосунка) для управління файловими архівами на сервері, складається з 4 розділів, містить 69 сторінки тексту, 16 рисунків, 6 таблиць, 20 джерел

Робота присвячена проектуванню та розробці безпечного портативного серверного веб-модуля для управління файловими ZIP-архівами. Метою дослідження є створення автономного програмного рішення для автоматизації процесів завантаження, архівації та декомпресії даних на віддалених веб-серверах із забезпеченням високого рівня захисту від поширених кіберзагроз. Під час розробки застосовано архітектурний підхід Single-File Application, де вся серверна логіка та генерація клієнтського інтерфейсу інкапсульовані в єдиному файлі мовою PHP.

Особливу увагу приділено механізмам кібербезпеки: імплементовано глибоку валідацію MIME-типів на базі розширення finfo, впроваджено систему перевірки CSRF-токенів, а також алгоритми протидії атакам типу Zip Slip та захисту від вичерпання ресурсів сервера. Графічний інтерфейс користувача побудовано з використанням семантичної розмітки HTML5, каскадних таблиць стилів CSS3 та нативного JavaScript, що дозволило створити адаптивне та ергономічне середовище для адміністратора.

Окрім технічної реалізації, у роботі проведено комплексний аналіз умов праці оператора системи та розроблено рекомендації з охорони праці, які враховують психофізіологічні фактори та сучасні безпекові виклики. Практичне значення отриманих результатів полягає у створенні повністю готового до експлуатації інструменту, який значно оптимізує процеси резервного копіювання, міграції та адміністрування веб-проєктів на серверах без попередньо встановлених панелей керування.

ANNOTATION

Structure and scope of work. Explanatory note of the qualification work of a higher education applicant of the IStaT group 2022-1, specialty 126 "Information Systems and Technologies" by Kirill Vyacheslavovich Yelistratov on the topic Design and development of an interactive web application (web application) for managing file archives on the server, consists of 4 sections, contains 69 pages of text, 16 figures, 6 tables, 20 sources

The work is devoted to the design and development of a secure portable server web module for managing file ZIP archives. The purpose of the research is to create an autonomous software solution for automating the processes of downloading, archiving and decompression of data on remote web servers while ensuring a high level of protection against common cyber threats. During the development, the Single-File Application architectural approach was used, where all server logic and client interface generation are encapsulated in a single file in the PHP language.

Special attention is paid to cybersecurity mechanisms: deep validation of MIME types based on the finfo extension has been implemented, a CSRF token verification system has been introduced, as well as algorithms for countering Zip Slip attacks and protecting against server resource exhaustion. The graphical user interface has been built using HTML5 semantic markup, CSS3 cascading style sheets, and native JavaScript, which has allowed creating an adaptive and ergonomic environment for the administrator. In addition to technical implementation, the work has conducted a comprehensive analysis of the system operator's working conditions and developed recommendations for labor protection that take into account psychophysiological factors and modern security challenges. The practical significance of the results obtained lies in creating a fully operational tool that significantly optimizes the processes of backup, migration, and administration of web projects on servers.

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

CMS (Content Management System) - інформаційна система управління контентом, що використовується для спільного процесу створення, редагування та управління текстовими і мультимедійними документами (наприклад, WordPress, Joomla). [1]

CSRF (Cross-Site Request Forgery) - міжсайтова підробка запиту; вид хакерської атаки, за якої зловмисник змушує браузер жертви виконати небажану дію на довіреному сайті, де користувач наразі авторизований. [1]

DOM (Document Object Model) - об'єктна модель документа; програмний інтерфейс для HTML та XML документів, що представляє сторінку як дерево об'єктів. [1]

FTP (File Transfer Protocol) - стандартний мережевий протокол, призначений для передачі файлів між клієнтом і сервером у комп'ютерній мережі. [1]

GUI (Graphical User Interface) - графічний інтерфейс користувача; система засобів для взаємодії користувача з комп'ютером, заснована на представленні всіх доступних об'єктів у вигляді графічних елементів (вікон, кнопок, меню).

IDE (Integrated Development Environment) - інтегроване середовище розробки; комплексне програмне забезпечення для програмістів, що поєднує в собі редактор коду, компілятор/інтерпретатор та засоби налагодження (наприклад, Visual Studio Code, PhpStorm). [1]

MIME-тип (Multipurpose Internet Mail Extensions) - стандарт, що описує тип та формат даних у файлі або мережевому з'єднанні (наприклад, application/zip, text/html). [1]

PHP (Hypertext Preprocessor) - поширена скриптова мова програмування загального призначення з відкритим вихідним кодом, спеціально сконструйована для веб-розробки та інтеграції в HTML. [1]

SPL (Standard PHP Library) - стандартна бібліотека PHP; набір вбудованих інтерфейсів та класів, призначених для вирішення стандартних завдань, таких як обхід директорій (ітератори) та робота зі структурами даних. [1]

SSH (Secure Shell) - мережевий протокол прикладного рівня, що дозволяє проводити віддалене управління операційною системою та тунелювання TCP-з'єднань із шифруванням. [1]

UI/UX (User Interface / User Experience) - дизайн інтерфейсу користувача та користувацький досвід; комплексний підхід до проектування програмного забезпечення, що враховує як його зовнішній вигляд, так і зручність використання логіки. [1]

UML (Unified Modeling Language) - уніфікована мова моделювання; стандартизована мова графічного опису для об'єктного моделювання в галузі розробки програмного забезпечення. [2]

VPS / VDS (Virtual Private Server / Virtual Dedicated Server) - послуга надання віртуального виділеного сервера, який емулює роботу фізичного сервера з правами адміністратора (root-доступом). [2]

Zip Slip - критична вразливість процесу декомпресії, яка дозволяє зловмиснику розпакувати файли поза межами цільової директорії шляхом маніпуляцій зі шляхами (наприклад, використанням символів ../), що може призвести до перезапису системних файлів. [2]

Zip-бомба (Архівна бомба) - шкідливий архівний файл невеликого розміру, який містить величезну кількість високостиснутих даних. Призначений для виклику відмови в обслуговуванні (DoS) шляхом вичерпання дискового простору або оперативної пам'яті під час спроби його розпакувати. [2]

ЗМІСТ

ВСТУП

В умовах стрімкого розвитку веб-технологій та постійного збільшення обсягів цифрових даних, ефективне управління файлами на віддалених серверах є одним із ключових завдань веб-розробників та системних адміністраторів. Традиційні методи передачі великої кількості дрібних файлів (наприклад, через протокол FTP) є вкрай повільними та неефективними через великі накладні витрати на встановлення мережевих з'єднань. Оптимальним рішенням є передача даних у вигляді єдиного стиснутого архіву з його подальшим розпакуванням безпосередньо на сервері.

Однак використання сторонніх скриптів-разархіваторів або панелей керування часто пов'язане з високими ризиками інформаційній безпеці. Багато існуючих рішень не мають належного захисту від таких вразливостей, як завантаження шкідливого програмного забезпечення (веб-шелів), атаки типу «Zip Slip»[2] (вихід за межі цільової директорії при розпакуванні) та «Zip-бомб»[2] (вичерпання ресурсів сервера при розпакуванні рекурсивних архівів). Саме тому розробка власного, вузькоспеціалізованого, автономного та максимально захищеного веб-модуля для роботи з ZIP-архівами є актуальним та практично значущим завданням.

Об'єктом дослідження є процес серверної обробки, передачі та управління стиснутими файловими даними в сучасному веб-середовищі.

Предметом дослідження виступають методи, алгоритми та програмні засоби забезпечення безпечної архівації, декомпресії та управління файловими структурами з використанням мови програмування PHP.

Мета роботи полягає у проектуванні та розробці безпечного серверного веб-модуля мовою PHP для автоматизації процесів завантаження, архівації, розпакування та управління ZIP-архівами з інтегрованими механізмами захисту від поширених веб-вразливостей.

Для досягнення поставленої мети були визначені наступні завдання. По-перше, проаналізувати існуючі методи обробки стиснутих даних на стороні

сервера та визначити їхні слабкі місця. По-друге, розробити алгоритми безпечного завантаження файлів з валідацією MIME-типів та перевіркою реального вмісту архіву за допомогою класу ZipArchive. По-третє, імплементувати комплексний захист системи, що включає впровадження CSRF-токенів[2] для протидії підробці міжсайтових запитів, встановлення жорстких лімітів на кількість файлів та обсяг оперативної пам'яті як захист від Zip-бомб[2], а також реалізацію блокування потенційно небезпечних розширень файлів. Крім того, передбачено розробку алгоритму створення кастомних архівів із вибраних користувачем директорій з використанням рекурсивного ітератора. Останнім завданням є створення сучасного, адаптивного графічного інтерфейсу користувача в стилі мінімалізму, без використання важких зовнішніх бібліотек, об'єднавши логіку та інтерфейс в одному портативному файлі.

Практичне значення отриманих результатів та сфери використання проєкту полягають у тому, що розроблений програмний продукт має високу практичну цінність і може бути використаний у низці сценаріїв. Зокрема, він є ефективним інструментом для швидкого розгортання веб-проєктів шляхом миттєвого завантаження та розпакування дистрибутивів систем управління контентом або фреймворків на хостинг. Також модуль оптимізує процес резервного копіювання даних, дозволяючи вибірково створювати архіви з критично важливими файлами сайту для їх подальшого безпечного завантаження. Зрештою, розробка значно спрощує адміністрування серверів без панелі керування, забезпечуючи зручне управління файловою структурою в середовищах, де немає встановлених cPanel, Plesk чи доступу через SSH, вимагаючи лише наявності стандартного веб-сервера із PHP.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Дослідження предметної області

У сучасний період глобальної цифровізації обсяги інформації, що генеруються, передаються та обробляються в комп’ютерних мережах, зростають у геометричній прогресії. Це створює суттєве навантаження на канали зв’язку та дискові підсистеми серверів. Для оптимізації процесів зберігання та транспортування даних широко застосовуються технології стиснення інформації. Відповідно, програмні засоби для пакування та розпакування даних (архіватори та розархіватори) стали невід’ємною складовою інфраструктури будь-якої сучасної інформаційної системи, зокрема у сфері веб-розробки та адміністрування серверів.

Основні поняття та термінологія

Для глибокого розуміння предметної області необхідно визначити базові терміни:

Визначення 1.1. Стиснення даних (Data compression) [1] - це процедура перекодування даних з метою зменшення їхнього обсягу. Цей процес базується на усуненні статистичної надмірності інформації.

Визначення 1.2. Файловий архів[1] - це спеціальний файл, який містить у собі один або декілька інших файлів (разом із їхніми метаданими: іменами, структурою каталогів, правами доступу, мітками часу), які зазвичай зберігаються у стиснутому вигляді.

Визначення 1.3. Архіватор[1] - комп’ютерна програма, яка здійснює об’єднання файлів в єдиний архів або серію архівів для зручності перенесення чи зберігання.

Визначення 1.4. Розархіватор (Unzipper) [1] - програмний модуль або самостійний додаток, основним завданням якого є виконання зворотного

процесу: читання архіву, декомпресія даних та відновлення початкової файлової структури на цільовому носії.

Залежно від цілей застосовуються два основних методи стиснення: із втратами та без втрат. У контексті файлових архівів (текстових документів, програмного коду, баз даних) використовується виключно стиснення без втрат (lossless compression). Воно гарантує, що розпакований файл буде побітово ідентичним оригіналу до моменту стиснення.

1.1.1 Опис процесу діяльності формату ZIP та алгоритми компресії

Серед великої кількості форматів архівації (RAR, TAR.GZ, 7z) стандартом де-факто для кросплатформної взаємодії став формат ZIP. Він був розроблений у 1989 році Філом Кацом і сьогодні підтримується на рівні операційних систем без необхідності встановлення стороннього програмного забезпечення.

В основі формату ZIP лежить алгоритм DEFLATE, який є комбінацією двох математичних підходів:

Алгоритм LZ77: шукає послідовності байтів, що повторюються, і замінює їх посиланнями на попередні входження.

Кодування Гаффмана (Huffman coding): використовує змінну довжину кодів для представлення символів - ті символи, що зустрічаються найчастіше, отримують найкоротші коди.

Особливістю формату ZIP є те, що кожен файл в архіві стискається окремо. Це дозволяє розархіваторам видобувати конкретні файли без необхідності розпаковування всього архіву цілком (читання так званого «Центрального каталогу» - Central Directory, який знаходиться в кінці файлу)[3].

Необхідність розархіваторів у веб-середовищі

Особливого значення розархіватори набувають у сфері веб-програмування та адміністрування серверів. Розгортання сучасних веб-додатків, фреймворків чи систем управління контентом (CMS) часто

передбачає перенесення десятків тисяч дрібних файлів (скриптів, стилів, зображень).

Передача такої кількості файлів через стандартний протокол FTP (File Transfer Protocol) є вкрай неефективною. На передачу кожного окремого файлу витрачається час на встановлення з'єднання, авторизацію та підтвердження отримання пакетів. Як наслідок, копіювання проекту може тривати годинами.

Рішенням цієї проблеми є використання архівації на стороні клієнта: всі файли проекту стискаються в один єдиний ZIP-архів. Передача одного великого файлу на сервер займає лічені секунди. Однак після завантаження виникає необхідність розпакувати цей архів безпосередньо в операційній системі сервера. Саме для цього і розробляються серверні веб-розархіватори - спеціальні скрипти (наприклад, написані мовою PHP), які приймають архів і дають команду серверу розпакувати його локально (рис. 1.1).

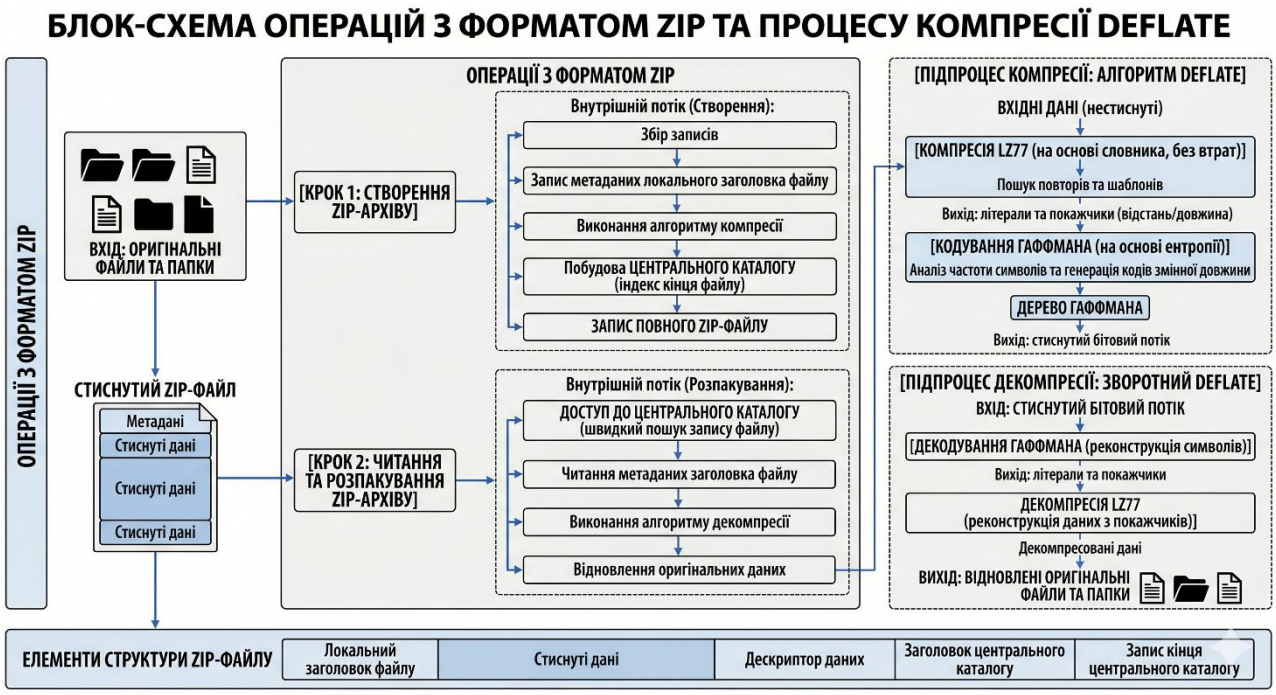

Рисунок 1.1 – Опис процесу діяльності формату ZIP та алгоритми компресії

1.1.2 Опис функціональності моделі серверного розархіватора

Процес програмної декомпресії веб-додатком включає наступні етапи:

1. Ініціалізація та валідація: скрипт перевіряє факт завантаження файлу, його розмір та відповідність MIME-типу (захист від завантаження виконуваних файлів під виглядом архівів).

