POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki
Filia w Gdańsku

Langmesser Adam
Nr albumu s27119
Nazwa specjalizacji: Aplikacje Internetowe
Redosz Mateusz
Nr albumu s27094
Nazwa specjalizacji: Aplikacje Internetowe
Oziemczuk Stanisław
Nr albumu s26982
Nazwa specjalizacji: Aplikacje Internetowe
Badek Kacper
Nr albumu s29168
Nazwa specjalizacji: Aplikacje Internetowe

Aplikacja webowa: spoty-na-drony.pl

Rodzaj pracy
inżynierska
Imię i nazwisko promotora
mgr Adam Urbanowicz

Gdańsk, miesiąc, 2100 obrony

Streszczenie: Celem niniejszej pracy było stworzenie w pełni funkcjonalnej i działającej aplikacji internetowej pozwalającej na szybkie wyszukiwanie spotów w okolicy oraz dzielenie się zdjęciami, filmami oraz doświdczeniem z innymi użytkownikami. W ramach pracy stworzono system składający się z trzech komponentów: Frontendu, Backendu oraz bazy-danych. Aplikacja internetowa została wykonana przy pomocy Frameworka React w językach Javascript oraz Typescript, do styli został użyty Tailwind. Serwis backendowy został stworzony w języku Java oraz biblioteki Spring Boot. Baza danych to PostgreSql.
Komunikacja między komponentami odbywała się zgodnie ze standardem REST. Projekt został zrealizowany w podejściu ewolucyjno-przyrostowym z elementami Kanban.
Słowa kluczowe: — brak —

POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Karta projektu

Temat projektu: Aplikacja webowa: spoty-na-drony.pl **Temat projektu po angielsku:** Web application: spoty-na-drony.pl	**Akronim:** Merkury **Data ustalenia tematu** 2023-10-10
Promotor: mgr Adam Urbanowicz	**Konsultanci:** 1. — brak —
Cele projektu: Stworzenie w pełni funkcjonalnej aplikacji internetowej do rozwijania hobby(latania dronem).	
Rezultaty projektu: Aplikacja Internetowa, Dokumentacja Interaktywna mapa z wyświetlanymi spotami oraz pogodą. Zaawansowana wyszukiwarka spotów. Forum do dzielenia się informacjami na temat dronów. Chat jednoosobowy oraz grupowy. Konto użytkownika z możliwością zapisania ulubionych spotów.	
Miary sukcesu: Gotowa do wdrożenia aplikacja. Realizacja w terminie zgodnym z wymaganiami.	
Ograniczenia: Budżetowe: brak środków na wdrożenie. Zawodowe: brak doświadczenia. Czasowe: trzy semestry (09.2024 - 02.2026). Ludzkie: czteroosobowy zespół.	

Wykonawcy	**Numer albumu**	**Specjalizacja**	**Tryb studiów**
Langmesser Adam	s27119	Aplikacje Internetowe	Stacjonarny
Redosz Mateusz	s27094	Aplikacje Internetowe	Stacjonarny
Oziemczuk Stanisław	s26982	Aplikacje Internetowe	Stacjonarny
Badek Kacper	s29168	Aplikacje Internetowe	Stacjonarny

Data ukończenia projektu:	**Recenzent:**
27 listopada 2025	dr Elżbieta Puźniakowska-Gałuch

Spis treści

Rozdział 1

Wstęp

1.1 O projekcie

1.2 Cel i zakres prac

1.3 Geneza pomysłu

Rozdział 2

Opis problemu

2.1 Rich picture

2.2 Udziałowcy

KARTA UDZIAŁOWCA	
Identyfikator:	UO1
Nazwa:	Zespół projektowy
Opis:	Zespół czterech studentów odpowiedzialnych za analizę, projekt, implementację, testy oraz dokumentację systemu.
Typ udziałowca:	ożywiony, bezpośredni
Punkt widzenia:	techniczna, wykonawcza
Ograniczenia:	Ograniczone zasoby czasowe i doświadczenie komercyjne.
Wymagania:	Wymagania funkcjonalne i techniczne systemu, możliwość realizacji w ramach projektu dyplomowego.

Tabela 2.1: Zespół projektowy

KARTA UDZIAŁOWCA	
Identyfikator:	UO2
Nazwa:	Promotor
Opis:	Osoba nadzorująca przebieg projektu, weryfikująca poprawność merytoryczną i zgodność z wymaganiami uczelni.
Typ udziałowca:	ożywiony, pośredni
Punkt widzenia:	merytoryczna, formalna, jakościowa
Ograniczenia:	Nie odpowiada za implementację; rekomenduje, opiniuje i zatwierdza.
Wymagania:	Czytelna dokumentacja, zgodność z wytycznymi kierunku i dobry poziom techniczny rozwiązania.

Tabela 2.2: Promotor

KARTA UDZIAŁOWCA	
Identyfikator:	UO3
Nazwa:	Droniarze
Opis:	Główna grupa docelowa systemu – osoby latające dronami rekreacyjnie lub półprofesjonalnie, szukające miejsc do lotów i wymiany doświadczeń.
Typ udziałowca:	ożywiony, bezpośredni
Punkt widzenia:	użytkownik końcowy: prostota obsługi, rzetelne informacje o spotach, wygodne dzielenie się treściami.
Ograniczenia:	Brak wpływu na architekturę techniczną systemu; oczekują intuicyjnego interfejsu.
Wymagania:	Lista spotów, informacje o ograniczeniach prawnych, oceny/komentarze, dodawanie treści, podstawowe funkcje społecznościowe.

Tabela 2.3: Droniarze

2.3 Istniejące rozwiązania

2.4 Wizja rozwiązania

2.5 Aspekty społeczne i biznesowe

2.5.1 Aspekty społeczne

2.5.2 Aspekty biznesowe

Rozdział 3

Planowanie

3.1 Metodologia pracy

3.1.1 Przegląd rozważanych podejść

Przy wyborze metodologii pracy rozważono trzy podejścia do prowadzenia projektu informatycznego:

- klasyczny Agile (w praktyce: Scrum),
- model kaskadowy (Waterfall),
- Disciplined Agile Delivery - Lean Life Cycle.

3.1.2 Odrzucone podejścia

„Klasyczny Agile" (Scrum). Mimo elastyczności i popularności zakłada pracę w iteracjach 2–4 tygodni oraz stały zestaw ceremonii (planowanie, przegląd, retrospektywa). Ze względu na nierównomierną dostępność zasobów w kolejnych miesiącach studiów nie zapewniono możliwości utrzymania stałej kadencji sprintów, dlatego z podejścia zrezygnowano.

Model kaskadowy (Waterfall). Przewiduje sekwencyjne przechodzenie przez z góry określone etapy i ogranicza bieżącą weryfikację wymagań w trakcie prac deweloperskich. W projekcie wymagano możliwości częstych rewizji założeń oraz

wprowadzania istotnych zmian w docelowej wizji rozwiązania; dlatego z podejścia zrezygnowano.

