МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

Кваліфікаційна робота магістра

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДВОФАКТОРНОЇ АВТОРИЗАЦІЇ ІЗ ЗАСТОСУВАННЯМ TELEGRAM»

Здобувач освіти гр. Ік.мз-13с Ростислав КОВАЛЕНКО

Науковий керівник,
к.т.н Борис КУЗІКОВ

В.о. завідувача кафедри
доцент, к.т.н. Ігор ШЕЛЕХОВ

Суми 2023

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«122 - Комп'ютерні науки»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Коваленко Ростиславу Анатолійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія двофакторної авторизації із застосуванням Telegram*

затверджую наказом по інституту від "__" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____

_____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз та огляд літератури за темою; 2) Постановка завдання; 3) Аналіз методу двофакторної авторизації за допомогою Telegram; 4) Опис веб-ресурсу; 5) Розробка веб-ресурсу з використанням методу двофакторної авторизації за допомогою Telegram; 5) Аналіз результатів;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____

_____

_____

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання_____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналіз проблеми та огляд літератури за темою.		
2.	Постановка задачі та формування завдань дослідження.		
3.	Опис архітектури веб-ресурсу.		
4.	Розробка веб-ресурсу з використанням методу двофакторної автентифікації.		
5.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи.		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 54 стор., 11 рис., 2 табл., 1 додаток., 13 джерел.

Об'єкт дослідження – інформаційна технологія двофакторної автентифікації із застосуванням Telegram.

Мета роботи - є дослідження існуючих методів аутентифікації та підвищення рівня захисту інформації Веб-ресурсів за допомогою використання технології двофакторної автентифікації через Telegram.

Методи дослідження - інформаційний аналіз, метод моделювання, методи розробки: мови програмування Python та PHP, фреймворк Angular.

Результати — реалізована модель доступу до даних веб-ресурсу на базі методу двофакторної автентифікації із застосуванням Telegram. Метод покращить авторизацію користувача і створить більш стійку систему автентифікації, що зменшить ризик несанкціонованого доступу до інформації.

АВТОРИЗАЦІЯ, ВЕРИФІКАЦІЯ, ДВОФАКТОРНА АВТЕНТИФІКАЦІЯ, TELEGRAM, 2FA

ЗМІСТ

ВСТУП

У наш час відбувається швидкий розвиток інформаційних технологій. Прогнозується, що значна частина всіх буденних справ, буде переведена в цифрову сферу. Вже зараз ми можемо робити покупки, дивитись новини, читати книги та навіть виконувати роботу просто маючи доступ до мережі Інтернет. Прикладів того, як діджиталізація полегшує наше життя, дуже багато і їх кількість постійно зростає.

Кожного дня створюються тисячі нових веб-ресурсів та додатків, які мають принести покращення в наше життя у різних сферах. Однак разом із цим зростає і кількість випадків махінацій з даними користувачів в мережі Інтернет. Питання захисту інформації користувачів останнім часом все частіше і частіше на слуху. Викрадення особистої інформації користувачів і організацій, таких як номери ID, паролі, дані кредитних карток чи номерів соціального страхування, можуть бути використанні шахраями в подальшому для різноманітних незаконних цілей, таких як отримання кредитів, проведення онлайн-покупок чи з метою отримання доступу до фінансових чи медичних відомостей жертви.

Для уникнення подібних ситуації потрібно дотримуватись ряду простих правил, як зі сторони користувача, так і з боку веб-ресурсу. Для останнього є важливим надійно зберігати персональні дані користувача та унеможливити доступ до них третіх осіб. Найпершим та найпростішим способом для цього є використання на ресурсах особистих кабінетів для користувачів, які будуть захищені багатофакторним захистом. Він характеризується тим, що користувачу для доступу до ресурсу потрібно не просто ввести свій логін та пароль, які можуть стати легкою здобиччю для шахраїв, а й, для прикладу, код, який надходить на його телефон у вигляді смс чи в спеціальному додатку кожного разу, коли відбувається авторизується в системі. Таким чином сильно знижується ймовірність не санкціонованого доступу до персонального кабінету користувача.

Основним завдання роботи є розробка системи двофакторної авторизації із застосуванням Telegram на основі веб-ресурсу, що буде простою та зрозумілою для користувача та забезпечить надійний захист.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Загальна інформація про Веб-ресурси

Веб-ресурс — це будь-який ідентифікований ресурс (цифровий, фізичний або абстрактний), наявний у Всесвітній павутині або підключений до неї. Ресурси ідентифікуються за допомогою уніфікованих ідентифікаторів ресурсів (URI) [1].

Принцип роботи Інтернету можна подати у спрощеному вигляді: коли ви переглядаєте веб-сторінку у веб-браузері на комп’ютері чи телефоні, комп’ютери, підключені до Інтернету, називаються клієнтами взаємодіють з серверами, які зазвичай знаходяться віддалено. Спрощена схема їх взаємодії відображена на рисунку 1.1.

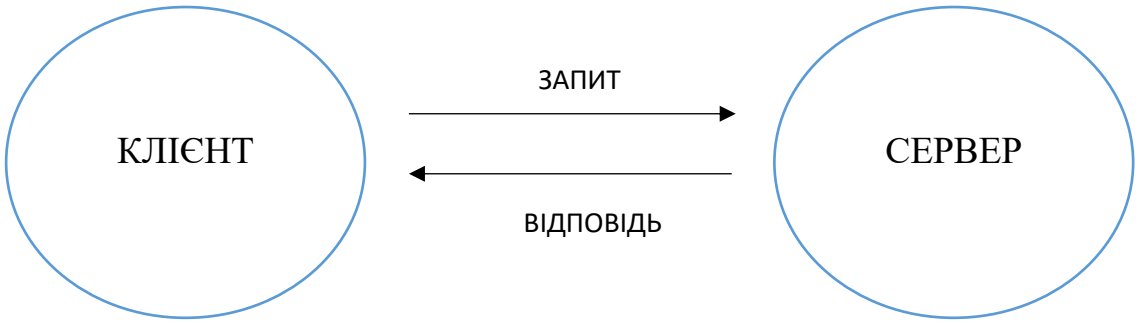

Рисунок 1.1

При їх взаємодії основними складовими є:

- Клієнти – це типові підключені до Інтернету пристрої користувача Інтернету (наприклад, ваш комп'ютер, підключений до мережі Wi-Fi, або ваш телефон, підключений до мобільної мережі) та програмне забезпечення для доступу до Інтернету, доступне на цих пристроях (зазвичай веб-браузер, такий як Firefox або Chrome).
- Сервери — це комп'ютери, на яких зберігаються веб-сторінки, веб-сайти або програми. Коли клієнтський пристрій хоче отримати доступ до веб-сторінки, копія веб-сторінки завантажується з сервера на клієнтський комп'ютер для відображення у веб-браузері користувача.
- Ваше підключення до Інтернету: дозволяє надсилати та отримувати дані в Інтернеті.

- TCP/IP: протокол керування передачею та інтернет-протокол – це протоколи зв'язку, які визначають, як дані повинні передаватися через Інтернет.
- DNS: система доменних імен подібна до адресної книги для веб-сайтів. Коли ви вводите веб-адресу у своєму браузері, браузер переглядає DNS, щоб знайти IP-адресу веб-сайту, перш ніж він зможе отримати веб-сайт. Браузеру необхідно з'ясувати, на якому сервері знаходиться веб-сайт, щоб він міг відправляти HTTP-повідомлення в потрібне місце. Це схоже на пошук адреси магазину, щоб ви могли отримати доступ до нього. Реальні веб-адреси — це не красиві рядки, які ви вводите в адресний рядок, щоб знайти свої улюблені веб-сайти. Це спеціальні номери, які мають такий вигляд: 62.240.210.10. Це IP-адреса і є унікальним розташуванням в мережі.
- HTTP: протокол передачі гіпертексту — це прикладний протокол, що визначає мову, якою клієнти та сервери спілкуються один з одним.
- Файли компонентів: веб-сайт складається з безлічі різних файлів, які подібні до різних частин товарів, які ви купуєте в магазині. Ці файли бувають двох основних типів:
    - Файли коду. Веб-сайти в основному створюються на основі HTML, CSS та JavaScript;
    - Активи: це збірна назва для всіх інших матеріалів, що становлять веб-сайт, таких як зображення, музика, відео, документи Word та PDF-файли.

Отже коли ви вводите адресу у браузері (за аналогією з походом до магазину):

- Браузер звертається до DNS-сервера та знаходить реальну адресу сервера, на якому знаходиться веб-сайт (ви знаходите адресу магазину).
- Браузер відправляє повідомлення HTTP-запиту на сервер із проханням надіслати копію веб-сайту клієнту (ви йдете до магазину та замовляєте свої товари). Це повідомлення та всі інші дані, що передаються між клієнтом та сервером, надсилаються через ваше інтернет-з'єднання з використанням TCP/IP.
- Якщо сервер схвалює запит клієнта, сервер надсилає клієнту повідомлення «200 OK», що означає «Звичайно, ви можете переглянути цей веб-сайт! Ось він», а потім починає відправляти файли веб-сайту в браузер у вигляді

серії з невеликих шматків, які називають пакетами даних (магазин дає вам ваші товари, і ви приносите їх назад до себе додому) [2].

- Браузер збирає невеликі фрагменти у повноцінну веб-сторінку та відображає її вам (товари доставляються до ваших дверей).

Порядок, у якому аналізуються файли компонентів:

1. Коли браузери надсилають запити на сервери для HTML-файлів, ці файли HTML часто містять елементи <link>, що посилаються на зовнішні таблиці стилів CSS, і елементи <script>, що посилаються на зовнішні сценарії JavaScript.
2. Браузер спочатку аналізує HTML файл, і це призводить до того, що браузер розпізнає будь-які посилання елемента <link> на зовнішні таблиці стилів CSS і будь-які посилання елемента <script> на скрипти.
3. Коли браузер аналізує HTML, він надсилає запити назад на сервер для будь-яких файлів CSS, які він знайшов в елементах <link>, та будь-яких файлів JavaScript, які він знайшов в елементах <script>, та з них, а потім аналізує CSS та JavaScript.
4. Браузер створює дерево DOM у пам'яті з проаналізованого HTML, створює в пам'яті структуру CSSOM із проаналізованого CSS, а також компілює та виконує проаналізований JavaScript [2].
5. Коли браузер будує DOM-дерево, застосовує стилі з CSSOM-дерева та виконує JavaScript, візуальне подання сторінки відображається на екрані, і користувач бачить вміст сторінки та може почати з ним взаємодіяти [3]. Таким чином відбувається створення кожного веб-ресурсу, яким ми користуємось в мережі Інтернет.

З кожним днем кількість веб-ресурсів зростає в прогресії й разом з цим зростає кількість кіберзагроз. Компанія Symantec у своєму звіті Global Internet Security Threat Report (ISTR) показує, що шахраї при зломі веб-ресурсів зазвичай користуються вразливими місцями веб-сайтів, що працюють на сервері, або ж користуються вразливостями операційної системи, на якій працюють ці ресурси. Для прикладу, кіберзлочинці можуть перенаправити запит від користувача на шкідливий веб-ресурс за допомогою XSS-атаки, а використовуючи SQL-ін’єкцію – витягнути з бази даних конфіденційну інформацію користувачів [4].