2. Аналіз структури: за допомогою внутрішніх бібліотек (в PHP це розширення ZipArchive) програма зчитує Central Directory архіву для отримання списку файлів та шляхів.

3. Виділення пам'яті та декомпресія: потокове читання стиснутих байтів та застосування зворотного алгоритму DEFLATE.

4. Реконструкція файлової системи: програма послідовно створює директорії на сервері та записує розпаковані байти у відповідні файли, відновлюючи первісну архітектуру проекту.

Проблематика та виклики

Попри очевидні переваги, серверна обробка архівів є потенційним вектором кібератак. При дослідженні предметної області слід виділити основні загрози, з якими стикаються розархіватори:

Вразливість Zip Slip: якщо розархіватор сліпо довіряє шляхам всередині архіву (які можуть містити символи ../), розпаковані файли можуть бути записані поза цільовою директорією, перезаписуючи критичні системні файли.

Завантаження веб-шелів: зловмисники можуть помістити в архів шкідливі скрипти (наприклад, .php, .phtml), які після розпакування дадуть їм повний контроль над сервером.

Атаки типу «Zip-бомба» (Zip Bomb): спеціально створені архіви, які мають невеликий розмір (кілька кілобайтів), але при розпакуванні займають петабайти пам'яті, що призводить до відмови в обслуговуванні (DDoS) сервера через вичерпання ресурсів оперативної пам'яті або дискового простору.

З огляду на вищезазначене, розробка сучасного розархіватора вимагає не просто реалізації базового функціоналу розпакування, але й впровадження жорстких політик безпеки, контролю шляхів, обмежень розмірів та фільтрації розширень файлів [4].

1.2 Аналіз існуючих програмних рішень та їх можливостей

Ринок програмного забезпечення для роботи зі стиснутими даними розвивається вже понад три десятиліття. На сьогоднішній день існує велика кількість розархіваторів, які різняться за архітектурою, цільовою платформою та функціональними можливостями. Для комплексного дослідження доцільно класифікувати існуючі рішення на три основні категорії: десктопні (настільні) додатки, консольні серверні утиліти та веб-орієнтовані файлові менеджери.

Десктопні архіватори (клієнтські додатки)

До цієї категорії належать класичні програми, що встановлюються на персональні комп'ютери користувачів під управлінням операційних систем Windows, macOS або настільних дистрибутивів Linux. Найяскравішими представниками є 7-Zip (рис. 1.2), WinRAR, WinZip та PeaZip.

Переваги та можливості: Десктопні рішення пропонують максимальну продуктивність завдяки прямому доступу до апаратних ресурсів комп’ютера (багатопотоковість процесора, оперативна пам'ять). Вони підтримують десятки форматів архівів, мають зручний графічний інтерфейс (GUI), функції шифрування за стандартом AES-256 та можливість створення саморозпакувальних (SFX) архівів.

Недоліки в контексті веб-розробки: Основною проблемою цих програм є неможливість їх використання для прямого розпакування даних на віддаленому сервері. Якщо веб-майстер хоче завантажити проект на хостинг, використання десктопного розархіватора змушує його розпакувати файли локально, а потім передавати кожен файл окремо через FTP, що є вкрай повільним і неефективним процесом.

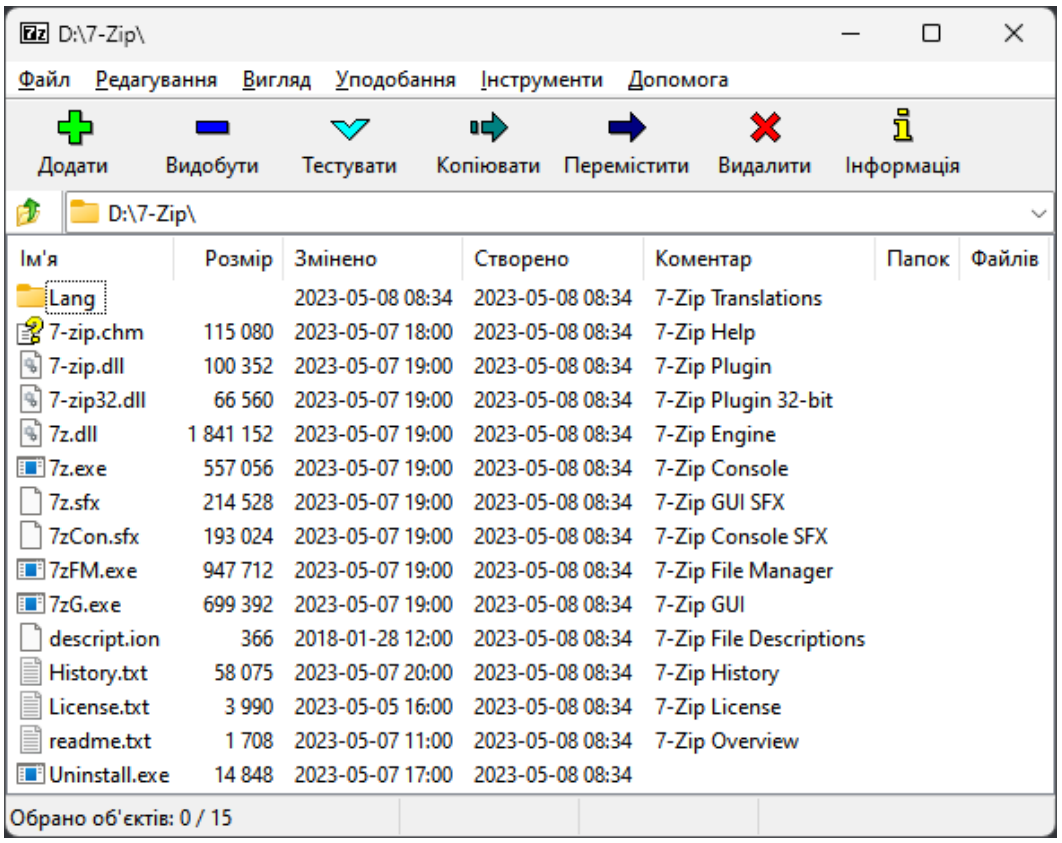

Рисунок 1.2 – Приклад архіватора від 7ZIP

Консольні серверні утиліти (CLI)

Для роботи безпосередньо на серверах сімейства UNIX/Linux стандартом є консольні утиліти, такі як unzip, tar, gzip та bzip2. Взаємодія з ними відбувається через інтерфейс командного рядка (Command Line Interface) за допомогою SSH-з'єднання (рис. 1.3).

Переваги та можливості: Це найшвидші та найбільш надійні інструменти для роботи з архівами на сервері. Вони споживають мінімум оперативної пам'яті, здатні обробляти архіви розміром у сотні гігабайтів і легко інтегруються в bash-скрипти для автоматизації процесів резервного копіювання (бекапів) або розгортання CI/CD (Continuous Integration / Continuous Deployment).

Обмеження: Головним недоліком є високий поріг входження. Користувач повинен володіти знаннями термінальних команд Linux та синтаксису параметрів утиліт. Крім того, на багатьох тарифах віртуального хостингу (Shared Hosting) провайдери з міркувань безпеки взагалі не надають клієнтам

доступу до SSH. Відповідно, використання цих потужних інструментів стає неможливим для пересічного власника сайту.

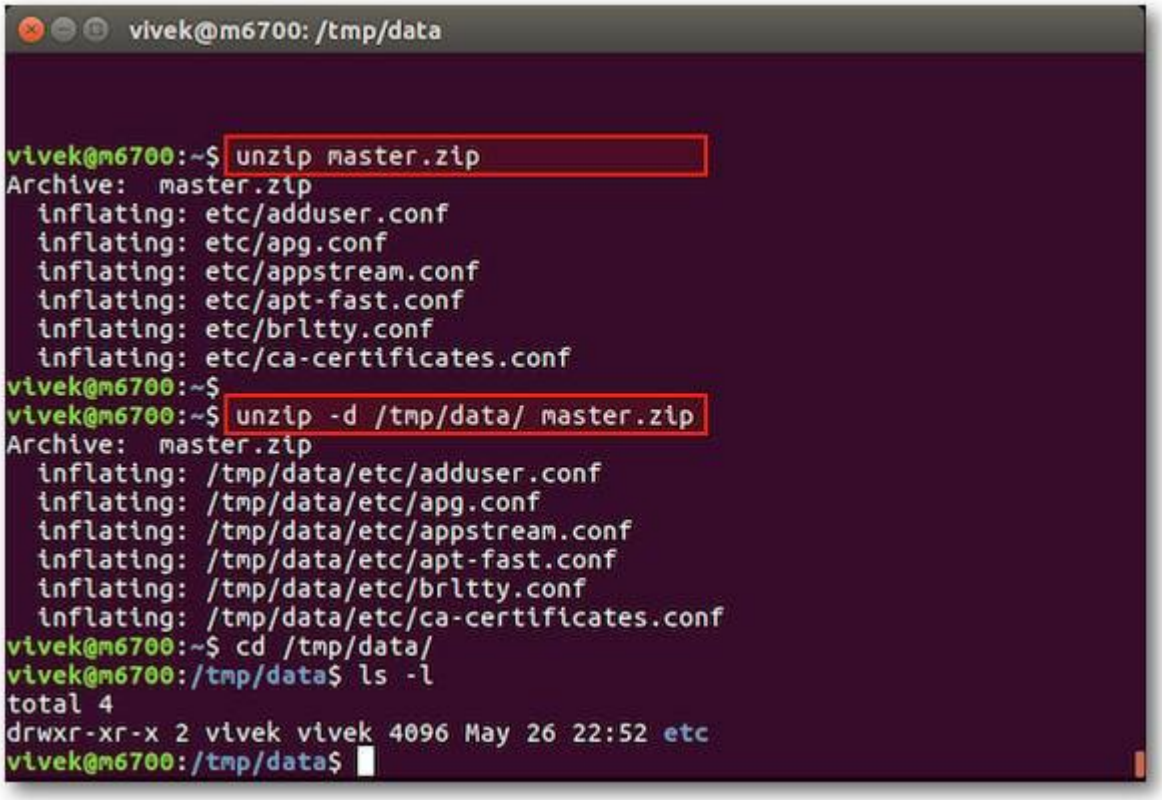

Рисунок 1.3 – Консольній архіватор на ОС Linux

Серверні веб-панелі та файлові менеджери

Для вирішення проблеми доступу до файлів без SSH були створені веб-панелі керування хостингом (cPanel, ISPmanager, Plesk) та інтегровані в них файлові менеджери, які працюють безпосередньо через браузер (за протоколами HTTP/HTTPS).

Переваги: Вони надають графічний інтерфейс, подібний до звичайного Провідника Windows. Користувач може завантажити архів на сервер і розпакувати його натисканням однієї кнопки у браузері. Ці системи також дозволяють редагувати код, керувати правами доступу (CHMOD) та адмініструвати бази даних.

Недоліки: Панелі керування є важким, комерційним програмним забезпеченням. Вони вимагають придбання дорогих ліцензій, споживають

багато системних ресурсів і потребують root-доступу для встановлення. Якщо розробник працює з "чистим" сервером (VPS/VDS) або безкоштовним хостингом без панелі керування, розпакування файлів знову стає проблемою.

Автономні PHP-скрипти для розпакування (Аналоги)

Окрему нішу займають спеціалізовані веб-скрипти, написані мовою PHP, які завантажуються на сервер і працюють як міні-додатки. Прикладами можуть слугувати TinyFileManager (рис. 1.4), eXtplorer або старі відкриті скрипти типу «PHP Unzipper». Саме вони є найближчими аналогами розроблюваного програмного продукту.

Переваги: Вони портативні, не вимагають встановлення (достатньо завантажити файли скрипта на сервер) та працюють на будь-якому веб-сервері, де налаштована підтримка PHP. Вони вирішують проблему відсутності SSH та панелей керування.

Критичні недоліки існуючих рішень: Більшість доступних у відкритому доступі PHP-розархіваторів мають ряд суттєвих вразливостей або архітектурних проблем:

Застарілий код та надмірність (Bloatware): Такі системи як eXtplorer містять тисячі рядків коду, складну структуру папок та модулів, що ускладнює їх аудит та використання лише заради однієї функції – розпакування.

Проблеми безпеки: Багато легковагових скриптів не мають вбудованого захисту від CSRF-атак (Cross-Site Request Forgery), не перевіряють MIME-типи файлів перед завантаженням та не фільтрують розширення файлів у самому архіві. Це дозволяє зловмисникам обходити захист веб-сервера.

Відсутність захисту від перевантажень: Більшість аналогів намагаються вивантажити архів у пам'ять цілком або не мають обмежень на кількість файлів. У разі завантаження «Zip-бомби» такий скрипт призводить до падіння веб-сервера через помилку Fatal error: Allowed memory size exhausted.

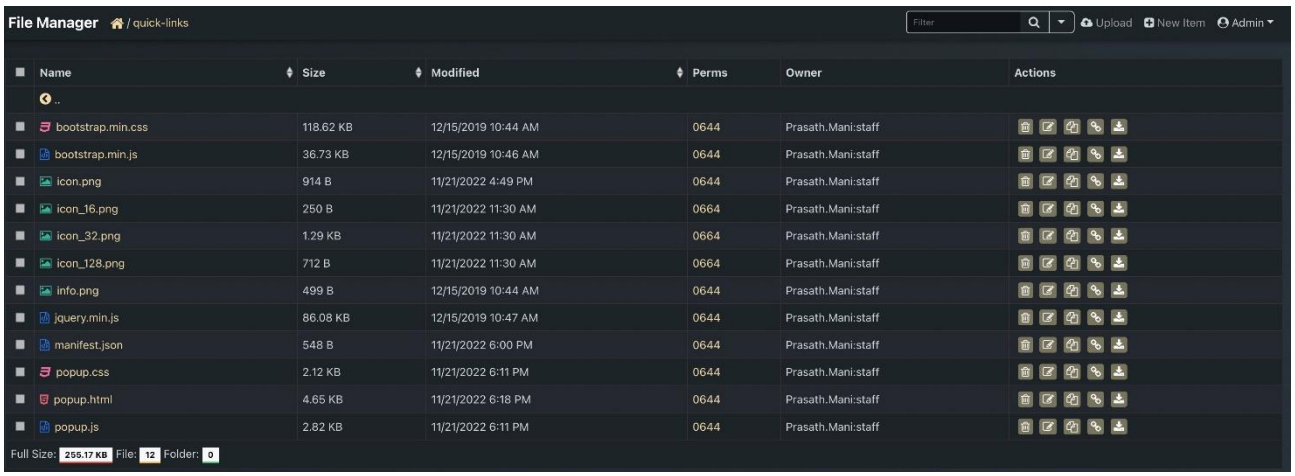

Рисунок 1.4 - Автономний PHP архіватор «TinyFileManager UI»

1.3 Порівняння існуючих рішень

Розглянувши аналоги створенного разархіватора, складемо таблицю з наочним порівнянням аналогічних додатків за характеристиками переглянувши таблицю1.1:

Таблиця 1.1 – Порівняння існуючих рішень

Критерії оцінювання	Десктопні додатки	Консольні утіліти	Панелі керування хостингом	Існуючі автономні PHP скрипти	Розроблений Веб-модуль
1	2	3	4	5	6
Середовище виконання	Локальний ПК	Віддалений сервер	Віддалений сервер	Віддалений сервер	Віддалений сервер
Вимоги до розгортання	Встановлений в ОС	SSH-доступ, права Root	Придбання ліцензії, права Root	Підтримка PHP	Підтримка PHP
Тип інтерфейсу	Графічний	Командний рядок	Веб-графічнний	Веб-графічнний	Веб-графічнний
Автономніть\ портативність	Низька – прив’язаність до ПК	Висока – вбудована в ОС	Низька – важка система	Середня – часто мають багато папок	Абсолютна – один файл
Захист від веб вразливостей	Не застосовується	Не застосовується	Високий	Низбкий\ відсутній	Високий

Продовження таблиці 1.1

1	2	3	4	5	6
Захист від ZIP-вірусів та ліміти	Залежить від ПК	Обмеження на рівні ОС	Присутній	Відсутній	Жорстко регламентований у коді
Валідація MIME-типів файлів	Не застосовується	Не вимагається	Присутня	Часто ігнорується	Комплексна
Поріг входження для користувача	Низький	Дуже високйи	Низький	Низький	Низький

Як видно з наведеної таблиці, десктопні додатки та консольні утиліти, попри свою потужність, не відповідають вимогам швидкого та зручного віддаленого веб-адміністрування через відсутність прямого доступу до середовища (у випадку десктопу) або надто високий поріг входження (у випадку CLI).

