

Politechnika Świętokrzyska

Wydział Elektrotechniki, Automatyki i Informatyki

Studia stacjonarne II stopnia

Bezpieczeństwo aplikacji internetowych

Temat projektu:

Aplikacja do wspomagania funkcjonowania biblioteki.

Grupa 1ID24B

Borcuch Mateusz
Kwiatkowska Aleksandra
Sikora Wiktoria

Spis treści

1. Wstęp	3
2. Wykorzystane technologie oraz narzędzia	3
2.1. Python.....	3
2.2. Flask.....	3
2.3. HTML.....	4
2.4. Style zewnętrzne CSS	4
2.5. MySQL	5
2.6. Środowisko Visual Studio Code	5
2.7. Narzędzie XAMPP	5
3. Implementacja	6
3.1. Implementacja backend'u	6
3.2. Widoki	21
3.3. Zabezpieczenia przed podatnościami	27
4. Testy aplikacji	28
5. Podział pracy.....	35
6. Podsumowanie	35
7. Spis listingów.....	36
8. Spis rysunków	37

1. Wstęp

Dokumentacja ta ma na celu przedstawienie szczegółowych informacji dotyczących naszej aplikacji webowej do zarządzania biblioteką. Będzie ona stanowić przydatne źródło informacji dla administratorów, pracowników biblioteki oraz innych użytkowników zainteresowanych korzystaniem z naszego oprogramowania. Nasze oprogramowanie zostało stworzone z myślą o ułatwieniu codziennej pracy w bibliotece.

2. Wykorzystane technologie oraz narzędzia

W rozdziale drugim skupimy się na kluczowych elementach technologicznych, które stanowią fundament naszego projektu. Poczynając od języka programowania Python oraz framework Flask. Rozdział ten poświęcony będzie również językowi znaczników HTML oraz zewnętrznym arkuszom stylów CSS, które wspólnie tworzą fundament struktury i wyglądu naszej aplikacji. W dalszej części skoncentrujemy się na bazie danych MySQL, środowisku programistycznym Visual Studio Code oraz narzędziu XAMPP.

2.1. Python

Python to wszechstronny i czytelny język programowania ogólnego przeznaczenia, który został stworzony w 1991 roku przez Guido van Rossuma. Jego składnia jest znana ze swojej czytelności, co ułatwia naukę i zrozumienie kodu. Język ten jest wieloparadygmatowy, co oznacza, że wspiera różne podejścia programistyczne, takie jak programowanie obiektowe, proceduralne i funkcyjne. Python charakteryzuje się dynamicznym typowaniem, co pozwala na elastyczne przypisywanie typów danych w trakcie działania programu. Posiada również obszerną bibliotekę standardową, która oferuje gotowe moduły do wielu zastosowań, co ułatwia tworzenie oprogramowania bez konieczności implementacji wszystkiego od zera.

Python jest platformnie niezależny, co oznacza, że kod napisany w tym języku może być uruchamiany na różnych systemach operacyjnych. Znajduje szerokie zastosowanie w dziedzinach takich jak rozwój webowy, analiza danych, sztuczna inteligencja, uczenie maszynowe, automatyzacja, administracja systemem, a także w tworzeniu gier komputerowych.

2.2. Flask

Flask to mikroframework do tworzenia aplikacji webowych napisany w języku Python. Jest określany jako mikro-framework, ponieważ nie narzuca konkretnych narzędzi czy bibliotek, co daje programiście dużą swobodę w wyborze komponentów do swojego projektu. Co ważne, Flask nie dostarcza warstwy abstrakcji dla funkcji takich jak obsługa baz danych czy walidacja

formularzy, w przeciwieństwie do niektórych innych frameworków, które korzystają z istniejących bibliotek stron trzecich.

Mimo braku tych wbudowanych funkcji, Flask pozwala na rozszerzenia, które umożliwiają dodawanie funkcji do aplikacji, tak jakby były one integralną częścią frameworka. Istnieją rozszerzenia obsługujące mapowanie obiektowo-relacyjne, walidację formularzy, obsługę przesyłania plików, różne metody uwierzytelniania i wiele innych, co pozwala dostosować framework do konkretnych potrzeb projektu.

2.3. HTML

HTML (HyperText Markup Language) jest językiem znaczników używanym do strukturyzowania treści na stronach internetowych. Składa się z różnych znaczników, które definiują elementy na stronie, takie jak nagłówki, paragrafy, listy, linki, obrazy i formularze. Każdy dokument HTML zaczyna się od znacznika `<!DOCTYPE html>` i zawiera główny znacznik `<html>`, obejmujący treść strony. Znaczniki mogą zawierać atrybuty, dostarczające dodatkowych informacji o elemencie, i mogą być zagnieżdżane, co oznacza, że jeden element może zawierać inne elementy. HTML wprowadza semantykę, co oznacza, że znaczniki są używane do określania znaczenia zawartości, na przykład znaczniki nagłówków, akapitów, list, itp. Obecnie najnowszą wersją HTML jest HTML5, wprowadzając nowe elementy, atrybuty i funkcje, zwiększając możliwości tworzenia interaktywnych stron internetowych.

2.4. Style zewnętrzne CSS

Style zewnętrzne w CSS to technika, która pozwala na oddzielenie definicji stylów od treści strony HTML. Zamiast umieszczać style bezpośrednio w kodzie HTML, są one przechowywane w osobnym pliku CSS. Kluczowym elementem tego podejścia jest wykorzystanie tagu `<link>` w sekcji `<head>` strony HTML, aby podłączyć zewnętrzny arkusz stylów.

Zalety korzystania z zewnętrznych stylów CSS obejmują zwiezłość, czytelność kodu oraz możliwość ponownego użycia tych samych stylów na wielu stronach. Dodatkowo, gdy zachodzi potrzeba aktualizacji wizualnej strony, zmiany można wprowadzać bezpośrednio w pliku CSS, co ułatwia zarządzanie i utrzymanie spójności wyglądu witryny.

W praktyce, plik CSS zewnętrzny zawiera definicje stylów dla różnych elementów strony, takich jak tło, czcionka, marginesy czy kolor tekstu. Podłączenie tego pliku do strony HTML umożliwia jednolite zastosowanie stylów na wszystkich stronach, co przyczynia się do spójnego i profesjonalnego wyglądu całej witryny.

2.5. MySQL

MySQL to system zarządzania relacyjnymi bazami danych (RDBMS), stworzony pierwotnie przez firmę MySQL AB i obecnie rozwijany przez Oracle Corporation. Jest jednym z najbardziej popularnych systemów baz danych na świecie i szeroko stosowany w aplikacjach internetowych, systemach zarządzania treścią, e-commerce i innych dziedzinach. MySQL opiera się na relacyjnej strukturze danych, gromadząc dane w tabelach i ustanawiając relacje między nimi za pomocą kluczy obcych. Język zapytań SQL jest używany do manipulowania danymi w bazie, obejmując operacje takie jak pobieranie, aktualizacja, wstawianie i usuwanie danych.

System jest wieloplatformowy, co oznacza, że działa na różnych systemach operacyjnych, takich jak Windows, Linux i macOS. Ponadto, MySQL jest darmowy i dostępny na zasadach licencji open source, co umożliwia użytkownikom korzystanie, modyfikowanie i dystrybuowanie go bez kosztów licencji. MySQL obsługuje różne typy danych, indeksowanie, transakcje oraz oferuje zaawansowane funkcje, w tym triggerzy i procedury składowane. Dzięki wszechstronności, jest popularny w projektach o zróżnicowanych wymaganiach. Dodatkowo, MySQL umożliwia równoczesne połączenia wielu użytkowników, co jest istotne w aplikacjach, gdzie wymagana jest współpraca wielu osób na bazie danych. Ogólnie rzecz biorąc, MySQL jest często wybierany jako backend dla dynamicznych stron internetowych i aplikacji, ze względu na swoją niezawodność, wydajność i prostotę użycia.

2.6. Środowisko Visual Studio Code

Visual Studio Code, często skracane do VS Code, to bezpłatne i lekkie środowisko programistyczne stworzone przez Microsoft. Zostało zoptymalizowane dla efektywnej pracy programistów, oferując szeroką gamę funkcji i narzędzi. Kluczowe aspekty tego środowiska obejmują: bezpłatność i otwartość źródła, wsparcie dla wielu języków programowania, rozszerzenia, integrację z systemami kontroli wersji, możliwość debugowania, inteligentne uzupełnianie kodu, obsługę wielu systemów operacyjnych, a także aktywną społeczność użytkowników i dostępność licznych zasobów, tutoriali i rozszerzeń. W skrócie, Visual Studio Code to zaawansowane, intuicyjne i elastyczne środowisko programistyczne, które zdobyło uznanie ze względu na swoje możliwości i łatwość użycia.

2.7. Narzędzie XAMPP

XAMPP to darmowy, otwartoźródłowy zestaw oprogramowania, który umożliwia łatwą instalację i konfigurację środowiska do lokalnego hostowania stron internetowych. Nazwa "XAMPP" pochodzi od skrótu pierwszych liter używanych komponentów w tym zestawie:

- X (Cross-platform): Oznacza, że XAMPP jest dostępny na różnych platformach, takich jak Windows, Linux, Mac itp.
- A (Apache): Serwer HTTP Apache, popularny serwer internetowy, który obsługuje protokół HTTP.
- M (MySQL): System zarządzania bazą danych MySQL, który umożliwia przechowywanie i zarządzanie danymi.
- P (PHP): Język skryptowy PHP, używany często do programowania stron internetowych. XAMPP zawiera także Perl.
- P (Perl): Język programowania Perl, który jest także skryptowym językiem programowania.

XAMPP jest używany głównie do celów developerskich i testowych. Zapewnia gotowe środowisko do uruchamiania i testowania skryptów PHP, aplikacji internetowych opartych na MySQL, oraz innych projektów webowych lokalnie na komputerze, zanim zostaną one wdrożone na serwerze internetowym. Dzięki temu programiści mogą pracować nad swoimi projektami bez konieczności dostępu do rzeczywistego serwera internetowego.

