

PAINT - Projekt

Zespół: Samochodoza

Dawid Markowski 325302

Marcin Lusawa 325298

Filip Wiśniewski 325335

Politechnika Warszawska

Wydział Elektroniki i Technik Informatycznych

Kierunek: Telekomunikacja

Semestr: 2024Z

Warszawa, 26 Stycznia 2025

Elementy dokumentacji projektów

1. Określenie tematu Projektu

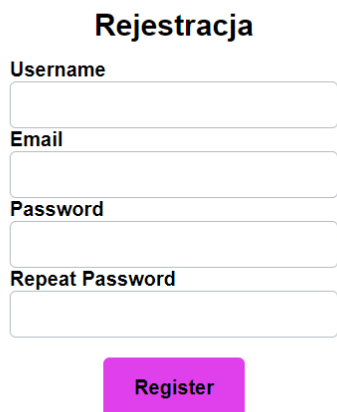
- Tematem projektu jest stworzenie sklepu internetowego z częściami samochodowymi służącej do zarządzania elementami sklepowymi, w tym obsługą koszyka, zamówieniami oraz systemem użytkowników.
- Projekt obejmuje implementację panelu administratora, sekcji użytkowników oraz funkcji sklepowych, takich jak przeglądanie produktów, dodawanie ich do koszyka, realizacja zamówień oraz rejestracja użytkowników.
- Dodatkowo, aplikacja oferuje moduł zarządzania rolami użytkowników (np. admin, klient) oraz mechanizmy uwierzytelniania i resetowania hasła.

2. Zespół i opis ról w Projekcie

- **Programista Backend (Dawid Markowski):** Odpowiada za implementację logiki biznesowej aplikacji, integrację z bazą danych oraz tworzenie API, zapewniając płynne działanie funkcjonalności aplikacji.
- **Programista Frontend (Marcin Lusawa):** Tworzy interfejs użytkownika, implementuje strony HTML, CSS oraz logikę frontendową. Dbą o estetykę oraz intuicyjność korzystania z aplikacji.
- **Analitik Danych (Filip Wiśniewski):** Wyszukuje, analizuje i interpretuje dane związane z projektem. Przygotowuje raporty i wizualizacje danych, wspomaga decyzje projektowe.

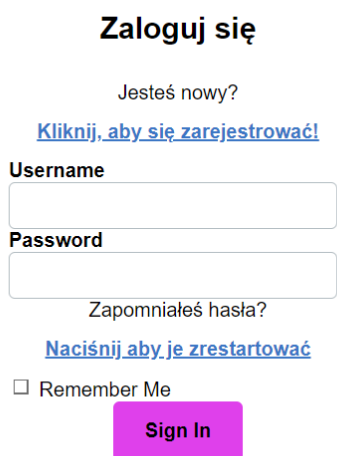
3. Opis szczegółowych założeń funkcjonalnych aplikacji (Produkt Projektu)

- **Rejestracja i logowanie użytkowników:** Aplikacja umożliwia tworzenie kont użytkowników za pomocą formularza rejestracyjnego oraz logowanie do systemu przy użyciu adresu e-mail i hasła. W przypadku utraty hasła dostępna jest funkcjonalność resetowania hasła za pomocą linku przesyłanego na adres e-mail.



The diagram shows a registration form titled "Rejestracja". It contains four input fields stacked vertically, each with a label above it: "Username", "Email", "Password", and "Repeat Password". Below these fields is a purple button labeled "Register".

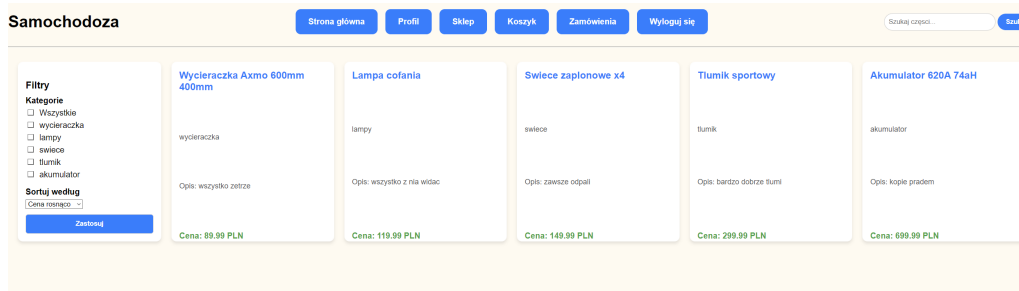
Rysunek 1: Formularz rejestracji



The diagram shows a login form titled "Zaloguj się". At the top, it asks "Jesteś nowy?" and provides a blue link "Kliknij, aby się zarejestrować!". Below this are two input fields labeled "Username" and "Password". Under the "Password" field is a link "Zapomniałeś hasła?" and another blue link "Naciśnij aby je zresetować". At the bottom left is a checkbox labeled "Remember Me". A purple button labeled "Sign In" is positioned at the bottom right.