Згідно з даними від IBM Security, опублікованих в X-Force Threat Intelligence Index 2022, можна побачити, що найпоширенішим видом кіберзагрози було програмне забезпечення вимагач, такі як REvil та SOdinokibi. Дослідження показало, що Microsoft, Apple і Google були такими трьома

найпопулярнішими марками, які злочинці намагалися імітувати. Ці мегабренди неодноразово з'являлися у наборах для фішингу, причому зловмисники, ймовірно, прагнули отримати вигоду з їхньої популярності та довіри багатьох споживачів до них. 41% атак були з використанням фішингу для початкового доступу і стали основним шляхом компрометації. На 33% зросла кількість інцидентів за минулий рік і на додаток до цього, чотири з п'яти з найпопулярніших уразливостей були новими [5]. Це говорить про те, що кіберзлочинці постійно шукають нові можливості та слабкі місця у програмному забезпеченні з метою проведення шахрайських операцій. Саме тому необхідно постійно докладати зусилля для покращення захисту веб-ресурсів від можливих атак.

У відповідь на часті зломи систем безпеки був створений консорціум Open Web Application Security Project. OWASP – це відкритий проект по забезпеченню безпеки веб-ресурсів. Однак і фахівці, і зловмисники, продовжують знаходити слабкі місця у веб-ресурсах, що можуть привести до серйозних проблем з боку бізнесу. Основною причиною більшості зломів є написаний розробниками програмний код. Розробники можуть робити помилки при написанні коду або не використовувати прийоми безпечного програмування, не усвідомлюючи важливість їх використання. Все це стає причиною появи слабких місць [4].

1.2 Авторизація та автентифікація

Автентифікація та авторизація є двома життєво важливими процесами захисту інформації, які адміністратори використовують для захисту систем та інформації. Автентифікація перевіряє особу користувача або служби, а авторизація визначає їхні права доступу. Хоча ці два терміни звучать схоже, вони відіграють різні, але однаково важливі ролі у захисті програм і даних. Розуміння різниці має вирішальне значення. У сукупності вони визначають безпеку системи. Ви не можете мати безпечне рішення, якщо ви не налаштували як автентифікацію, так і авторизацію правильно.

Автентифікація - це процес, який підтверджує, що хтось або щось є тим, за кого себе видає. Технологічні системи зазвичай використовують певну форму автентифікації для захисту доступу до програми або її даних. Наприклад, коли вам потрібно отримати доступ до онлайн-сайту або служби, зазвичай потрібно ввести своє ім'я користувача/пошту/номер телефона, або ж логін, та пароль.

Потім, за лаштунками, він порівнює ім’я користувача та пароль, які ви ввели, із записом у своїй базі даних. Якщо подана вами інформація збігається, система вважатиме вас дійсним користувачем і надасть вам доступ. Системна автентифікація в цьому прикладі передбачає, що лише ви знаєте правильне ім’я користувача та пароль. Таким чином, він засвідчує вашу автентичність, використовуючи принцип того, що знаєте лише ви.

Мета автентифікації полягає в тому, щоб підтвердити, що хтось або щось є тим, ким або тим, за кого себе видає. Існує багато форм автентифікації. Наприклад, у світі мистецтва є процеси та інституції, які підтверджують, що картина чи скульптура є роботою певного художника. Так само уряди використовують різні методи автентифікації, щоб захистити свою валюту від підробки. Як правило, автентифікація захищає цінні предмети, а в епоху інформації вона захищає системи та дані.

Автентифікація особи – це процес перевірки ідентичності користувача або служби. На основі цієї інформації система надає користувачеві відповідний доступ.

Авторизація – це процес безпеки, який визначає рівень доступу користувача або служби. У технології ми використовуємо авторизацію, щоб надати користувачам або службам дозвіл на доступ до деяких даних або виконання певної дії. Важливо звернути увагу на різницю між автентифікацією та авторизацією. Автентифікація перевіряє користувача, перш ніж дозволити йому доступ, а авторизація визначає, що вони можуть робити після того, як система надасть їм доступ.

Системи авторизації існують у багатьох формах у типовому технологічному середовищі. Наприклад, списки контролю доступу (ACL) визначають, які користувачі або служби можуть отримати доступ до певного цифрового середовища. Вони здійснюють цей контроль доступу шляхом застосування правил дозволу або заборони на основі рівня авторизації користувача. Наприклад, у будь-якій системі зазвичай є звичайні користувачі та суперкористувачі або адміністратори. Якщо звичайний користувач хоче внести зміни, які впливають на його безпеку, ACL може заборонити доступ. З іншого боку, адміністратори мають право вносити зміни в систему безпеки, тому ACL дозволяє їм це робити.

Ще один поширений тип авторизації – доступ до даних. У будь-якому корпоративному середовищі зазвичай є дані з різними рівнями чутливості. Наприклад, у вас можуть бути загальнодоступні дані, які ви знайдете на веб-сайті

компанії, внутрішні дані, доступ до яких мають лише співробітники, і конфіденційні дані, доступ до яких мають лише кілька осіб. У цьому прикладі авторизація визначає, які користувачі можуть отримати доступ до різних типів інформації.

Як згадувалося, автентифікація та авторизація можуть звучати однаково, але кожна з них відіграє різну роль у захисті систем і даних. На жаль, люди часто використовують обидва терміни як синоніми, оскільки обидва вони стосуються доступу до системи. Однак це різні процеси. Простіше кажучи, один перевіряє особу користувача або служби перед наданням їм доступу, а інший визначає, що вони можуть робити, отримавши доступ.

Найкращий спосіб проілюструвати різницю між цими двома термінами – простий приклад. Припустимо, ви вирішили піти в гості до друга. Прийшовши, ви стукаєте у двері, і ваш друг відкриває. Вона впізнає вас (автентифікація) і вітає вас. Оскільки ваш друг підтвердив вашу автентичність, тепер вона спокійно пускає вас у свій дім. Однак, виходячи з ваших стосунків, є певні речі, які ви можете робити, а інші – ні (авторизація). Наприклад, ви можете зайти на кухню, але ви не можете зайти в її особистий кабінет. Іншими словами, ви маєте дозвіл входити на кухню, але доступ до її особистого кабінету заборонений.

Підсумовуючи основні відмінності автентифікації та авторизації зазначені у таблиці 1.1.

Таблиця 1.1

Автентифікація	Авторизація
Автентифікація перевіряє, ким є користувач.	Авторизація визначає, до яких ресурсів може отримати доступ користувач.
Автентифікація працює за допомогою паролів, одноразових пін-кодів, біометричної інформації та іншої інформації, наданої або введеної користувачем.	Авторизація працює через налаштування, які впроваджуються та підтримуються організацією.
Автентифікація – це перший крок хорошого процесу керування ідентифікацією та доступом.	Авторизація завжди відбувається після аутентифікації.

Автентифікацію видно користувачеві і частково може бути змінено користувачем.	Авторизація не видно і не може бути змінена користувачем.
Приклад. Підтвердивши свою особистість, співробітники можуть отримати доступ до програми HR, яка включає їх особисту інформацію про заробітну плату, час відпустки та дані 401K.	Приклад. Після авторизації рівня доступу співробітники та менеджери з персоналу можуть отримувати доступ до різних рівнів даних залежно від дозволів, встановлених організацією.

1.3 Огляд методів автентифікації та практичних реалізацій

Автентифікація — це процес ідентифікації користувачів, що запитують доступ до системи, мережі або пристрою. Контроль доступу часто визначає особу користувача відповідно до облікових даних, що маються в системі, таких як ім’я користувача та пароль. Автентифікація користувача – це метод, який запобігає доступу неавторизованих користувачів до конфіденційної інформації [6]. Для прикладу, існує користувач №1, котрий має доступ до однієї групи інформації та в цей самий час він не може бачити конфіденційну інформацію користувача №2.

Кібершахраї можуть дістати доступ до системи і викрасти певні дані, якщо процес автентифікації користувача є незахищеним. Порушення даних, з якими стикаються такі компанії, як Adobe, Equifax і Yahoo, є прикладами того, що відбувається, коли організаціям не вдається забезпечити автентифікацію своїх користувачів. Хакери отримали доступ до облікових записів користувачів Yahoo, щоб викрасти контакти, календарі та особисту електронну пошту в період з 2012 по 2016 рік. Витік даних Equifax у 2017 році відкрив дані кредитних карток понад 147 мільйонів споживачів. При відсутності безпечного процесу авторизації жодна організація не може відчувати себе в безпеці [7].

Кількість серйозних атак,що постійно збільшується, приводять нас до висновку, що незалежно від того, чи є ви малим підприємством, чи великою корпорацією, автентифікація з використанням найкращих методів безпеки є обов’язковою умовою, щоб залишатися стабільним в цьому технологічному середовищі.

Коли справа доходить до автентифікації та безпеки, існує величезне різноманіття варіантів автентифікації на вибір. Перш ніж прийняти або вибрати

будь-який із методів автентифікації, ви повинні ознайомитися з кількома ключовими факторами, які допоможуть вам вибрати найбільш відповідний для вас метод автентифікації:

- Можливість безпеки цього методу автентифікації;
- Зручність інтерфейсу;

Детальніше розглянемо доступні методи автентифікації:

- Вхід на основі пароля.

Найпоширеніша звичайна система автентифікації входу, яку ви використовуватимете щодня під час користування онлайн-службою, — це вхід на основі пароля. Під час використання техніки автентифікації на основі пароля вам потрібно ввести комбінацію вашого імені користувача/номера мобільного телефону та пароля. Особа є авторизованою лише тоді, коли обидва ці елементи були перевірені. Однак, оскільки сучасні клієнти користуються кількома онлайн-сервісами (програмами та веб-сайтами), важко відстежувати всі їхні імена користувачів і паролі. У результаті цього кінцеві користувачі вчиняють неетичну поведінку, наприклад забувають паролі, використовують той самий пароль для кількох служб тощо. Кіберзлочинці проникають у цей момент і починають такі дії, як фішинг, витік даних тощо. Це фундаментальна причина, чому стандартна автентифікація на основі пароля втрачає прихильність і все більше організацій звертаються до розширених додаткових факторів автентифікації безпеки.

- Багатофакторна автентифікація.

Багатофакторна автентифікація — це метод автентифікації, за якого особа повинна пройти кілька факторів, щоб отримати доступ до служби чи мережі. Це додатковий рівень безпеки на додаток до стандартного входа на основі пароля. Особи також повинні надіслати другий фактор у формі одноразового коду, який вони отримають телефоном або електронною поштою на додаток до свого імені користувача та пароля.

Ви можете швидко налаштувати кілька методів багатофакторної автентифікації, щоб забезпечити додатковий рівень безпеки для своїх ресурсів. Отримання коду через SMS чи електронну пошту, push-сповіщення, апаратний маркер і мобільний автентифікатор — це приклади методів багатофакторної автентифікації (Google, Microsoft, Authy тощо). Ви можете вибрати будь-яку техніку і застосувати її для організаційної безпеки відповідно до ваших потреб і вимог. Після традиційного входу на основі пароля, багатофакторна автентифікація є найбільш надійним механізмом автентифікації. Для

покращення безпеки традиційна автентифікація на основі пароля та методи багатофакторної автентифікації зазвичай використовуються одночасно.

- Біометрична автентифікація.