3.1.3 Wybrane podejście: Disciplined Agile Delivery (Lean Life Cycle)

Podjęto decyzję o zastosowaniu **Disciplined Agile Delivery** [1] w wariancie **Lean Life Cycle** [2], ponieważ podejście to łączy pożądane cechy Agile i Waterfall, a jednocześnie eliminuje stałe sprinty na rzecz pracy w ciągłym przepływie.

Kluczowe argumenty wyboru:

- **Brak sprintów.** Zastosowano przepływ ciągły, co pozwala dopasować tempo do zmiennej dostępności zespołu i unikać sztucznego „domykania" iteracji.
- **Rozbudowana faza startowa.** Na początku przewidziano większy wysiłek planistyczny: doprecyzowanie zakresu, wstępna wizja architektury, identyfikacja ryzyk, plan publikacji oraz kryteria jakości – bez zamrażania szczegółów.
- **Ciągła weryfikacja wymagań.** W trakcie realizacji przewidziano bieżące doprecyzowywanie backlogu, regularny feedback promotora oraz możliwość korygowania kierunku bez kosztów „przeskakiwania" między fazami.
- **Praktyki Lean i koncentracja na wartości.** Priorytetyzacja wartości biznesowej, wizualizacja pracy, małe partie dostaw.
- **Lekka governance i kamienie milowe.** Zastosowano lekkie mechanizmy nadzoru (peer review, prezentacje postępów) zapewniające przejrzystość bez nadmiernej biurokracji.

3.1.4 Narzędzia i komunikacja

Do zarządzania zadaniami zastosowana została **Jira** (monitorowanie postępu prac oraz ewidencja zadań członków zespołu). Komunikację w zespole zaplanowano w

formie regularnych spotkań oraz asynchronicznie z wykorzystaniem **Discorda** oraz **Messengera**.

3.1.5 Podział ról w zespole

- **Adam - fullstack developer, lider zespołu**
- **Stanisław - fullstack developer**
- **Kacper - fullstack developer**
- **Mateusz - fullstack developer**

Każdy z członków zespołu uczestniczy również w przygotowaniu dokumentacji.

3.2 Harmonogram projektu

W poniższym harmonogramie przedstawiono plan prac nad poszczególnymi częściami projektu, rozłożony na miesiące.

Rok 2024

Czerwiec
- Zebranie zespołu.
- Rozważenie potencjalnych pomysłów.

Lipiec
- Wybór technologii.
- Wstępne założenia architektoniczne.

Sierpień
- *(do uzupełnienia)*
- *(do uzupełnienia)*

Wrzesień
- *(do uzupełnienia)*
- *(do uzupełnienia)*

Październik
- *(do uzupełnienia)*
- *(do uzupełnienia)*

Listopad • *(do uzupełnienia)*

- *(do uzupełnienia)*

Grudzień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Rok 2025

Styczeń • *(do uzupełnienia)*

- *(do uzupełnienia)*

Luty • *(do uzupełnienia)*

- *(do uzupełnienia)*

Marzec • *(do uzupełnienia)*

- *(do uzupełnienia)*

Kwiecień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Maj • *(do uzupełnienia)*

- *(do uzupełnienia)*

Czerwiec • *(do uzupełnienia)*

- *(do uzupełnienia)*

Lipiec • *(do uzupełnienia)*

- *(do uzupełnienia)*

Sierpień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Wrzesień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Październik • *(do uzupełnienia)*

- *(do uzupełnienia)*

Listopad • *(do uzupełnienia)*

- *(do uzupełnienia)*

Grudzień • *(do uzupełnienia)*

- *(do uzupełnienia)*

Rok 2026

Styczeń • *(do uzupełnienia)*

- *(do uzupełnienia)*

3.3 Technologie i narzędzia

Do realizacji projektu wykorzystano wiele technologii oraz narzędzi informatycznych. Przy wyborze technologii kierowaliśmy się ich popularnością, dostępnością dokumentacji oraz artykułów, a także łatwością użycia. Narzędzia zostały dopasowane do wybranych technologii i specyfikacji zadań. Poniżej przedstawiono opis wybranych opcji.

3.3.1 Technologie

3.3.2 Narzędzia

Do niektórych płatnych narzędzi mieliśmy bezpłatny dostęp za pośrednictwem uczelni, w innych mogliśmy założyć konta edukacyjne, które oferowały dostęp do wszystkich funkcji narzędzia. Gdy żadna z wymienionych opcji nie była udostępniona, wybieraliśmy rozwiązania darmowe.

- **IntelliJ IDEA Ultimate**

  Jest to IDE od firmy JetBrains. Dzięki licznie dostępnym pluginom oferuje obsługę wielu języków programowania oraz innych składni. Pozwala również

na integrację z repozytorium. Używaliśmy go do programowania zarówno frontendu, jak i backendu oraz tworzenia dokumentacji w LaTeX.

- **Docker Desktop**

  To narzędzie do zarządzania obrazami, kontenerami oraz wolumenami Docker. Zawiera w sobie również silnik tej technologii. Wykorzystywaliśmy je do lokalnego uruchamiana bazy danych oraz serwisu do cachowania.

- **One Drive**

  Usługa dysku chmurowego oferowana przez firmę Microsoft. Przechowywaliśmy tam dokumenty oraz obrazy diagramów.

- **Azure Blob Storage**

  To rozwiązanie chmurowe Microsoft, służące do bezpiecznego przechowywania dużej ilości danych nieustrukturyzowanych, takich jak pliki multimedialne, dokumenty czy kopie zapasowe. Dane są dostępne poprzez interfejs REST API usługi Azure Storage. Wykorzystaliśmy go do przechowywania zdjęć profilowych użytkownika oraz multimedii (zdjęcia i filmy) ze spotów i forum.

- **Jira**

  To narzędzie firmy Atlassian do zarządzania pracami nad projektem w metodykach zwinnych. Do Backlogu wpisywaliśmy zadania, a na tablicy Kanbanowej rejestrowaliśmy ich statusy oraz poświęcony czas.

- **GitHub**

  Zdalne repozytorium służące do przechowywania i wersjonowania kodu aplikacji. Zamieściliśmy tam kod naszego projektu. Do każdego zadania tworzyliśmy osobną gałąź z właściwą nazwą, a po zakończeniu prac przeprowadzaliśmy review kodu. Następnie łączyliśmy ją do głównej gałęzi deweloperskiej.

- **GitHub Actions**

To narzędzie do implementacji procesów CI/CD na platformie GitHub, które umożliwiają automatyczne testowanie lub wdrażanie kodu. Uruchamiają się w reakcji na różne operacje w repozytorium, na przykład przesłanie zmian na wybraną gałąź. Stosowaliśmy je do automatycznego testowania i budowania projektu po każdorazowym wprowadzeniu zmian.