3. Implementacja

3.1. Implementacja backend'u

Poniżej znajdują się najważniejsze fragmenty aplikacji.

```
@app.route('/register', methods=['POST'])
def register():
    session.pop('logged_in', None)
    session.pop('verified', None)
    imie = escape(request.form['imie'])
    nazwisko = escape(request.form['nazwisko'])
    numer_telefonu = escape(request.form['numer_telefonu'])
    email = escape(request.form['email'])
    nazwa_uzytkownika = escape(request.form['nazwa_uzytkownika'])
    haslo = escape(request.form['haslo'])

    error = None

    if not validate_name(imie) or not validate_name(nazwisko):
        error = "Niepoprawne imię lub nazwisko!"

    if not validate_phone_number(numer_telefonu):
        error = "Niepoprawny numer telefonu!"
```

```

if not validate_email(email):
    error = "Niepoprawny adres email!"

if not validate_password(haslo):
    error = "Niepoprawne hasło! Powinno posiadać conajmniej 8 znaków, w
tym przynajmniej jedną małą literę, dużą literę oraz cyfrę."

if error:
    return render_template('register.html', error=error)

salt = os.urandom(32)
hash = hashlib.pbkdf2_hmac("sha256", haslo.encode('utf-8'), salt, 10000)
hexhash = (salt + hash).hex()

cursor = mydb.cursor()
query = "SELECT * FROM uzytkownicy WHERE nazwa_uzytkownika = %s OR email
= %s"
values = (nazwa_uzytkownika, email)
cursor.execute(query, values)
result = cursor.fetchone()

if result:
    return render_template('register.html', error="Użytkownik o podanym
loginie lub emailu już istnieje!")
else:
    query = "INSERT INTO uzytkownicy (imie, nazwisko, numer_telefonu,
email, nazwa_uzytkownika, haslo) VALUES (%s, %s, %s,%s, %s, %s)"
    values = (imie, nazwisko, numer_telefonu, email, nazwa_uzytkownika,
hexhash)
    cursor.execute(query, values)
    mydb.commit()

    return redirect(url_for('login'))

```

Listing 3.1. Rejestracja.

Kod (listing 3.1) definiuje trasę `"/register"` obsługującą żądania POST. Na początku usuwane są zmienne sesyjne. Następnie pobierane są dane z formularza rejestracyjnego, takie jak imię, nazwisko, numer telefonu, email, nazwa użytkownika i hasło. Następnie są walidowane. Jeśli dane są niepoprawne, ustawia zmienną `error` z odpowiednim komunikatem. Kolejno jest hashowane hasło. Generuje unikalną sól, hashowanie hasła i zapisuje wynik w formie szesnastkowego ciągu znaków. Sprawdzana jest baza danych w celu istnienia użytkownika o podanej nazwie użytkownika lub adresie email. Jeśli użytkownik nie istnieje, dodawany jest do bazy danych. Po pomyślnej rejestracji następuje przekierowanie do strony logowania.

```

@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    session.pop('verified', None)
    session.pop('logged_in', None)

    if request.method == 'POST':
        nazwa_uzytkownika = escape(request.form['nazwa_uzytkownika'])
        haslo = escape(request.form['haslo'])

        cursor = mydb.cursor()
        query = "SELECT * FROM uzytkownicy WHERE nazwa_uzytkownika = %s"
        values = (nazwa_uzytkownika,)
        cursor.execute(query, values)
        result = cursor.fetchone()

        if result:
            if is_account_blocked(cursor, nazwa_uzytkownika):
                error = "Twoje konto jest zablokowane. Spróbuj ponownie za
kilka minut."
            else:
                stored_hash = result[6]
                salt = bytes.fromhex(stored_hash[:64])
                expected_hash = stored_hash[64:]

                hash = hashlib.pbkdf2_hmac("sha256", haslo.encode('utf-8'),
salt, 10000)

                hexhash = hash.hex()

                if hexhash == expected_hash:
                    session['logged_in'] = True
                    session['nazwa_uzytkownika'] = nazwa_uzytkownika

                    reset_login_attempts(cursor, nazwa_uzytkownika)

                    cursor.execute("SELECT email FROM uzytkownicy WHERE
nazwa_uzytkownika = %s", (nazwa_uzytkownika,))
                    user = cursor.fetchone()

                    totp = pyotp.TOTP("tajnehaslotajnehaslo", interval=30)
                    pyotp_code = totp.now()

                    session['verification_code'] = {
                        'code': pyotp_code,
                        'expiration_time': time.time() + 30
                    }

                    query = "UPDATE uzytkownicy SET verification_code = %s
WHERE nazwa_uzytkownika = %s"

```



```

        values = (pyotp_code, nazwa_uzytkownika)
        cursor.execute(query, values)
        mydb.commit()

        sender_email = "kiegarnia8@gmail.com"
        receiver_email = user[0]
        password_email = "sfhxleosktquvdzx"
        message = MIMEMultipart()
        message["From"] = sender_email
        message["To"] = receiver_email
        message["Subject"] = "Twój kod weryfikacyjny"
        body = f"Kod: {pyotp_code}"
        message.attach(MIMEText(body, "plain"))
        with smtplib.SMTP("smtp.gmail.com", 587) as server:
            server.starttls()
            server.login(sender_email, password_email)
            text = message.as_string()
            server.sendmail(sender_email, receiver_email, text)

        return redirect(url_for('weryfikacja'))
    else:
        blocked_until = increase_login_attempts(cursor,
nazwa_uzytkownika)
        if blocked_until:
            error = f"Nieprawidłowe dane logowania. Twoje konto
zostało zablokowane."
        else:
            error = "Dane są nieprawidłowe, spróbuj jeszcze
raz."

    else:
        error = "Dane są nieprawidłowe, spróbuj jeszcze raz."

    else:
        session.pop('logged_in', None)
        session.pop('nazwa_uzytkownika', None)

    return render_template('login.html', error=error)

```

Listing 3.2. Logowanie.

Kod (listing 3.2) definiuje trasę '/login' dla żądań GET i POST. Na początku usuwane są zmienne sesyjne 'verified' i 'logged_in'. W przypadku żądania POST pobierane są dane z formularza logowania (nazwa użytkownika i hasło). Następnie sprawdzane jest istnienie użytkownika w bazie danych. Jeśli użytkownik istnieje, kod sprawdza, czy konto nie jest zablokowane. Jeśli nie jest zablokowane, porównuje hasło wprowadzone przez użytkownika z hasłem z bazy danych. Jeśli hasło jest poprawne, ustawia zmienną sesyjną 'logged_in' na True,

resetuje liczbę nieudanych prób logowania, generuje kod weryfikacyjny TOTP, aktualizuje go w bazie danych i wysyła na email użytkownika. Następnie przekierowuje użytkownika do strony weryfikacji. Jeśli hasło jest nieprawidłowe, zwiększa liczbę nieudanych prób logowania, a w przypadku zablokowania konta informuje użytkownika. Jeśli użytkownik nie istnieje, również informuje o błędnych danych logowania. W przypadku żądania GET, kod obsługuje stronę logowania poprzez usuwanie zmiennych sesyjnych 'logged_in' i 'nazwa_uzytkownika'. Ostatecznie, szablon 'login.html' jest renderowany z ewentualnym komunikatem błędu.

```
@app.route('/weryfikacja', methods=['GET', 'POST'])
def weryfikacja():
    error = None
    if not session.get('logged_in'):
        return redirect(url_for('login'))
    if request.method == 'POST':
        kod_weryfikacyjny = escape(request.form['kod_weryfikacyjny'])
        nazwa_uzytkownika = session.get('nazwa_uzytkownika')

        stored_verification_code = session.get('verification_code')

        if stored_verification_code and time.time() <
stored_verification_code['expiration_time']:
            if kod_weryfikacyjny == stored_verification_code['code']:
                session['verified'] = True
                return redirect(url_for('strona_glowna'))
            else:
                error = "Nieprawidłowy kod weryfikacyjny!"
        else:
            error = "Błąd weryfikacji lub kod wygasł!"

    return render_template('weryfikacja.html', error=error)
```

Listing 3.3. Weryfikacja.

Kod (listing 3.3) definiuje trasę '/weryfikacja' dla żądań GET i POST. Sprawdza, czy użytkownik jest zalogowany, a następnie, w przypadku żądania POST, pobiera kod weryfikacyjny z formularza i nazwę użytkownika z sesji. Następnie sprawdza, czy wprowadzony kod zgadza się z przechowywanym w sesji oraz czy kod nie wygasł. Jeśli kod jest poprawny i nie wygasł, oznacza sesję jako zweryfikowaną i przekierowuje użytkownika na stronę główną. W przypadku błędnego kodu lub wygaśnięcia, ustawia komunikat błędu i renderuje szablon 'weryfikacja.html'.

```
@app.route('/change_pass', methods=['GET', 'POST'])
def change_pass():
    session.pop('logged_in', None)
```

```

session.pop('verified', None)
error = None

if request.method == 'POST':
    email = escape(request.form['email'])

    if not validate_email(email):
        error = "Niepoprawny adres email!"
    if error:
        return render_template('change_pass.html', error=error)

    cursor = mydb.cursor()
    cursor.execute("SELECT COUNT(*) FROM uzytkownicy WHERE email = %s",
(email,))
    result = cursor.fetchone()

    if result[0] > 0:
        haslo_gen = ''.join(random.choices(string.ascii_letters +
string.digits, k=8))
        if any(c.islower() for c in haslo_gen) and any(c.isupper() for c
in haslo_gen) and any(c.isdigit() for c in haslo_gen):
            sender_email = "kiegarnia8@gmail.com"
            receiver_email = email
            password_email = "sfhxleosktquvdzx"
            message = MIMEMultipart()
            message["From"] = sender_email
            message["To"] = receiver_email
            message["Subject"] = "Twoje nowe hasło"
            body = f"Hasło: {haslo_gen}"
            body_bytes = body.encode('utf-8')
            message.attach(MIMEText(body_bytes, "plain", "utf-8"))
            with smtplib.SMTP("smtp.gmail.com", 587) as server:
                server.starttls()
                server.login(sender_email, password_email)
                server.send_message(message)

            salt = os.urandom(32)
            hash = hashlib.pbkdf2_hmac("sha256", haslo_gen.encode('utf-
8'), salt, 10000)
            hexhash = (salt + hash).hex()

            cursor.execute("UPDATE uzytkownicy SET haslo = %s WHERE
email = %s", (hexhash, email))
            mydb.commit()

            return redirect(url_for('login'))
        else:
            error = "Email nie istnieje"

```

```
return render_template('change_pass.html', error=error)
```

Listing 3.4. Resetowanie hasła.