Індивідуальні фізичні атрибути, такі як відбитки пальців, долоні, сітківка, голос, обличчя та розпізнавання голосу, використовуються в біометричній автентифікації. Біометрична аутентифікація працює таким чином: спочатку фізичні характеристики людей зберігаються в базі даних. Щоразу, коли користувач хоче отримати доступ до будь-якого пристрою або фізично увійти в будь-яке приміщення (організація, школа, коледжі, робоче місце), фізичні особливості окремих осіб перевіряються на відповідність даним, що містяться в базі даних. Технологія біометричної автентифікації в основному використовується приватними організаціями, аеропортами та пунктами пропуску, де безпека є пріоритетом. Завдяки своїй здатності створювати високий рівень безпеки та зручний безперебійний потік, біометрія є однією з найбільш часто використовуваних технологій безпеки. Серед найпоширеніших методів біометричної аутентифікації:

1. Відбиток пальця: щоб увімкнути доступ, автентифікація за відбитком пальця відповідає унікальному шаблону відбитка особи. У деяких вдосконалених системах автентифікації за відбитками пальців також визначається судинна структура пальця. Оскільки це одна з найбільш зручних і точних біометричних систем, автентифікація за відбитками пальців наразі є найпоширенішою біометричною технологією для звичайних клієнтів. Популярність біометрії може бути пов’язана з тим, що ви регулярно використовуєте свої мобільні телефони з відбитками пальців, а також компаніями чи установами, які використовують автентифікацію за відбитками пальців.
2. Сітківка та райдужна оболонка ока: сканери просвічують яскраве світло в око та шукають характерні візерунки в барвистому кільці навколо зіниці ока в цій біометрії. Після цього сканований шаблон порівнюється з даними, записаними в базі даних. Коли людина носить окуляри або контактні лінзи, автентифікація на основі очей може бути неточною.
3. Обличчя: під час автентифікації обличчя сканується кілька аспектів обличчя людини, коли вона намагається отримати доступ до певного ресурсу. Під час порівняння облич під різними кутами або схожих осіб, наприклад членів сім’ї, результати розпізнавання облич можуть бути суперечливими.

4. Розпізнавання голосу: тон вашого голосу зберігається зі стандартизованим секретним кодом так само як і вищезгаданий підхід. Перевірка відбувається, оскільки ви повинні говорити щоразу, коли хочете отримати доступ.

- Аутентифікація на основі сертифіката.

Автентифікація на основі сертифіката ідентифікує людей, сервери, робочі станції та пристрої за допомогою електронної цифрової ідентифікації. У нашому повсякденному житті цифровий сертифікат функціонує подібно до водійських прав чи паспорта. Сертифікат складається з цифрової ідентифікації користувача, яка містить відкритий ключ і цифровий підпис центру сертифікації. Цей сертифікат підтверджує, що відкритий ключ і особа, яка видала сертифікат, є однією особою. Коли користувач намагається увійти на сервер, він повинен спочатку надати свій цифровий сертифікат. Сервер перевіряє ідентичність і достовірність цифрового сертифіката, підтверджуючи, що користувач має правильно пов’язаний закритий ключ із сертифікатом за допомогою криптографії.

- Автентифікація на основі маркерів.

Автентифікація на основі маркерів дає можливість користувачам вводити свої облікові дані один раз і в результаті отримати єдиний оригінальний рядок, що є зашифрований. Він також називається токен. Після цього не буде потреби вводити свої дані для входа кожного разу, коли є потреба увійти або отримати доступ. Токен гарантує, що вам уже надано доступ. Більшість випадків використання, наприклад Restful API, доступ до яких здійснюється багатьма фреймворками та клієнтами, потребують автентифікації на основі токенів.

Кожен метод має переваги, однак деякі типи автентифікації ефективніші за інші. Наприклад, паролі самі по собі забезпечують найменшу безпеку з усіх цих методів. Найнадійніший метод автентифікації поєднує багато з цих функцій, щоб запропонувати найнадійніший захист.

Найчастішим рішенням для підвищення безпеки є використання багатофакторної автентифікації за допомогою одноразового пароля, що надходить на телефон, пошту чи відображається в спеціальному додатку.

Розглянемо основні рішення, які зараз використовуються найчастіше:

Посилання SMS з паролем – це одна з перших найбільш поширених технологій для відправлення одноразових паролів. SMS – це дуже поширений канал зв’язку, що присутній на всіх телефонах і регулярно використовується

значною кількістю користувачів. Доставлення одноразових паролів через SMS відносно просте у своїй реалізації, але володіє низкою недоліків, через що їх використання зменшується з кожним роком. По-перше, їх доставлення менш надійну, оскільки в ланцюг доставки додається мобільний оператор, у якого можуть бути проблеми з покриттям, також це додатковий фактор ризику для безпеки. По-друге, необхідно мати при собі телефон, на який будуть надходити SMS, і у випадку його втрати або викрадення зловмисником - він може легко скористатись ним у своїх цілях і з цим нічого не вийде вдіяти. Дуже важливим моментом є те, що надсилання таких сповіщень не є безпланим. Вартість однієї SMS може коливатись від 30 до 60 копійок за одну SMS у випадку надсилання від імені компанії, що зареєстрована в України, а для закордонних компаній вартість одного SMS становить від 4 до 6 грн за повідомлення.

Google Authenticator – це найпопулярніша програма для двофакторної аутентифікації. Вона є простою та не пропонує жодних надмірностей. На відміну від Microsoft Authenticator, Google Authenticator не додає жодних спеціальних параметрів для власних служб. У Google Authenticator немає онлайн резервної копії кодів ваших облікових записів, але ви можете імпортувати їх зі старого телефону на новий, якщо у вас під рукою є перший. Google Authenticator надає вам код автентифікації в реальному часі, який змінюється кожні 30 секунд. Google пропонує його для всіх ваших облікових записів Google. Однак його можна використовувати для багатьох інших веб-сайтів. Для його використання не потрібне підключення до Інтернету Він абсолютно безплатний, чистий, функціональний і має велику базу користувачів. Зрештою ви зможете додати численні облікові записи до цієї програми. Завантажте звідси для Android та iOS. Він також доступний як розширення Chrome [8].

Додаток не використовує значки, що ускладнює швидкий пошук кодів, особливо якщо у вас десятки облікових записів. Він часто затримує оновлення програмного забезпечення, коли виходить нове оновлення мобільної операційної системи, особливо на телефонах Apple, що спричиняло проблеми з відкриттям програми в минулому [9].

Microsoft Authenticator - включає безпечну генерацію паролів і дозволяє входити в облікові записи Microsoft одним натисканням кнопки. Відновлення облікового запису є важливою функцією, яку слід увімкнути, якщо ви використовуєте цю програму. Таким чином, коли ви отримаєте новий телефон, ви побачите можливість відновити, увійшовши в обліковий запис Microsoft і надавши додаткові підтвердження. Як і програма 2FA, Microsoft Authenticator

пропонує інший рівень безпеки: ви можете вимагати розблокування телефону за допомогою PIN-коду або біометричної перевірки, щоб побачити коди.

Microsoft Authenticator має кольорові піктограми для кожної служби, що спрощує швидкий пошук потрібного імені для входу.

2FA Authenticator - чудовий варіант, якщо вам потрібне елегантне програмне забезпечення для автентифікації. Для шестизначної автентифікації TOTP це чудова програма. Програма генерує тимчасові одноразові паролі, які зберігаються на телефоні користувача без доступу до Інтернету. Це рішення значно підвищує безпеку входу. 2FA також пропонує таку функцію, як автентифікація на основі QR-коду, які роблять вхід легшим і безпечнішим. Це дозволяє повністю уникнути проблеми ненавмисного неправильного введення та економить час. На відміну від Google Authenticator у ньому розроблена функція служби віддаленого резервного копіювання, яка дозволяє зберігати службові ключі в безпечній хмарній резервній копії, щоб ви могли відновити їх у разі втрати телефону. Ця функція також дозволяє генерувати коди на різних пристроях.

2FAS не потрібно знати ваш номер телефону або навіть вимагати від вас створення облікового запису, тому він не схильний до шахрайства із заміною SIM-карти. Ви можете встановити PIN-код для доступу до програми, і на iPhone він може використовувати FaceID або TouchID.

Додаток 2FAS Authenticator є безплатним і доступним для користувачів Android та iOS через Google Play або App Store.Крім того, цей надійний інструмент автентифікації сумісний із понад 500 соціальними та іншими веб-сайтами.

У якості додатка для двофакторної авторизації також може бути використаний Telegram. Telegram messenger – це безплатна, глобально доступна, зашифрована, хмарна та централізована служба для миттєвого обміну повідомлення. Він використовується для обміну текстовими, голосовими чи відеоповідомленнями, файлами, фотографія та наліпками. У Telegram можливо створювати власні групи та канали і проводити в них голосові та відеоконференції.

Сервери Telegram знаходяться по всьому світу з основними п’ятьма дата-центрами, що знаходяться у різних частинах світу, а головний офіс знаходиться в Дубаї, ОАЕ. Telegram доступний для Android, операційних систем Windows, Linux та macOS. Але для реєстрації необхідний телефон і наявний номер телефону. Месенджер має відкритий вихідний код, через що існує значна

кількість офіційних та неофіційних клієнтів. Єдина частина програми, яка недоступна для спільноти, це серсев, оскільки він має закритий вихідний код і належить компанії. Telegram надає опціональні наскрізні зашифровані чати. Хмарні чати і групи мають шифрування між клієнтом і сервером, саме через це провайдери та інші треті сторони не мають змоги отримати доступ до даних, що забезпечую гарну безпеку для месенджера.

Telegram є одним з найпопулярніших додатків для обміну повідомлень і має більше ніж 700 мільйонів активних користувачів щомісяця.

Для месенджера був створений протокол MTProto, що передбачає використання декількох протоколів шифрування. Під час авторизації та аутентифікації використовуються алгоритми RSA-2048, DH-2048 для шифрування, під час передачі повідомлень протоколу в мережу вони шифруються AES з ключем, відомим клієнту і серверу. Також застосовуються криптографічні хеш-алгоритми SHA-1 та MD5 [10].

На цей час активно набирає популярність побудова чат-ботів у Telegram. Кожен користувач має змогу зробити свій чат-бот для своїх потреб. Боти можуть бути використання для виконання різних завдань, наприклад пошук інформації в мережі Інтернет чи в державних реєстрах, проведення покупок та платежів, для модерації груп або розваг.

1.4 Постановка задачі

Метою даної роботи є розгляд та реалізація методу двофакторної авторизації за допомогою Telegram на основі веб-ресурсу. Метод забезпечить користувача додатковим рівнем захисту його особистих даних та буде зручним у користуванні.

Основними вимогами до ресурсу є:

- Організація роботи бази даних;
- Можливість реєстрації та авторизації користувача на веб-ресурсі;
- Використання методу двофакторної авторизації;
- Реалізація чат-боту в Telegram;
- Стабільність роботи веб-ресурсу;
- Зручність використання веб-ресурсу.

2 ПРАКТИЧНА РЕАЛІЗАЦІЯ

2.1 Вибір мови програмування

Для розробки веб-додатку було вибрано фреймворк Angular - JavaScript фреймворк з відкритим кодом, що був створений Angular Team в Google в колаборації з приватними розробниками і розповсюджений спільнотою, щоб допомогти всім охочим створювати front-end програми. Згідно з даними GitHub, який використовують 73 мільйони інженерів для розміщення та нагляду за проектами з відкритим кодом і бізнес-проектами, у них зберігається близько 28 тисяч репозиторіїв з Angular і більше чим 1,6 мільйона користувачів використовували дані репозиторії у своїй роботі, а рівень задоволення від їх використання становив в близько 83.65% [11].