- **GitHub Copilot**

  To narzędzie sztucznej inteligencji będące asystentem programisty. W projekcie analizuje plik oraz pliki powiązane. Wykorzystywaliśmy go podczas review codu. Copilot skanował wszystkie pliki i w komentarzach opisywał sugerowane zmiany lub potencjalne błędy.

- **Discord**

  Darmowa platforma komunikacyjna. Umożliwia udostępnienie obrazu z ekranu, komunikację głosową oraz tekstową, jak i również przesyłanie plików. Stosowaliśmy go do spotkań, na których omawialiśmy sprawy dotyczące projektu.

- **Messenger**

  Komunikator będący usługą Facebooka. Daje możliwość tworzenia czatów grupowych lub prywatnych, a także udostępniania plików. Używaliśmy go do ustalania spotkań na Discordzie oraz szybkiej komunikacji.

- **Postman**

  To narzędzie służące do testowania endpointów API. Pozwala grupować zapytania w kolekcje, wysyłać ich różne typy oraz analizować odpowiedzi z serwera. Wykorzystywaliśmy go do testowania stworzonych endpointów oraz debugowania.

- **Figma**

  Narzędzie chmurowe do projektowania interfejsów użytkownika (UI). Umożliwia zespołowe tworzenie w pełni interaktywnych prototypów. Wykonaliśmy w nim projekty ekranów naszej aplikacji.

- **Visual Paradigm**

  To narzędzie do tworzenia różnych diagramów stosowanych w inżynierii oprogramowania, takich jak UML([3]) czy BPMN([4]). Zrobiliśmy w nim diagram przypadków użycia.

3.4 Zasoby i ograniczenia

3.4.1 Zasoby

- **Specjalizacja członków zespołu** — wszyscy członkowie zespołu projektowego specjalizują się w aplikacjach internetowych.
- **Dostęp do przedstawiciela grupy docelowej** — jeden z członków zespołu (Adam) jest droniarzem foto/video.
- **Status studenta** — fakt bycia studentem zapewnia dostęp do wersji premium wielu usług (Figma Education, GitHub PRO).
- **Oprogramowanie zapewniane przez PJATK** - uczelnia zapewnia dostęp do pakietu JetBrains oraz usług firmy Microsoft (OneDrive).

3.4.2 Ograniczenia

- **Ograniczenia czasowe** — projekt jest ograniczony harmonogramem akademickim i terminem oddania pracy dyplomowej, co wymagało wysokiego tempa realizacji oraz sprawnej komunikacji w zespole.
- **Ograniczenia budżetowe** — projekt nie posiada finansowania i w związku z tym korzystano z rozwiązań darmowych oraz open source.

3.4.3 Usługi zewnętrzne

Usługa	GitHub Actions (CI) [5]

Opis	Uruchomienia pipeline'ów CI/CD dla repozytorium GitHub.
Limit	3000 min/mies.

Tabela 3.1: Usługa zewnętrzna: GitHub Actions (CI)

Usługa	Azure Blob Storage [6]
Opis	Magazyn plików (m.in. zdjęcia spotów, załączniki z czatu).
Limit	1 GB/mies.

Tabela 3.2: Usługa zewnętrzna: Azure Blob Storage

Usługa	Mailtrap [7]
Opis	Środowisko testowe SMTP oraz Email API do wysyłki maili.
Limit	150 maili/dzień

Tabela 3.3: Usługa zewnętrzna: Mailtrap

Usługa	LocationIQ [8]
Opis	Geokodowanie adresu przy dodawaniu nowych spotów.
Limit	5 000 zapytań/dzień

Tabela 3.4: Usługa zewnętrzna: LocationIQ

Usługa	Google Maps (Maps URLs) [9]
Opis	Otwieranie nawigacji w aplikacji Map Google (deep link/URL).
Limit	Brak limitu w ramach dokumentowanego sposobu użycia.

Tabela 3.5: Usługa zewnętrzna: Google Maps (Maps URLs)

Usługa	OpenFreeMap [10, 11]
Opis	Publiczny serwer kafelków do renderu mapy na froncie.
Limit	30 000 zapytań/mies.

Tabela 3.6: Usługa zewnętrzna: OpenFreeMap

Usługa	Open-Meteo [12]
Opis	Prognozy pogody wyświetlane dla spotów.
Limit	10 000 zapytań/dzień

Tabela 3.7: Usługa zewnętrzna: Open-Meteo

Usługa	Tenor GIF API [13]
Opis	Wyszukiwanie GIF-ów w czacie.
Limit	1 zapytanie na sekundę; brak ogólnego limitu dziennego.

Tabela 3.8: Usługa zewnętrzna: Tenor GIF API

Usługa	Where the ISS at? [14]
Opis	HTTP API z bieżącą pozycją satelity, używane pomocniczo.
Limit	1 zapytanie na sekundę; brak ogólnego limitu dziennego.

Tabela 3.9: Usługa zewnętrzna: Where the ISS at?

3.5 Analiza ryzyka

Rozdział 4

Analiza wymagań

W niniejszym rozdziale przedstawiono analizę wymagań stawianych aplikacji. Punktem wyjścia była identyfikacja i opracowanie przypadków użycia systemu (zarówno w postaci diagramów, jak i szczegółowych scenariuszy), które stanowią podstawę do dalszej specyfikacji zachowania systemu.

Na tej podstawie zdefiniowano kolejno:

- wymagania ogólne oraz dziedzinowe,
- wymagania funkcjonalne,
- wymagania pozafunkcjonalne,
- wymagania dotyczące interfejsu z otoczeniem,
- wymagania na środowisko docelowe.

Do określania priorytetów realizacji poszczególnych wymagań wykorzystano technikę MoSCoW. Metoda ta wyróżnia cztery kategorie:

M (Must have) wymagania kluczowe, które muszą zostać zrealizowane, aby system mógł zostać uznany za działający poprawnie;

S (Should have) wymagania bardzo istotne, jednak niewpływające bezpośrednio na minimalną zdolność operacyjną systemu;

C (Could have) wymagania opcjonalne, podnoszące wygodę użytkowania lub wartość biznesową rozwiązania;

W (Won't have this time) wymagania odłożone na przyszłość, które w bieżącej iteracji nie będą realizowane.

Tak przeprowadzona analiza pozwala w sposób uporządkowany opisać zarówno zakres funkcjonalny systemu, jak i ograniczenia oraz oczekiwania niefunkcjonalne, a także zewnętrzne uwarunkowania jego działania.

4.1 Przypadki użycia

4.1.1 Aktorzy

Użytkownik niezalogowany Gość przeglądający publiczne treści (mapa, spoty, forum): może się zarejestrować lub zalogować.

Użytkownik (nie premium) Zarejestrowany użytkownik: zarządza kontem i ulubionymi spotami, dodaje treści/komentarze, korzysta z czatu.