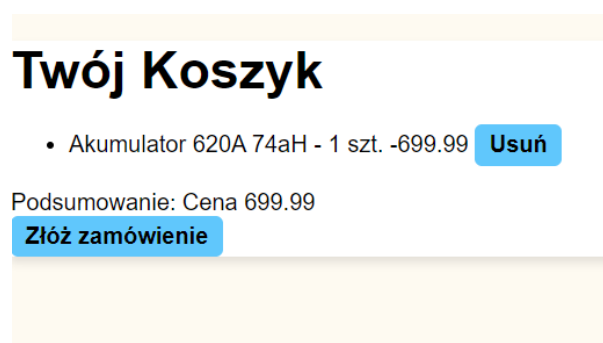
Rysunek 2: Formularz logowania

- **Zarządzanie rolami użytkowników:** System obsługuje różne poziomy uprawnień, takie jak administrator, sprzedawca i klient. Administrator ma pełny dostęp do zarządzania aplikacją, w tym do edycji produktów, zarządzania zamówieniami oraz przeglądania raportów.
- **Przeglądanie produktów:** Użytkownicy mogą przeglądać dostępne produkty z podziałem na kategorie. Dla każdego produktu widoczne są szczegóły, takie jak opis, cena, zdjęcia oraz dostępność w magazynie.



Rysunek 3: Widok listy produktów

- **Koszyk zakupowy:** Aplikacja umożliwia użytkownikom dodawanie produktów do koszyka, edycję ich ilości oraz usuwanie ich z koszyka. Koszyk przechowuje informacje o produktach do czasu finalizacji zamówienia.



Rysunek 4: Widok koszyka

- **Proces składania zamówień:** Użytkownicy mogą złożyć zamówienie, wybierając metodę płatności i dostawy. Aplikacja obsługuje różne formy płatności, takie jak przelew bankowy, płatności kartą oraz systemy płatności elektronicznej (np. PayPal).

Checkout

Proszę wypełnić dane zamówienia:

Adres:

Miasto:

Kod pocztowy:

Metoda płatności:

Metoda dostawy:

Numer karty:

Data ważności:

CVV:

Złóż zamówienie

Rysunek 5: Widok składania zamówienia

- **Panel administracyjny:** Administrator ma dostęp do panelu zarządzania produktami (dodawanie, edytowanie i usuwanie), zamówieniami oraz użytkownikami.

Admin

Home

User









Part

Order

List (4)

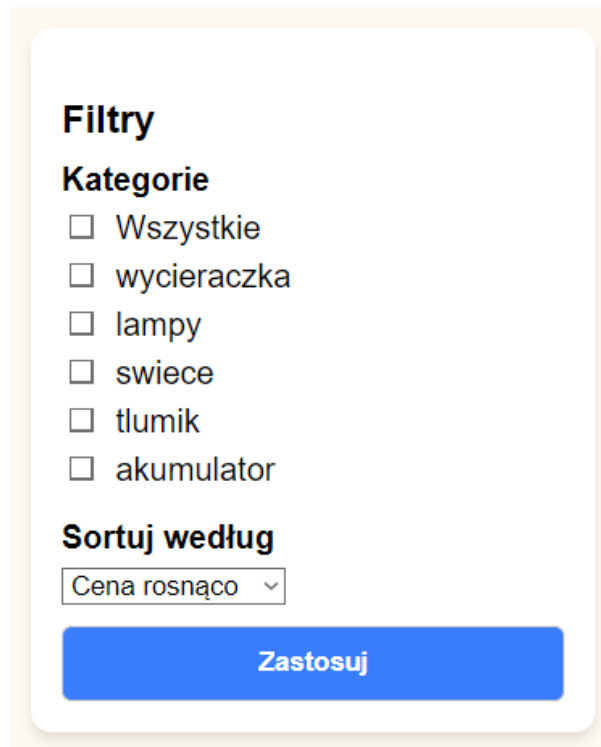
Create

With selected

<input type="checkbox"/>		Username	Email	Is Admin	Address	City	Postal Code
<input type="checkbox"/>	 	admin1	admin@example.com	<input checked="" type="radio"/>			
<input type="checkbox"/>	 	user1	user1@example.com	<input type="radio"/>			
<input type="checkbox"/>	 	user2	user2@example.com	<input type="radio"/>			
<input type="checkbox"/>	 	1234	1234@op.pl	<input type="radio"/>			