Kod (listing 3.4) definiuje trasę '/zmien_haslo' dla żądań GET i POST. Usuwa zmienne sesyjne 'logged_in' i 'verified'. W przypadku żądania POST pobiera adres email, sprawdza poprawność, generuje nowe hasło, wysyła je na email użytkownika, aktualizuje je w bazie danych po zahashowaniu, i przekierowuje do strony logowania. W przypadku błędnych danych renderuje szablon 'change_pass.html' z odpowiednim komunikatem błędu.

```
@app.route('/zmien_haslo', methods=['POST'])
def zmien_haslo():
    if not session.get('verified'):
        return redirect(url_for('login'))

    stare_haslo = escape(request.form['stare_haslo'])
    nowe_haslo = escape(request.form['nowe_haslo'])

    error = None

    if not validate_password(stare_haslo):
        error = "Niepoprawne stare hasło! Powinno posiadać co najmniej 8 znaków, w tym przynajmniej jedną małą literę, dużą literę oraz cyfrę."

    if not validate_password(nowe_haslo):
        error = "Niepoprawne nowe hasło! Powinno posiadać co najmniej 8 znaków, w tym przynajmniej jedną małą literę, dużą literę oraz cyfrę."

    if error:
        return profil(error_message=error)

    cursor = mydb.cursor()
    query = "SELECT * FROM uzytkownicy WHERE nazwa_uzytkownika = %s"
    values = (session['nazwa_uzytkownika'],)
    cursor.execute(query, values)
    result = cursor.fetchone()

    if result:
        stored_hash = result[6]
        salt = bytes.fromhex(stored_hash[:64])
        expected_hash = stored_hash[64:]

        hash = hashlib.pbkdf2_hmac("sha256", stare_haslo.encode('utf-8'),
salt, 10000)
        hexhash = hash.hex()

        if hexhash == expected_hash:
            new_salt = os.urandom(32)
```

```

        new_hash = hashlib.pbkdf2_hmac("sha256", nowe_haslo.encode('utf-
8'), new_salt, 10000)
        new_hexhash = (new_salt + new_hash).hex()

        query = "UPDATE uzytkownicy SET haslo = %s WHERE
nazwa_uzytkownika = %s"
        values = (new_hexhash, session['nazwa_uzytkownika'])
        cursor.execute(query, values)
        mydb.commit()
        return profil(success_message="Hasło zostało pomyślnie
zaktualizowane!")
    else:
        error = "Niepoprawnie wpisane obecne hasło, spróbuj jeszcze
raz!"

        return profil(error_message=error)
    else:
        return redirect(url_for('login'))

```

Listing 3.5. Zmiana hasła.

Kod (listing 3.5) obsługuje trasę '/zmien_haslo' dla żądań POST. Sprawdza, czy użytkownik jest zweryfikowany; jeśli nie, przekierowuje go do strony logowania. Następnie pobiera i waliduje dane z formularza zmiany hasła. Jeżeli dane są nieprawidłowe, renderuje stronę profilu z odpowiednim komunikatem błędu. Jeśli dane są poprawne, kod sprawdza poprawność starego hasła. Jeżeli hasło jest poprawne, generuje nową sól i hashowane hasło na podstawie nowego hasła użytkownika, a następnie aktualizuje te dane w bazie danych. Po pomyślnej zmianie hasła przekierowuje użytkownika do strony profilu z komunikatem sukcesu. W przypadku niepoprawnego starego hasła, informuje użytkownika o błędzie i renderuje stronę profilu z odpowiednim komunikatem.

```

@app.route('/edytuj_dane', methods=['POST'])
def edytuj_dane():
    if not session.get('verified'):
        return redirect(url_for('login'))

    imie = escape(request.form['imie'])
    nazwisko = escape(request.form['nazwisko'])
    numer_telefonu = escape(request.form['numer_telefonu'])
    email = escape(request.form['email'])
    nazwa_uzytkownika = escape(request.form['nazwa_uzytkownika'])

    error = None

    if not validate_name(imie) or not validate_name(nazwisko):
        error = "Niepoprawne imię lub nazwisko!"

```

```

if not validate_phone_number(numer_telefonu):
    error = "Niepoprawny numer telefonu!"

if not validate_email(email):
    error = "Niepoprawny adres email!"

if error:
    return profil(error_message=error), 200

cursor = mydb.cursor()
query = "UPDATE uzytkownicy SET imie = %s, nazwisko = %s, numer_telefonu
= %s, email = %s WHERE nazwa_uzytkownika = %s"
values = (imie, nazwisko, numer_telefonu, email, nazwa_uzytkownika)
cursor.execute(query, values)
mydb.commit()

return redirect(url_for('profil'))

```

Listing 3.6. Edycja danych użytkownika.

Kod (listing 3.6) definiuje trasę '/edytuj_dane' dla żądań POST. Sprawdza, czy użytkownik jest zweryfikowany. Pobiera dane z formularza edycji, takie jak imię, nazwisko, numer telefonu, email i nazwa użytkownika, a następnie waliduje je. Po walidacji aktualizuje dane użytkownika w bazie danych. W przypadku sukcesu przekierowuje użytkownika do strony profilu, a w przypadku błędów renderuje stronę profilu z odpowiednim komunikatem.

```

@app.route('/dodaj_ksiazke', methods=['GET', 'POST'])
def dodaj_ksiazke():
    if not session.get('verified'):
        return redirect(url_for('login'))

    if request.method == 'POST':
        tytul = escape(request.form['tytul'])
        autor = escape(request.form['autor'])
        wydawnictwo = escape(request.form['wydawnictwo'])
        seria = escape(request.form['seria'])
        oprawa = escape(request.form['oprawa'])
        rok_wydania = escape(request.form['rok_wydania'])
        ilosc_stron = escape(request.form['ilosc_stron'])
        rzad = escape(request.form['rzad'])
        regal = escape(request.form['regal'])
        polka = escape(request.form['polka'])

        errors = []

        if not validate(autor):
            errors.append("Nieprawidłowy autor książki")

```

```

if not validate_name(oprawa):
    errors.append("Nieprawidłowy rodzaj oprawy książki!")

if not validate_book_year(rok_wydania):
    errors.append("Niepoprawny rok wydania książki!")

if not validate_positive_number(ilosc_stron):
    errors.append("Nieprawidłowa liczba stron!")

if not validate_positive_number(rzad):
    errors.append("Nieprawidłowy numer rzędu!")

if not validate_positive_number(regal):
    errors.append("Nieprawidłowy numer regału!")

if not validate_positive_number(polka):
    errors.append("Nieprawidłowy numer półki!")

if errors:
    return render_template('dodaj_ksiazke.html', errors=errors)

cursor = mydb.cursor()
query = "INSERT INTO ksiazki (tytul, autor, wydawnictwo, seria,
oprawa, rok_wydania, ilosc_stron, rzad, regal, polka) VALUES (%s, %s, %s,
%s, %s, %s, %s, %s, %s, %s)"
values = (tytul, autor, wydawnictwo, seria, oprawa, rok_wydania,
ilosc_stron, rzad, regal, polka)
cursor.execute(query, values)
mydb.commit()

return redirect(url_for('strona_glowna'))
else:
    return render_template('dodaj_ksiazke.html')

```

Listing 3.7. Dodawanie książki.

Kod (listing 3.7) definiuje trasę '/dodaj_ksiazke' dla żądań GET i POST. Sprawdza, czy użytkownik jest zweryfikowany. W przypadku żądania POST pobiera dane z formularza dodawania książki, takie jak tytuł, autor, wydawnictwo, seria, oprawa, rok wydania, ilość stron, rząd, regał i półka. Następnie dokonuje walidacji tych danych, sprawdzając poprawność różnych pól. Jeśli dane są nieprawidłowe, renderuje stronę 'dodaj_ksiazke.html' z komunikatem błędu. W przypadku poprawnych danych, dodaje informacje o książce do bazy danych i przekierowuje użytkownika na stronę główną. W przypadku żądania GET renderuje stronę 'dodaj_ksiazke.html' umożliwiającą dodanie kolejnej książki.

```

@app.route('/usun_ksiazke/<int:book_id>', methods=['POST'])
def usun_ksiazke(book_id):

```

```

if not session.get('verified'):
    return redirect(url_for('login'))

cursor = mydb.cursor()
query = "DELETE FROM ksiazki WHERE id = %s"
values = (book_id,)
cursor.execute(query, values)
mydb.commit()

return redirect(url_for('strona_glowna'))

```

Listing 3.8. Usuwanie książki.

Kod (listing 3.8) definiuje trasę '/usun_ksiazke/int:book_id' dla żądań POST. Sprawdza, czy użytkownik jest zweryfikowany. Następnie usuwa książkę z bazy danych na podstawie przekazanego identyfikatora (book_id). Po pomyślnym usunięciu książki przekierowuje użytkownika na stronę główną.

```

@app.route('/edytuj_ksiazke/<int:ksiazka_id>', methods=['GET', 'POST'])
def edytuj_ksiazke(ksiazka_id):
    if not session.get('verified'):
        return redirect(url_for('login'))

    cursor = mydb.cursor()
    errors = []

    if request.method == 'POST':
        tytul = escape(request.form['tytul'])
        autor = escape(request.form['autor'])
        wydawnictwo = escape(request.form['wydawnictwo'])
        seria = escape(request.form['seria'])
        oprawa = escape(request.form['oprawa'])
        rok_wydania = escape(request.form['rok_wydania'])
        ilosc_stron = escape(request.form['ilosc_stron'])
        rzad = escape(request.form['rzad'])
        regal = escape(request.form['regal'])
        polka = escape(request.form['polka'])

        if not validate(autor):
            errors.append('Nieprawidłowy autor książki!')

        if not validate_name(oprawa):
            errors.append('Nieprawidłowy rodzaj oprawy książki!')

        if not validate_book_year(rok_wydania):
            errors.append('Nieprawidłowy rok wydania książki!')

        if not validate_positive_number(ilosc_stron):

```



```

        errors.append('Nieprawidłowa liczba stron!')

    if not validate_positive_number(rzad):
        errors.append('Nieprawidłowy numer rzędu!')

    if not validate_positive_number(regal):
        errors.append('Nieprawidłowy numer regału!')

    if not validate_positive_number(polka):
        errors.append('Nieprawidłowy numer półki!')

    if errors:
        for error in errors:
            flash(error, 'error')
        return redirect(url_for('edytuj_ksiazke',
ksiazka_id=ksiazka_id))

    query = "UPDATE ksiazki SET tytul = %s, autor = %s, wydawnictwo =
%s, seria = %s, oprawa = %s, rok_wydania = %s, ilosc_stron = %s, rzad = %s,
regal = %s, polka = %s WHERE id = %s"
    values = (tytul, autor, wydawnictwo, seria, oprawa, rok_wydania,
ilosc_stron, rzad, regal, polka, ksiazka_id)
    cursor.execute(query, values)
    mydb.commit()

    return redirect(url_for('strona_glowna'))
else:
    query = "SELECT * FROM ksiazki WHERE id = %s"
    values = (ksiazka_id,)
    cursor.execute(query, values)
    book = cursor.fetchone()

    return render_template('edytuj_ksiazke.html', book=book)