Панелі керування надають відмінний функціонал та безпеку, проте є занадто громіздкими та недоступними на бюджетних або нестандартних серверах. Існуючі автономні PHP-аналоги вирішують проблему портативності, але їхнім критичним недоліком є нехтування правилами безпеки: відсутність CSRF-токенів, ігнорування перевірки реального типу завантаженого файлу (MIME) та схильність до вичерпання ресурсів пам'яті сервера (відсутність захисту від переповнення).

На відміну від аналогів, розроблений веб-модуль займає унікальну нішу. Він поєднує абсолютну портативність (розгортання відбувається шляхом копіювання лише одного файлу unzipper.php) та інтуїтивно зрозумілий графічний інтерфейс із потужними серверними механізмами безпеки, що притаманні лише комерційним панелям керування. Впроваджені ліміти розміру (MAX_UNZIP_BYTES), захист від виконання шкідливих скриптів та валідація шляхів розпакування роблять його оптимальним інструментом для сучасних веб-розробників.

1.4 Постановка задачі

На основі проведеного аналізу предметної області та виявлених недоліків існуючих програмних рішень, виникає об’єктивна необхідність у створенні власного, вузькоспеціалізованого програмного продукту.

Головною метою даної кваліфікаційної роботи є проектування, розробка та тестування серверного веб-модуля для автоматизації процесів завантаження, архівації, розпакування та управління файловими даними у форматі ZIP з інтегрованою багаторівневою системою захисту від серверних вразливостей.

Для досягнення поставленої мети необхідно чітко формалізувати вимоги до майбутнього програмного забезпечення, визначити технологічний стек, окреслити необхідний рівень компетенцій розробника та сформувати перелік конкретних завдань.

Вимоги до функціональних можливостей (Functional Requirements)

Програмний продукт повинен забезпечувати виконання наступних базових функцій в межах єдиного інтерфейсу:

Завантаження архівів на сервер: Модуль має надавати користувачеві можливість вибору локального ZIP-архіву та його завантаження на сервер через метод POST (форма multipart/form-data). Повинна бути реалізована перевірка успішності завантаження та автоматичне перейменування файлів у разі збігу імен (уникнення перезапису існуючих архівів).

Визначення 1.5. Декомпресія (розпакування) даних - це забезпечення читання вмісту завантажених архівів та їх безпечне розпакування у спеціально відведену директорію (наприклад, unzipped/). Система повинна вміти створювати необхідне дерево каталогів відповідно до внутрішньої структури архіву.

Визначення 1.6. Створення нових архівів (компресія)[5] – це модуль має дозволяти користувачеві вибирати окремі файли або цілі директорії на сервері

за допомогою чекбоксів (графічного інтерфейсу) та генерувати з них новий ZIP-архів. Цей процес вимагає реалізації рекурсивного обходу папок.

Файловий менеджмент: Реалізація базових операцій управління об'єктами, зокрема безпечного видалення завантажених архівів (функція unlink) та рекурсивного видалення розпакованих директорій з усім їхнім вмістом. [5]

Завантаження файлів клієнту: Модуль повинен формувати коректні HTTP-заголовки (Content-Type, Content-Disposition) для можливості прямого завантаження (скачування) створених або наявних на сервері архівів на локальний комп'ютер користувача.

Вимоги до безпеки та нефункціональних характеристик (Non-Functional Requirements)

Оскільки робота з файловою системою сервера є потенційним вектором для кібератак, ключовий акцент у постановці задачі робиться на безпеку. Програма повинна задовольняти наступним критеріям:

Абсолютна портативність (Single-file architecture): Весь бекенд-код (PHP), фронтенд-логіка (JavaScript) та стилі (CSS) повинні бути інкапсульовані в один єдиний файл (наприклад, unzipper.php). Проект не повинен залежати від зовнішніх баз даних (MySQL тощо) або пакетних менеджерів (Composer). [6]

Протидія міжсайтовій підробці запитів (CSRF): Кожна дія, що змінює стан сервера (завантаження, видалення, розпакування), повинна захищатися унікальним криптографічним токеном (CSRF-token), згенерованим через random_bytes() і прив'язаним до сесії користувача.

Багаторівнева валідація завантажень: Забороняється довіряти виключно розширенню файлу. Необхідно реалізувати перевірку MIME-типу завантаженого файлу за допомогою розширень finfo (File Information) та mime_content_type.

Захист від шкідливого коду (Web Shells): Під час розпакування скрипт повинен перевіряти розширення кожного файлу всередині архіву. Файли з

небезпечними розширеннями (.php, .phtml, .exe, .sh, .phar) мають блокуватися, а процес розпакування - перериватися. [7]

Захист від атак «Zip Slip»: Програмно повинна перевірятися цілісність шляхів. Кінцевий шлях розпакованого файлу (після обробки символів ../) обов'язково має знаходитися в межах дозволеної базової директорії.

Захист від «Zip-бомб» (Resource Exhaustion): Встановлення жорстких констант-лімітів на максимальну кількість файлів в архіві (наприклад, 50 000) та максимальний загальний обсяг розпакованих даних (наприклад, 3 ГБ), щоб уникнути відмови в обслуговуванні (DDoS) та вичерпання оперативної пам'яті.

Ергономічність (UI/UX): Графічний інтерфейс має бути адаптивним (коректно відображатися на мобільних пристроях), сучасним (з використанням принципів мінімалізму, відсутності зайвих елементів) та надавати користувачеві чіткий зворотний зв'язок (повідомлення про успішні дії або зрозумілі описи помилок). [8]

Перелік завдань розробки (Етапи виконання роботи)

Відповідно до сформульованих вимог, виконання кваліфікаційної роботи розбивається на наступні логічні завдання:

Спроектувати загальну архітектуру скрипта, визначити константи безпеки (ліміти розмірів, списки заборонених розширень, дозволені MIME-типи).

Розробити блок ініціалізації та управління сесіями, реалізувати механізм генерації та перевірки CSRF-токенів.

Написати допоміжні функції для роботи з файловою системою (наприклад, безпечне рекурсивне видалення директорій, форматування розміру файлів у зручний для читання вигляд).

Імплементувати алгоритм безпечного розпакування (unzipFileSafe), який включатиме перевірку внутрішніх шляхів (захист від Zip Slip) та контроль розмірів даних «на льоту».

Імплементувати алгоритм створення кастомних архівів (createCustomZip), що дозволить динамічно пакувати вибрані користувачем директорії та файли.

Розробити алгоритм валідації та безпечного завантаження файлів на сервер із детальним перехопленням та обробкою кодів помилок масиву $_FILES.

Створити фронтенд-частину (HTML-каркас та CSS-стилі) з дотриманням сучасних UI/UX стандартів, реалізувати форму для завантаження та списки для відображення існуючих файлів.

Провести комплексне тестування розробленого модуля на предмет коректної роботи алгоритмів декомпресії та стійкості до штучно змодельованих атак (спроби завантажити PHP-скрипт під виглядом ZIP-архіву, спроби передачі архівів з некоректними шляхами).

Очікуваний результат: У результаті виконання перелічених завдань має бути створений готовий до використання, повністю автономний веб-модуль мовою PHP, який вирішує проблему швидкого та безпечного управління файловими архівами на сервері без необхідності використання SSH-доступу або важких панелей керування.

Висновки до розділу

У першому розділі було досліджено теоретичні основи роботи зі стиснутими даними у веб-середовищі. З'ясовано, що використання ZIP-архівів є критично важливим для швидкого завантаження проєктів на сервер. Порівняльний аналіз існуючих рішень (десктопних програм, консольних утиліт та веб-скриптів) показав, що більшість із них або занадто складні у розгортанні, або не мають належного захисту від специфічних серверних вразливостей (Zip-вірусів, завантаження шкідливого коду). Це повністю обґрунтовує необхідність створення власного програмного продукту -

безпечного та портативного однофайлового PHP-розархіватора, процес розробки якого буде детально описано у наступному розділі.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

Цільова аудиторія застосунку – це веб-розробники, вони є основна категорія користувачів продукту. Розробникам часто доводиться розгортати готові веб-сайти, системи управління контентом (WordPress, OpenCart) або PHP-фреймворки (Laravel, Symfony) на клієнтських хостингах. Передача тисяч файлів CMS через FTP є неефективною, тому розробник може завантажити один ZIP-архів і миттєво розпакувати його розробленим модулем. Також інструмент є ідеальним для швидкого перенесення сайтів між різними серверами.

2.1 Функціональні вимоги

Для успішного виконання поставлених завдань розроблений програмний модуль повинен забезпечувати виконання наступних функцій:

а) Завантаження архівів: Можливість безпечного завантаження локальних ZIP-файлів на віддалений сервер через графічний веб-інтерфейс.

б) Відображення стану файлової системи: Автоматичне сканування робочої директорії та виведення списку наявних ZIP-архівів і раніше розпакованих папок.

в) Безпечне розпакування (Декомпресія): Видобування вмісту архівів у спеціально відведену директорію (наприклад, unzipped/) зі збереженням оригінального дерева каталогів.

г) Вибіркова архівація (Компресія): Можливість генерації нових ZIP-архівів безпосередньо на сервері з обраних користувачем файлів та папок.

д) Завантаження клієнту (Download): Забезпечення прямого скачування будь-якого наявного архіву з сервера на локальний пристрій користувача.

є) Управління архівами: Функція безповоротного видалення обраних ZIP-файлів із дискового простору сервера.

ж) Управління розпакованими даними: Можливість рекурсивного видалення цілих папок із видобутими файлами для звільнення місця.

з) Зворотний зв'язок (Обробка статусів): Виведення інформативних повідомлень для користувача про успішне завершення операції або причини виникнення помилок (наприклад, перевищення ліміту розміру).

і) Фільтрація вмісту: Автоматичне розпізнавання та блокування розпакування заборонених або виконуваних файлів

2.2 Нефункціональні вимоги

Для забезпечення надійності, безпеки та зручності експлуатації розроблюваного веб-модуля встановлено такі нефункціональні вимоги:

а) Безпека та стійкість до атак: Програма повинна мати вбудований захист від міжсайтової підробки запитів (CSRF), виходу за межі цільової директорії при розпакуванні (Zip Slip) та атак, спрямованих на вичерпання ресурсів сервера (Zip-бомби).

б) Абсолютна портативність (Автономність): Увесь програмний код (PHP-логіка, HTML-розмітка та CSS-стилі) має бути інкапсульований в один єдиний файл. Програма не повинна залежати від сторонніх бібліотек (наприклад, Composer) чи баз даних (MySQL).

в) Ергономічність (UI/UX): Графічний інтерфейс має бути адаптивним (коректно працювати на мобільних пристроях і ПК), мінімалістичним та забезпечувати швидкий відгук системи без зайвих перезавантажень сторінки.

г) Сумісність та вимоги до середовища: Модуль повинен коректно функціонувати на будь-якому стандартному веб-сервері з підтримкою мови PHP (версії 7.4 або новішої) та активованим базовим розширенням ZipArchive.

д) Надійність та обробка винятків (Fault Tolerance): Програма не повинна завершувати роботу фатальною помилкою (Fatal Error) при спробі завантажити пошкоджений архів або файл, що перевищує серверні ліміти

(upload_max_filesize). Замість цього система має перехоплювати винятки та виводити користувачеві зрозумілі текстові повідомлення.

2.3 Технічний опис архітектури

Розроблений програмний продукт функціонує в рамках класичної клієнт-серверної архітектури веб-додатків. Відмінною рисою даного рішення є його монолітна однофайлова структура (Single-File Application) [10]. Увесь життєвий цикл обробки запиту - від конфігурації сервера та бізнес-логіки до генерації графічного інтерфейсу – інкапсульований в єдиному файлі unzipper.php. Такий архітектурний підхід обрано навмисно для забезпечення максимальної портативності модуля та незалежності від сторонніх фреймворків чи систем управління пакетами (наприклад, Composer).

З технічної точки зору, програма використовує змішану парадигму програмування:

Процедурний підхід: застосовується для ініціалізації сесій, маршрутизації HTTP-запитів (GET, POST) [5] та виконання базових операцій (генерація CSRF-токенів, перевірка розмірів).

Об'єктно-орієнтований підхід (ООП): використовується під час взаємодії з ядром мови PHP, зокрема виклик методів вбудованого класу ZipArchive для компресії/декомпресії та використання класів Стандартної бібліотеки PHP (SPL), таких як RecursiveDirectoryIterator та RecursiveIteratorIterator, для обходу файлових дерев дерева.

Маршрутизація в системі реалізована за допомогою патерну Front Controller, де єдиною точкою входу є сам файл, а дія, яку потрібно виконати, визначається через передачу параметра action у тілі POST-запиту або GET-параметрах. Зв'язок між клієнтом (браузером) і сервером здійснюється за протоколом HTTP, а стан користувача (для захисту від міжсайтової підробки запитів) зберігається за допомогою механізму PHP-сесій (session_start())[11].

2.3.1 Компоненти застосунку

Попри те, що програма фізично складається з одного файлу, логічно її архітектуру розділено на п'ять взаємопов'язаних компонентів (модулів), які виконуються послідовно зверху вниз:

Компонент конфігурації та безпеки (Security & Config Module):

Відповідає за сувору типізацію (declare(strict_types=1)), відображення помилок та старт сесії.

Містить генератор та валідатор CSRF-токенів (csrf_check).

Ініціалізує глобальні константи безпеки: максимальні ліміти файлів (MAX_UNZIP_ENTRIES, MAX_UNZIP_BYTES) [11], дозволені MIME-типи та масив заборонених розширень (BLOCKED_EXTS).

Модуль файлових операцій (File Operations Module):

Включає допоміжні функції для роботи з файловою системою: рекурсивне видалення директорій (rrmdir), форматування розміру файлів у зручний для читання вигляд (human_size), а також функцію глибокої перевірки MIME-типу файлу (is_real_zip), яка використовує розширення finfo.

Ядро обробки архівів (Core Archive Engine): Ключовий бекенд-компонент, який відповідає за бізнес-логіку програми:

Функція unzipFileSafe: реалізує безпечну декомпресію. Включає перевірку на спроби обходу директорії (Zip Slip) та контроль розміру "на льоту".

Функція createCustomZip: забезпечує рекурсивну компресію обраних користувачем файлів і папок у новий архів.

Маршрутизатор запитів (Request Router / Controller):

Блок логіки, що перехоплює суперглобальні масиви $_POST та $_GET[11].