Angular допомагає створювати веб-додатки на основі HTML, CSS і JavaScript. Розробники можуть створювати персоналізовані компоненти об'єктної моделі документа за допомогою AngularJS [12]. Синтаксис складається з HTML і TypeScript. Це діалект програмування з відкритим кодом, створений і підтримуваний корпорацією Майкрософт, який включає статичні варіанти написання. HTML використовується в синтаксисі шаблону Angular, і практично весь HTML можна використовувати повторно. Додатки Angular складаються з контейнерів блоків коду, оскільки вони складаються з модулів NgModules, призначених для обсягу, робочого процесу або тісно пов'язаних функцій. Модулі забезпечують контекст компіляції вмісту, і вони можуть включати компоненти та служби в Provider та інші файли.

Життєвий цикл в Angular керує тривалістю життя компонента. Він складається з трьох фаз: завантаження, компіляції та виконання. Фаза завантаження — це перша фаза життєвого циклу AngularJS. Фаза початкового завантаження відбувається, коли бібліотека AngularJS JavaScript завантажується у ваш браузер. Далі AngularJS налаштовує необхідні компоненти, а потім повідомляє модулю, де їх знайти. Далі модулі складаються в стек і всі залежності вбудовані в модуль є доступні для коду в ньому. Фаза компіляції HTML є другою фазою життєвого циклу веб-розробки. Коли веб-сторінка завантажується, створюється статична структура DOM і вставляється в браузер. Статичний DOM замінюється на динамічний DOM, який відображає вигляд AngularJS під час виконання етапу компіляції. Фаза виконання програми AngularJS триває, доки користувач не перезавантажить сторінку або не вийде з неї. Будь-які зміни в

області відображаються в той самий час. Крім того, будь-які зміни явно оновлюються в межах вмісту, завдяки чому він виглядає як єдине джерело інформації для подання.

AngularJS поводиться інакше, ніж традиційні методи зв'язування даних. Традиційні методи змішують шаблон із даними механізму перед маніпулюванням DOM кожного разу, коли дані змінюються. AngularJS компілює DOM лише один раз, а потім за потреби підключає створений шаблон, заощаджуючи час у порівнянні зі старішими методами.

AngularJs може використовувати сторонні бібліотеки, встановлюючи та імпортуючи за допомогою наданих функцій NPM. Крім того, бібліотеки сторонніх розробників розширюють простоту використання Angular, додаючи завершений стиль і функціональність. Однією з найпопулярніших бібліотек стилів для Angular є Angular Material. Angular Material — це набір готових компонентів, розроблених з урахуванням матеріального дизайну. Вони розроблені для продуктивності та бездоганно працюють з Angular.

Фреймворк у своїй роботі базується на Typescript, Html5, CSS3. TypeScript додає до JavaScript додатковий синтаксис для підтримки більш тісної інтеграції з вашим редактором. Код TypeScript перетворюється на JavaScript, який виконується будь-де, де працює JavaScript: у браузері, на Node.js або Deno та у ваших програмах. TypeScript розуміє JavaScript і використовує визначення типу, щоб надати вам чудові інструменти без додаткових Отримала відзнаку як друга найулюбленішу мову програмування в опитуванні розробників Stack Overflow 2020. TypeScript використовували 78% респондентів State of JS у 2020 році, а 93% сказали, що використали б його знову.

HTML 5 — це остання версія HTML, яка обіцяє надати ряд корисних функцій і покращити досвід роботи в Інтернеті. Він долає обмеження HTML 4, додаючи параметри, які ще більше розширюють функціональність. Хоча наразі він підтримується не всіма браузерами, але коли це буде зроблено, він може змінити спосіб створення веб-сайтів. HTML значно полегшив декларування документа, яке тепер можна зробити як <!doctype html>. Присутня функція API геолокації – вона допомагає визначати місцеперебування користувачів і, таким чином, корисна для онлайн-бізнесу, щоб відстежувати своїх потенційних відвідувачів. Це особливо корисно, якщо користувачі використовують мобільні телефони для серфінгу. Однак необхідно, щоб користувач надав необхідний дозвіл на доступ до інформації про місцеперебування. Не вимагаючи жодний додатковий плагін чи налаштування, HTML 5 дозволяє легко включати аудіо- та

відеоелементи на сторінку. Таким чином, він перевершує HTML 4. HTML 5 дозволяє зберігати файли або веб-додатки та робить їх доступними без підключення до Інтернету.

Проблемними зонами використання HTML 5 є те, що він ще не повністю розроблений і тому схильний до змін. Через це його прийняття розробниками може бути трохи відкладено.

Так само як HTML 5, CSS 3 також залишив свій слід завдяки багатьом функціям, які не лише підвищують естетичну привабливість, але й покращують функціональність.

Ось деякі особливості CSS 3, які діють на його користь:

- Кілька фонів: це ще одна корисна функція CSS 3, за допомогою якої досить легко створити кілька фонів для одного елемента;
- Кольорові схеми плюс непрозорість: це одна з найбільш цінних функцій, оскільки вона дозволяє розробникам вибирати з кількох варіантів кольорів, будь то RGB, CMYK або HSL, а також легко визначати рівень непрозорості або прозорості прямо в коді. За допомогою CSS 3 код можна записати як RGBa або HSLa;
- Заокруглені кути: це ще одна дуже корисна функція CSS 3, яка дозволяє отримувати закруглені кути без додаткових зусиль з боку розробників. За допомогою властивості Border Radius CSS 3 легко створити круглі краї для коробки;
- Зображення межі - CSS 3 дозволяє дизайнерам гнучко використовувати зображення для межі з його властивістю border image і таким чином підтримує їхню творчість;

Проблемні зони:

- Підтримка браузера - як і у випадку з HTML 5, деякі браузери не підтримують CSS3;
- Змінні відсутні - оскільки CSS 3 не має змінних, він потребує значних зусиль, навіть якщо потрібно внести незначні зміни на сайті;
- Все ще потребує вдосконалення, щоб називатися мовою макета. CSS 3 можна вважати високоефективною мовою стилів, але щоб стати мовою верстки, вона повинна мати більше контролю та зробити макети гнучкими.

Back-end частина була реалізована за допомогою PHP та Python.

PHP, також відомий як Hypertext Preprocessor, — це сценарна мова з відкритим вихідним кодом, що була створена Расмусом Лердорфом приблизно три десятиліття тому, і в основному використовується для створення веб-

сторінок. Останні версії цієї мови сценаріїв, такі як PHP 7 і PHP 8, є об'єктно-орієнтованими, що допомагає створювати безпечні структури та функції, які є динамічними та придатними для повторного використання [13].

Плюси використання PHP є:

- Відкрите джерело: повністю відкритий кодом, тобто користувачам не потрібно витрачати гроші на використання фреймворків, виконання коду, використання бази даних, інструментів налагодження тощо;
- Велика бібліотека: бібліотеки мов програмування мають попередньо написані й оптимізовані коди для безперебійного виконання завдань і зниження продуктивності;
- Швидкість: PHP майже втричі швидша за Python, а поточні версії (№7 і новіші) навіть швидші за попередні;
- Кросплатформенність: PHP може без проблем бути використаний на багатьох платформах - Microsoft Windows, MacOS, Linux та різних варіаціях системи Unix;
- Доступ до бази даних: За допомогою PHP можливо створити доступ до більшості баз даних, а також працювати з інформацією в них, використовуючи дані користувача з мережі. За його допомогою можуть бути виведені PDF-файли , зображення і навіть Flash- фільми.

Недоліки використання PHP:

- Обмежені засоби налагодження: однією з поширених скарг більшості розробників на PHP є те, що він має обмежені інструменти налагодження. Він погано обробляє помилки, особливо в порівнянні з іншими мовами сценаріїв;
- Неможливо змінити основну поведінку: якщо ви працюєте над завданням і ваша головна мета — зробити його якомога креативнішим — тоді PHP — не підходящий для вас інструмент. Це пояснюється тим, що PHP як мова сценаріїв не допускає будь-яких модифікацій або змін у рамках основної поведінки веб-додатків.

Підсумовуючи основні переваги та недоліки PHP відображені у таблиці 2.1.

Таблиця 2.1

Переваги PHP	Недоліки PHP
Надзвичайна масштабованість	Не універсальність
Безплатний для використання	Обмежені засоби налагодження
Підтримка спільноти	Неможливо змінити основну поведінку
Легкий у використанні	Безпека не найкраща
Блискавична швидкість	Є мови, які простіше використовувати

Python — це універсальна, інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою.

Згідно з даними опитування stackoverflow.com Python 4 за популярністю мова програмування, а рівень задоволення від її використання становить 67,3%.

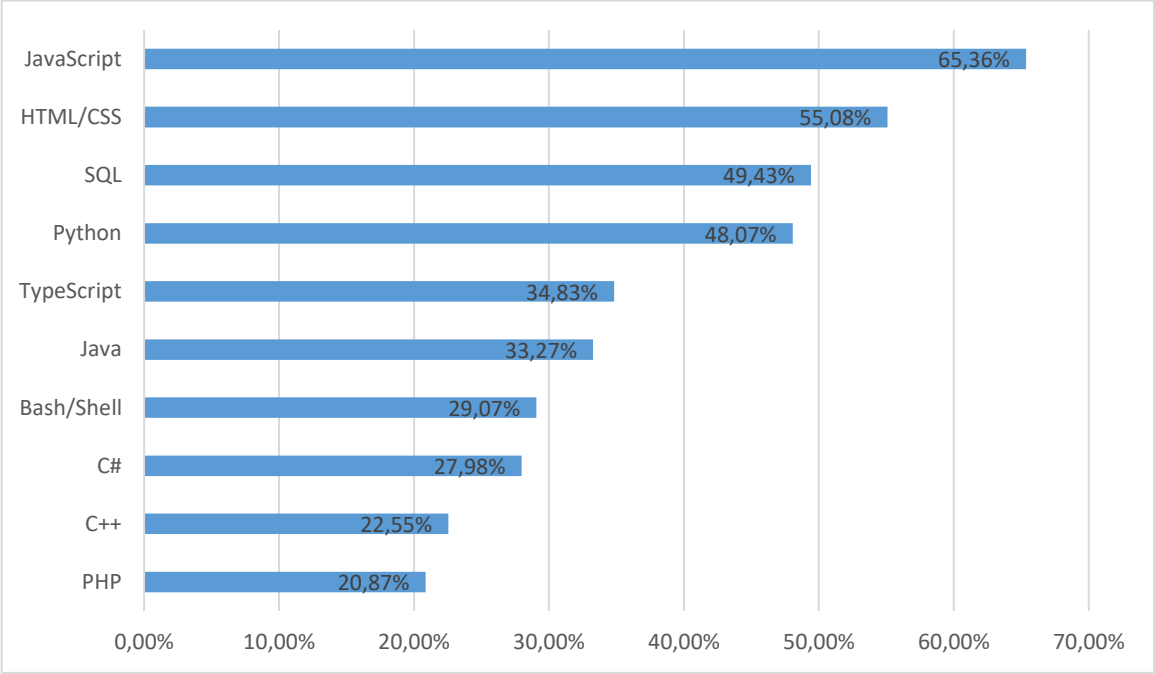

Рисунок 2.1 - Частота використання різних мов програмування

У Python вихідний код компілюється в проміжний формат, званий байт-кодом. Ця компактна мова низького рівня працює на віртуальній машині Python (PVM), яка є програмним забезпеченням, яке імітує роботу справжнього апаратного забезпечення. PVM, у свою чергу, діє як інтерпретатор: він

перетворює інструкції байт-коду одну за одною в машинний код безпосередньо під час виконання. Хоча в Python є етап компіляції, він не такий, як у компільованих мовах, таких як C++ або Swift. Останній компілюється безпосередньо в машинний код одним фрагментом, перед виконанням, без проміжного кроку.