```

Listing 3.9. Edycja książki

Kod (listing 3.9) definiuje trasę '/edytuj_ksiazke/int:ksiazka_id' obsługującą żądania GET i POST. Sprawdza, czy użytkownik jest zweryfikowany, a następnie, w przypadku żądania POST, pobiera i waliduje dane z formularza edycji książki. Jeżeli dane są nieprawidłowe, dodaje komunikaty błędów do mechanizmu Flash i przekierowuje użytkownika z powrotem do formularza edycji. W przypadku poprawnych danych, aktualizuje informacje o książce w bazie danych i przekierowuje użytkownika na stronę główną. W przypadku żądania GET, pobiera informacje o edytowanej książce i renderuje formularz edycji z tymi danymi.

```

@app.route('/wyszukaj_ksiazke', methods=['GET', 'POST'])
def wyszukaj_ksiazke():
    if not session.get('verified'):

```

```

        return redirect(url_for('login'))

if request.method == 'POST':
    tytul = escape(request.form['tytul']).lower()
    autor = escape(request.form['autor']).lower()
    wydawnictwo = escape(request.form['wydawnictwo']).lower()
    seria = escape(request.form['seria']).lower()
    oprawa = escape(request.form['oprawa']).lower()
    rok_wydania = escape(request.form['rok_wydania'])
    ilosc_stron = escape(request.form['ilosc_stron'])
    rzad = escape(request.form['rzad'])
    regal = escape(request.form['regal'])
    polka = escape(request.form['polka'])

    errors = []

    if autor and not validate(autor):
        errors.append("Nieprawidłowy autor książki")

    if oprawa and not validate_name(oprawa):
        errors.append("Nieprawidłowy rodzaj oprawy książki!")

    if rok_wydania and not validate_book_year(rok_wydania):
        errors.append("Niepoprawny rok wydania książki!")

    if ilosc_stron and not validate_positive_number(ilosc_stron):
        errors.append("Nieprawidłowa liczba stron!")

    if rzad and not validate_positive_number(rzad):
        errors.append("Nieprawidłowy numer rzędu!")

    if regal and not validate_positive_number(regal):
        errors.append("Nieprawidłowy numer regału!")

    if polka and not validate_positive_number(polka):
        errors.append("Nieprawidłowy numer półki!")

    if errors:
        return render_template('wyszukaj_ksiazke.html', errors=errors)

    cursor = mydb.cursor()
    query = "SELECT * FROM ksiazki WHERE 1=1"
    values = []

    if tytul:
        query += " AND LOWER(tytul) LIKE %s"
        values.append('%' + tytul + '%')

    if autor:

```

```

        query += " AND LOWER(autor) LIKE %s"
        values.append('%' + autor + '%')

    if wydawnictwo:
        query += " AND LOWER(wydawnictwo) LIKE %s"
        values.append('%' + wydawnictwo + '%')

    if seria:
        query += " AND LOWER(seria) LIKE %s"
        values.append('%' + seria + '%')

    if oprawa:
        query += " AND LOWER(oprawa) LIKE %s"
        values.append('%' + oprawa + '%')

    if rok_wydania:
        query += " AND rok_wydania LIKE %s"
        values.append('%' + rok_wydania + '%')

    if ilosc_stron:
        query += " AND ilosc_stron LIKE %s"
        values.append('%' + ilosc_stron + '%')

    if rzad:
        query += " AND LOWER(rzad) LIKE %s"
        values.append('%' + rzad + '%')

    if regal:
        query += " AND LOWER(regal) LIKE %s"
        values.append('%' + regal + '%')

    if polka:
        query += " AND LOWER(polka) LIKE %s"
        values.append('%' + polka + '%')

    cursor.execute(query, values)
    books = cursor.fetchall()

    return render_template('wyszukaj_ksiazke.html', books=books)
else:
    return render_template('wyszukaj_ksiazke.html', books=[])

```

Listing 3.10. Wyszukiwanie książek

Kod (listing 3.10) definiuje trasę '/wyszukaj_ksiazke' dla żądań GET i POST. Sprawdza, czy użytkownik jest zweryfikowany, a następnie, w przypadku żądania POST, pobiera i waliduje dane z formularza wyszukiwania książki. Jeżeli dane są nieprawidłowe, renderuje szablon 'wyszukaj_ksiazke.html' z komunikatami błędu. Następnie przeprowadza

wyszukiwanie książek w bazie danych na podstawie różnych kryteriów, takich jak tytuł, autor, wydawnictwo, seria, oprawa, rok wydania, ilość stron, rząd, regał i półka. Renderuje szablon 'wyszukaj_ksiazke.html' z wynikami wyszukiwania lub, w przypadku żądania GET, z pustym formularzem.

```
class TokenBucket:
    def __init__(self, tokens, time_unit, forward_callback, drop_callback):
        self.tokens = tokens
        self.time_unit = time_unit
        self.forward_callback = forward_callback
        self.drop_callback = drop_callback
        self.bucket = tokens
        self.last_check = time.time()

    def handle(self, request):
        current = time.time()
        time_passed = current - self.last_check
        self.last_check = current

        self.bucket = self.bucket + time_passed * (self.tokens /
self.time_unit)

        if self.bucket > self.tokens:
            self.bucket = self.tokens

        if self.bucket < 1:
            self.drop_callback(request)
            return True
        else:
            self.bucket -= 1
            self.forward_callback(request)
            return False

    def forward(self, request):
        print("Żądanie przekazane: " + str(request))

    def drop(self, request):
        print("Żądanie odrzucone: " + str(request))
```

Listing 3.11 TokenBucket

Kod (listing 3.11) definiuje klasę TokenBucket, która implementuje algorytm token bucket. Token bucket to algorytm ograniczający dostęp, kontrolujący przepustowość żądań na podstawie puli dostępnych tokenów. Klasa TokenBucket posiada metodę __init__, inicjalizującą wartości dla puli tokenów, jednostki czasu, funkcji przekazywania i odrzucania

zadań, aktualnej puli oraz ostatniego sprawdzenia czasu. Metoda handle obsługuje żądanie, aktualizując pulę tokenów w zależności od upływu czasu i podejmując działania (przekazywanie lub odrzucanie żądania). Funkcje forward i drop obsługują przekazywanie i odrzucanie żądań.

3.2. Widoki

Poniżej zostały przedstawione wszystkie dostępne widoki w aplikacji. Rysunek 1 przedstawia stronę główną aplikacji, z której możemy się zalogować lub zarejestrować.



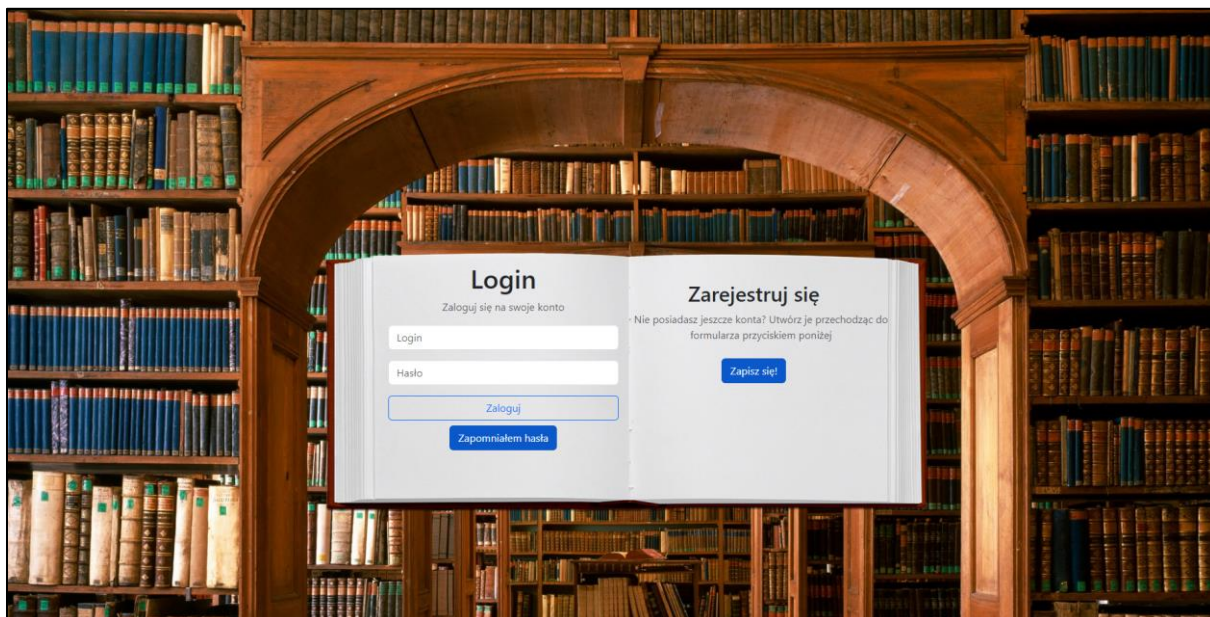
Rysunek 1. Strona główna.

Z rejestracji (rysunek 2) można przenieść się do logowania oraz utworzyć konto podając: imię, nazwisko, numer telefonu, email, nazwę użytkownika oraz hasło.