Rysunek 6: Widok panelu administratora

- **Zaawansowane wyszukiwanie produktów:** Aplikacja pozwala użytkownikom na filtrowanie produktów według kategorii, ceny, ocen oraz innych parametrów, co ułatwia odnalezienie interesujących przedmiotów.



Filtry

Kategorie

- ☐ Wszystkie
- ☐ wycieraczka
- ☐ lampy
- ☐ swiece
- ☐ tłumik
- ☐ akumulator

Sortuj według

Cena rosnąco ▼

Zastosuj

Rysunek 7: Widok filtracji produktów

4. Opis szczegółowych założeń architektonicznych aplikacji

- **Wybrane technologie i wzorce architektoniczne:**
 - Aplikacja wykorzystuje framework **Flask** jako podstawę backendu, zapewniając obsługę logiki biznesowej, zarządzanie użytkownikami oraz realizację procesów zakupowych.
 - **SQLAlchemy** jest używane jako ORM (Object-Relational Mapping) do zarządzania bazą danych, umożliwiając łatwe wykonywanie zapytań i utrzymanie spójności danych.
 - System autoryzacji i uwierzytelniania opiera się na **Flask-Login**, co pozwala na zarządzanie sesjami użytkowników oraz uprawnieniami.
 - Frontend aplikacji integruje się z backendem za pomocą szablonów **Jinja2**, co pozwala na dynamiczne generowanie stron HTML.

5. Opis wykorzystania narzędzi programowych w powiązaniu z ww. założeniami funkcjonalnymi i architektonicznymi

• Wykorzystane narzędzia programistyczne:

- **Flask**: Lekki framework webowy użyty jako podstawa backendu aplikacji. Flask pozwala na implementację logiki biznesowej, obsługę routingu, autoryzacji i uwierzytelniania użytkowników.
- **SQLAlchemy**: ORM (Object-Relational Mapping) używany do zarządzania bazą danych. Umożliwia łatwe definiowanie modeli danych oraz wykonywanie zapytań do bazy.
- **Flask-Login**: Biblioteka odpowiedzialna za zarządzanie sesjami użytkowników oraz obsługę logowania i wylogowywania.
- **Flask-Mail**: Narzędzie służące do wysyłania powiadomień e-mailowych, takich jak resetowanie hasła, potwierdzenia zamówień czy komunikaty systemowe.
- **Flask-Migrate**: Narzędzie do zarządzania migracjami bazy danych w oparciu o Alembic. Umożliwia łatwe aktualizacje schematu bazy danych w miarę rozwoju projektu.
- **WTForms**: Framework używany do tworzenia i walidacji formularzy w aplikacji. Umożliwia łatwe zbieranie i przetwarzanie danych od użytkowników.
- **Jinja2**: Silnik szablonów, pozwalający na dynamiczne generowanie HTML-a w oparciu o dane przesyłane z backendu.
- **Python-Dotenv**: Narzędzie do obsługi plików `.env`, które przechowują zmienne środowiskowe, takie jak klucze API, dane dostępu do bazy danych czy ustawienia serwera.
- **PostgreSQL/SQLite**: Używana baza danych, odpowiedzialna za przechowywanie informacji o użytkownikach, produktach, zamówieniach i innych kluczowych danych aplikacji.
- **IDE i edytory kodu**: Projekt został rozwijany z wykorzystaniem nowoczesnych środowisk programistycznych, takich jak **PyCharm** czy **Visual Studio Code**, które wspierają debugowanie i refaktoryzację kodu.