Обробляє події завантаження нового архіву на сервер (з детальною валідацією кодів помилок $_FILES), видалення існуючих файлів, команди розпакування та скачування. Результатом роботи цього компонента є генерація текстового повідомлення ($message) про статус виконання операції.

Інтерфейс користувача (Presentation Component / View):

Фронтенд-частина, реалізована мовою HTML5 із вбудованими CSS3-стилями.

Використовує CSS-змінні (Custom Properties) для задання кольорової схеми, CSS Grid та Flexbox для адаптивної верстки.

Включає мінімалістичний JavaScript-модуль (Vanilla JS) для клієнтської взаємодії: управління станами чекбоксів (функції "Вибрати все", "Зняти виділення") та динамічного підрахунку обраних елементів.

2.4 Проектування програми

На основі сформованих вимог була спроектована загальна діаграма варіантів використання (рис. 2.1). Для зручності деякі варіанти використання були об’єднанні між собою.

Актор системи – користувач, який може використовувати усі функції програми

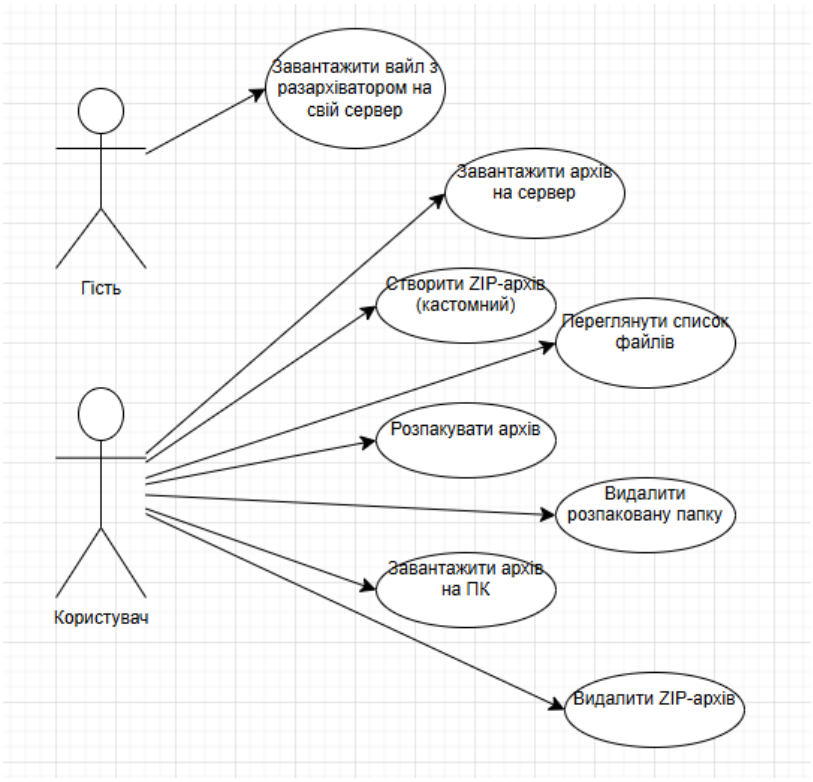

Рисунок 2.1 – Загальна діаграма варіантів використання

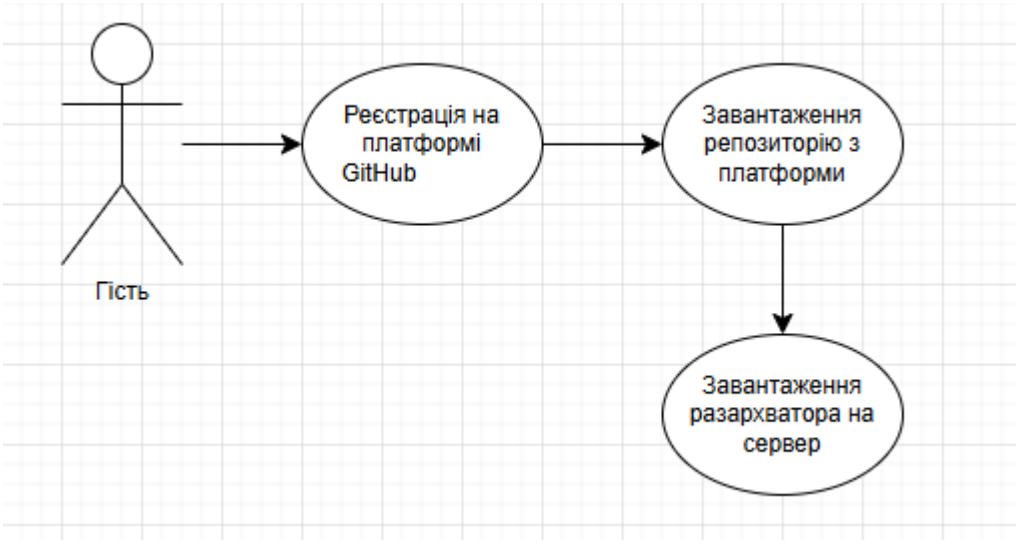

Рисунок 2.2 – Завантаження разархіватора для нового користувача

```
// Создание ZIP из выбранных элементов
function createCustomZip(string $rootDir, array $items): string {
    if (empty($items)) return "❌ Не выбраны файлы/папки.";

    $zipName = "custom_" . date("Y-m-d_H-i-s") . ".zip";
    $zipPath = $rootDir . DIRECTORY_SEPARATOR . $zipName;

    $zip = new ZipArchive();
    if ($zip->open($zipPath, ZipArchive::CREATE) !== TRUE) {
        return "❌ Ошибка создания архива.";
    }

    $scriptName = basename(__FILE__);
    $unzippedPath = $rootDir . DIRECTORY_SEPARATOR . "unzipped" . DIRECTORY_SEPARATOR;

    foreach ($items as $rel) {
        $rel = ltrim($rel, '/\\');
        $fullPath = realpath($rootDir . DIRECTORY_SEPARATOR . $rel);
        if ($fullPath === false) continue;

        // Не пакуем сам скрипт и содержимое папки распаковки
        if (basename($fullPath) === $scriptName) continue;
        if (strpos($fullPath, $unzippedPath) === 0) continue;

        if (is_dir($fullPath)) {
            $iterator = new RecursiveIteratorIterator(
                new RecursiveDirectoryIterator($fullPath, FilesystemIterator::SKIP_DOTS),
                RecursiveIteratorIterator::SELF_FIRST
            );
            foreach ($iterator as $node) {
                $nodePath = $node->getPathname();
                if (strpos($nodePath, $unzippedPath) === 0) continue;
                if (basename($nodePath) === $scriptName) continue;

                $local = substr($nodePath, strlen($rootDir) + 1);
                if ($node->isDir()) {
                    $zip->addEmptyDir($local);
                } else {
                    $zip->addFile($nodePath, $local);
                }
            }
        } else {
            $local = substr($fullPath, strlen($rootDir) + 1);
            $zip->addFile($fullPath, $local);
        }
    }

    $zip->close();
    return "✅ Архив создан: <a href=\"" . htmlspecialchars($zipName) . "\" download>" . htmlspecialchars($zipName) . "</a>";
```

Рисунок 2.3 – код створення архіву на сервері

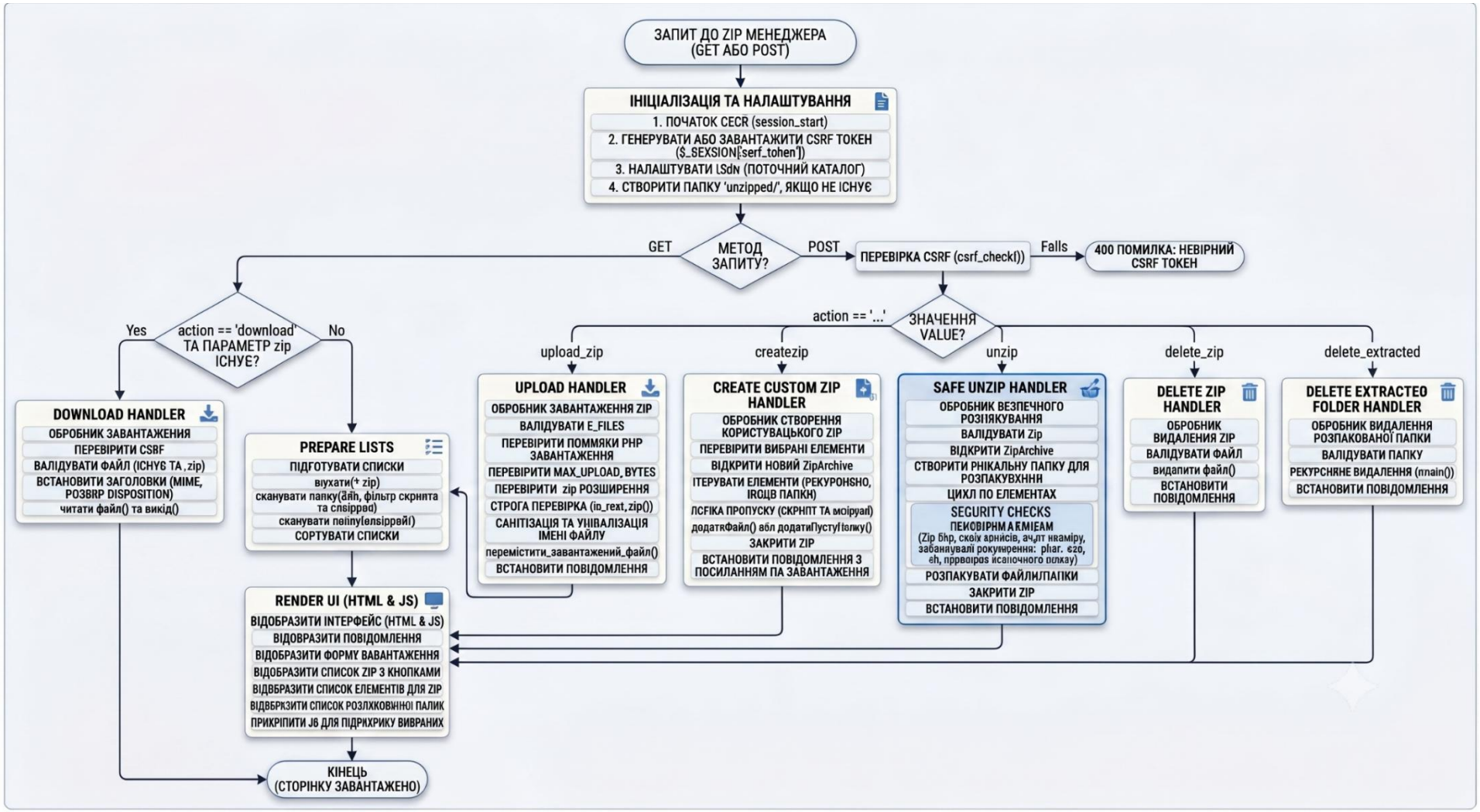

Рисунок 2.4 - Блок схема роботи разархіватора

Висновки до розділу

У процесі дослідження було складено вичерпний перелік функціональних та нефункціональних вимог. Визначено, що модуль повинен не лише забезпечувати базові операції з архівами (завантаження, створення, розпакування), але й відповідати суворим критеріям безпеки (захист від CSRF, Zip Slip, Zip-бомб) та портативності. Обґрунтовано використання монолітної архітектури у вигляді єдиного файлу (Single-File Application). Описано технічну структуру скрипта та виділено п'ять ключових логічних компонентів (модуль безпеки, файлових операцій, ядро обробки, маршрутизатор та інтерфейс користувача), що забезпечують його автономну роботу. Крім цього було складено діаграму варіантів використання, побудовано розгорнуту блок-схему роботи коду, яка візуалізує внутрішню логіку маршрутизації (GET/POST запитів), процеси валідації даних та послідовність виклику внутрішніх підпрограм.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Вибір операційної системи

Розробка програмного забезпечення мовою PHP має суттєву перевагу у вигляді апаратної та програмної незалежності (кросплатформеності). Оскільки PHP є інтерпретованою мовою, розроблений веб-модуль не компілюється під конкретну архітектуру процесора чи ядро операційної системи, а виконується безпосередньо середовищем інтерпретатора на стороні веб-сервера. Це означає, що продукт є універсальним і концептуально розрахованим на роботу в будь-якій операційній системі, яка здатна підтримувати роботу веб-серверів, таких як Apache, Nginx або IIS [12].

Основним цільовим середовищем для практичної експлуатації даного веб-розархіватора є серверні операційні системи сімейства UNIX та GNU/Linux, зокрема такі популярні дистрибутиви, як Ubuntu, Debian, CentOS або AlmaLinux. Саме ці операційні системи є стандартом де-факто у сфері веб-хостингу, і саме на них розміщується переважна більшість сучасних веб-сайтів. Водночас процес розробки та локального тестування скрипта може вільно здійснюватися на персональних комп'ютерах під управлінням Windows або macOS із використанням локальних серверних рішень (наприклад, XAMPP, MAMP чи Docker-контейнерів) [13], що підтверджує високий рівень портативності модуля.

Незважаючи на загальну кросплатформеність мови, цільова операційна система все ж таки має певний вплив на специфіку розробки, особливо у проектах, які тісно взаємодіють із дисковою підсистемою. Найбільш суттєвою відмінністю між Windows та UNIX-подібними системами є форматування шляхів до файлів: Windows використовує зворотний слеш як роздільник каталогів, тоді як Linux - прямий. Для того, щоб цей фактор не впливав на працездатність продукту, у програмному коді веб-модуля повністю виключено

використання жорстко заданих слешів; замість них застосовується вбудована константа DIRECTORY_SEPARATOR [14], яка автоматично адаптується під поточну операційну систему.

Ще одним важливим аспектом впливу операційної системи на розробку є механізм розподілу прав доступу. Серверні системи Linux суворо контролюють права на читання, запис та виконання (CHMOD). Тому під час розробки алгоритмів декомпресії та створення нових папок через функцію mkdir було враховано необхідність явної передачі параметрів прав доступу (наприклад, 0777 або 0755), щоб гарантувати коректну роботу модуля у суворих серверних середовищах і запобігти помилкам типу "Permission denied". Таким чином, розроблений модуль враховує специфіку різних операційних систем, залишаючись при цьому універсальним інструментом.

3.2 Вибір середовища

Для ефективного написання програмного коду, його налагодження та тестування критично важливим етапом є правильний вибір інтегрованого середовища розробки (IDE [1] Integrated Development Environment) або сучасного редактора коду. Професійні інструменти надають розробнику такі життєво необхідні функції, як інтелектуальне підсвічування синтаксису, автодоповнення функцій, перевірку на наявність синтаксичних помилок у реальному часі та зручну роботу з файловою системою.

Оскільки об'єктом розробки у даній роботі є однофайловий веб-модуль мовою PHP, який містить змішаний код (серверну логіку на PHP [3], фронтенд на HTML5 [1], CSS3-стилі та клієнтські скрипти на JavaScript), до середовища висуваються специфічні вимоги. Воно повинно швидко працювати, коректно розпізнавати переходи між різними мовами в межах одного документа і не бути перевантаженим зайвим функціоналом, який зазвичай потрібен лише для важких фреймворків чи масштабних архітектур.

На сучасному ринку програмного забезпечення для веб-розробки виділяються три основні рішення: повноцінна комерційна система PhpStorm, популярний редактор Visual Studio Code та мінімалістичний Sublime Text. Їхнє детальне порівняння за ключовими для проекту критеріями наведено у Таблиці 3.1.