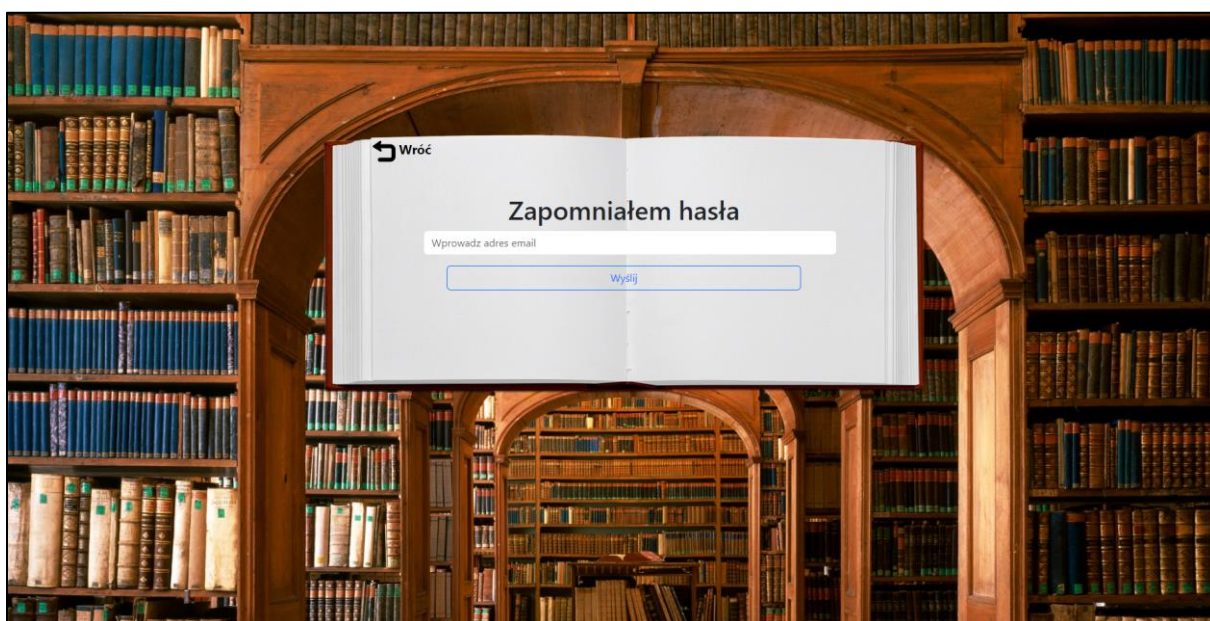
Rysunek 2. Rejestracja

Aby zalogować się należy podać login oraz hasło. Również z tego ekranu można przenieść się do rejestracji oraz do zresetowania hasła, kiedy użytkownik nie pamięta swojego.



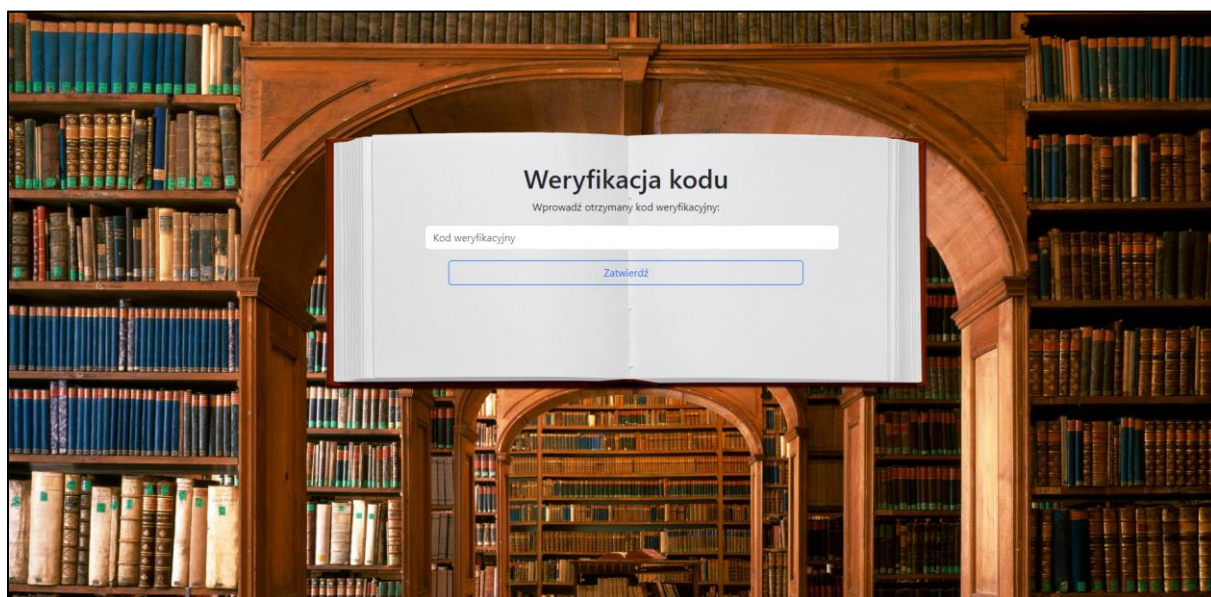
Rysunek 3. Logowanie.

W widoku resetowania hasła należy podać email, na który zostanie wysłane nowe hasło.



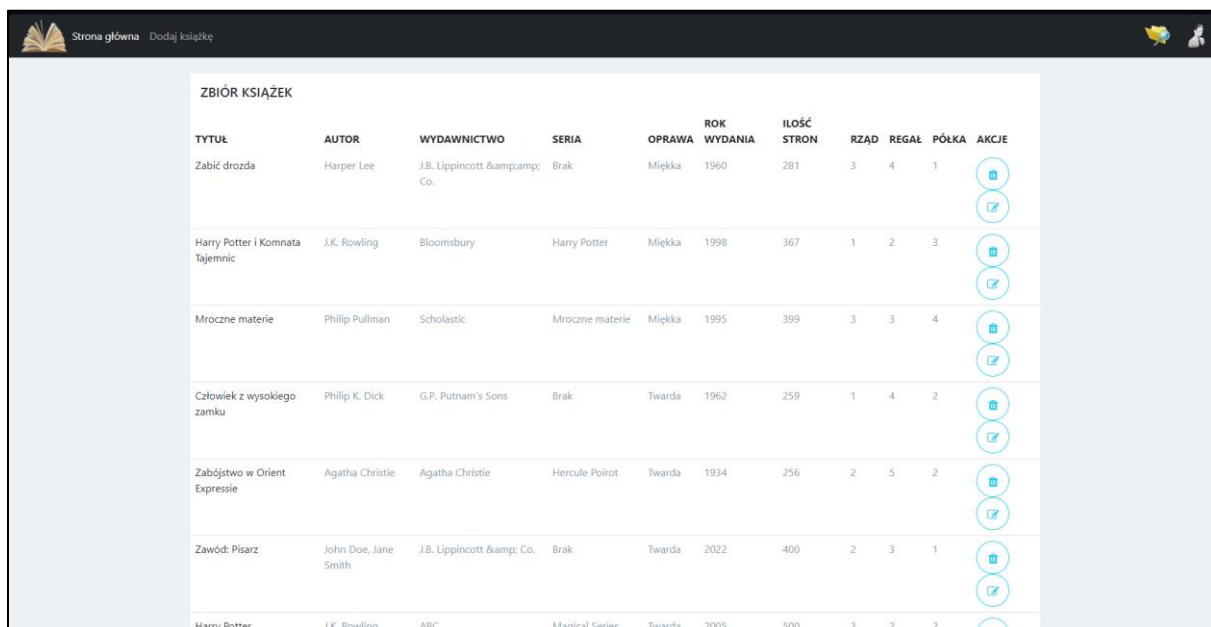
Rysunek 4. Resetowanie hasła.















W weryfikacji należy podać kod , który został wysłany na email użytkownika.



Rysunek 5. Weryfikacja.

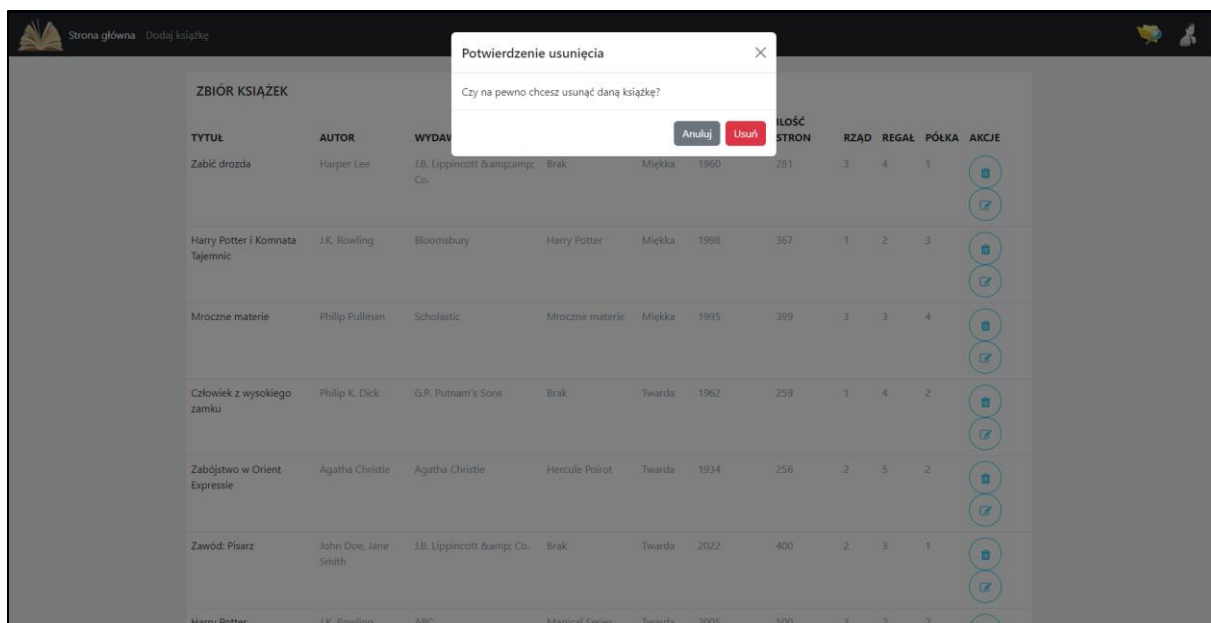
Po zalogowaniu zostaje wyświetlony spis wszystkich książek. Z widoku tego można przejść do wyszukiwania książek, dodawania, edycji oraz usuwania i edycji danych użytkownika.



ZBIÓR KSIĄŻEK										
TYTUŁ	AUTOR	WYDAWNICTWO	SERIA	OPRAWA	ROK WYDANIA	IŁOŚĆ STRON	RZĄD	REGAL	PÓŁKA	AKCJE
Zabić drozda	Harper Lee	J.B. Lippincott & amp; Co.	Brak	Miękka	1960	281	3	4	1	 
Harry Potter i Komnata Tajemnic	J.K. Rowling	Bloomsbury	Harry Potter	Miękka	1998	367	1	2	3	 
Mroczne materie	Philip Pullman	Scholastic	Mroczne materie	Miękka	1995	399	3	3	4	 
Człowiek z wysokiego zamku	Philip K. Dick	G.P. Putnam's Sons	Brak	Twarda	1962	259	1	4	2	 
Zabójstwo w Orient Expressie	Agatha Christie	Agatha Christie	Hercule Poirot	Twarda	1934	256	2	5	2	 
Zawód: Pisarz	John Doe, Jane Smith	J.B. Lippincott & amp; Co.	Brak	Twarda	2022	400	2	3	1	 
Harry Potter	J.K. Rowling	ABC	Magical Series	Twarda	2005	500	3	2	2	 

Rysunek 6. Spis książek.