Użytkownik premium Użytkownik z wykupioną subskrypcją: ma dostęp do funkcji premium (np. oznaczenie stref PANSA na mapie, rozbudowana prognoza pogody).

Moderator Moderacja treści: posty na forum, komentarze spotów.

Deweloper Rozwija i utrzymuje system.

Aktorzy będący zewnętrznymi usługami Poniżej wymieniono aktorów, których opisy zamieszczono w rozdziale poświęconym integracji z zewnętrznymi usługami 3.4.3.

- Usługa mailowa (Mailtrap)
- Dostawca API do map (OpenFreeMap)
- Nawigacja (Google Maps)

- Dostawca API pogodowego (Open-Meteo)
- Dostawca API GIFów (Tenor)
- Dostawca API do określania strefy czasowej spota ("Where the ISS at?")
- Dostawca API do geolokalizacji (LocationIQ)
- Azure Blob Storage
- Dostawca OAuth (Google)
- Dostawca OAuth (GitHub)

4.1.2 Diagram przypadków użycia

4.1.3 Scenariusze przypadków użycia

Scenariusze przypadków użycia – funkcje ogólne

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU1
Nazwa:	Rejestracja użytkownika
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik zakłada konto poprzez formularz rejestracji.
Warunki wstępne:	Użytkownik znajduje się na stronie z formularzem rejestracji.
Warunki końcowe:	Użytkownik posiada konto w systemie.

Główny przepływ zdarzeń:	1. Użytkownik wypełnia formularz rejestracyjny. 2. Użytkownik naciska przycisk rejestracji. 3. System tworzy konto użytkownika. 4. System loguje użytkownika i przenosi go na ostatnio dowiedzoną podstronę.
Alternatywne przepływy zdarzeń:	1a. Podane dane są niepoprawne – system wyświetla komunikat o błędzie oraz podświetla pola wymagające poprawy. 2a. Nazwa użytkownika jest już zajęta – system wyświetla komunikat o błędzie. 2b. Adres email jest już zajęty – system wyświetla komunikat o błędzie.

Tabela 4.1: Scenariusz przypadku użycia: Rejestracja użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU2
Nazwa:	Logowanie użytkownika
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik loguje się do systemu, podając login i hasło.
Warunki wstępne:	Użytkownik znajduje się na stronie logowania.
Warunki końcowe:	Użytkownik jest zalogowany i przeniesiony na ostatnio dowiedzoną podstronę.

Główny przepływ zdarzeń:	1. Użytkownik wypełnia formularz logowania. 2. Użytkownik naciska przycisk logowania. 3. System loguje użytkownika i przenosi go na ostatnio dowiedzoną podstronę.
Alternatywne przepływy zdarzeń:	2a. Podane dane są niepoprawne – system wyświetla komunikat o błędzie.

Tabela 4.2: Scenariusz przypadku użycia: Logowanie użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU3
Nazwa:	Wykupienie subskrypcji premium
Priorytet:	Niski
Aktorzy:	Użytkownik zalogowany, Bramka płatnicza, System finansowo-księgowy
Opis:	Użytkownik opłaca subskrypcję premium w celu uzyskania dodatkowych funkcji.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w module subskrypcji.
Warunki końcowe:	Subskrypcja premium jest aktywna, a użytkownik ma dostęp do funkcji premium.

Główny przepływ zdarzeń:	1. Użytkownik wybiera plan subskrypcji. 2. Użytkownik przechodzi do bramki płatniczej. 3. Użytkownik podaje dane płatnicze i zatwierdza transakcję. 4. Bramka płatnicza przetwarza płatność i zwraca wynik do systemu. 5. System zapisuje informację o opłaconej subskrypcji i aktualizuje uprawnienia. 6. System generuje wpis w systemie finansowo-księgowym.
Alternatywne przepływy zdarzeń:	4a. Płatność nie powiodła się – system informuje użytkownika i umożliwia ponowną próbę. 5a. W czasie aktualizacji subskrypcji wystąpił błąd – system cofa zmiany i wyświetla komunikat o problemie.

Tabela 4.3: Scenariusz przypadku użycia: Wykupienie subskrypcji premium

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU4
Nazwa:	Resetowanie hasła
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany, Usługa SMTP
Opis:	Użytkownik inicjuje reset hasła, aby odzyskać dostęp do konta.
Warunki wstępne:	Użytkownik znajduje się na ekranie resetu hasła.
Warunki końcowe:	Użytkownik otrzymuje wiadomość e-mail z linkiem do ustawienia nowego hasła.

Główny przepływ zdarzeń:	1. Użytkownik wpisuje adres e-mail powiązany z kontem. 2. Użytkownik zatwierdza żądanie resetu hasła. 3. System generuje token resetu hasła. 4. System wysyła e-mail z linkiem do zmiany hasła.
Alternatywne przepływy zdarzeń:	2a. Nie istnieje konto dla podanego adresu – system wyświetla komunikat o błędzie. 4a. Występuje błąd połączenia z usługą SMTP – system informuje użytkownika o problemie technicznym.

Tabela 4.4: Scenariusz przypadku użycia: Resetowanie hasła

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU5
Nazwa:	Zmiana hasła w ustawieniach konta
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik zmienia hasło do konta z poziomu ustawień profilu.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się na ekranie zmiany danych konta.
Warunki końcowe:	Hasło do konta użytkownika zostało zaktualizowane.

Główny przepływ zdarzeń:	1. Użytkownik wpisuje aktualne hasło. 2. Użytkownik wpisuje nowe hasło i powtarza je. 3. Użytkownik zatwierdza formularz zmiany hasła. 4. System zapisuje nowe hasło i informuje o powodzeniu operacji.
Alternatywne przepływy zdarzeń:	3a. Aktualne hasło jest nieprawidłowe – system wyświetla komunikat i nie zapisuje zmian. 3b. Nowe hasło nie spełnia wymagań bezpieczeństwa – system informuje o błędzie i podświetla pola do poprawy.

Tabela 4.5: Scenariusz przypadku użycia: Zmiana hasła w ustawieniach konta

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU6
Nazwa:	Wylogowanie użytkownika
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik wylogowuje się z aplikacji.
Warunki wstępne:	Użytkownik jest zalogowany.
Warunki końcowe:	Sesja użytkownika została zakończona, użytkownik widzi stronę główną dla niezalogowanych.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję wylogowania z menu. 2. System unieważnia token dostępu użytkownika. 3. System przenosi użytkownika na stronę główną aplikacji.

Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.6: Scenariusz przypadku użycia: Wylogowanie użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU7
Nazwa:	Przeglądanie powiadomień
Priorytet:	Niski
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik przegląda listę powiadomień.
Warunki wstępne:	Użytkownik jest na ekranie centra powiadomień.
Warunki końcowe:	Powiadomienia zostały wyświetlone, a wybrane oznaczone jako przeczytane.
Główny przepływ zdarzeń:	1. System wyświetla powiadomienia w odwróconym porządku chronologicznym. 2. Użytkownik otwiera wybrane powiadomienie. 3. System oznacza powiadomienie jako przeczytane i ewentualnie przenosi użytkownika do powiązanego widoku.
Alternatywne przepływy zdarzeń:	1a. System nie może pobrać powiadomień (błąd serwera) – użytkownik otrzymuje komunikat o błędzie i może spróbować ponownie.

Tabela 4.7: Scenariusz przypadku użycia: Przeglądanie powiadomień

Scenariusze przypadków użycia dla funkcji premium

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU8
Nazwa:	Przeszukiwanie historii czatu
Priorytet:	Niski
Aktorzy:	Użytkownik premium
Opis:	Użytkownik wyszukuje konkretne wiadomości w historii czatu.
Warunki wstępne:	Użytkownik jest zalogowany jako użytkownik premium i znajduje się w widoku czatu.
Warunki końcowe:	Wiadomości spełniające kryteria wyszukiwania zostały wyświetlone.
Główny przepływ zdarzeń:	1. Użytkownik otwiera pole wyszukiwania historii w czacie. 2. Użytkownik wpisuje frazę lub filtr (np. zakres dat, autor). 3. System filtruje wiadomości zgodnie z kryteriami. 4. System prezentuje listę dopasowanych fragmentów rozmowy.
Alternatywne przepływy zdarzeń:	4a. Nie znaleziono wiadomości spełniających kryteria – system informuje o braku wyników wyszukwiania.

Tabela 4.8: Scenariusz przypadku użycia: Przeszukiwanie historii czatu

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU9
Nazwa:	Przeglądanie wysłanych plików na czacie

Priorytet:	Niski
Aktorzy:	Użytkownik premium, Usługa do przechowywania plików w chmurze
Opis:	Użytkownik przegląda listę plików wysłanych w ramach czatów.
Warunki wstępne:	Użytkownik jest zalogowany jako użytkownik premium.
Warunki końcowe:	Użytkownik widzi listę wysłanych plików i może przechodzić do powiązanych czatów.
Główny przepływ zdarzeń:	1. Użytkownik otwiera sekcję „Wysłane pliki”. 2. System pobiera metadane plików z usługi przechowywania. 3. System wyświetla listę plików z podstawowymi informacjami (nazwa, typ, data). 4. Użytkownik wybiera plik, aby otworzyć go lub przejść do powiązanego czatu.
Alternatywne przepływy zdarzeń:	4a. Doszło do błędu podczas pobierania pliku - system wyświetla odpowiedni komunikat.

Tabela 4.9: Scenariusz przypadku użycia: Przeglądanie wysłanych plików na czacie

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU10
Nazwa:	Zmiana typu mapy
Priorytet:	Niski

Aktorzy:	Użytkownik premium, Usługa do wyświetlania mapy
Opis:	Użytkownik zmienia typ mapy (np. standardowa, satelitarna, hybrydowa).
Warunki wstępne:	Użytkownik premium jest na ekranie mapy.
Warunki końcowe:	Mapa jest wyświetlana w wybranym typie.
Główny przepływ zdarzeń:	1. Użytkownik otwiera ustawienia widoku mapy. 2. Użytkownik wybiera typ mapy z dostępnej listy. 3. System przełącza widok mapy na wybrany typ.
Alternatywne przepływy zdarzeń:	3a. Wybrany typ mapy nie jest dostępny (błąd usługi mapowej) – system przywraca poprzedni typ i informuje o błędzie.

Tabela 4.10: Scenariusz przypadku użycia: Zmiana typu mapy

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU11
Nazwa:	Przeglądanie stref PANSA
Priorytet:	Niski
Aktorzy:	Użytkownik premium, Usługa do wyświetlania mapy
Opis:	Użytkownik wyświetla na mapie strefy przestrzeni powietrznej PANSA.
Warunki wstępne:	Użytkownik premium ma otwarty moduł mapy.

Warunki końcowe:	Strefy PANSA zostały zwizualizowane na mapie.
Główny przepływ zdarzeń:	1. Użytkownik włącza warstwę „Strefy PANSA”. 2. System pobiera dane o strefach. 3. System nakłada kontury stref na mapę.
Alternatywne przepływy zdarzeń:	2a. Dane o strefach są chwilowo niedostępne – system komunikuje problem i nie włącza warstwy.

Tabela 4.11: Scenariusz przypadku użycia: Przeglądanie stref PANSA

Scenariusze przypadków użycia dla wyszukiwarki

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU12
Nazwa:	Wyszukiwanie spota w globalnej wyszukiwarce
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany, Usługa do wyświetlania mapy, Usługa do pogody
Opis:	Użytkownik wyszukuje spoty za pomocą globalnej wyszukiwarki w aplikacji.
Warunki wstępne:	Użytkownik znajduje się na stronie głównej z wyszukwiarką.
Warunki końcowe:	Użytkownik otrzymuje listę znalezionych spotów.

Główny przepływ zdarzeń:	1. Użytkownik wpisuje frazę w globalnej wyszukiwarce. 2. System wyszukuje spoty spełniające kryteria. 3. System wyświetla listę wyników.
Alternatywne przepływy zdarzeń:	3a. System informuje o braku wyników spełniających kryteria.

Tabela 4.12: Scenariusz przypadku użycia: Wyszukiwanie spota w globalnej wyszukiwarce

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU13
Nazwa:	Przejście do spota na mapie z wyszukiwarki
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik przechodzi z wyników wyszukiwarki do widoku mapy ustawionego na konkretny spot.
Warunki wstępne:	Wyświetlona jest lista wyników wyszukiwania spotów.
Warunki końcowe:	Mapa jest przybliżona do wybranego spota, a jego szczegóły są dostępne.
Główny przepływ zdarzeń:	1. Użytkownik wybiera spota z listy wyników. 2. System przełącza widok na moduł mapy. 3. System ustawia mapę na lokalizację spota i otwiera jego szczegóły.

Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.13: Scenariusz przypadku użycia: Przejście do spota na mapie z wyszukiwarki

Scenariusze przypadków użycia dla mapy

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU14
Nazwa:	Przeglądanie mapy spotów
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany, Usługa do wyświetlania mapy
Opis:	Użytkownik przegląda mapę spotów.
Warunki wstępne:	Użytkownik znajduje się w module mapy.
Warunki końcowe:	Mapa ze spotami została wyświetlona, a użytkownik może przybliżać, oddalać i przesuwać widok.
Główny przepływ zdarzeń:	1. System inicjuje widok mapy z domyślnym obszarem. 2. System pobiera listę spotów. 3. System rysuje znaczniki spotów na mapie. 4. Użytkownik przesuwa lub skaluje mapę. 5. System pobiera spoty dla nowego zakresu.
Alternatywne przepływy zdarzeń:	2a. Usługa mapy jest niedostępna – system wyświetla komunikat o błędzie.