- **Powiązanie narzędzi z wymaganiami projektowymi:**

- **Obsługa użytkowników i ról:** Dzięki Flask-Login i WTForms aplikacja umożliwia rejestrację, logowanie, resetowanie hasła oraz zarządzanie profilami użytkowników.
- **Zarządzanie danymi:** SQLAlchemy w połączeniu z Flask-Migrate zapewnia łatwą obsługę modeli danych oraz migracji schematów bazy danych, co pozwala na elastyczne rozwijanie aplikacji.
- **Powiadomienia e-mailowe:** Flask-Mail umożliwia wysyłanie automatycznych powiadomień, spełniając wymagania dotyczące informowania użytkowników o ważnych wydarzeniach w systemie.
- **Responsywne generowanie stron:** Dzięki Jinja2 aplikacja generuje dynamiczne strony HTML, dostosowujące się do danych wprowadzonych przez użytkowników i danych przechowywanych w bazie.
- **Rozwijalność aplikacji:** Wykorzystanie Flask-Migrate i SQLAlchemy pozwala na łatwe wprowadzanie nowych funkcjonalności, takich jak dodatkowe pola w modelach danych czy nowe tabele w bazie.

6. Opis realizacji (opis procesu powstawania aplikacji, problemy, ich rozwiązania, modyfikacje założeń itp.)

- **Proces tworzenia aplikacji krok po kroku:**

- **Krok 1: Analiza wymagań i przygotowanie dokumentacji.** Zespół zebrał wymagania funkcjonalne i нефункционалне, zdefiniował kluczowe funkcje aplikacji, a następnie przygotował dokumentację projektową.
- **Krok 2: Wybór technologii i narzędzi.** Na podstawie wymagań projektu wybrano framework Flask jako backend, SQLAlchemy jako ORM, oraz Flask-Login do zarządzania użytkownikami. Wybrano również narzędzia wspierające rozwój, takie jak PyCharm i Git.
- **Krok 3: Projektowanie architektury.** Zespół opracował diagramy architektoniczne, w tym strukturę modułów aplikacji, przepływ danych.
- **Krok 4: Implementacja backendu.** Zaimplementowano modele danych w SQLAlchemy, przygotowano routy obsługujące logikę biznesową oraz system autoryzacji użytkowników.
- **Krok 5: Implementacja frontendu.** Stworzono szablony HTML za pomocą Jinja2, uwzględniając dane przekazywane z backendu.
- **Krok 6: Dokumentacja końcowa.** Przygotowano instrukcję instalacji i obsługi dla użytkowników.

7. Opis techniczny („wewnętrzny”) Produktu – czyli opis poszczególnych modułów

- **Struktura modułów aplikacji:**

- **Moduł główny (main):**

- * Zawiera funkcje odpowiedzialne za podstawowe operacje aplikacji, takie jak wyświetlanie strony głównej, profilu użytkownika oraz przeglądanie listy produktów.
 - * Obsługuje routy takie jak `/index`, `/user/{username}` i `/edit_profile`, które zarządzają interakcjami użytkownika z aplikacją.

- * **Moduł uwierzytelniania (auth):**

- Realizuje funkcje rejestracji, logowania i resetowania hasła użytkowników.
 - Wykorzystuje biblioteki Flask-Login oraz Flask-WTF do obsługi sesji i formularzy logowania.
 - Zapewnia bezpieczną walidację danych oraz ochronę dostępu do zasobów aplikacji.

- * **Moduł koszyka (cart):**

- Zarządza funkcjonalnościami związanymi z koszykiem użytkownika, w tym dodawaniem produktów, aktualizacją ilości oraz usuwaniem przedmiotów.
 - Integruje się z modułem produktów, aby weryfikować dostępność towarów.

- * **Moduł realizacji zamówień (checkout):**

- Obsługuje proces składania zamówienia, w tym wybór metody płatności i dostawy.
 - Integruje się z zewnętrznymi systemami płatności, aby realizować transakcje.
 - Przechowuje szczegóły zamówień w bazie danych i generuje potwierdzenia e-mailowe.

- * **Moduł administracyjny (admin):**

- Udostępnia panel administracyjny, w którym można zarządzać produktami, użytkownikami oraz zamówieniami.
 - Zawiera funkcje edytowania, dodawania i usuwania produktów oraz przeglądania raportów sprzedaży.

- * **Moduł obsługi błędów (errors):**

- Obsługuje strony błędów, takie jak 404 (Nie znaleziono) i 500 (Błąd serwera).
 - Zapewnia odpowiednie komunikaty dla użytkownika w przypadku problemów technicznych.

- * **Moduł produktów (shop):**

- Zajmuje się zarządzaniem produktami, w tym wyświetlaniem szczegółów, filtrowaniem i wyszukiwaniem produktów.
 - Wspiera generowanie dynamicznych widoków dla produktów przy użyciu Jinja2.