Kiedy jest wyświetlony spis książek, przy każdej pozycji są dwie opcje. Jedna z nich to usuwanie książki. Po kliknięciu ikony kosza, użytkownik zostaje poproszony o potwierdzenie czynności.



Rysunek 7. Usuwanie książek.

Druga ikona dotyczy edycji danych książki. Należy zmienić dane oraz nacisnąć przycisk *Zapisz zamiany*.


Panel edycji książki:

Tytuł: Zabić drożdża	Rok wydania: 1960
Autor: Harper Lee	Ilość stron: 281
Wydawnictwo: J.B. Lippincott & Pincott Co.	Rząd: 3
Seria: Brak	Regał: 4
Oprawa: Miękka	Półka: 1

[Zapisz zmiany](#)



Rysunek 8. Edycja książki.

Podczas dodawania książki trzeba uzupełnić cały formularz, a następnie nacisnąć przycisk *Dodaj książkę*.



Strona główna

Dodaj książkę

Wprowadź dane aby dodać książkę do systemu:

Tytuł:

Rok wydania:

Autor:

Ilość stron:

Wydawnictwo:

Rząd:

Seria:

Regał:


Oprawa:

Półka:

Dodaj książkę



Rysunek 9. Dodawanie książki.

W widoku wyszukiwania książek, można wyświetlić wszystkie pozycje po kliknięciu w przycisk *Wyszukaj*. Aby znaleźć książkę, który nas interesuje np. po autorze czy serii, należy uzupełnić pole, a następnie wyszukać. Istnieje możliwość zresetowania całego wyszukiwania.



Strona główna

Dodaj książkę

Tytuł:

Rok wydania:

Autor:

Ilość stron:

Wydawnictwo:

Rząd:

Seria:

Regał:

Oprawa:

Półka:

Wyszukaj

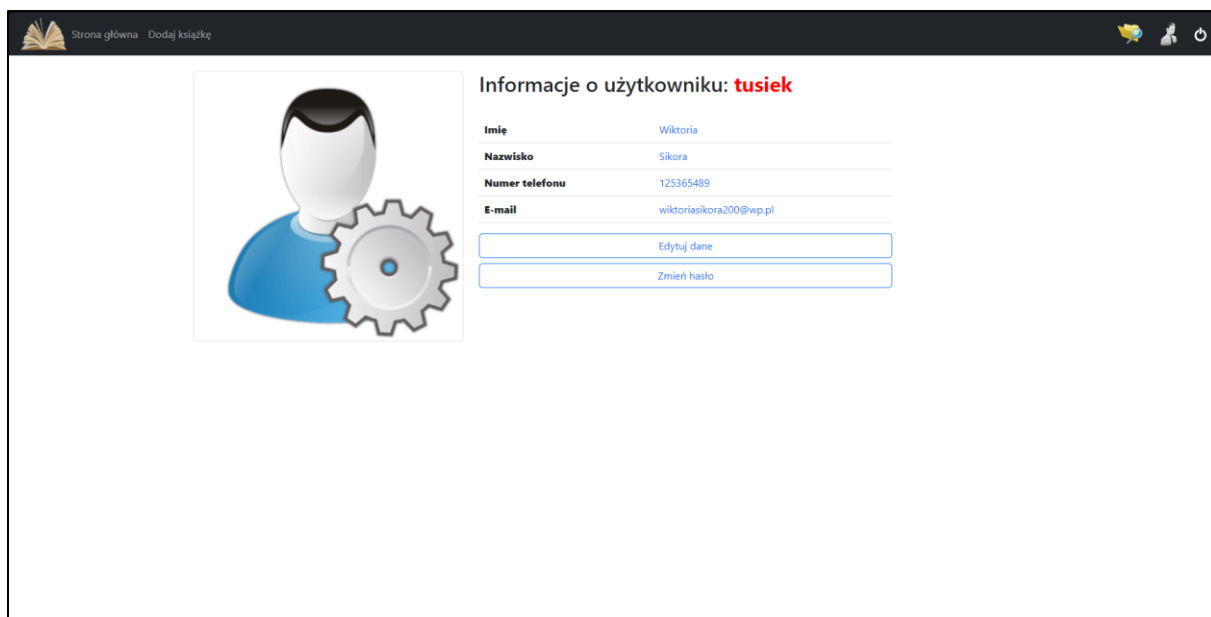
Resetuj wyszukiwanie

Wyniki wyszukiwania:

TYTUŁ	AUTOR	WYDAWNICTWO	SERIA	OPRAWA	ROK WYDANIA	ILOŚĆ STRON	RZĄD	REGAŁ	PÓŁKA
Zabić drozda	Harper Lee	J.B. Lippincott & Pincott Co.	Brak	Miękka	1960	281	3	4	1
Harry Potter i Komnata Tajemnic	J.K. Rowling	Bloomsbury	Harry Potter	Miękka	1998	367	1	2	3
Mroczne materie	Philip Pullman	Scholastic	Mroczne materie	Miękka	1995	399	3	3	4
Człowiek z wysokiego zamku	Philip K. Dick	G.P. Putnam's Sons	Brak	Twarda	1962	259	1	4	2
Zabójstwo w Orient Expressie	Agatha Christie	Agatha Christie	Hercule Poirot	Twarda	1934	256	2	5	2
Zawód: Pisarz	John Doe, Jane Smith	J.B. Lippincott & Pincott Co.	Brak	Twarda	2022	400	2	3	1
Harry Potter	J.K. Rowling	ABC	Magical Series	Twarda	2005	500	3	2	2
Harry Potter	J.K. Rowling	ABC	Magical Series	Twarda	2005	500	3	2	2
Harry Potter	J.K. Rowling	ABC	Magical Series	Twarda	2005	500	3	2	2
Harry Potter	J.K. Rowling	ABC	Magical Series	Twarda	2005	500	3	2	2

Rysunek 10. Wyszukiwanie książek.

W panelu użytkownika, są wyświetlone wszystkie dane o użytkowniku oraz istnieje możliwość zmiany tych danych oraz hasła.



Rysunek 11. Panel użytkownika.

Po kliknięciu przycisku *Edytuj dane* zostaje wyświetlony formularz, w którym można zmienić swoje dane. Po kliknięciu w *Zmień hasło*, należy podać obecne oraz nowe hasło, a następnie zaakceptować.

Imię	Wiktor
Nazwisko	Sikora
Numer telefonu	125365489
E-mail	wiktoriasikora200@wp.pl
<input type="button" value="Edytuj dane"/>	
Edycja danych	
Imię:	<input type="text" value="Wiktor"/>
Nazwisko:	<input type="text" value="Sikora"/>
Numer telefonu:	<input type="text" value="125365489"/>
Email:	<input type="text" value="wiktoriasikora200@wp.pl"/>
<input type="button" value="Zapisz zmiany"/>	
<input type="button" value="Zmień hasło"/>	
Proces zmiany hasła	
Obecne hasło:	<input type="text" value="Obecne hasło"/>
Nowe hasło:	<input type="text" value="Nowe hasło"/>
<input type="button" value="Zmień hasło"/>	

Rysunek 12. Edycja danych użytkownika.

3.3. Zabezpieczenia przed podatnościami

Wstrzykiwanie SQL (SQL Injection) - aby zapobiec SQL Injection, stosujemy parametryzację zapytań. W ten sposób unikamy wstrzykiwania złośliwego kodu SQL do zapytań. Parametryzowanie zapytań polega na przekazywaniu parametrów zapytania w sposób, który nie pozwala na wstrzykiwanie złośliwego kodu SQL. Zamiast bezpośrednio umieszczać dane wejściowe w treści zapytania, używane są parametry, które są następnie zabezpieczone przed ewentualnym wstrzykiwaniem.

Cross-Site Scripting (XSS) - aby zapobiec temu rodzajowi ataku, stosujemy odpowiednie metody zabezpieczeń, takie jak escapowanie danych wprowadzanych przez użytkowników do stron i formularzy. To zapobiega interpretacji kodu JavaScript w użytkowniku i zabezpiecza przed XSS.

Cross-Site Request Forgery (CSRF) - tworzony jest unikalny, ukryty oraz trudny do przewidzenia token który jest generowany dla każdej sesji użytkownika. Jest on dołączany do żądań użytkownika i porównywany z wartością przechowywaną na serwerze. Jeśli token nie jest zgodny, żądanie wyświetli komunikat.

Nieprawidłowe Zarządzanie Hasłami - przechowujemy hasła użytkowników w postaci zahaszowanej, a nie w czystej postaci. Podczas procesu rejestracji lub zmiany hasła, hasło wprowadzone przez użytkownika jest haszowane przy użyciu algorytmu skrótu, takiego jak SHA-256. Haszowanie hasła sprawia, że jest ono nieczytelne nawet dla administratorów systemu, co zwiększa bezpieczeństwo w przypadku ewentualnego naruszenia bazy danych

Niewłaściwe Zarządzanie Autoryzacją - TOTP (Time-based One-Time Password) to metoda dwuskładnikowego uwierzytelniania, w której generowany jest jednorazowy kod uwierzytelniający. Wprowadzenie TOTP jako drugiego etapu uwierzytelniania zwiększa bezpieczeństwo, ponieważ nawet gdyby hasło użytkownika było skradzione, atakujący potrzebowałby również fizycznego dostępu do urządzenia generującego kody TOTP. Również wprowadzono blokadę konta po 3 niepowodzeniach logowań pod rząd na 5 minut.

Nieprawidłowe Walidowanie Danych - przyjmujemy praktyki walidacji danych, aby upewnić się, że użytkownicy wprowadzają poprawne informacje. Przykłady walidacji obejmują sprawdzanie poprawności adresów e-mail, numerów telefonu, numerów półki, regału, rzędu, ilości stron czy danych dotyczących oprawy itd. Wykorzystujemy odpowiednie wyrażenia regularne lub wbudowane funkcje walidacji, aby upewnić się, że dane są zgodne z określonym formatem.