Tabela 4.14: Scenariusz przypadku użycia: Przeglądanie mapy spotów

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU15
Nazwa:	Otwarcie szczegółów spota
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany, Użytkownik zalogowany
Opis:	Użytkownik otwiera widok szczegółów wybranego spota.
Warunki wstępne:	Użytkownik widzi mapę spotów.
Warunki końcowe:	Wyświetlony został widok szczegółów spota z podstawowymi informacjami oraz jego lokalizacją na mapie.
Główny przepływ zdarzeń:	1. Użytkownik wybiera spota z mapy. 2. System pobiera dane szczegółowe spota (informacje opisowe, lokalizacja). 3. System otwiera widok szczegółów spota. 4. System prezentuje informacje o spocie oraz mapę przybliżoną do jego lokalizacji.
Alternatywne przepływy zdarzeń:	2a. Spot nie istnieje (został usunięty lub ukryty) – system informuje użytkownika i powraca do poprzedniego widoku. 2b. Wystąpił błąd podczas pobierania danych spota – system wyświetla komunikat o błędzie i umożliwia ponowną próbę.

Tabela 4.15: Scenariusz przypadku użycia: Otwarcie szczegółów spota

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU16
Nazwa:	Przeglądanie komentarzy do spota

Priorytet:	Średni
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik czyta komentarze pod wybranym spotem.
Warunki wstępne:	Wyświetlany jest widok szczegółów spota.
Warunki końcowe:	Lista komentarzy do spota została wyświetlona.
Główny przepływ zdarzeń:	1. System pobiera komentarze powiązane ze spotem. 2. System wyświetla komentarze w kolejności chronologicznej lub według popularności. 3. Użytkownik przewija listę komentarzy.
Alternatywne przepływy zdarzeń:	1a. Spot nie ma jeszcze komentarzy – system wyświetla odpowiednią informację.

Tabela 4.16: Scenariusz przypadku użycia: Przeglądanie komentarzy do spota

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU17
Nazwa:	Przeglądanie pogody na spocie
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany, Usługa danych pogodowych
Opis:	Użytkownik sprawdza prognozę pogody dla lokalizacji spota.
Warunki wstępne:	Wyświetlany jest widok szczegółów spota.

Warunki końcowe:	Prognoza pogody dla spota została wyświetlona.
Główny przepływ zdarzeń:	1. Użytkownik otwiera zakładkę pogody. 2. System wysyła zapytanie do usługi pogodowej z lokalizacją spota. 3. System odbiera prognozę i prezentuje ją (temperatura, prędkość wiatru, opady).
Alternatywne przepływy zdarzeń:	2a. Usługa pogodowa jest niedostępna – system wyświetla komunikat o braku danych pogodowych.

Tabela 4.17: Scenariusz przypadku użycia: Przeglądanie pogody na spocie

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU18
Nazwa:	Wyszukiwanie spota na mapie
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik wyszukuje spota po nazwie korzystając z pola wyszukiwania na mapie.
Warunki wstępne:	Użytkownik widzi mapę spotów.
Warunki końcowe:	Mapa zostaje ustawiona na wybranego spota lub listę dopasowań.

Główny przepływ zdarzeń:	1. Użytkownik wpisuje frazę w polu wyszukiwania na mapie. 2. System podpowiada listę pasujących spotów. 3. Użytkownik wybiera spota z listy. 4. System przybliża mapę do wybranego spota i podświetla jego znacznik.
Alternatywne przepływy zdarzeń:	2a. Brak wyników dla podanej frazy – system informuje użytkownika o braku dopasowań.

Tabela 4.18: Scenariusz przypadku użycia: Wyszukiwanie spota na mapie

Scenariusze przypadków użycia dla czatu

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU19
Nazwa:	Utworzenie prywatnego czatu
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik tworzy prywatną konwersację z innym użytkownikiem.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w zakładce społeczność.
Warunki końcowe:	Nowy czat prywatny został utworzony i wyświetlony użytkownikowi.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję utworzenia nowego czatu. 2. System tworzy nowy czat (jeśli nie istnieje). 3. System otwiera widok nowego czatu.

Alternatywne przepływy zdarzeń:	1a. Taki czat już istnieje – system zamiast tworzyć nowy, otwiera istniejącą konwersację.

Tabela 4.19: Scenariusz przypadku użycia: Utworzenie prywatnego czatu

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU20
Nazwa:	Otworzenie czatu
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik otwiera wybrany czat, aby wyświetlić historię rozmowy i móc wysyłać kolejne wiadomości.
Warunki wstępne:	Użytkownik jest zalogowany i widzi listę swoich czatów lub otrzymał powiadomienie prowadzące do czatu.
Warunki końcowe:	Wybrany czat został otworzony, a historia rozmowy jest widoczna dla użytkownika.
Główny przepływ zdarzeń:	1. Użytkownik wybiera czat z listy czatów lub z powiadomienia. 2. System pobiera dane czatu (uczestników, ostatnie wiadomości). 3. System oznacza nieprzeczytane wiadomości na czacie jako przeczytane. 4. System wyświetla widok czatu wraz z historią rozmowy.

Alternatywne przepływy zdarzeń:	2a. Czat nie jest już dostępny (np. został usunięty lub użytkownik utracił do niego dostęp) – system wyświetla komunikat o braku dostępu i powraca do listy czatów. 2b. Wystąpił błąd podczas pobierania danych czatu – system wyświetla komunikat o błędzie i umożliwia ponowną próbę.

Tabela 4.20: Scenariusz przypadku użycia: Otworzenie czatu

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU21
Nazwa:	Utworzenie czatu grupowego
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik tworzy nowy czat grupowy z kilkoma uczestnikami.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się na dowolnym czacie prywatnym.
Warunki końcowe:	Czat grupowy został utworzony i wyświetlony na ekranie.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję utworzenia czatu grupowego. 2. Użytkownik wybiera uczestników grupy. 3. Użytkownik zatwierdza utworzenie czatu. 4. System tworzy czat grupowy i dodaje do niego wskazanych użytkowników. 5. System otwiera widok nowego czatu grupowego.

Alternatywne przepływy zdarzeń:	3a. System nie może utworzyć czatu – aplikacja informuje o błędzie.