Таблиця 3.1 – Порівняння різних серидовищ розробки

Критерій оцінювання	PhpStorm	Visual Studio Code (VS Code)	Sublime Text
Тип програмного забезпечення	Повноцінна IDE	Редактор коду з розширеннями	Легковаговий редактор тексту
Ліцензія та вартість	Комерційна (Платна)	Open Source (Безкоштовна)	Умовно-безкоштовна (Shareware)
Споживання системних ресурсів	Високе (на базі Java)	Середнє (на базі Electron)	Дуже низьке (написано на C++)
Підтримка PHP «з коробки»	Максимальна (глибокий аналіз коду)	Базова (розширюється плагінами)	Лише підсвічування синтаксису
Робота зі змішаним кодом (PHP + HTML + JS)	Відмінна	Відмінна	Добра
Наявність вбудованого термінала	Присутній	Присутній	Відсутній (потрібні плагіни)
Доцільність для однофайлового проєкту	Занизька (надмірний функціонал)	Найвища (оптимальний баланс)	Середня (не вистачає інструментів аналізу)

Здійснивши аналіз наведених варіантів, можна зробити висновок, що оптимальним вибором для проектування та написання даної кваліфікаційної роботи є редактор Visual Studio Code. Використання важкої інтегрованої системи рівня PhpStorm для створення портативного скрипта є нераціональним кроком, який призведе до невиправданого споживання оперативної пам'яті комп'ютера та перевантаження інтерфейсу зайвими інструментами. З іншого боку, базові редактори на кшталт Sublime Text надто прості і вимагають ручного встановлення багатьох доповнень для отримання підказок по функціям PHP. Visual Studio Code виступає ідеальною золотою

серединою: цей продукт розповсюджується абсолютно безкоштовно, має високу швидкість відгуку, бездоганно розуміє перемикання між PHP та HTML-розміткою в межах одного файлу, а також містить зручний вбудований термінал. Це дозволяє миттєво запускати локальний PHP-сервер і тестувати процес розпакування архівів, не залишаючи вікна розробки.

3.3 Вибір мови програмування

Основним інструментом для реалізації серверної логіки розроблюваного веб-модуля було обрано скриптову мову програмування PHP (Hypertext Preprocessor). Вибір саме цієї мови зумовлений її специфікою: PHP була від самого початку створена для веб-розробки і на сьогоднішній день залишається беззаперечним лідером у сегменті серверних технологій, на яких працює переважна більшість сучасних веб-сайтів та систем управління контентом.

Розглядаючи ключові характеристики PHP, які роблять її ідеальною для даного проєкту, варто насамперед відзначити її абсолютну портативність та простоту розгортання. На відміну від мов на кшталт Node.js, Python чи Java, які вимагають налаштування віртуального середовища, встановлення залежностей через пакетні менеджери або запуску окремих демонів, PHP-скрипт готовий до виконання одразу після копіювання у робочу директорію веб-сервера. Це повністю відповідає головній нефункціональній вимозі нашого проєкту - створенню монолітного однофайлового застосунку, який можна легко переносити між серверами.

Крім того, мова PHP має надзвичайно потужну стандартну бібліотеку, яка містить усі необхідні інструменти «з коробки». Для роботи зі стиснутими даними використовується вбудоване розширення ZipArchive, що дозволяє виконувати компресію та декомпресію на рівні ядра мови з максимальною продуктивністю, не викликаючи зовнішніх консольних команд. Робота з файловою системою, рекурсивний обхід каталогів та безпечна перевірка

MIME-типів забезпечуються базовими функціями та ітераторами стандартної бібліотеки (SPL).

Важливою перевагою сучасних версій PHP (починаючи з 7.x та 8.x) є підтримка суворої типізації, яка активується директивою declare(strict_types=1). Це дозволяє писати надійний, передбачуваний та безпечний код, мінімізуючи ризики виникнення вразливостей, пов'язаних із неявним перетворенням типів даних під час обробки HTTP-запитів. Разом із вбудованими механізмами управління сесіями, це забезпечує міцний фундамент для побудови безпечної архітектури.

Таким чином, PHP є не просто одним із можливих варіантів, а найбільш об'єктивним та раціональним вибором для створення серверного файлового менеджера та разархіватора. Вона поєднує в собі низький поріг розгортання, глибоку інтеграцію з веб-серверами, багатий набір вбудованих функцій для роботи з файлами та сучасні механізми забезпечення безпеки коду.

3.4 Архітектура програми

Архітектура розробленого програмного модуля побудована за принципом монолітного однофайлового застосунку, де вся серверна логіка, налаштування безпеки та генерація користувацького інтерфейсу інкапсульовані у єдиному файлі. Такий підхід гарантує максимальну портативність системи, дозволяючи розгорнути її простим копіюванням на цільовий сервер. Логічно код поділено на послідовні блоки, кожен з яких виконує строго визначену функцію та передає управління наступному, формуючи надійний конвеєр обробки даних.

Першочерговим етапом роботи скрипта є блок ініціалізації та управління сесіями. Використання конструкції строгої типізації declare(strict_types=1) на самому початку файлу є критично важливим архітектурним рішенням, яке змушує інтерпретатор PHP жорстко контролювати типи даних, що передаються у функції, мінімізуючи ризик прихованих помилок під час

виконання алгоритмів. Одразу після старту сесії реалізується механізм захисту від підробки міжсайтових запитів (CSRF). Програма генерує унікальний криптографічний токен за допомогою вбудованої функції random_bytes(32). Цей токен записується у сесію користувача та згодом вбудовується у вигляді прихованого поля в усі HTML-форми. Будь-яка дія, що змінює стан сервера, проходить через допоміжну функцію csrf_check, яка порівнює отриманий токен із сесійним, надійно відхиляючи несанкціоновані запити.

Наступним важливим компонентом є блок визначення глобальних обмежень, який виступає першим ешелоном захисту від вичерпання ресурсів сервера. У коді оголошено константи MAX_UNZIP_ENTRIES [16] та MAX_UNZIP_BYTES [16], які обмежують кількість файлів у архіві та їхній сумарний розпакований розмір на рівні трьох гігабайтів. Це архітектурне рішення є прямою протидією атакам типу «Zip-бомба», коли зловмисник завантажує малий архів, що при декомпресії здатен повністю заповнити дисковий простір або оперативну пам'ять хостингу. Разом із масивом заборонених розширень BLOCKED_EXTS [16], цей блок створює безпечне середовище для подальших маніпуляцій.

Ядром системи валідації виступає функція is_real_zip, яка реалізує глибоку перевірку завантажених файлів. Архітектурно неправильно довіряти лише розширенню файлу, оскільки зловмисники можуть перейменувати шкідливий PHP-скрипт на архів. Тому в коді використовується розширення finfo_open для зчитування магічних байтів файлу та визначення його реального MIME-типу на апаратному рівні. Лише після того, як система переконається, що це дійсно бінарний ZIP-файл, і клас ZipArchive зможе його успішно відкрити, файл допускається до подальшої обробки.

Найскладнішим та найвідповідальнішим компонентом програми є ядро декомпресії, реалізоване у функції unzipFileSafe. Її архітектура побудована на принципі поступового читання архіву у циклі for, а не сліпого розпакування всього вмісту. Під час ітерацій скрипт аналізує шлях кожного запису за допомогою регулярного виразу для виявлення спроб виходу за межі дозволеної

директорії, таких як використання символів ../. Це надійно захищає сервер від вразливості Zip Slip. Крім того, на кожній ітерації відбувається акумуляція розміру розпакованих файлів, і якщо ліміт перевищується, процес негайно переривається, а створені тимчасові файли видаляються допоміжною функцією рекурсивного очищення rrmdir.

Завершує бекенд-архітектуру маршрутизатор запитів, який аналізує суперглобальні масиви. Він розділяє логіку на обробку POST-запитів, де конструкція розгалуження керує викликом відповідних функцій завантаження чи видалення, та перехоплення GET-запитів для ініціалізації прямого скачування файлів. Після виконання бізнес-логіки управління передається фронтенд-компоненту на базі HTML та CSS, який сканує робочі директорії і формує адаптивний візуальний інтерфейс користувача з відображенням відповідних статусних повідомлень.

3.5 Реалізація програми

Етап програмної реалізації передбачає перенесення розробленої логічної архітектури у площину програмного коду та створення візуального представлення (View) веб-додатку. Оскільки головною вимогою до системи є її однофайлова монолітна структура, генерація інтерфейсу користувача відбувається безпосередньо в основному файлі unzipper.php, одразу після виконання блоку серверної логіки (маршрутизатора).

Реалізація візуальної частини (Фронтенд)

Код графічного інтерфейсу розташований у нижній частині файлу програми. Він базується на семантичній розмітці HTML5. Для стилізації елементів не використовуються сторонні важкі фреймворки (наприклад, Bootstrap), замість цього застосовано нативні каскадні таблиці стилів (CSS3), інкапсульовані у тег <style>.

Інтерфейс розроблено з урахуванням сучасних тенденцій UI/UX дизайну (у стилістиці Apple-мінімалізму). Це реалізовано завдяки використанню CSS-

змінних (Custom Properties) для визначення глобальної кольорової палітри (наприклад, --bg: #f5f5f7 для фону, --card-bg: #ffffff для карток), м'яких тіней (box-shadow) та плавних заокруглень (border-radius: 12px). Для забезпечення адаптивності на різних розмірах екранів (Responsive Design) використовується технологія CSS Flexbox.

Інтерактивність інтерфейсу на стороні клієнта (у браузері) реалізована за допомогою легкої анонімної самовикличної функції (IIFE) на нативному JavaScript (Vanilla JS), яка розміщена перед закриваючим тегом </body>. Цей скрипт взаємодіє з DOM-деревом документа, виконуючи такі задачі:

- Збирає масив усіх активних чекбоксів на сторінці (document.querySelectorAll('.chk')).
- Відстежує події зміни їхнього стану (change).
- Динамічно оновлює лічильник обраних файлів для архівації (selCount.textContent).
- Керує подіями для кнопок масового виділення («Вибрати все» та «Зняти виділення»).

Демонстрація роботи програмного модуля

Практична взаємодія користувача з системою відбувається через інтуїтивно зрозумілі графічні панелі. При першому зверненні до скрипта через GET-запит (відкриття сторінки), програма сканує серверну директорію та формує головний екран. Верхня частина містить панель для завантаження нових архівів (рис 3.1).

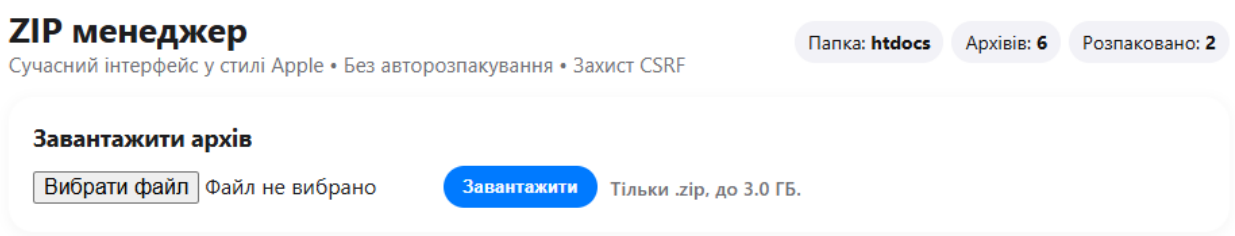

Рисунок 3.1 – Головний екран: панель завантаження архівів

Після успішного завантаження файлу на сервер (з проходженням валідації MIME-типу та перевіркою CSRF-токена), програма оновлює сторінку та відображає завантажений архів у блоці «Доступні ZIP-архіви». Кожен запис у цьому списку супроводжується інформацією про розмір файлу (сформованою серверною функцією human_size) та кнопками управління: «Розпакувати», «Скачати» та «Видалити» (Рис 3.2).

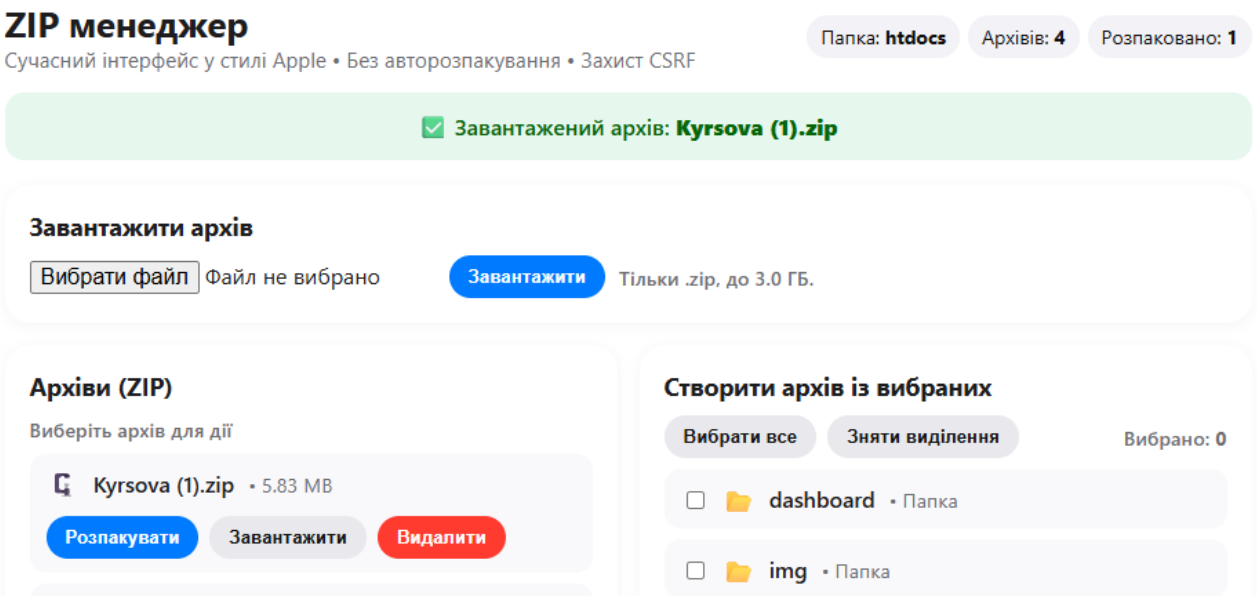

Рисунок 3.2 – Завантаження архіву на сервер

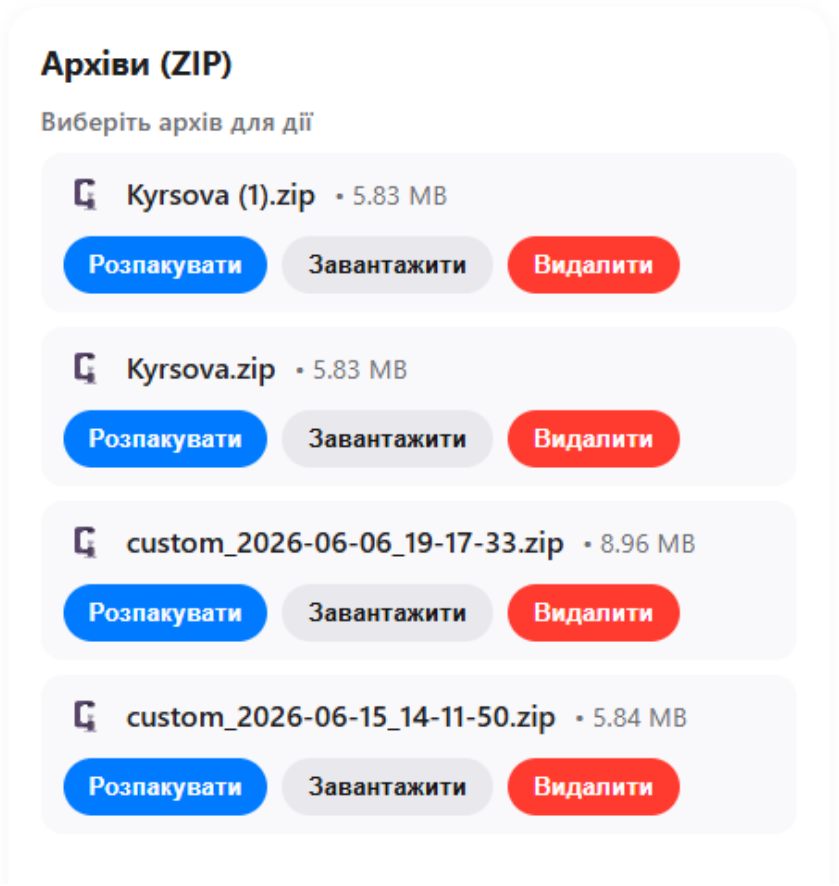

Рисунок 3.3 – Відображення списку доступних архівів та панелі керування ними

При натисканні на кнопку «Розпакувати», скрипт ініціює POST-запит з параметром action="unzip". Програма здійснює безпечну декомпресію даних у директорію unzipped/. У разі успішного завершення операції, у верхній частині екрана з'являється зелене інформаційне повідомлення (Рис 3.4), а нова папка з'являється у нижньому блоці інтерфейсу - «Розпаковані папки». Звідси адміністратор може швидко видалити усю папку з її вмістом для очищення дискового простору (Рис. 3.5).