DDoS – zastosowaliśmy zabezpieczenie typu Token Bucket, które polega na kontrolowaniu ilości żądań, które serwer musi obsłużyć w określonym czasie. Dzięki temu nawet w przypadku ataku, system jest w stanie skutecznie zarządzać przepływem ruchu i utrzymać dostępność usług dla prawidłowych użytkowników, minimalizując wpływ ataku DDoS.

4. Testy aplikacji

W niniejszym rozdziale skupimy się na dwóch głównych rodzajach testów: testach manualnych i testach jednostkowych. Obejmują one zarówno aspekty automatyzacji, jak i interakcje z oprogramowaniem w sposób przeprowadzany ręcznie przez testera. Testy jednostkowe koncentrują się na weryfikacji poprawności pojedynczych komponentów, funkcji lub modułów kodu źródłowego. Z drugiej strony, testy manualne angażują ludzką interwencję w celu oceny interakcji użytkownika z oprogramowaniem. W przeciwieństwie do testów jednostkowych, są bardziej skoncentrowane na doświadczeniu użytkownika, a ich celem jest sprawdzenie funkcji, wydajności oraz ogólnej użyteczności oprogramowania.

W testach manualnych, została sprawdzona reakcja aplikacji na różne akcje. Reaguje poprawnie na podawane dane, szybkość odpowiedzi z bazy danych jest natychmiastowa oraz sesja działa odpowiednio. Aplikacja reaguje od strony backendu oraz frontu. Podczas rejestracji może wystąpić komunikat o źle złożonym hasle (rysunek 13), złym numerze telefonu, emailu, imieniu, nazwisku oraz że użytkownik istnieje już o podanym emailu czy nazwie użytkownika (rysunek 14).

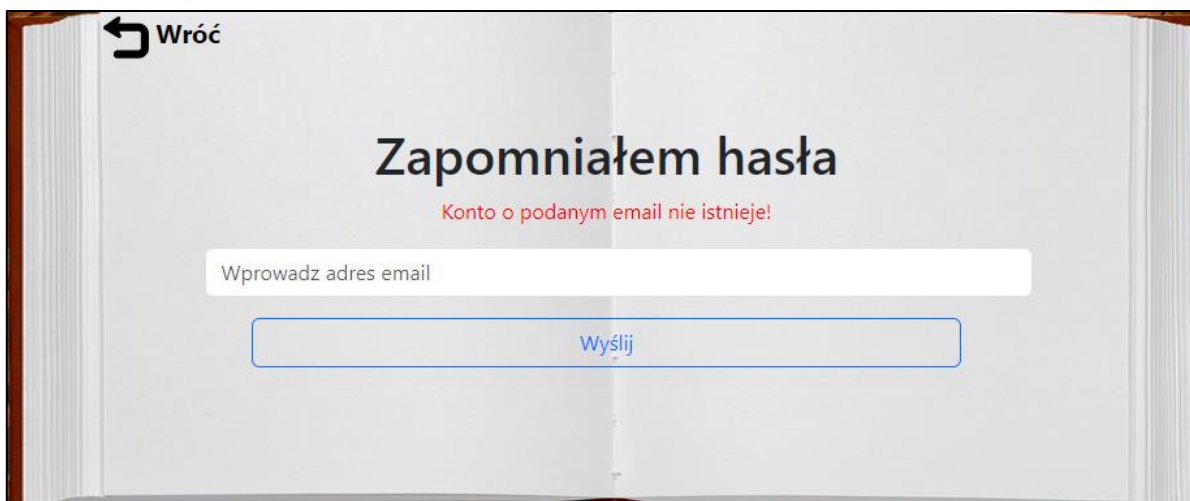
The image shows a registration form titled "Rejestracja". It has two columns of input fields. The left column contains fields for "Imię:", "Nazwisko:", and "Numer telefonu:". The right column contains fields for "Email:", "Nazwa użytkownika:", and "Hasło:". Below the fields, a red error message states: "Niepoprawne hasło! Powinno posiadać conajmniej 8 znaków, w tym przynajmniej jedną małą literę, dużą literę oraz cyfrę." Below the message is a blue button labeled "Zarejestruj". At the bottom, there is a link: "Masz już konto? [Zaloguj się](#)".

Rysunek 13. Komunikat dotyczący hasła podczas rejestracji.

The image shows the same registration form titled "Rejestracja" as in Figure 13. It has the same layout of input fields for "Imię:", "Nazwisko:", "Numer telefonu:", "Email:", "Nazwa użytkownika:", and "Hasło:". Below the fields, a red error message states: "Użytkownik o podanym loginie lub emailu już istnieje!". Below the message is a blue button labeled "Zarejestruj". At the bottom, there is a link: "Masz już konto? [Zaloguj się](#)".

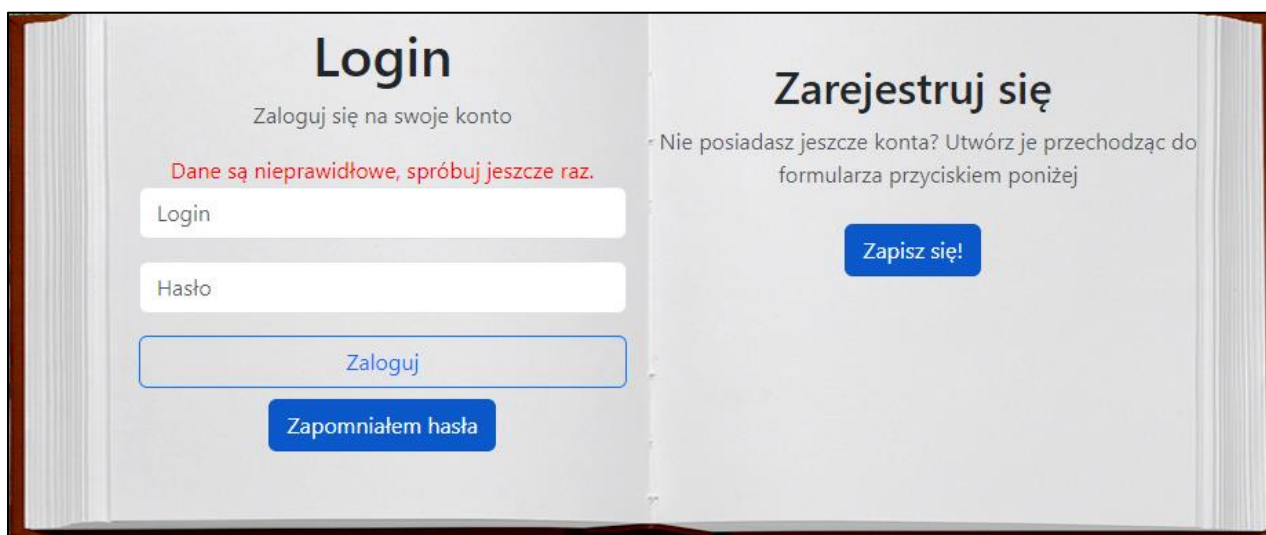
Rysunek 14. Komunikat podczas rejestracji dotyczący już istnienia takiego użytkownika.

Kiedy użytkownik zapomniał hasło ma możliwość jego zresetowania. Podczas tej akcji może uzyskać komunikat, że nie istnieje użytkownik o podanym emailu (rysunek 15).



Rysunek 15. Komunikat o nie istnieniu takiego użytkownika

Podczas logowania użytkownik może dostać komunikat, że dane są nieprawidłowe wtedy gdy użytkownik o takim loginie nie istnieje lub hasło do użytkownika nie pasuje (rysunek 16).



Rysunek 16. Komunikat, że dane są nieprawidłowe.

Użytkownik, który przeprowadzi pod rząd trzy niepoprawne logowania, ma zablokowane konto na 5 minut. Dostaje komunikat jak na rysunku 17.

Login

Zaloguj się na swoje konto

Nieprawidłowe dane logowania. Twoje konto zostało zablokowane.

Zarejestruj się

Nie posiadasz jeszcze konta? Utwórz je przechodząc do formularza przyciskiem poniżej

Rysunek 17. Zablokowane konto.

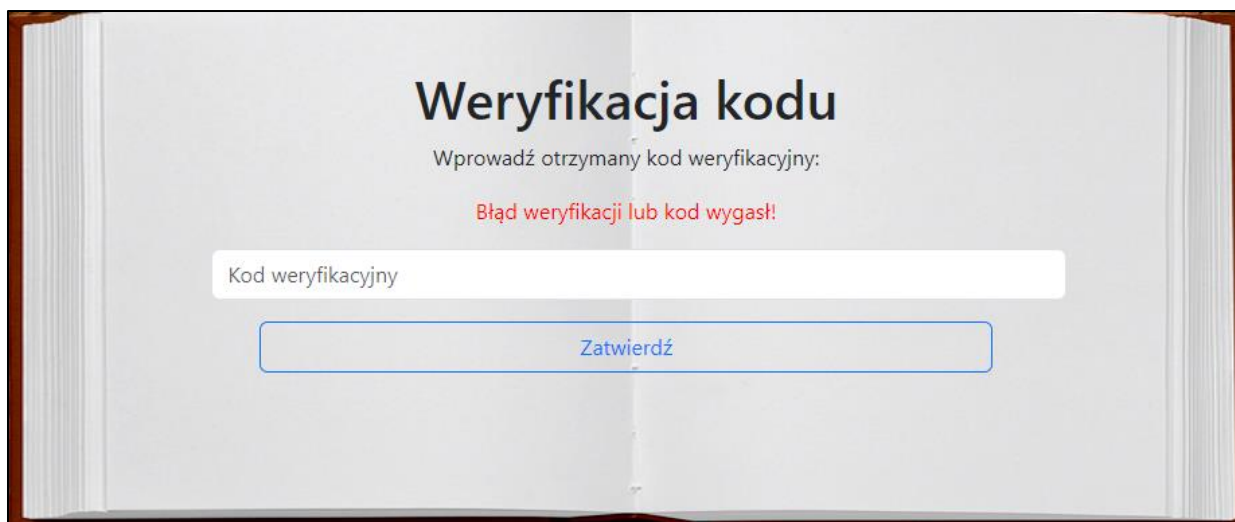
Kiedy użytkownik poda poprawne dane logowania, na email dostanie kod do weryfikacji. Użytkownik może uzyskać dwa komunikaty o niepoprawnym kodzie weryfikacji (rysunek 18), lub błędzie podczas weryfikacji i wygaśnięciu kodu (rysunek 19).

Weryfikacja kodu

Wprowadź otrzymany kod weryfikacyjny:

Nieprawidłowy kod weryfikacyjny!