Рисунок 2.2 – Відмінність компільованої та інтепретованої мови програмування

Подібно до Java, Ruby, C++ та багатьох інших популярних мов програмування, Python підтримує об’єктно-орієнтоване програмування (ООП), яке зосереджується на сутностях або об’єктах, з якими працюватимуть розробники. Об’єкт представляє річ реального життя або абстрактну одиницю зі своїми унікальними властивостями (станом) і поведінкою (методами).

Однак Python, який часто розглядають як чисту мову ООП, дозволяє функціональне програмування, яке зосереджується на тому, що потрібно зробити (функціях). У великих кодових базах можуть бути перемикання між двома парадигмами, залежно від конкретної мети.

Python належить до мов програмування високого рівня, які нагадують природну мову і не прив’язані до конкретного процесора комп’ютера.

Python використовує динамічну типізацію, що означає, що розробникам не потрібно оголошувати тип змінної. Його буде автоматично розпізнано під час виконання на основі значення, присвоєного відповідній змінній. Одна і та ж змінна може багато разів змінювати свій тип під час виконання програми, що неможливо для мов зі статичною типізацією.

Плюси використання Python:

- плавна крива навчання: Python дуже зручний для початківців і відносно простий у вивченні. Завдяки своєму спрощеному синтаксису, подібному до англійської, він дозволяє новачкам зосередитися на основах програмування, концепціях і хороших практиках кодування, а не на тонкощах структури мови;
- висока швидкість розвитку: чіткий, лаконічний синтаксис спрощує та прискорює не лише вивчення Python, але й створення програмного забезпечення за його допомогою. Крім того, його стандартна бібліотека надає безліч готових функцій, які дозволяють програмістам працювати з Інтернет-протоколами, керувати операційними системами, маніпулювати даними або інтегрувати веб-сервіси з меншими зусиллями.
- Python також є компонентом стеку LAMP, який означає Linux, Apache, MySQL і Python, PHP або Perl (усі мови з динамічною типізацією). Він забезпечує стандартний і зрілий спосіб створення веб-додатків, заощаджуючи час розробки.
- переносимість і розширюваність іншими мовами: Python не залежить від платформи: ви можете запускати той самий вихідний код у різних операційних системах, будь то macOS, Windows або Linux. Переносимість досягається за рахунок байт-коду та віртуальної машини Python (PVM), які служать посередниками між розробником і фактичним ЦП, що виконує програму.Крім того, Python легко об’єднується з іншими мовами за допомогою таких розширень, як Cython для C, Gython для Go, Jython для Java та IronPython для .Net. Вони дозволяють розробникам змішувати мови, запозичувати функціональність, якої не вистачає їхній основній технології, і запускати код у своїх програмах.
- автоматизація тестування: Python має репутацію великої кількості тестових фреймворків, які прискорюють перевірку якості вашого коду на кожному етапі життєвого циклу розробки програмного забезпечення.

Основні недоліки, про які слід пам’ятати, вибираючи Python:

- обмеження швидкості: Python не може похвалитися такими ж результатами в швидкості виконання, відстаючи від C++ і Java. Програма інтерпретується під час виконання, рядок за рядком, замість того, щоб компілюватися до машинного коду одним фрагментом. Хоча це приносить переваги з точки зору налагодження, це відбувається ціною продуктивності під час виконання. Іншим властивим фактором затримки є динамічна семантика. Це спрощує роботу програмістів, оскільки їм не потрібно оголошувати типи змінних і можна обійтися меншою кількістю рядків коду. Замість розробника, інтерпретатор повинен перевіряти та призначати типи під час виконання програми, що робить її повільнішою.
- відсутність багато поточності: Python використовує механізм під назвою Global Interpreter Lock, або скорочено GIL, який дозволяє одночасно виконувати лише одну послідовність інструкцій байт-коду (потік).
- велике споживання пам'яті: об’єкти мають величезні накладні витрати та можуть поглинути в десять разів більше пам’яті, ніж потрібно для зберігання інформації, яка нам насправді потрібна. Хоча Python має збирач сміття для керування пам’яттю, він не повертає ресурси системі одразу після того, як об’єкт стає непотрібним. Крім того, якщо ваш код містить будь-які посилання на цей застарілий об’єкт, він взагалі не буде видалений.
- Проблеми з розробкою для мобільних пристроїв і інтерфейсу: жоден смартфон не працює на Python: нативна розробка додатків для Android виконується за допомогою Java або Kotlin, тоді як iOS працює з Objective C і Swift.

2.2 Середовище розробки

Програмування з PHP не вимагає наявності редактору коду, адже PHP файл складається зі звичайного тексту і може редагуватись в будь-якому текстовому редакторі. Однак використання спеціально розробленого середовища для програмування допоможе полегшити навігацію в коді, пришвидшить його написання завдяки автозавершенню та зробить його перевірку зручнішою через автоматичне підсвічування помилок. Використання IDE робить процес розробки інтерактивним та ефективним.

PHPStorm – це спеціалізоване повноцінне інтегроване середовище розробки, розробкою якого займається JetBrains.

Плюсами використання PHPStorm IDE є:

- Завершення коду PHP та автоматичне підсвічування помилок;
- Підтримка Smarty та PHPDoc
- Підтримка VCS , таких як: SVN, Git, локальна;
- Розумний рефакторинг: при перейменуванні файлу – зміни будуть автоматично внесені по всьому проекту; при перейменуванні глобальної змінної – вона автоматично буде перейменована в усіх файлах, що її використовують;
- Візуальне налагодження всередині IDE;
- Веб орієнтована IDE, що означає підтримання швидкого редагування HTML, CSS і JS;
- Широкий вибір плагінів;

Недоліком даної IDE можна вважати високі вимоги до обладнання, адже велика кількість різнопланового функціонала використовують значну кількість ресурсів, та неможливість редагування одного окремого файлу - необхідно включати його в наявного проекту або створювати новий.

Написання чат-бота в Telegram найкраще реалізовувати за допомогою мови програмування Python. Для написання та запуску коду використовувався Python Intellij IDEA, який має широкий спектр інструментів для розробників.

Основними перевагами Python Intellij IDEA є:

- Якісний редактор вихідного коду: підсвітка синтаксису, переходи до визначення, автозавершення з врахування контексту, автоматичне додавання потрібних імпортів, що дає сильний приріст до продуктивності;
- Інтеграція з Git та іншими системами контролю версій, вбудований налагоджувач, вбудоване визначення ступеню покриття коду тестами;
- Автоматичне форматування коду;
- Зручний для рефакторингу: легко та ефективно роблять зміни як в глобальних, так і у локальних змінних;

2.3 Обгрунтування вибору Telegram для методу двофакторної автентифікації

У наш час оптимальним підходом для отримання доступу до певного ресурсу є реєстрація в ньому та подальше використання логіну та пароля для авторизації. Однак ця інформація може бути злмана та стати доступною для кіберзлочинців за лічені хвилини, через що особисті та фінансові дані користувачів знаходяться під постійною загрозою.

Двофакторна авторизація, також відома як 2FA, додає другий рівень безпеки, цим самим доповнює модель «логін та пароль» кодом, до якого має доступ тільки користувач, який намагається авторизуватися. Зазвичай даний код надсилається на телефон за допомогою SMS або ж відображається у спеціально призначеному для цього додатку. Такий підхід до автентифікації можна схарактеризувати як комбінацію «того, що ти знаєш і того, що ти маєш».

2FA використовуючи мобільний пристрій в останні роки являється галузевим стандартом, адже переважна більшість людей мають мобільний телефон при собі. Це практично у використанні, адже паролі,що генеруються динамічно, постійно надходять на пристрій і можуть бути використані користувачем для авторизації.

Рисунок 2.3 – Робота двофакторної автентифікаї за допомогою Telegram

Процес авторизації являє собою автентифікацію за допомогою логіну і паролю, коли веб-ресурс перевіряє чи існує користувач з такими даними, та введення цифрового коду, що відображається в додатку чи отриманий в SMS за відведений проміжок часу. Тільки після виконання цих умов користувач зможе потрапити на веб-ресурс.

Telegram – це глобальний багатоплатформовий хмарний месенджер. Він входить в десятку найбільш завантажуваних додатків у світі, яким користуються мільярди користувачів. Основними перевагами Telegram є:

- Швидкість роботи: дата центри розподілені по всьому світі, що забезпечує швидку передачу повідомлень;
- Синхронізація: можна отримати доступ до додатка одночасно з декількох пристроїв, в тому числі з комп’ютера за допомогою десктопної або

вебверсії. При цьому завжди можна перевірити кількість підключених акаунтів у Telegram і за потреби відімкнути будь-який за потреби;

- Безпека: захист даних відбувається завдяки використанню комбінації 256-бітного симетричного шифрування AES, 2048-бітного шифрування RSA і алгоритму безпечного обміну ключами Діффі-Гелмана;
- Простота використання: не зважаючи на широкий вибір функціонала та можливостей - додаток залишається простим та інтуїтивно зрозумілим під час використання;
- Доступність: можна користуватись безплатно, без підписок та реклами;
- Конфіденційність: Telegram не передає інформацію третім особам;
- Чат-боти: можливо самостійно створювати чат-боти для особистого та спільного використання. Саме це буде використано в роботі для використання Telegram, як застосунку для двофакторної автентифікації.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Підхід до побудови проекту

Розроблений веб-ресурс являє собою форму авторизації, яка відображає можливості інформаційної технології двофакторної авторизації за допомогою Telegram. Сам веб-ресурс не має жодного контенту для того, щоб відобразити,що дана інформаційна технологія може бути використана не залежно від направлення та сфери користування.

Першим етапом створення веб-ресурсу є планування та проектування. В нашому випадку було реалізовано:

- Сторінка реєстрація;
- Сторінка входу;
- Сторінка вводу пароля отриманого в чат-боті;
- Сторінка контенту.

Додатково розроблене підключення бази даних, для збереження інформації про користувачів, та чат-бот в Telegram, у якому можна отримати код, необхідний для успішної авторизації.