Tabela 4.21: Scenariusz przypadku użycia: Utworzenie czatu grupowego

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU22
Nazwa:	Przeglądanie listy czatów
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik przegląda listę swoich czatów prywatnych i grupowych.
Warunki wstępne:	Użytkownik jest zalogowany i otwiera moduł czatu.
Warunki końcowe:	Lista czatów użytkownika została wyświetlona.
Główny przepływ zdarzeń:	1. System pobiera listę czatów użytkownika. 2. System wyświetla listę czatów z podstawowymi informacjami. 3. Użytkownik wybiera czat z listy. 4. System otwiera widok wybranego czatu.
Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.22: Scenariusz przypadku użycia: Przeglądanie listy czatów

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU23
Nazwa:	Wysyłanie wiadomości na czacie
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik wysyła wiadomość tekstową na czacie.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w widoku konkretnego czatu.
Warunki końcowe:	Nowa wiadomość jest zapisana i widoczna w historii czatu.
Główny przepływ zdarzeń:	1. Użytkownik wpisuje treść wiadomości. 2. Użytkownik wysyła wiadomość. 3. System zapisuje wiadomość i dostarcza ją do uczestników czatu. 4. System wyświetla wiadomość na liście wiadomości.
Alternatywne przepływy zdarzeń:	2a. Treść wiadomości jest pusta – system blokuje wysłanie i pozostaje w tym samym widoku.

Tabela 4.23: Scenariusz przypadku użycia: Wysyłanie wiadomości na czacie

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU24
Nazwa:	Wysyłanie GIF-a na czacie
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany, Usługa GIF-ów
Opis:	Użytkownik wysyła animację GIF w konwersacji czatowej.

Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w widoku czatu.
Warunki końcowe:	Wybrany GIF został dodany jako wiadomość w czacie.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję dodania GIF-a. 2. System otwiera okno wyszukiwarki GIF-ów. 3. Użytkownik wybiera lub wyszukuje GIF-a. 4. Użytkownik zatwierdza wysłanie GIF-a. 5. System dodaje GIF-a jako wiadomość na czacie.
Alternatywne przepływy zdarzeń:	2a. Usługa GIF-ów jest niedostępna – system informuje o braku możliwości wysłania GIF-a.

Tabela 4.24: Scenariusz przypadku użycia: Wysyłanie GIF-a na czacie

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU25
Nazwa:	Wysyłanie pliku na czacie
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany, Usługa do przechowywania plików w chmurze
Opis:	Użytkownik wysyła plik (np. zdjęcie, film) na czacie.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w widoku czatu.
Warunki końcowe:	Plik został zapisany w chmurze i powiązany z wiadomością na czacie.

Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję dodania pliku. 2. Użytkownik wybiera plik z urządzenia. 3. System przesyła plik do usługi przechowywania w chmurze. 4. System tworzy wiadomość z odnośnikiem do pliku. 5. System wyświetla wiadomość na liście czatu.
Alternatywne przepływy zdarzeń:	3a. Przesyłanie pliku nie powiodło się – system informuje użytkownika i umożliwia ponowną próbę.

Tabela 4.25: Scenariusz przypadku użycia: Wysyłanie pliku na czacie

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU26
Nazwa:	Edycja ustawień czatu
Priorytet:	Niski
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik modyfikuje ustawienia czatu (np. nazwę, avatar, tryb powiadomień).
Warunki wstępne:	Użytkownik jest zalogowany i ma uprawnienia do edycji danego czatu.
Warunki końcowe:	Zaktualizowane ustawienia czatu są zapisane i widoczne dla uczestników.
Główny przepływ zdarzeń:	1. Użytkownik otwiera panel ustawień czatu. 2. Użytkownik wprowadza zmiany (np. nazwę, opis, avatar). 3. Użytkownik zapisuje zmiany. 4. System waliduje dane i aktualizuje konfigurację czatu.

Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów poza walidacją pól.

Tabela 4.26: Scenariusz przypadku użycia: Edycja ustawień czatu

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU27
Nazwa:	Dodanie członka do czatu grupowego
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik z uprawnieniami administratora dodaje nowego uczestnika do czatu grupowego.
Warunki wstępne:	Użytkownik jest zalogowany, znajduje się w czacie grupowym i ma prawo zarządzać członkami.
Warunki końcowe:	Nowy uczestnik został dodany do czatu grupowego.
Główny przepływ zdarzeń:	1. Użytkownik otwiera listę uczestników czatu grupowego. 2. Użytkownik wybiera opcję dodania nowego członka. 3. Użytkownik wskazuje użytkownika do dodania i zatwierdza wybór. 4. System dodaje wskazanego użytkownika do czatu grupowego.
Alternatywne przepływy zdarzeń:	3a. Operacja nie powiodła się – system informuje o błędzie.

Tabela 4.27: Scenariusz przypadku użycia: Dodanie członka do czatu grupowego

Scenariusze przypadków użycia dla forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU28
Nazwa:	Przeglądanie postów na forum
Priorytet:	Wysoki
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik przegląda listę postów na forum.
Warunki wstępne:	Użytkownik znajduje się w module forum.
Warunki końcowe:	Lista postów forum jest wyświetlona, a użytkownik może przechodzić do szczegółów.
Główny przepływ zdarzeń:	1. System pobiera listę postów. 2. System wyświetla posty z podstawowymi informacjami. 3. Użytkownik wybiera post, który chce przeczytać. 4. System otwiera szczegółowy widok posta.
Alternatywne przepływy zdarzeń:	3a. System nie może pobrać szczegółów posta – system wyświetla komunikat o błędzie.

Tabela 4.28: Scenariusz przypadku użycia: Przeglądanie postów na forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU29
Nazwa:	Dodanie posta na forum
Priorytet:	Wysoki

Aktorzy:	Użytkownik zalogowany, Usługa do przechowywania plików w chmurze
Opis:	Użytkownik publikuje nowy post na forum.
Warunki wstępne:	Użytkownik znajduje się w module forum.
Warunki końcowe:	Nowy post jest widoczny na forum.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję dodania nowego posta. 2. Użytkownik wpisuje tytuł i treść posta. 3. (Opcjonalnie) Użytkownik dodaje załączniki (zdjęcia-/filmy) do posta. 4. Użytkownik publikuje posta. 5. System zapisuje posta (oraz załączniki w chmurze) i wyświetla go na liście postów.
Alternatywne przepływy zdarzeń:	3a. Załącznik nie może zostać zapisany – system informuje o błędzie i pozwala opublikować posta bez pliku. 4a. Formularz zawiera błędne lub niekompletne dane – system wyświetla komunikat i prosi o poprawę.

Tabela 4.29: Scenariusz przypadku użycia: Dodanie posta na forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU30
Nazwa:	Dodanie komentarza na forum
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik dodaje komentarz pod postem na forum.

Warunki wstępne:	Użytkownik jest zalogowany i widzi szczegóły posta.
Warunki końcowe:	Nowy komentarz został zapisany i widoczny pod postem.
Główny przepływ zdarzeń:	1. Użytkownik wpisuje treść komentarza w formularzu pod postem. 2. Użytkownik publikuje komentarz. 3. System zapisuje komentarz i odświeża listę komentarzy.
Alternatywne przepływy zdarzeń:	2a. Treść komentarza jest niepoprawna – system wyświetla komunikat o błędzie.