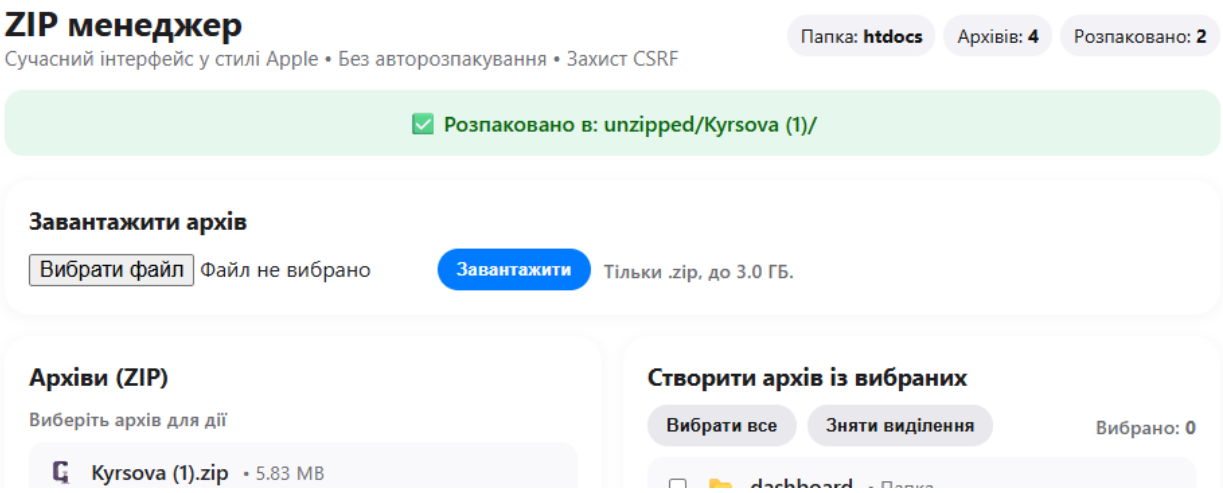

Рисунок 3.4 – Результат розпакування

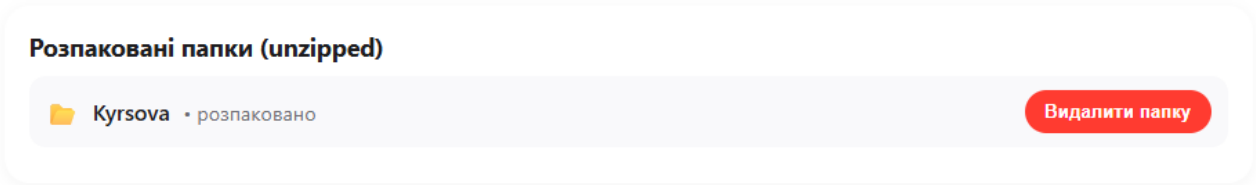

Рисунок 3.5 – Відображення розпакованих архівів

Окремою функціональною можливістю є блок створення користувацьких архівів (компресія). Завдяки написаному клієнтському JavaScript-коду, користувач може відмітити чекбоксами потрібні файли, а система в режимі реального часу підрахує кількість обраних елементів. Після підтвердження система згенерує новий архів під назвою custom_archive.zip. (Рис. 3.6).

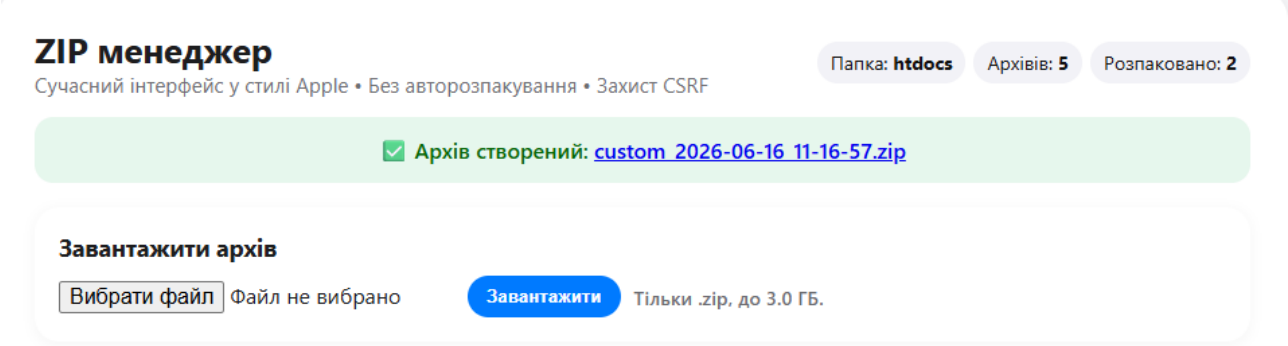

Рисунок 3.6 – Створення архіву з файлів на сервері

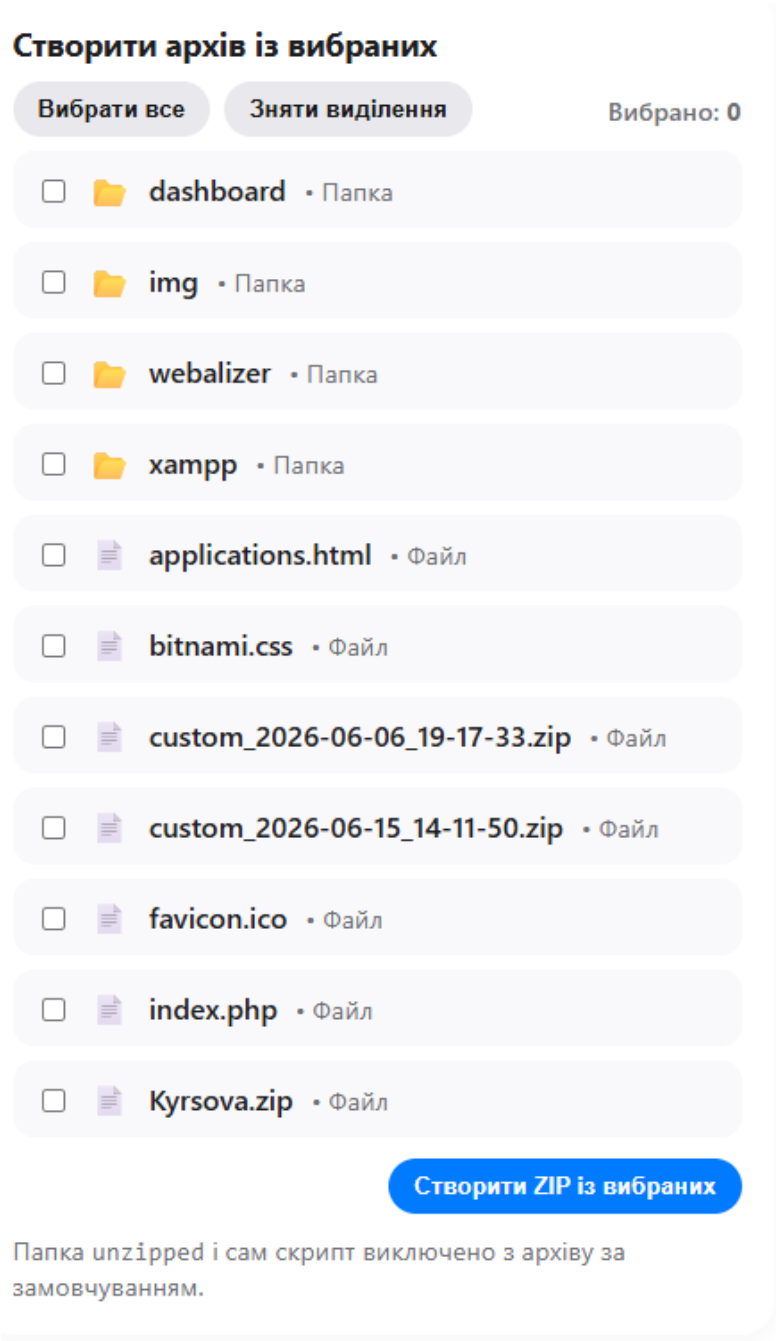

Рисунок 3.7 – Відображення директорій з видобутими даними

Таким чином, розроблений графічний інтерфейс повністю відповідає вимогам ергономіки, приховує складну серверну логіку від кінцевого користувача та надає зручний інструмент для швидкого адміністрування файлових структур віддаленого сервера.

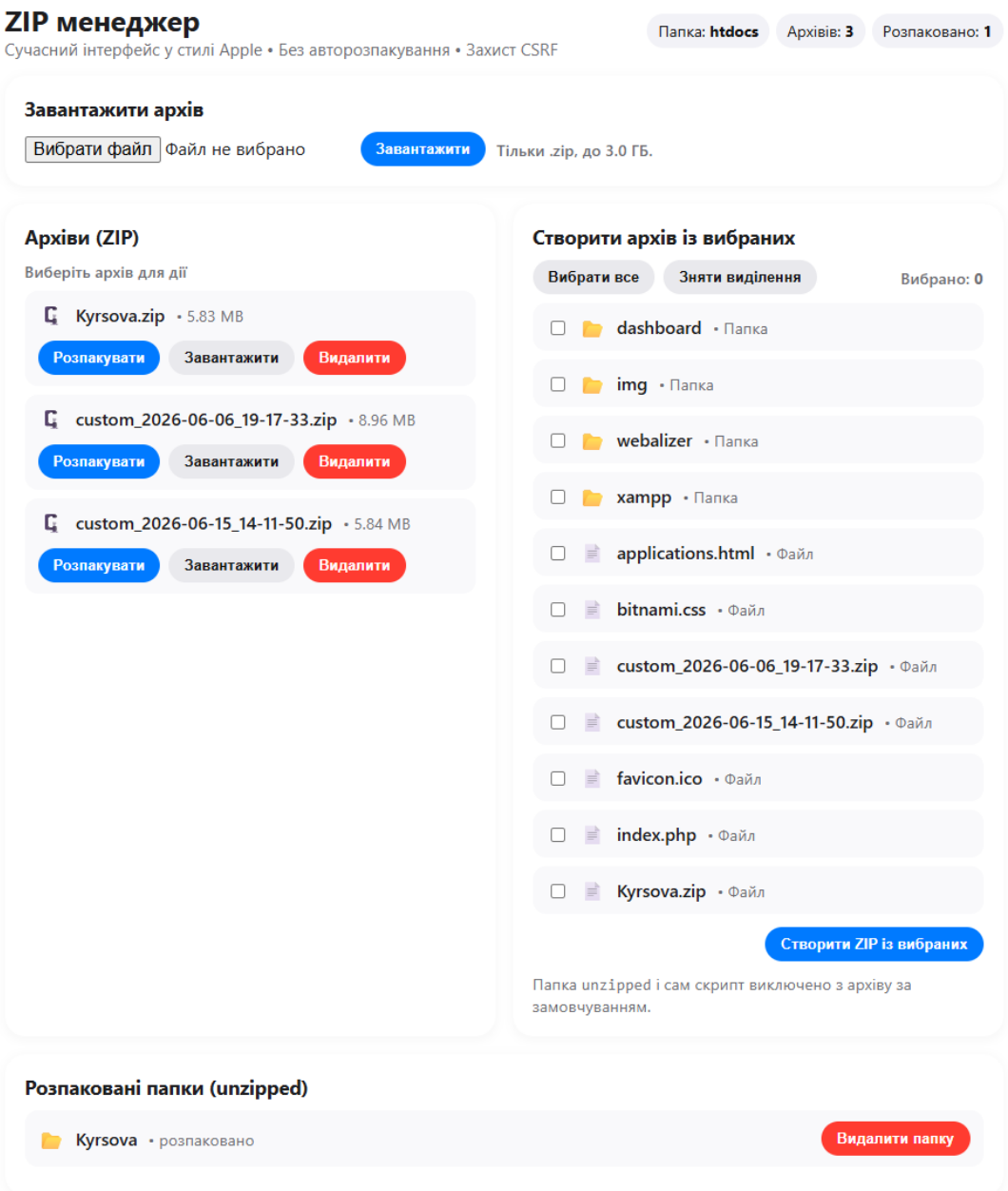

Рисунок 3.8 – Повне відображення архіватора

3.6 Функціональне тестування застосунку

Функціональне тестування є невід'ємним етапом життєвого циклу розробки програмного забезпечення, який дозволяє перевірити відповідність реальної поведінки системи закладеним вимогам. Для перевірки розробленого веб-модуля було обрано метод тестування «чорного ящика», за якого увага зосереджується на вхідних та вихідних даних без прямого втручання у внутрішній код під час виконання. Процес тестування охоплює всі ключові

сценарії взаємодії користувача з інтерфейсом, а також реакцію серверної частини на нестандартні або потенційно небезпечні дії.

Результати проведених перевірок зведені у загальну таблицю тестування, яка демонструє реакцію системи на різні типи запитів та коректність роботи захисних механізмів.

Таблиця 3.2 - Результати функціонального тестування веб-модуля

Назва тесту	Очікуваний результат	Отриманий результат	Статус перевірки
Завантаження валідного ZIP-архіву	Файл завантажується на сервер, проходить перевірку і з'являється у списку доступних архівів	Файл успішно збережено на диску та відображено в інтерфейсі	Пройдено
Завантаження файлу з підробленим розширенням	Відмова у завантаженні через невідповідність реального MIME-типу формату ZIP	Система видає помилку та блокує збереження небезпечного файлу	Пройдено
Розпакування стандатного архіву	Створення нової директорії та видобування туди всіх файлів із збереженням структури	Файли розпаковано коректно, нова папка з'явилася внизу сторінки	Пройдено
Створення архіву з обраних файлів	Генерація нового ZIP-файлу, що містить лише відмічені користувачем елементи	Архів custom_archive.zip успішно створено та додано до списку	Пройдено
Видалення архіву або папки	Фізичне видалення обраного об'єкта з диска сервера та зникнення його з інтерфейсу	Об'єкт безповоротно видалено, інтерфейс оновлено без помилок	Пройдено
Виконання запиту без CSRF-токена	Негайна зупинка виконання скрипта із поверненням HTTP-коду помилки 400	Запит відхилено системою безпеки, операцію заблоковано	Пройдено

Проведене тестування підтверджує повну працездатність програмного продукту та його готовність до практичного використання. Система

безпомилково обробляє стандартні операції з файлами, генерує коректні візуальні відгуки для користувача та успішно відбиває спроби маніпуляцій із типами файлів чи запитами. Всі закладені на етапі проектування архітектурні рішення довели свою ефективність на практиці.

Висновки до розділу

У третьому розділі було здійснено ключовий перехід від теоретичного проектування системи до її безпосередньої практичної реалізації. Фундаментом для успішного написання коду став обґрунтований вибір технологічного стека та середовища розробки. Використання мови програмування PHP гарантувало продукту абсолютну портативність, незалежність від операційної системи сервера та надало потужні вбудовані інструменти для маніпуляцій з архівами. Своєю чергою, редактор Visual Studio Code дозволив ефективно організувати роботу зі складним змішаним кодом в межах одного документа.