- **Interfejsy i interakcje pomiędzy modułami:**

- * Moduł **main** pełni rolę centralnego punktu komunikacji dla użytkowników i integruje się z modułami **auth**, **cart**, **checkout** i **shop**.
- * Moduł **auth** zarządza uwierzytelnianiem użytkowników i zapewnia odpowiednie sesje, które są wykorzystywane w innych modułach.
- * Moduł **cart** wymienia dane z modułem **shop**, aby zweryfikować dostępność produktów, oraz z modułem **checkout**, aby przysyłać szczegóły zamówień.
- * Moduł **checkout** współpracuje z zewnętrznymi usługami płatności oraz modułem **cart**, aby zapewnić poprawne przetwarzanie zamówień.
- * Moduł **admin** integruje się z pozostałymi modułami w celu zarządzania danymi aplikacji, w tym użytkownikami, produktami i zamówieniami.
- * Moduł **errors** jest uniwersalny i może być wywoływany przez każdy inny moduł w przypadku wystąpienia problemów technicznych.

8. Opis instalacji Produktu (instrukcja instalacji – step-by-step)

- **Sposób 1: Tradycyjna instalacja.**

- **Krok 1: Skopiuj repozytorium projektu.**

- * Wykonaj klonowanie repozytorium za pomocą polecenia:

- ```
git clone https://github.com/dawid-markowski/ProjPAINT.git
```

- **Krok 2: Utwórz i aktywuj środowisko wirtualne.**

- \* Utwórz środowisko wirtualne, korzystając z polecenia:

- ```
python -m venv venv
```

- * Aktywuj środowisko:

- ```
source venv/bin/activate # Linux/MacOS
```

- ```
venv\Scripts\activate # Windows
```

- **Krok 3: Zainstaluj wymagane pakiety.**

- * Zainstaluj pakiety wymienione w pliku `requirements.txt`:

- ```
pip install -r requirements.txt
```

- **Krok 4: Skonfiguruj bazę danych.**

- \* Wykonaj migrację bazy danych za pomocą polecenia:

- ```
flask db upgrade
```

- **Krok 5: Uzupełnij bazę danych przykładowymi danymi.**

- * Dodaj przykładowe dane do bazy danych:

- ```
python populate.py
```

- **Krok 6: Uruchom aplikację.**

- \* Włącz aplikację lokalnie za pomocą:

- ```
python stronka.py
```

- * Aplikacja będzie dostępna w przeglądarce pod domyślnym adresem:

- ```
http://127.0.0.1:5000.
```

- **Sposób 2: Instalacja za pomocą Dockera.**

- **Krok 1: Skopiuj repozytorium projektu.**

- \* Wykonaj klonowanie repozytorium za pomocą polecenia:

- ```
git clone https://github.com/dawid-markowski/ProjPAINT.git
```

- **Krok 2: Pobierz obraz Dockerowy.**

- * Pobierz gotowy obraz aplikacji z rejestru:

- ```
docker pull ghcr.io/dawid-markowski/samo:latest
```

- **Krok 3: Uruchom kontener.**

- \* Uruchom aplikację w kontenerze Dockera za pomocą polecenia:

- ```
docker run -d -p 5000:5000 --name samochodoza ghcr.io/dawid-markowski/projpaint/samo:latest
```

- **Krok 4: Otwórz aplikację.**

- * Otwórz przeglądarkę i przejdź do adresu: <http://localhost:5000>.

- **Lista wymaganych komponentów:**

- W przypadku tradycyjnej instalacji: Python (3.8+), Git, narzędzia developer-skie.
 - W przypadku instalacji Dockera: Docker i Docker Compose.

9. Opis użytkowania Produktu (instrukcja użytkownika)

- Użytkownik rozpoczyna korzystanie z aplikacji poprzez rejestrację konta w sekcji **/register** lub logowanie w sekcji **/login**, podając adres e-mail i hasło. W przypadku zapomnienia hasła można je zresetować za pomocą funkcji resetowania dostępnej na stronie logowania.
- Po zalogowaniu użytkownik może przeglądać produkty na stronie głównej lub w sekcji sklepu (**/shop**). Produkty można filtrować według kategorii, ceny lub dostępności, a szczegóły każdego produktu są dostępne po kliknięciu na niego.
- Produkty można dodawać do koszyka, edytować ich ilość lub usuwać. Po zakończeniu wyboru produktów użytkownik przechodzi do procesu składania zamówienia, wybierając metodę płatności i dostawy.
- Panel administracyjny jest dostępny dla użytkowników z odpowiednimi uprawnieniami i umożliwia zarządzanie produktami, zamówieniami oraz użytkownikami.