Rysunek 18. Nieprawidłowy kod weryfikacji.



Rysunek 19. Błąd podczas weryfikacji lub nieważność kodu.

Podczas operacji takich jak dodawanie książki (rysunek 20), edycji książki (rysunek 21) oraz wyszukiwania książek (rysunek 22), użytkownik może dostać komunikat o niepoprawnie wprowadzonych danych. Dla przykładu numery półki, rzędu czy regału nie mogą być ujemne. Rok nie może być większy niż obecny. Ilość stron musi być dodatnia. Imię i nazwisko autora nie może posiadać cyfr czy też oprawa książki.

Rysunek 20. Komunikaty podczas dodawania książki.

Strona główna

Panel edycji książki:

Tytuł: Harry Potter

Autor: J.K. Rowling

Wydawnictwo: ABC

Seria: Magical Series

Oprawa: Twarda

Rok wydania: 2005

Ilość stron: 500

Rząd: 3

Regał: 2

Półka: 2

Nieprawidłowy rok wydania książki!
Nieprawidłowa liczba stron!

Zapisz zmiany

Rysunek 21. Komunikaty podczas edycji książki.

Strona główna Dodaj książkę

Tytuł:

Autor: 123

Wydawnictwo:

Seria:

Oprawa:

Rok wydania: 2028

Ilość stron: -200

Rząd: -2

Regał: 0

Półka: 10

Wyszukaj

Resetuj wyszukiwanie

Nieprawidłowy autor książki!
Niepoprawny rok wydania książki!
Nieprawidłowa liczba stron!
Nieprawidłowy numer rzędu!
Nieprawidłowy numer regału!

Obecnie nie wyszukiujesz żadnej książki.

Rysunek 22. Komunikaty podczas wyszukiwania książek.

Podczas zmiany danych, przez użytkownika może wystąpić komunikat o niepoprawnym numerze telefonu (rysunek 23), emailu czy imieniu i nazwisku (rysunek 24).

Imię	Wiktoria
Nazwisko	Sikora
Numer telefonu	125365489
E-mail	wiktoriasikora200@wp.pl
Edytuj dane	
Niepoprawny numer telefonu!	

Rysunek 23. Niepoprawny numer telefonu.

Imię	Wiktoria
Nazwisko	Sikora
Numer telefonu	125365489
E-mail	wiktoriasikora200@wp.pl
<input type="button" value="Edytuj dane"/>	
Niepoprawne imię lub nazwisko!	

Rysunek 24. Niepoprawne imię lub nazwisko.

Użytkownik ma możliwość zmiany hasła na nowe. Podczas tej operacji, podając obecne jako złe aplikacja poinformuje go komunikatem (rysunek 25). Natomiast kiedy nowe hasło nie spełni wymagań wyświetli się komunikat, że hasło musi zawierać co najmniej 8 znaków w tym jedną małą literę, dużą literę oraz cyfrę.

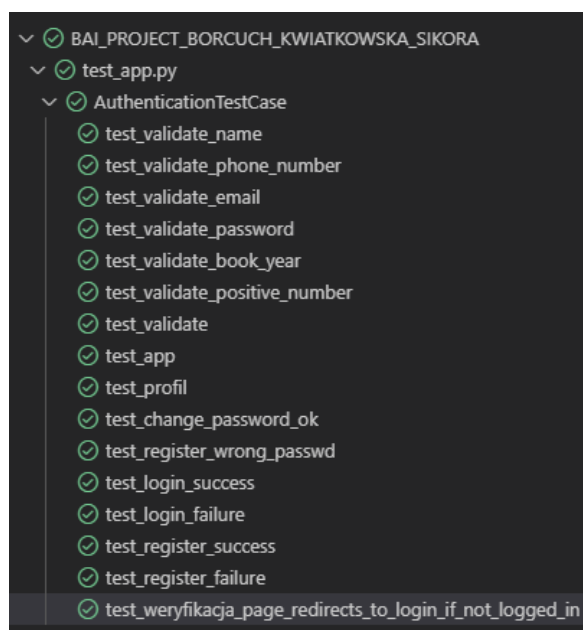
Imię	Wiktoria
Nazwisko	Sikora
Numer telefonu	125365489
E-mail	wiktoriasikora200@wp.pl
<input type="button" value="Edytuj dane"/>	
Niepoprawnie wpisane obecne hasło, spróbuj jeszcze raz!	
<input type="button" value="Zmień hasło"/>	

Rysunek 25. Niepoprawne obecne hasło podczas zmiany hasła.

Imię	Wiktoria
Nazwisko	Sikora
Numer telefonu	125365489
E-mail	wiktoriasikora200@wp.pl
<input type="button" value="Edytuj dane"/>	
Niepoprawne nowe hasło! Powinno posiadać co najmniej 8 znaków, w tym przynajmniej jedną małą literę, dużą literę oraz cyfrę.	
<input type="button" value="Zmień hasło"/>	

Rysunek 26. Nowe hasło nie spełnia wymagań.

Zostały przeprowadzone 16 testów jednostkowych, który zostały zdane.



Rysunek 27. Zdane testy jednostkowe.

5. Podział pracy

Nazwisko i imię	Udział procentowy
Borcuch Matusz	Strona główna, weryfikacja, profil, dodawanie książki, wyszukiwanie książki
Kwiatkowska Aleksandra	Rejestracja, resetowanie hasła, zmiana hasła, usuwanie książki, TokenBucket
Sikora Wiktoria	Logowanie, spis książek, edycja danych, edycja książki, walidacja danych

6. Podsumowanie

Zabezpieczenia w aplikacjach webowych stanowią fundament ochrony przed różnorodnymi zagrożeniami cybernetycznymi. W kontekście podatności, które mogą być wykorzystane przez atakujących, kluczowe jest implementowanie skutecznych środków obronnych.

W dzisiejszym środowisku online, gdzie aplikacje webowe obsługują różnorodne dane i interakcje użytkowników, bezpieczeństwo staje się priorytetem. Wstrzykiwanie SQL, będące jednym z najpoważniejszych zagrożeń, może prowadzić do naruszenia bazy danych. Jednym z

efektywnych sposobów zabezpieczenia przed tym atakiem jest stosowanie parametryzacji zapytań, co pozwala na oddzielenie danych wejściowych od samego kodu SQL.

Kolejną podatnością wartą uwagi jest Cross-Site Scripting (XSS), gdzie złośliwe skrypty są osadzone w treści strony. Aby temu przeciwdziałać, konieczne jest escapowanie danych, czyli konwersja znaków HTML na bezpieczne formy, co uniemożliwia wykonanie szkodliwego kodu.

CSRF, czyli atak wykorzystujący zalogowaną sesję użytkownika, może prowadzić do nieautoryzowanych działań. W celu zabezpieczenia przed CSRF, stosuje się tokeny CSRF, które są unikalnymi identyfikatorami generowanymi dla każdej sesji użytkownika, sprawdzanymi przed przetworzeniem żądania.

Nieprawidłowe zarządzanie hasłami stanowi kolejne zagrożenie, zwłaszcza gdy hasła są przechowywane w postaci niezaszyfrowanej. Bezpieczne haszowanie, z dodaniem soli do hasła, stanowi skuteczną obronę przed próbami złamania hasła.

W przypadku niewłaściwego zarządzania autoryzacją, implementacja bezpiecznego uwierzytelniania opartego na TOTP, czyli jednorazowych kodach czasowych, znacznie wzmacnia ochronę dostępu do kont użytkowników.

Wreszcie, nieprawidłowe walidowanie danych może prowadzić do różnych ataków. Stosowanie rzetelnej walidacji, ograniczanie dostępu do pól tylko do koniecznych informacji oraz stosowanie mechanizmów kontroli długości i typów danych to kluczowe elementy budowania solidnej obrony.

Podsumowując, zabezpieczenia aplikacji webowych to skomplikowany obszar, który wymaga holistycznego podejścia. Świadomość ryzyka, zastosowanie najlepszych praktyk programistycznych oraz regularne przeglądy bezpieczeństwa są kluczowe dla utrzymania integralności i poufności danych w dzisiejszym dynamicznym środowisku cybernetycznym.

7. Spis listingów

Listing 3.1. Rejestracja.....	7
Listing 3.2. Logowanie.....	9
Listing 3.3. Weryfikacja.	10
Listing 3.4. Resetowanie hasła.	12
Listing 3.5. Zmiana hasła.	13
Listing 3.6. Edycja danych użytkownika.	14

Listing 3.7. Dodawanie książki.....	15
Listing 3.8. Usuwanie książki.....	16
Listing 3.9. Edycja książki.....	17
Listing 3.10. Wyszukiwanie książek.....	19
Listing 3.11 TokenBucket.....	20

8. Spis rysunków

Rysunek 1. Strona główna.....	21
Rysunek 2. Rejestracja.....	21
Rysunek 3. Logowanie.....	22
Rysunek 4. Resetowanie hasła.....	22
Rysunek 5. Weryfikacja.....	23
Rysunek 6. Spis książek.....	23
Rysunek 7. Usuwanie książek.....	24
Rysunek 8. Edycja książki.....	24
Rysunek 9. Dodawanie książki.....	25
Rysunek 10. Wyszukiwanie książek.....	25
Rysunek 11. Panel użytkownika.....	26
Rysunek 12. Edycja danych użytkownika.....	26
Rysunek 13. Komunikat dotyczący hasła podczas rejestracji.....	29
Rysunek 14. Komunikat podczas rejestracji dotyczący już istnienia takiego użytkownika. ...	29
Rysunek 15. Komunikat o nie istnieniu takiego użytkownika.....	30
Rysunek 16. Komunikat, że dane są nieprawidłowe.....	30
Rysunek 17. Zablockowane konto.....	31
Rysunek 18. Nieprawidłowy kod weryfikacji.....	31
Rysunek 19. Błąd podczas weryfikacji lub nieważność kodu.....	32
Rysunek 20. Komunikaty podczas dodawania książki.....	32
Rysunek 21. Komunikaty podczas edycji książki.....	33
Rysunek 22. Komunikaty podczas wyszukiwania książek.....	33
Rysunek 23. Niepoprawny numer telefonu.....	33
Rysunek 24. Niepoprawne imię lub nazwisko.....	34
Rysunek 25. Niepoprawne obecne hasło podczas zmiany hasła.....	34
Rysunek 26. Nowe hasło nie spełnia wymagań.....	34

Rysunek 27. Zdane testy jednostkowe.	35
---	----