Втілена в життя монолітна архітектура повністю виправдала себе, дозволивши елегантно поєднати сувору серверну логіку з адаптивним графічним інтерфейсом. Особливу увагу під час реалізації було приділено механізмам безпеки, що дозволило створити не просто зручний файловий менеджер, а захищений інструмент, стійкий до підробки запитів та спроб вичерпання серверних ресурсів. Завершальним акордом розробки стало комплексне функціональне тестування, яке на практиці довело безпомилкову реакцію скрипта як на стандартні дії користувача, так і на змодельовані загрози. У результаті було отримано повністю готовий, стабільний та автономний програмний продукт, який ідеально підходить для швидкого та безпечного адміністрування віддалених веб-серверів.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

4.1 Організаційно правові основи забезпечення безпеки праці

У системі сучасного виробництва та управління підприємством охорона праці посідає чільне місце, оскільки вона безпосередньо пов'язана із захистом найвищої цінності - життя та здоров'я людини.[22] Забезпечення безпечних і нешкідливих умов праці є не лише гуманітарним обов'язком роботодавця, а й важливою передумовою стабільного економічного розвитку будь-якої організації чи підприємства. Організаційно-правові основи охорони праці становлять фундамент, на якому будується вся внутрішня система безпеки, визначаючи зобов’язання, права та відповідальність усіх учасників виробничого процесу.

Роль охорони праці у забезпеченні безпеки та захисту здоров'я працівників

Соціально-економічне значення охорони праці полягає в мінімізації ризиків виробничого травматизму та професійних захворювань, що, у свою чергу, безпосередньо впливає на продуктивність праці та стабільність виробничих процесів. Створення безпечного робочого середовища дозволяє досягти кількох критично важливих цілей:

Гуманітарна та соціальна функція: збереження фізичного та психічного здоров'я працівників, запобігання інвалідності та передчасній смертності внаслідок дії небезпечних чи шкідливих виробничих факторів.

Економічна функція: зниження матеріальних витрат підприємства на виплату компенсацій за роботу у шкідливих умовах, лікування та реабілітацію постраждалих, а також уникнення збитків від простою обладнання та ліквідації наслідків аварій.

Технологічна функція: інтеграція вимог безпеки безпосередньо у технологічні процеси, що вимагає впровадження більш сучасного, надійного та автоматизованого обладнання.

Ефективне функціонування системи охорони праці на підприємстві можливе лише за умови системного підходу, який поєднує правові норми, організаційні заходи, технічні засоби захисту та належний рівень культури безпеки серед персоналу.

Законодавча та нормативно-правова база України у сфері охорони праці

Правове регулювання відносин у сфері охорони праці в Україні базується на чіткій ієрархічній системі нормативних актів. Вона гармонізована з міжнародними стандартами, зокрема з конвенціями Міжнародної організації праці (МОП) та директивами Європейського Союзу. [22] [23]

Основним спеціальним законом, який детально регламентує цю сферу, є Закон України «Про охорону праці» [23][24]. Він поширюється на всі підприємства, установи та організації незалежно від форми власності та видів їхньої діяльності. Цей документ визначає, що державна політика у сфері охорони праці базується на принципі пріоритету життя і здоров'я працівників відносно результатів виробничої діяльності підприємства.

Законодавство чітко розмежовує сфери відповідальності та встановлює, що фінансування заходів з охорони праці здійснюється виключно за рахунок роботодавця. Крім того, нормативна база передбачає обов'язкове створення служби охорони праці на підприємствах із кількістю працюючих 50 і більше осіб, а також регулярне проведення навчання та перевірки знань посадових осіб і робітників з питань безпеки.

Організаційні заходи забезпечення безпеки на рівні підприємства

Правові норми реалізуються через конкретні організаційні механізми всередині підприємства. Сюди належить побудова Системи управління охороною праці (СУОП), яка є невіддільною частиною загальної системи менеджменту організації.

Організаційна робота включає такі ключові елементи:

Проведення інструктажів та навчання: кожен працівник під час прийняття на роботу та періодично в процесі діяльності має проходити відповідні інструктажі (вступний, первинний, повторний, позаплановий, цільовий). Для робіт із підвищеною небезпекою передбачено обов'язкове щорічне спеціальне навчання.

Атестація робочих місць за умовами праці: комплексний аналіз робочого місця для виявлення шкідливих та небезпечних факторів, визначення права працівників на пільги та компенсації, а також розробки заходів щодо покращення умов праці.

Забезпечення засобами захисту: безкоштовне надання працівникам спецодягу, спецвзуття та інших засобів індивідуального (ЗІЗ) та колективного захисту відповідно до встановлених норм.

Медичні огляди: організація обов'язкових попередніх (під час прийняття на роботу) та періодичних медичних оглядів для категорій працівників, зайнятих на важких роботах або роботах із шкідливими чи небезпечними умовами.

Важливий аспект: Сучасний підхід до організації безпеки праці зміщує акцент із ліквідації наслідків нещасних випадків на оцінку та управління ризиками. Це означає, що ідентифікація небезпек має відбуватися ще до того, як вони призведуть до травмування чи погіршення стану здоров'я персоналу.[24]

4.2 Характеристика об’єкта та виявлення потенційних небезпек

4.2.1 Характеристика об’єкта проектування та робочого місця користувача

Об'єктом дослідження в даному розділі є робоче середовище кінцевого користувача розроблюваного вебзастосунка – системи управління файловими архівами. Кінцевим користувачем виступає оператор системи (архівіст, діловод

або ІТ-спеціаліст підприємства), який здійснює введення, систематизацію, пошук та обробку великих масивів даних і цифрових документів.

Робоче місце оператора розташоване у типовому офісному приміщенні підприємства-користувача. Приміщення характеризується наявністю природного (через вікна) та штучного (стельові світильники) освітлення, обладнане системами централізованого опалення та кондиціонування повітря або припливно-витяжною вентиляцією.

Безпосереднє робоче місце користувача складається з робочого столу, ергономічного крісла з можливістю регулювання висоти сидіння та кута нахилу спинки, а також комплекту обчислювальної техніки. Основним інструментом праці є персональний комп'ютер (або ноутбук) з одним чи двома моніторами для зручної роботи з електронними архівами, клавіатурою, маніпулятором типу «миша» та джерелом безперебійного живлення (ДБЖ). Також у зоні досяжності може знаходитися периферійне обладнання: принтери, сканери, маршрутизатори.

Характер роботи оператора вебзастосунка управління архівами відноситься до категорії розумової (інтелектуальної) праці, яка пов'язана з прийомом та переробкою великого обсягу інформації. Основні завдання користувача включають:

- моніторинг інтерфейсу вебзастосунку;
- завантаження та класифікацію файлів;
- перевірку цілісності даних;
- роботу з текстовою та графічною інформацією на екрані монітора.

Специфіка такої діяльності вимагає високого ступеня концентрації уваги, тривалого перебування у статичній сидячій позі та постійного зорового контакту з екраном монітора. Робота переважно виконується в однозмінному режимі (стандартний 8-годинний робочий день).

4.2.2 Аналіз та класифікація небезпечних і шкідливих виробничих факторів

Згідно з чинними санітарними нормами та міждержавним стандартом МОЗ [25], у процесі експлуатації розробленого вебзастосунка на робочому місці оператора можуть виникати різні потенційні небезпеки. Усі небезпечні та шкідливі виробничі фактори (НШВФ), що супроводжують даний вид діяльності, поділяються на чотири основні групи: фізичні, хімічні, біологічні та психофізіологічні.

1. Фізичні фактори: До цієї групи належать чинники матеріального середовища, які безпосередньо впливають на організм працівника:

Електромагнітне випромінювання та статична електрика: Основним джерелом є персональний комп'ютер, монітор та інша оргтехніка. Тривалий вплив електромагнітних полів може викликати розлади нервової та серцево-судинної систем.

Нераціональне освітлення: Недостатній рівень природного світла, неправильно підібране штучне освітлення, відсутність розсіювачів на лампах або наявність відблисків (прямого чи відбитого блискоутворення) на екрані монітора. Це призводить до швидкої втоми очей та зниження працездатності.

Порушення мікроклімату: Невідповідність температури, відносної вологості або швидкості руху повітря санітарним нормам. У літній період фактором ризику є протяги та переохолодження від кондиціонерів, у зимовий – пересушене повітря через роботу обігрівачів.

Підвищений рівень шуму: Генерується кулерами (вентиляторами) системних блоків комп'ютерів, серверами, принтерами, а також зовнішнім середовищем (вулиця, розмови колег). Тривалий монотонний шум знижує концентрацію уваги.

Небезпека ураження електричним струмом: Офісне приміщення насичене електричним обладнанням (розетки, подовжувачі, ПК, блоки живлення). Пошкодження ізоляції кабелів або несправність заземлення становить пряму загрозу життю.

2. Хімічні фактори: У звичайному офісному приміщенні хімічні небезпеки не є домінуючими, проте вони присутні. Основними джерелами є:

Виділення шкідливих речовин (фенол, формальдегід, озон) від пластикових корпусів нагрітої під час роботи оргтехніки, а також від порошку лазерних принтерів під час друку.

Наявність у повітрі робочої зони підвищеного вмісту побутового та паперового пилу, що може стати причиною алергічних реакцій або подразнення дихальних шляхів.

3. Біологічні фактори: Для оператора, який працює у колективі в замкненому просторі, біологічні фактори ризику включають:

Патогенні мікроорганізми (віруси, бактерії), які поширюються повітряно-крапельним шляхом у періоди сезонних епідемій (ГРВІ, грип, COVID-19).

Спори цвілевих грибів, що можуть накопичуватися у фільтрах систем кондиціонування при їх несвоєчасному технічному обслуговуванні, спричиняючи захворювання дихальної системи.

4. Психофізіологічні фактори: Для ІТ-спеціалістів та операторів баз даних ця група факторів є найбільш критичною:

Фізичні перевантаження (статичні): Гіподинамія (нестача рухової активності) та тривале підтримання вимушеної робочої пози призводять до захворювань опорно-рухового апарату (остеохондроз, сколіоз), застою крові в органах малого таза та нижніх кінцівках. Також характерним є перенапруження кистей рук (синдром зап'ястного каналу) через постійну роботу з мишею та клавіатурою.

Нервово-психічні перевантаження: Напруження зорового аналізатора через постійну необхідність вчитуватися в дрібний текст і аналізувати дані на екрані. Крім того, робота з важливими архівними даними передбачає високу відповідальність, що може супроводжуватися стресом, емоційним вигоранням та монотонністю праці.

4.2.3 Небезпеки загального характеру (пожежна та військова загроза)

Окрім факторів, пов'язаних безпосередньо з трудовим процесом, робоче місце оператора піддається небезпекам більш масштабного характеру:

Пожежна небезпека. Офісні приміщення відносяться до зон підвищеної пожежної небезпеки через велику кількість електронного обладнання та горючих матеріалів (паперові документи, дерев'яні меблі, пластик). Причиною пожежі може стати коротке замикання електропроводки, перевантаження електромережі, залишення техніки без нагляду або порушення правил пожежної безпеки працівниками. Наслідки виникнення пожежі є вкрай важкими: від термічних опіків та отруєння чадним газом до летальних випадків та знищення матеріальних цінностей (серверів з архівами).

Військова загроза. В умовах сучасних реалій та воєнного стану в Україні військова агресія є найбільш критичним та непередбачуваним фактором небезпеки. Для робочого місця оператора вона проявляється у вигляді:

- Загрози ракетних та бомбових ударів, атак безпілотних літальних апаратів.
- Ураження уламками скла та конструкцій будівлі внаслідок вибухової хвилі.
- Екстрених відключень електроенергії (блекаутів), що порушує роботу систем життєзабезпечення приміщення та може призвести до втрати незбережених масивів даних у застосунку.
- Постійного психологічного стресу та тривожності працівників через сигнали повітряної тривоги.

З огляду на це, невід'ємною частиною забезпечення безпеки праці сьогодні є наявність обладнаного укриття та чітко відпрацьований алгоритм дій під час повітряної тривоги.

4.2.4 Виявлення потенційних небезпек стосовно об'єкту проектування

Для систематизації виявлених загроз та визначення пріоритетних напрямків розробки заходів із охорони праці, зведемо результати аналізу до таблиці 4.1.

Таблиця 4.1 – Виявлення потенційних небезпек стосовно об'єкту проектування

№	Потенційна небезпека	Джерело небезпеки	Можливі наслідки
1	Ураження електричним струмом	Несправна ізоляція проводки, ПК, розетки, ДБЖ	Електротравми, опіки, шок, фібриляція серця, летальний наслідок
2	Підвищений рівень електромагнітного випромінювання	Монітор, системний блок ПК, Wi-Fi роутер	Головний біль, зниження імунітету, розлади нервової системи
3	Перенапруження зорового аналізатора	Тривала робота з екраном, дрібні шрифти інтерфейсу	Зниження гостроти зору, "синдром сухого ока", спазм акомодації, міопія
4	Недостатнє освітлення або відблиски	Нераціональне розміщення столу щодо вікон, погане штучне освітлення	Швидка втома очей, зниження концентрації, загальна перевтома
5	Статичне навантаження на опорно-руховий апарат (гіподинамія)	Тривале сидіння у вимушеній позі під час роботи в вебзастосунку	Остеохондроз, сколіоз, порушення кровообігу органів малого таза, радикуліт
6	Перенапруження кистей рук	Тривале використання клавіатури та комп'ютерної миші	Тунельний синдром (синдром зап'ястного каналу), біль у суглобах
7	Порушення мікроклімату	Системи опалення (перегрів/сухість), кондиціонери (протяги/переохолодження)	Застудні захворювання, зниження імунітету, тепловий дискомфорт
8	Підвищений рівень шуму	Кулери техніки, сервери, вуличний шум, офісна техніка	Зниження концентрації уваги, нервове подразнення, швидка втомлюваність
9	Нервово-емоційне напруження	Розумове навантаження, відповідальність за цілісність бази даних, монотонність	Емоційне вигоряння, стрес, безсоння, невротичні розлади
10	Біологічна небезпека (інфекції)	Контакти з колегами у закритому офісі, нечищені фільтри кондиціонерів	Поширення вірусних інфекцій (ГРВІ, грип), алергічні реакції
11	Виникнення пожежі	Коротке замикання мережі, скупчення паперу, порушення правил пожежної безпеки	Опіки, отруєння чадним газом, травми, втрата даних та майна
12	Військова загроза (бойові дії, обстріли)	Агресія з боку РФ, ракетні удари, падіння уламків збитих цілей	Осколкові поранення, баротравми, загроза життю, важкий ПТСР

4.3 Дослідження ризику реалізації потенційних небезпек на об’єкті проектування та розробка заходів щодо їх попередження

4.3.1 Теоретичні основи оцінювання ризиків

Сучасний підхід до охорони праці базується на концепції прийнятного ризику, яка передбачає перехід від реактивного реагування на нещасні випадки до проактивного управління безпекою. Оцінка ризику – це систематичний процес ідентифікації небезпек, аналізу ймовірності їх виникнення та визначення тяжкості можливих наслідків для здоров'я і життя працівників. [23]

Головною метою процедури оцінювання ризиків є створення безпечного робочого середовища шляхом своєчасного виявлення "слабких місць" у виробничому процесі. Основними задачами цього процесу є:

- виявлення всіх потенційних джерел небезпеки на робочому місці;
- визначення осіб, які можуть зазнати впливу цих небезпек;
- класифікація ризиків за рівнем їхньої критичності (від нехтуваних до неприпустимих);
- розробка пріоритетних превентивних заходів для усунення або мінімізації впливу шкідливих факторів.

Особливістю оцінки ризику для робочого місця оператора вебзастосунку (розумова праця в офісі) є те, що більшість небезпек мають прихований, накопичувальний характер (наприклад, погіршення зору або захворювання хребта), тому їх складніше ідентифікувати візуально порівняно з промисловим виробництвом.

4.3.2 Практична оцінка ризиків виявлених небезпек

Для проведення об'єктивного аналізу використаємо два взаємодоповнюючі методи: аналізування «дерева відмов» для дослідження причинно-наслідкових зв'язків критичних ситуацій та матрицю оцінювання ризиків для їх загальної класифікації.