3.2 Опис функціоналу та методу двофакторної авторизації

Для того, щоб мати можливість користуватись веб-ресурсом, необхідно зареєструватись на сайті. Для цього розроблена форма реєстрації.

```
<div fxLayout="row" fxLayoutAlign="center center" class="login-main">
  <mat-card >
    <form [formGroup]="registerForm">
    <mat-card-header>
          <mat-card-title>Зареєструватися</mat-card-title>
    </mat-card-header>
    <br>
    <mat-card-content fxLayout="column">
      <mat-form-field>
<input formControlName="email" name="Email" [(ngModel)]="Email" type="email" matInput placeholder="Пошта" required>
      </mat-form-field>
      <mat-form-field>
 <input formControlName="phone" name="Phone" [(ngModel)]="Phone" type="text" matInput placeholder="Номер телефону" required>
```

```
            </mat-form-field>
            <mat-form-field>
        <input formControlName="password" name="Password" [(ngModel)]="Password" type="password" matInput placeholder="Пароль" required>
            </mat-form-field>
            <a style="color: black;" routerLink="/login">Увійти</a>
          </mat-card-content>
          <mat-card-actions align="end">
        <button (click)="register()" mat-raised-button color="primary" [disabled]="!registerForm.valid">Реєстрація</button>
          </mat-card-actions>
        </form>
      </mat-card>
    </div>
```

При введенні даних на ній відбувається підключення до бази даних і валідація даних, чи користувач з такими даними вже не зареєстрований.

```
    <?php
      header('Access-Control-Allow-Origin: *');
      header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
      header('Access-Control-Allow-Headers: token, Content-Type');
      header('Content-Type: text/plain');
      include('db.php');
      $connect->set_charset("utf8");

        $postdata = file_get_contents('php://input');
        $request = json_decode($postdata);
        $phone = $request->phone;
        $email = $request->email;
        $password = $request->password;
        if(isset($phone) || isset($email) || isset($password))
        {
          $select = "SELECT id FROM users WHERE phone = '$phone' or email = '$email'";
          $result = $connect->query($select);
          if($result->num_rows > 0)
          {
            $message = "Користувач уже існує";
            $array = array("info" => $message);
            echo json_encode($array);
          }
          else
```

```
        {
            $create = "INSERT INTO users(phone, email, password, hash, time_hash) VALUES('$phone', '$email', '$password', '1', '0')";
            if($connect->query($create) === TRUE)
            {
                $select_2 = "SELECT id FROM users WHERE email = '$email' and password = '$password'";
                $result_2 = $connect->query($select_2);
                if($result_2->num_rows == 1)
                {
                        $row_2 = $result_2->fetch_assoc();
                        $id = $row_2['id'];
    $array = array("data" => $id, "phone" => $phone, "email" =>  $email);
                        echo json_encode($array);
                }
            }
            else
            {
                $message = $connect->error;
                $array = array("info" => "Сталася невідома помилка");
                echo json_encode($array);
            }
        }
    }
    else
    {
        $message = $connect->error;
        $array = array("info" => "Сталася невідома помилка");
        echo json_encode($array);
    }
?>
```

Після реєстрації користувач може ввійти під своїми даними на веб-ресурс на сторінці входу.

```
<div fxLayout="row" fxLayoutAlign="center center" class="login-main">
  <mat-card >
    <mat-card-header>
      <mat-card-title>Увійти</mat-card-title>
    </mat-card-header>
    <br>
    <form [formGroup]="loginForm">
    <mat-card-content fxLayout="column">
```

```
                <mat-form-field>
        <input formControlName="email" [(ngModel)]="Email" matInput placeholder="Пошта">
                </mat-form-field>
                <mat-form-field>
        <input formControlName="password" [(ngModel)]="Password" type="password" matInput placeholder="Пароль">
                </mat-form-field>
                <a style="color: black;" routerLink="/register">Зареєструватися</a>
              </mat-card-content>

                <mat-card-actions align="end">
        <button (click) = "login()" mat-raised-button [disabled]="!loginForm.valid" color="primary">Вхід</button>
                </mat-card-actions>
            </form>
            </mat-card>
        </div>
```

При введенні даних вони перевіряються в базі даних на відповідність, що дійсно користувач з такими обліковими даними існує.

```
        <?php
            header('Access-Control-Allow-Origin: *');
            header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
            header('Access-Control-Allow-Headers: token, Content-Type');
            header('Content-Type: text/plain');
            include('db.php');
            $connect->set_charset("utf8");
            $postdata = file_get_contents('php://input');
            $request = json_decode($postdata);
            $login = $request->login;
            $pass = $request->password;
            function generateRandomString($length) {
                return substr(str_shuffle(str_repeat($x='0123456789', ceil($length/strlen($x)) )),1,$length);
            }
            $time = time();
            $select = "SELECT*FROM users WHERE email = '$login' and password = '$pass'";
            $result = $connect->query($select);
            if($result->num_rows == 1 )
            {
```

```
            $hash = generateRandomString(6);
            $update = "UPDATE users SET hash = '$hash', time_hash = '$time' WHERE email =
'$login' and password = '$pass'";
            if($connect->query($update) === TRUE)
            {
               $row = $result->fetch_assoc();
               $id = $row['id'];
               $phone = $row['phone'];
               $email = $row['email'];
               $array = array("data" => $id, "phone" => $phone, "email" => $email, 'hash' =>
$hash);
               echo json_encode($array);
            }
         }
         else
         {
            $message = "error";
            $array = array("id" => $message);
            echo json_encode($array);
         }
      ?>
```

В разі введення коректних даних користувача, які існують в базі, він перенаправляється на сторінку введення коду. Також в цей самий момент генерується одноразовий пароль, який зберігається в базі даних і буде надісланий в чат-боті в Telegram.

```
<div fxLayout="row" fxLayoutAlign="center center" class="login-main">
   <mat-card >
      <mat-card-header>
         <mat-card-title>Увійти</mat-card-title>
      </mat-card-header>
      <br>
      <mat-card-content fxLayout="column">
         <mat-form-field>
               <input [(ngModel)]="Code" matInput placeholder="Код">
          </mat-form-field>
      </mat-card-content>
      <mat-card-actions align="end">
               <button (click) = "login()" mat-raised-button
color="primary">Вхід</button>
      </mat-card-actions>
   </mat-card>
```

```
</div>
```

Також в цей самий момент генерується одноразовий пароль, який зберігається в базі даних і буде надісланий в чат-боті в Telegram. Для цього відбувається підключення до чат-бота і передача йому одноразового коду.

```
<?php
   header('Access-Control-Allow-Origin: *');
   header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
   header('Access-Control-Allow-Headers: token, Content-Type');
   header('Content-Type: text/plain');
   include('db.php');
   $connect->set_charset("utf8");
      $postdata = file_get_contents('php://input');
      $request = json_decode($postdata);
      $phone = $request->phone;
      if(isset($phone))
      {
         $select = "SELECT*FROM users WHERE phone = '$phone'";
         $result = $connect->query($select);
         if($result->num_rows > 0)
         {
            $row = $result->fetch_assoc();
            $message = $row['hash'];
            $array = array("info" => $message);
            echo json_encode($array);
         }
      }
      else
      {
         $message = $connect->error;
         $array = array("info" => "Сталася невідома помилка");
         echo json_encode($array);
      }
?>
```

Боти в Telegram створюються за допомогою наявного і розробленого кампанією бота – BotFather, який відображений на рисунку 3.1. Робота починається з ведення команди /start.

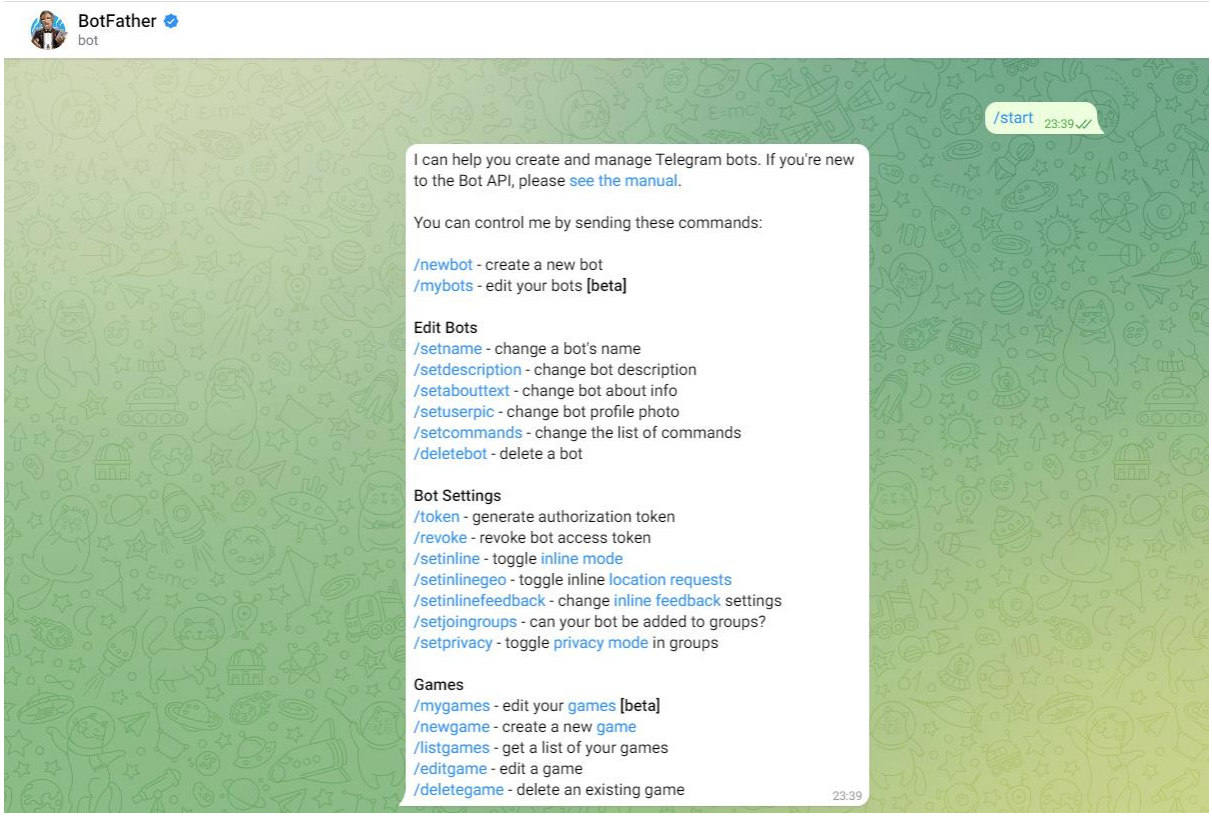

Рисунок 3.1 – BotFather

З переліку команд вибирається команда /newbot і створюється назва для бота, як показано на рисунку 3.2.