Застосування методу «Аналізування дерева відмов»

Цей метод дозволяє графічно і логічно визначити шляхи, що призводять до небажаної (кінцевої) події. Побудуємо дерево відмов для однієї з найбільш критичних небезпек офісного середовища – Ураження оператора електричним струмом (Рисунок 4.1).

Кінцева подія (А): Ураження оператора електричним струмом. Для того, щоб ця подія відбулася, необхідний збіг двох умов одночасно (використовується логічна операція «Та»):

Подія Б: Наявність небезпечної напруги на корпусі обладнання (ПК, периферії).

Подія В: Безпосередній фізичний контакт працівника з обладнанням та відсутність/несправність захисного заземлення.

У свою чергу, Подія Б може бути викликана однією з кількох причин (використовується логічна операція «Або»):

Подія Г: Механічне пошкодження ізоляції живильного кабелю.

Подія Д: Внутрішнє коротке замикання в блоці живлення комп'ютера.

Вигляд дерева можна побачити за рисунком 4.1

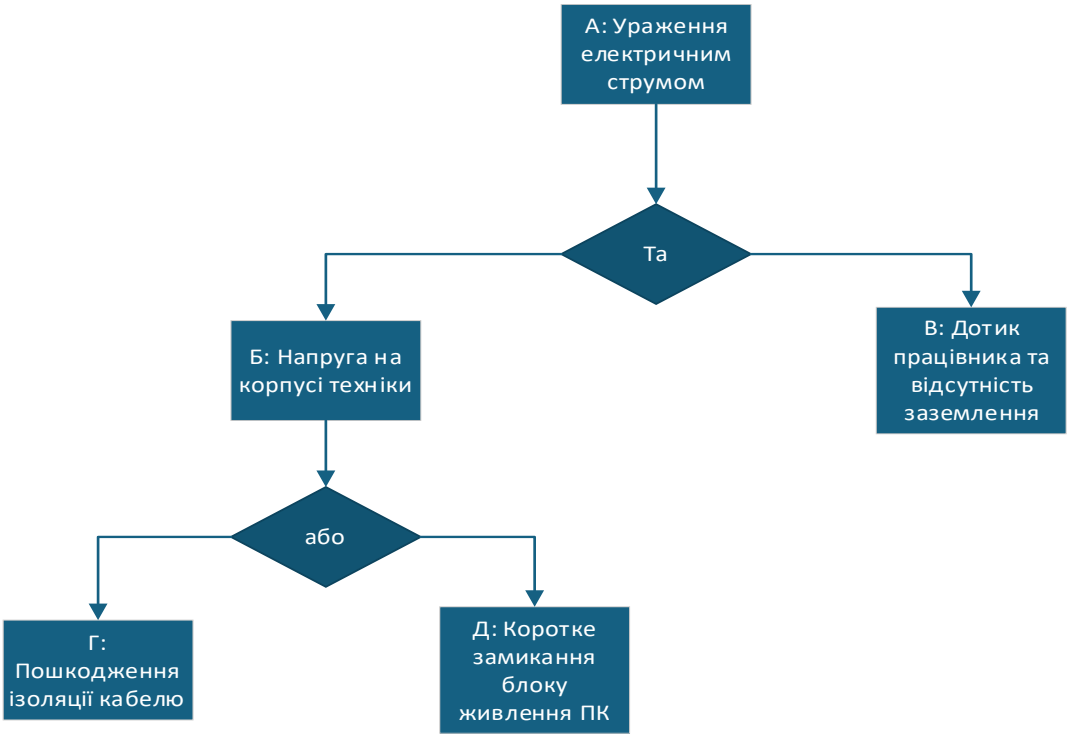

Рисунок 4.1 – Дерево відмов

Використання матриці оцінювання ризиків

Керуючись критеріями серйозності небезпеки та рівнями ймовірності, оцінимо найбільш актуальні ризики для оператора розроблюваного застосунку (таблиця 4.2).

Таблиця 4.2 – Класифікація та оцінка ризиків на робочому місці оператора

Назва небезпеки (небажаної події)	Визначення категорії серйозності	Визначення рівня ймовірності	Індекс та критерій ризику
Перенапруження зорового аналізатора	III (Гранична) – незначна травма, короткочасне захворювання (зниження гостроти зору, синдром сухого ока)	A (Часта) – велика ймовірність події при щоденній роботі з монітором	3A – Неприпустимий (надмірний) ризик
Статичне навантаження на опорно-руховий апарат	III (Гранична) – захворювання спини, остеохондроз	B (Можлива) – може трапитися кілька разів за життєвий цикл	3B – Небажаний (гранично допустимий) ризик
Ураження електричним струмом	II (Критична) – серйозна травма, суттєве пошкодження здоров'я	D (Віддалена) – малоймовірна, але можлива подія	2D – Небажаний (гранично допустимий) ризик
Військова загроза (наслідки обстрілів)	I (Катастрофічна) - загибель працівників, руйнування будівлі	C (Випадкова) - іноді може відбутися (залежить від регіону та інтенсивності бойових дій)	1C – Неприпустимий (надмірний) ризик

Найбільшої уваги потребують ризики з індексами 3A (перенапруження зору) та 1C (військова загроза), оскільки вони належать до категорії неприпустимих і вимагають негайного впровадження заходів захисту перед початком або продовженням роботи.

4.3.3 Розробка заходів щодо зниження ризиків

На основі проведеного дослідження та виявлених «вузьких місць» в організації праці оператора вебзастосунку, було розроблено комплекс технічних, організаційних та санітарно-гігієнічних заходів. Ці заходи спрямовані на переведення неприпустимих та небажаних ризиків у категорію припустимих. Перелік запропонованих рішень наведено у таблиці 4.3.

Таблиця 4.3 – Заходи щодо зниження ризиків

№	Небезпека	Запропонований захід	Очікуваний результат
1	2	3	4
1	Перенапруження зорового аналізатора (Ризик 3A)	1. Використання сучасних моніторів з технологіями Flicker-Free та Low Blue Light. 2. Регламентовані перерви по 10-15 хв кожні 2 години роботи для гімнастики очей. 3. Налаштування темної теми (Dark Mode) у розробленому вебзастосунку.	Зниження зорової втоми. Зниження індексу ризику до рівня 3D (Припустимий із перевіркою). Збереження працездатності.
2	Статичне навантаження на хребет (Ризик 3B)	1. Забезпечення оператора ергономічним офісним кріслом із підтримкою попереку та регулюванням підлокітників. 2. Розміщення монітора на рівні очей для уникнення нахилу шиї.	Попередження розвитку остеохондрозу та сколіозу. Зниження ризику до рівня 3D (Припустимий із перевіркою).
3	Ураження електричним струмом (Ризик 2D)	1. Регулярна перевірка ізоляції кабелів живлення. 2. Підключення всієї оргтехніки через розетки із захисним заземленням. 3. Встановлення пристроїв захисного відключення (ПЗВ) на лініях живлення офісу.	Усунення можливості протікання струму через тіло людини. Зниження ризику до рівня 2E (Припустимий із перевіркою).
4	Виникнення пожежі в приміщенні	1. Заборона використання кустарних обігрівачів. 2. Оснащення приміщення порошковими або вуглекислотними вогнегасниками. 3. Монтаж систем автоматичної пожежної сигналізації.	Своєчасне виявлення та локалізація загоряння на початковій стадії. Збереження життя персоналу та серверів.

Продовження таблиці 4.3

5	Військова загроза (Ризик 1С)	1. Організація роботи виключно в будівлях, що мають облаштоване укриття (бомбосховище). 2. Затвердження жорсткого алгоритму негайної евакуації при сигналі "Повітряна тривога". 3. Резервне копіювання архівів у хмарні сервіси на випадок руйнування техніки.	Захист життя працівників під час ракетних атак. Забезпечення безперервності роботи системи (збереження даних).

Висновки до четвертого розділу

У четвертому розділі кваліфікаційного проекту було проведено комплексне дослідження питань охорони праці та безпеки життєдіяльності на робочому місці кінцевого користувача розроблюваного вебзастосунка – оператора системи управління файловими архівами. На початку розділу було розглянуто організаційно-правові основи забезпечення безпеки праці, які регламентуються законодавством України. Встановлено, що ключову роль у створенні безпечного робочого середовища відіграє системний підхід до управління охороною праці з боку підприємства, який базується на абсолютному пріоритеті збереження життя та здоров'я працівника.

У ході роботи було детально проаналізовано специфіку об’єкта проектування та безпосередні умови праці оператора. Оскільки діяльність користувача пов'язана з інтенсивною розумовою працею, тривалим перебуванням у статичній позі та постійною взаємодією з обчислювальною технікою, було виявлено низку характерних небезпечних і шкідливих виробничих факторів. Встановлено, що найбільший вплив на працівника мають психофізіологічні чинники, такі як перенапруження зорового аналізатора та навантаження на опорно-руховий апарат, а також небезпеки, пов'язані з електричним обладнанням. Окрім специфічних професійних факторів, були враховані й загрози загального характеру, зокрема небезпека

виникнення пожежі та надзвичайно актуальні сьогодні ризики, пов'язані з військовою агресією.

Для об'єктивного розуміння рівня загроз було проведено оцінювання ризиків за допомогою методів побудови «дерева відмов» та матриці ризиків. Це дозволило класифікувати виявлені небезпеки та визначити ті, що мають неприпустимий характер і потребують першочергового реагування. На основі результатів цього аналізу було розроблено комплексний перелік рекомендацій щодо зниження впливу шкідливих факторів до прийнятного рівня.

Запропоновані заходи мають комплексний характер і охоплюють впровадження ергономічних рішень на робочому місці, використання захисних технічних засобів (наприклад, пристроїв захисного відключення електроенергії) та адаптацію інтерфейсу самого розробленого вебзастосунка (зокрема, інтеграцію темної теми для зниження навантаження на зір). Особливу увагу також приділено організаційним заходам: дотриманню регламентованих перерв у роботі, забезпеченню надійного резервного копіювання даних та розробці чіткого алгоритму дій персоналу під час сигналів повітряної тривоги.

ВИСНОВКИ

У роботі було вирішено актуальне науково-практичне завдання, яке полягає у проектуванні та розробці безпечного, портативного веб-модуля для управління файловими архівами на віддалених серверах. Проведене на початку дослідження предметної області підтвердило, що традиційні методи передачі великих масивів даних є неефективними, а більшість існуючих автономних рішень нехтують базовими правилами кібербезпеки. Це зумовило необхідність створення власного програмного продукту, здатного протистояти таким специфічним загрозам, як завантаження шкідливого коду, атаки формату «Zip Slip» та вичерпання ресурсів пам'яті через «Zip-бомби».

На етапі системного аналізу та проектування було сформовано вичерпний набір функціональних і нефункціональних вимог до майбутнього застосунка. Стратегічним рішенням став вибір монолітної архітектури у вигляді єдиного файлу, що гарантує абсолютну автономність скрипта без прив'язки до зовнішніх пакетних менеджерів чи баз даних. Завдяки побудові діаграм варіантів використання в нотації UML та розробці детальної алгоритмічної моделі, вдалося закласти надійний фундамент для інтеграції суворої серверної логіки маршрутизації та інтуїтивно зрозумілого графічного представлення.

Практична реалізація проєкту повністю підтвердила правильність обраних архітектурних рішень. Використання об'єктно-орієнтованих можливостей мови PHP у поєднанні з вбудованим класом ZipArchive дозволило створити потужне ядро для рекурсивної компресії та декомпресії даних. Вагомим досягненням стала реалізація багаторівневої системи захисту, що включає глибоку валідацію MIME-типів через розширення finfo, перевірку CSRF-токенів та жорсткий контроль шляхів розпакування. Інтерфейс користувача, написаний із застосуванням нативного JavaScript та сучасних CSS-підходів, забезпечив високу ергономічність і адаптивність системи.

Проведене функціональне тестування довело стабільність роботи модуля за будь-яких сценаріїв використання та його стійкість до змодельованих атак.

Окрім технічної складової, у роботі було ґрунтовно досліджено питання охорони праці та безпеки життєдіяльності. Проведений аналіз професійних ризиків оператора системи дозволив розробити комплекс організаційно-технічних заходів для мінімізації впливу шкідливих виробничих факторів. Особливий акцент було зроблено на інтеграції ергономічних рішень безпосередньо у візуальний інтерфейс розробленого застосунка задля зниження психофізіологічного навантаження, а також на адаптації робочих процесів до умов сучасних безпекових викликів.

Отримані результати дають підстави стверджувати, що мета роботи досягнута в повному обсязі, а всі поставлені завдання успішно виконані. Створений програмний продукт має високу практичну цінність, відповідає сучасним стандартам веб-розробки і є повністю готовим до впровадження як надійний інструмент для системних адміністраторів та розробників у процесах швидкого розгортання проєктів і резервного копіювання даних.

ПЕРЕЛІК ВИКОРИСТАИХ ДЖЕРЕЛ

1. Глосарій термінів з кібербезпеки OWASP (Open Web Application Security Project). Офіційний ресурс: https://owasp.org/www-community/vulnerabilities/**.**

2. MDN Web Docs (Mozilla Developer Network). Web technology for developers. Офіційний ресурс: https://developer.mozilla.org/

3. The PHP Group. PHP Manual: ZipArchive Class. URL: https://www.php.net/manual/en/class.ziparchive.php

4. The PHP Group. Standard PHP Library (SPL) and Iterators. URL: https://www.php.net/manual/en/book.spl.php

5. The PHP Group. File Information (finfo) Functions. URL: https://www.php.net/manual/en/book.finfo.php

6. The PHP Group. PHP Sessions and Security. URL: https://www.php.net/manual/en/features.session.security.php

7. PHP: The Right Way. A quick reference for PHP coding standards. URL: https://phptherightway.com/

8. MDN Web Docs (Mozilla Developer Network). HTML Forms Guide. URL: https://developer.mozilla.org/en-US/docs/Learn/Forms

9. MDN Web Docs. CSS Flexible Box Layout. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

10. MDN Web Docs. Using CSS Custom Properties (Variables). URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties

11. MDN Web Docs. Document Object Model (DOM) Manipulation. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

12. World Wide Web Consortium (W3C). HTML 5.2 Specification. URL: https://www.w3.org/TR/html52/

13. IETF. RFC 1951: DEFLATE Compressed Data Format Specification. URL: https://datatracker.ietf.org/doc/html/rfc1951

14. OWASP Foundation. OWASP Top 10 Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/

15. OWASP Foundation. Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

16. Snyk Security Research. Zip Slip Vulnerability Detailed Description. URL: https://snyk.io/research/zip-slip-vulnerability

17. MITRE Corporation. CWE-409: Improper Handling of Highly Compressed Data (Data Amplification / Zip Bomb). URL: https://cwe.mitre.org/data/definitions/409.html

18. OWASP Foundation. Unrestricted File Upload Vulnerabilities. URL: https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

19. Refactoring.Guru. Патерни проектування: Front Controller (Контролер фронтенду). URL: https://refactoring.guru/uk/design-patterns/front-controller

20. Object Management Group (OMG). Unified Modeling Language (UML) Specification. URL: https://www.omg.org/spec/UML/

21. Nielsen Norman Group. 10 Usability Heuristics for User Interface Design. URL: https://www.nngroup.com/articles/ten-usability-heuristics/

22. Конституція України. Відомості Верховної Ради України (ВВР), 1996, № 30, ст. 141. Режим доступу: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96-%D0%B2%D1%80#Text

23. Кодекс законів про працю України. Відомості Верховної Ради додаток до № 50, ст. 375. Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08

24. Закон України «Про охорону праці». Відомості Верховної Ради України (ВВР), 1992, № 49, ст. 668. Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12

25. Наказ МОЗ №248 від 08.04.2014 https://zakon.rada.gov.ua/laws/show/z0472-14#Text