Рисунок 3.2 – Створення ного бота

Після цього буде згенерований токен для API, який надалі буде використовуватись при написанні структури самого бота. Бот найзручніше налаштовувати використовуючи мову програмування Python.

```
import telebot
from telebot import types
import requests
bot = telebot.TeleBot('5762963978:AAG63NbCAw5YJC6q680tn7eX0pdC4H9C054')
url = 'http://logikstart.s-host.net/server/checkBot.php'
```

```
@bot.message_handler(commands=['start'])
def register(message):
    keyboard = types.ReplyKeyboardMarkup(one_time_keyboard=True)
    reg_button = types.KeyboardButton(text="Авторизуватися", request_contact=True)
    keyboard.add(reg_button)
    response = bot.send_message(message.chat.id,
                                "Авторизуватися",
                                reply_markup=keyboard)
@bot.message_handler(content_types=['contact'])
def contact_handler(message):
    myobj = {'phone': message.contact.phone_number}
    x = requests.post(url, json = myobj)
    data = x.json()
    if data['info'] == 1:
        bot.send_message(message.chat.id,"Немає активних авторизацій!")
    else:
        bot.send_message(message.chat.id,"Код: "+data['info']
if __name__ == '__main__':
    bot.polling(none_stop=True)
```

Далі користувач вводить отриманий одноразовий код і відбувається порівняння введеного коду з тим, що зберігається в базі даних.

```
<?php
    header('Access-Control-Allow-Origin: *');
    header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
    header('Access-Control-Allow-Headers: token, Content-Type');
    header('Content-Type: text/plain');
    include('db.php');
    $connect->set_charset("utf8");
    $postdata = file_get_contents('php://input');
    $request = json_decode($postdata);
    $code = $request->code;
    $login = $request->login;
    $time = time();
    $select = "SELECT*FROM users WHERE hash = '$code'";
    $result = $connect->query($select);
    $row = $result->fetch_assoc();
    if($time - $row['time_hash'] < 180)
    {
        if($result->num_rows == 1 )
        {
```

```
            $update = "UPDATE users SET hash = '1' WHERE email = '$login'";
            if($connect->query($update) === TRUE)
            {
               $id = $row['id'];
               $phone = $row['phone'];
               $email = $row['email'];
               $array = array("id" => $id, "phone" => $phone, "email" => $email);
               echo json_encode($array);
            }
         }
         else
         {
            $message = "error";
            $array = array("id" => $message);
            echo json_encode($array);
         }
      }
      else
      {
         $message = "time";
         $array = array("id" => $message);
         echo json_encode($array);
      }
   ?>
```

Після введення пароля користувач перенаправляється на головну сторінку, де зможе повноцінно користуватись всіма можливостями авторизованого доступу.

```
<img mat-card-image src="https://material.angular.io/assets/img/examples/shiba2.jpg" alt="Photo of a Shiba Inu">
<mat-card-content>
  <p>
     Ви на головній сторінці!<br><br>
     Пошта: {{email}}<br>
     Телефон:{{phone}}
  </p>
</mat-card-content>
<mat-card-actions>
  <button mat-button>Підтримати ЗСУ</button>
  <button (click)="exit()" mat-raised-button color="warn">Вийти</button>
</mat-card-actions>
</mat-card>
```

* *

3.3 Гайд для користувача

Користувач потрапляючи на сайт стикається з формою для входу в особистий кабінет. Для того, щоб отримати доступ до всього функціоналу ресурсу. Якщо користувач вперше на сайті, він має змогу зареєструватись.

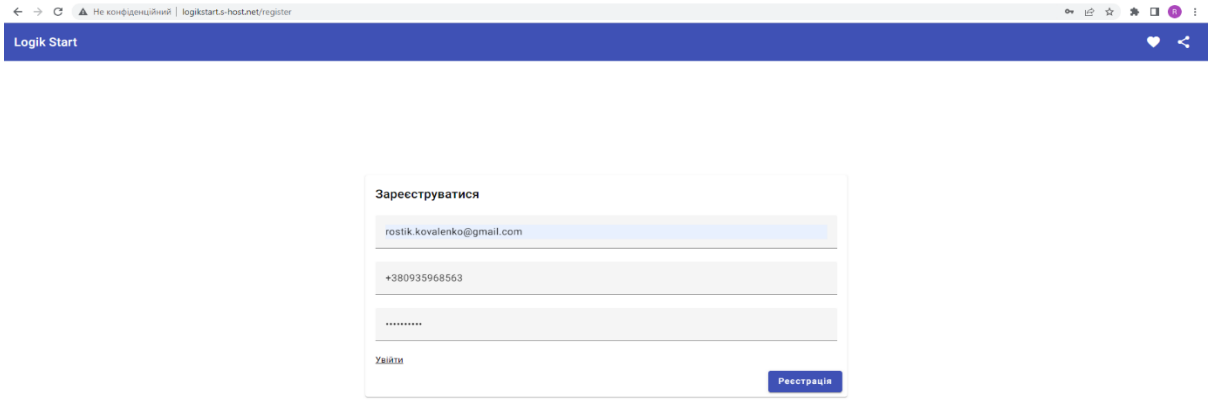

Рисунок 3.3 – Форма реєстрації

Після успішної реєстрації на сайті, користувач може входити в особистий кабінет, використовуючи дані вказані при реєстрації.

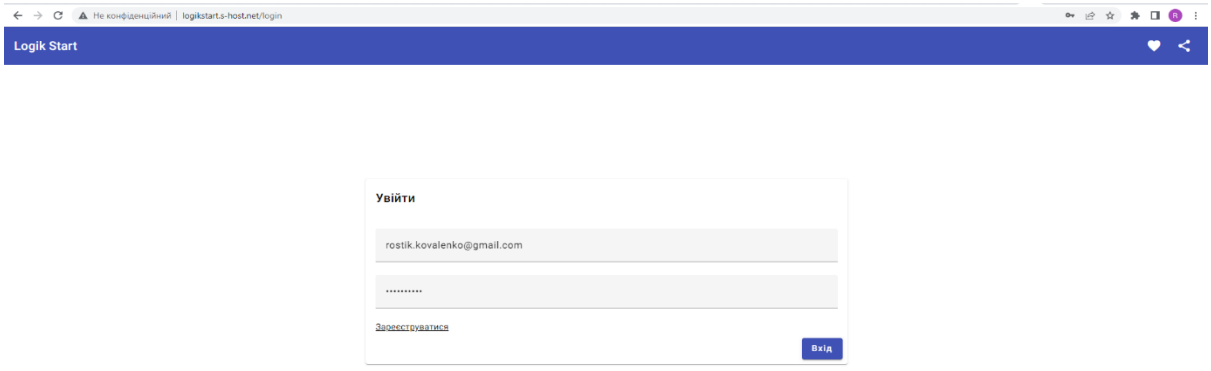

Рисунок 3.4 – Форма входу

У випадку введення коректних даних система перенаправляє користувача на сторінку введення коду, який генерується і зберігається в базі даних.

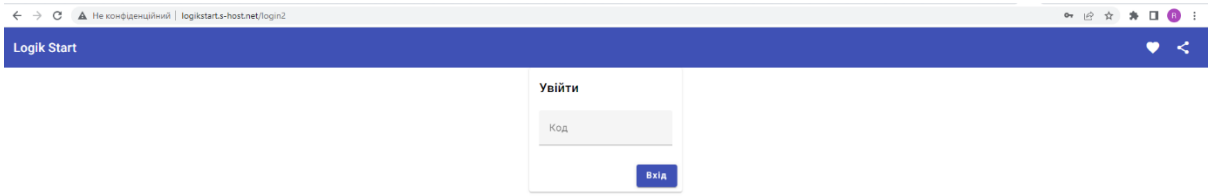

Рисунок 3.5 – Форма введення коду

Користувач повинен відкрити Telegram (на телефоні, веб- чи десктоп-версію) і в пошуку знайти бот, у нашому випадку він називається LogikStart. Прописуємо команду /start і авторизуємось в боті, надавши йому доступ до номера телефону, який був вказаний при реєстрації. У випадку якщо для цього номеру є активна спроба входу на цей момент, то буде відображений код.

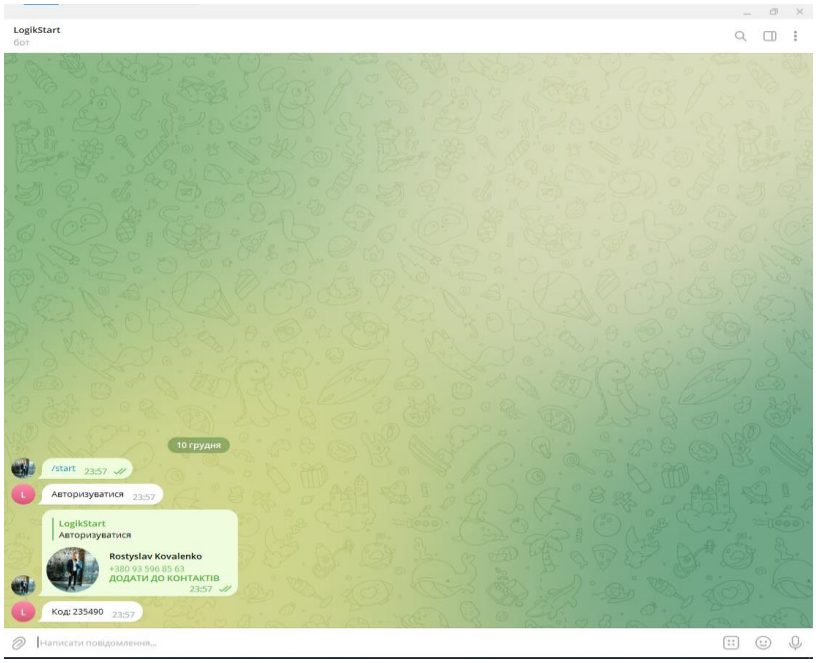

Рисунок 3.6 – Робота з ботом

Вводимо код на сторінці для автентифікації. У випадку коректно введеного коду, автентифікація користувача проходить успішно і він буде авторизований на сайті.

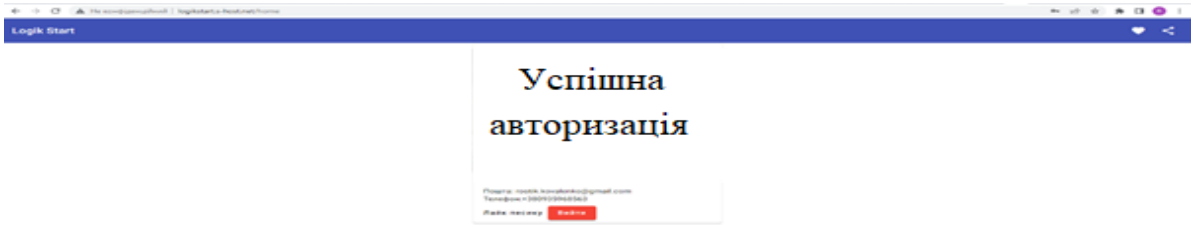

Рисунок 3.7 – Успішна авторизація

ВИСНОВКИ

Забезпечення безпеки інформації та даних користувачів залишається найбільш актуальним питанням веб-безпеки у наш час. Саме тому тисячі людей кожного дня розробляють системи, які допомагають користувачам почуватись в безпеці, а організаціям захищати свої мережі.

Результатом виконаної роботи є робоча модель захисту веб-ресурсів за допомогою інформаційної технології двофакторної авторизації за допомогою Telegram. Дана технологія забезпечує захист користувачів від викрадення їх особистих даних, а організацій – від ризиків несанкціонованого доступу зловмисниками з метою нанесення негативних наслідків. Двофакторна авторизація представлена перевіркою даних користувача (логін та пароль) та введенням підтвердження в Чат-Боті в Telegram.

Робоча модель може бути використана будь-яким веб-ресурсом для покращення своєї безпеки без особливих складностей та фінансових затрат. Саме використання Чат-Бота в Telegram є оптимальним рішенням, оскільки:

- Telegram використовує кодування даних в запитах;
- У нього є десктоп-, веб- та мобільна версія, що надає змогу підтвердити (або заблокувати) доступ навіть при втраті мобільного телефону;
- Використання Чат-Бота в Telegram є повністю безплатним.

Під час роботи над проектом, були застосовані практично такі технології, як двофакторна авторизація за допомогою Telegram, фреймворк Angular, мова програмування Python, скриптова мова загального призначення PHP, мова гіпертекстової розмітки HTML, каскадна таблиця стилів CSS.

СПИСОК ЛІТЕРАТУРИ

1. Вікіпедія – веб-ресурс. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Web_resource

2. Як працює Веб. [Електронний ресурс] – Режим доступу: https://dynamic-design.com.ua/novosti/uk/brauzer-ak-pracue-veb/

3. MDN web docs - How the web works. [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

4. ITBIZ – Захист веб-додатків: чому це важливо? [Електронний ресурс] – Режим доступу: https://itbiz.ua/statti-ta-obzori/zaxist-veb-dodatkiv-chomu-ce-vazhlivo/

5. IBM Security - X-Force Threat Intelligence Index 2022. [Електронний ресурс] – Режим доступу: https://www.ibm.com/downloads/cas/ADLMYLAZ

6. Chapmen N., Chapman J. Authentication and Authorization on the Web, 2012. – 246 p

7. Federal Trade Commision – Equifax Data Breach Settlment. [Електронний ресурс] – Режим доступу: https://www.ftc.gov/enforcement/refunds/equifax-data-breach-settlement

8. Acemyan Claudia, Philip Kortum – 2FA Might be secure, but it`s not usable: A summative usability assessment of Google`s Two-Factor Authentication methods, 2018

9. The New York Times: Wirecutter – The best two-factor Authentication App. [Електронний ресурс] – Режим доступу: https://www.nytimes.com/wirecutter/reviews/best-two-factor-authentication-app/

10. Telegram APIs. [Електронний ресурс] – Режим доступу: *core.telegram.org*.

11. React vs. Angular Framework Kakumani Eswar akanksh and Yadam Mohan sai Poleboina Naga Siva and Dangeti Chaitanya Sai and Thupakula Hemanth Goud. [Електронний ресурс] – Режим доступу: https://ijcrt.org/papers/IJCRT2205332.pdf

12. Фреймворк Angular. [Електронний ресурс] – Режим доступу: https://angular.io/

13. David R. Brooks – Programming in HTML and PHP, Undergraduate Topics in Computer Science, 2017

ДОДАТОК

checkBot.php

```
<?php
   header('Access-Control-Allow-Origin: *');
   header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
   header('Access-Control-Allow-Headers: token, Content-Type');
   header('Content-Type: text/plain');
   include('db.php');
   $connect->set_charset("utf8");
      $postdata = file_get_contents('php://input');
      $request = json_decode($postdata);
      $phone = $request->phone;
      if(isset($phone))
      {
         $select = "SELECT*FROM users WHERE phone = '$phone'";
         $result = $connect->query($select);
         if($result->num_rows > 0)
         {
            $row = $result->fetch_assoc();
            $message = $row['hash'];
            $array = array("info" => $message);
            echo json_encode($array);
         }
      }
      else
      {
         $message = $connect->error;
         $array = array("info" => "Сталася невідома помилка");
         echo json_encode($array);
      }
?>
```

checkCode.php

```
<?php
   header('Access-Control-Allow-Origin: *');
```

```
header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
header('Access-Control-Allow-Headers: token, Content-Type');
header('Content-Type: text/plain');
include('db.php');
$connect->set_charset("utf8");
$postdata = file_get_contents('php://input');
$request = json_decode($postdata);
$code = $request->code;
$login = $request->login;
$time = time();
$select = "SELECT*FROM users WHERE hash = '$code'";
$result = $connect->query($select);
$row = $result->fetch_assoc();
if($time - $row['time_hash'] < 180)
{
   if($result->num_rows == 1 )
   {
      $update = "UPDATE users SET hash = '1' WHERE email = '$login'";
      if($connect->query($update) === TRUE)
      {
         $id = $row['id'];
         $phone = $row['phone'];
         $email = $row['email'];
         $array = array("id" => $id, "phone" => $phone, "email" => $email);
         echo json_encode($array);
      }
   }
   else
   {
      $message = "error";
      $array = array("id" => $message);
      echo json_encode($array);
   }
}
else
```

```
        {
            $message = "time";
            $array = array("id" => $message);
            echo json_encode($array);
        }
    ?>
```

login.php

```
    <?php
        header('Access-Control-Allow-Origin: *');
        header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT, PATCH, OPTIONS');
        header('Access-Control-Allow-Headers: token, Content-Type');
        header('Content-Type: text/plain');
        include('db.php');
        $connect->set_charset("utf8");
        $postdata = file_get_contents('php://input');
        $request = json_decode($postdata);
        $login = $request->login;
        $pass = $request->password;
        function generateRandomString($length) {
            return substr(str_shuffle(str_repeat($x='0123456789', ceil($length/strlen($x)) )),1,$length);
        }
       $time = time();
        $select = "SELECT*FROM users WHERE email = '$login' and password = '$pass'";
        $result = $connect->query($select);
        if($result->num_rows == 1 )
        {
            $hash = generateRandomString(6);
            $update = "UPDATE users SET hash = '$hash', time_hash = '$time' WHERE email = '$login' and password = '$pass'";
            if($connect->query($update) === TRUE)
            {
                $row = $result->fetch_assoc();
                $id = $row['id'];
```

```
                $phone = $row['phone'];
                $email = $row['email'];
                $array = array("data" => $id, "phone" => $phone, "email" => $email,
'hash' => $hash);
                echo json_encode($array);
            }
        }
        else
        {
            $message = "error";
            $array = array("id" => $message);
            echo json_encode($array);
        }
    ?>
```

register.php

```
    <?php
        header('Access-Control-Allow-Origin: *');
        header('Access-Control-Allow-Methods: POST, GET, DELETE, PUT,
PATCH, OPTIONS');
        header('Access-Control-Allow-Headers: token, Content-Type');
        header('Content-Type: text/plain');
        include('db.php');
            $connect->set_charset("utf8");
            $postdata = file_get_contents('php://input');
            $request = json_decode($postdata);
            $phone = $request->phone;
            $email = $request->email;
            $password = $request->password;
            if(isset($phone) || isset($email) || isset($password))
            {
                $select = "SELECT id FROM users WHERE phone = '$phone' or email
= '$email'";
                $result = $connect->query($select);
                if($result->num_rows > 0)
                {
                    $message = "Користувач уже існує";
```

```
                $array = array("info" => $message);
                echo json_encode($array);
            }
            else
            {
                $create = "INSERT INTO users(phone, email, password, hash, time_hash) VALUES('$phone', '$email', '$password', '1', '0')";
                if($connect->query($create) === TRUE)
                {
                    $select_2 = "SELECT id FROM users WHERE email = '$email' and password = '$password'";
                    $result_2 = $connect->query($select_2);

                    if($result_2->num_rows == 1)
                    {
                        $row_2 = $result_2->fetch_assoc();
                        $id = $row_2['id'];
                        $array = array("data" => $id, "phone" => $phone, "email" => $email);

                        echo json_encode($array);
                    }
                }
                else
                {
                    $message = $connect->error;
                    $array = array("info" => "Сталася невідома помилка");
                    echo json_encode($array);
                }
            }
        }
        else
        {
            $message = $connect->error;
            $array = array("info" => "Сталася невідома помилка");
            echo json_encode($array);
        }
```

```
?>
```

bot.py

```
import telebot
from telebot import types
import requests
bot = telebot.TeleBot('5762963978:AAG63NbCAw5YJC6q680tn7eX0pdC4H9C054')
url = 'http://logikstart.s-host.net/server/checkBot.php'
@bot.message_handler(commands=['start'])
def register(message):
    keyboard = types.ReplyKeyboardMarkup(one_time_keyboard=True)
    reg_button = types.KeyboardButton(text="Авторизуватися", request_contact=True)
    keyboard.add(reg_button)
    response = bot.send_message(message.chat.id,
                                "Авторизуватися",
                                reply_markup=keyboard)
@bot.message_handler(content_types=['contact'])
def contact_handler(message):
    myobj = {'phone': message.contact.phone_number}
    x = requests.post(url, json = myobj)
    data = x.json()
    if data['info'] == 1:
        bot.send_message(message.chat.id,"Немає активних авторизацій!")
    else:
        bot.send_message(message.chat.id,"Код: "+data['info'])
if __name__ == '__main__':
    bot.polling(none_stop=True)
```

home.component.html

```
  <div style=" margin: auto;">
<mat-card class="example-card" style=" margin: auto;">
    <mat-card-header>

    </mat-card-header>
```

```
        <img                                                    mat-card-image
src="https://material.angular.io/assets/img/examples/shiba2.jpg" alt="Photo of a Shiba
Inu">
        <mat-card-content>
          <p>
            Ви на головній сторінці!<br><br>
            Пошта: {{email}}<br>
            Телефон:{{phone}}
          </p>
        </mat-card-content>
        <mat-card-actions>
          <button mat-button>Підтримати ЗСУ</button>
          <button (click)="exit()" mat-raised-button color="warn">Вийти</button>
        </mat-card-actions>
      </mat-card>
    </div>
```

home.component.scss

```
    .example-card {
      max-width: 400px;
    }
    .example-header-image {
      background-image:
url('https://material.angular.io/assets/img/examples/shiba1.jpg');
      background-size: cover;
    }
```

login.component.html

```
    <div fxLayout="row" fxLayoutAlign="center center" class="login-main">
      <mat-card >
        <mat-card-header>
          <mat-card-title>Увійти</mat-card-title>
        </mat-card-header>
        <br>
        <form [formGroup]="loginForm">
        <mat-card-content fxLayout="column">
          <mat-form-field>
```

```
                <input formControlName="email" [(ngModel)]="Email" matInput placeholder="Пошта">
                </mat-form-field>
                <mat-form-field>
                <input formControlName="password" [(ngModel)]="Password" type="password" matInput placeholder="Пароль">
                </mat-form-field>
                <a style="color: black;" routerLink="/register">Зареєструватися</a>
            </mat-card-content>
            <mat-card-actions align="end">
                <button (click) = "login()" mat-raised-button [disabled]="!loginForm.valid" color="primary">Вхід</button>
            </mat-card-actions>
        </form>
        </mat-card>
    </div>
```

login.component.scss

```
    margin-top: 10%;
    }
    mat-card{
    min-width: 40%;
    }
```

login2.component.html

```
        <mat-card-header>
            <mat-card-title>Увійти</mat-card-title>
        </mat-card-header>
        <br>
        <mat-card-content fxLayout="column">
            <mat-form-field>
                <input [(ngModel)]="Code" matInput placeholder="Код">
            </mat-form-field>
        </mat-card-content>
        <mat-card-actions align="end">
            <button (click) = "login()" mat-raised-button color="primary">Вхід</button>
        </mat-card-actions>
```

```
    </mat-card>
  </div>
```

register.component.html

```
  <div fxLayout="row" fxLayoutAlign="center center" class="login-main">
    <mat-card >
      <form [formGroup]="registerForm">
      <mat-card-header>
        <mat-card-title>Зареєструватися</mat-card-title>
      </mat-card-header>
      <br>
      <mat-card-content fxLayout="column">

        <mat-form-field>
          <input formControlName="email" name="Email" [(ngModel)]="Email" type="email" matInput placeholder="Пошта" required>
        </mat-form-field>
        <mat-form-field>
          <input formControlName="phone" name="Phone" [(ngModel)]="Phone" type="text" matInput placeholder="Номер телефону" required>
        </mat-form-field>
        <mat-form-field>
          <input formControlName="password" name="Password" [(ngModel)]="Password" type="password" matInput placeholder="Пароль" required>
        </mat-form-field>
        <a style="color: black;" routerLink="/login">Увійти</a>
      </mat-card-content>
      <mat-card-actions align="end">
        <button (click)="register()" mat-raised-button color="primary" [disabled]="!registerForm.valid">Реєстрація</button>
      </mat-card-actions>
     </form>
    </mat-card>
  </div>
```

register.component.scss

```
.login-main{
    margin-top: 10%;
    }
    mat-card{
    min-width: 40%;
    }
```