Tabela 4.30: Scenariusz przypadku użycia: Dodanie komentarza na forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU31
Nazwa:	Przeglądanie historii interakcji z postami
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik przegląda historię swoich aktywności na forum (dodane posty, komentarze, reakcje).
Warunki wstępne:	Użytkownik jest zalogowany.
Warunki końcowe:	Lista interakcji użytkownika z postami jest wyświetlona.

Główny przepływ zdarzeń:	1. Użytkownik przechodzi do sekcji historii aktywności. 2. System pobiera historię interakcji użytkownika. 3. System wyświetla listę interakcji z możliwością filtrowania.
Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.31: Scenariusz przypadku użycia: Przeglądanie historii interakcji z postami

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU32
Nazwa:	Zarządzanie komentarzami na forum
Priorytet:	Niski
Aktorzy:	Użytkownik zalogowany (autor posta lub moderator)
Opis:	Użytkownik zarządza komentarzami pod postami forum (edycja, usuwanie, przypinanie).
Warunki wstępne:	Użytkownik jest zalogowany i ma dostęp do danego wątku forum.
Warunki końcowe:	Komentarze zostały zaktualizowane zgodnie z działaniami użytkownika.
Główny przepływ zdarzeń:	1. Użytkownik otwiera widok komentarzy pod postem. 2. Użytkownik wybiera komentarz i odpowiednią akcję. 3. System weryfikuje uprawnienia użytkownika. 4. System wykonuje wybraną akcję i aktualizuje widok.

Alternatywne przepływy zdarzeń:	3a. Użytkownik nie ma wymaganych uprawnień – system blokuje operację i informuje o tym.

Tabela 4.32: Scenariusz przypadku użycia: Zarządzanie komentarzami na forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU33
Nazwa:	Zgłoszenie komentarza naruszającego regulamin
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik zgłasza komentarz na forum.
Warunki wstępne:	Użytkownik widzi komentarz w aplikacji.
Warunki końcowe:	Zgłoszenie komentarza zostało zapisane i trafiło do kolejki moderacyjnej.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję „Zgłoś komentarz”. 2. Użytkownik określa powód zgłoszenia. 3. System zapisuje zgłoszenie i wiąże je z komentarzem i zgłaszającym.
Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.33: Scenariusz przypadku użycia: Zgłoszenie komentarza naruszającego regulamin

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU34
Nazwa:	Zgłoszenie posta na forum
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik zgłasza post forum jako naruszający regulamin lub tematykę.
Warunki wstępne:	Wyświetlony jest widok posta na forum.
Warunki końcowe:	Zgłoszenie posta zostało zapisane i przekazane moderatorom.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję „Zgłoś post”. 2. Użytkownik wybiera kategorię naruszenia i potwierdza zgłoszenie. 3. System zapisuje zgłoszenie i oznacza post jako zgłoszony.
Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.34: Scenariusz przypadku użycia: Zgłoszenie posta na forum

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU35
Nazwa:	Przeglądanie komentarzy pod postem
Priorytet:	Średni
Aktorzy:	Użytkownik niezalogowany, Użytkownik zalogowany

Opis:	Użytkownik przegląda komentarze dodane pod wybranym postem na forum.
Warunki wstępne:	Wyświetlany jest szczegółowy widok posta na forum.
Warunki końcowe:	Lista komentarzy powiązanych z postem została wyświetlona.
Główny przepływ zdarzeń:	1. System pobiera komentarze powiązane z wybranym postem. 2. System wyświetla komentarze (np. w kolejności chronologicznej lub według popularności). 3. Użytkownik przewija listę komentarzy i zapoznaje się z ich treścią.
Alternatywne przepływy zdarzeń:	1a. Post nie ma jeszcze komentarzy – system wyświetla informację o braku komentarzy. 1b. Wystąpił błąd podczas pobierania komentarzy – system wyświetla komunikat o błędzie i umożliwia ponowną próbę.

Tabela 4.35: Scenariusz przypadku użycia: Przeglądanie komentarzy pod postem

Scenariusze przypadków użycia dla panelu użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU36
Nazwa:	Dodanie spota w panelu użytkownika
Priorytet:	Wysoki

Aktorzy:	Użytkownik zalogowany, Usługa do wyświetlania mapy, Usługa do przechowywania plików w chmurze
Opis:	Użytkownik dodaje nowy spot poprzez panel.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w widoku panelu użytkownika.
Warunki końcowe:	Nowy spot został zapisany i jest widoczny na mapie oraz w panelu użytkownika.
Główny przepływ zdarzeń:	1. Użytkownik wybiera opcję „Dodaj spota”. 2. Użytkownik uzupełnia podstawowe informacje o spocie (nazwa, opis, tagi). 3. Użytkownik wskazuje lokalizację spota na mapie. 4. Użytkownik dodaje zdjęcia/filmy do spota. 5. Użytkownik zapisuje spota. 6. System zapisuje dane spota (oraz pliki w chmurze) i aktualizuje mapę.
Alternatywne przepływy zdarzeń:	3a. Podane dane wejściowe są niepoprawne – system wyświetla komunikat i zaznacza wymagające poprawy pola.

Tabela 4.36: Scenariusz przypadku użycia: Dodanie spota w panelu użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU37
Nazwa:	Przeglądanie profilu użytkownika
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany

Opis:	Użytkownik przegląda swój profil (lista spotów, media, podstawowe dane).
Warunki wstępne:	Użytkownik jest zalogowany.
Warunki końcowe:	Wyświetlony jest widok profilu użytkownika wraz z jego zawartością.
Główny przepływ zdarzeń:	1. Użytkownik otwiera swój profil. 2. System pobiera dane profilu (informacje podstawowe, spoty, media). 3. System wyświetla dane w odpowiednich sekcjach (spoty, zdjęcia, filmy, komentarze).
Alternatywne przepływy zdarzeń:	2a. Wystąpił błąd podczas pobierania danych użytkownika – system wyświetla informację o błędzie.

Tabela 4.37: Scenariusz przypadku użycia: Przeglądanie profilu użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU38
Nazwa:	Przeglądanie profilu innego użytkownika
Priorytet:	Średni
Aktorzy:	Użytkownik niezalogowany
Opis:	Użytkownik ogląda profil innego użytkownika (np. z mapy, forum lub społeczności).
Warunki wstępne:	Użytkownik jest zalogowany i ma dostęp do odnośnika do profilu innego użytkownika.

Warunki końcowe:	Profil innego użytkownika został wyświetlony.
Główny przepływ zdarzeń:	1. Użytkownik wybiera odnośnik do profilu innego użytkownika. 2. System pobiera dane profilu docelowego użytkownika. 3. System wyświetla profil (media, podstawowe informacje).
Alternatywne przepływy zdarzeń:	2a. Wystąpił błąd podczas pobierania danych użytkownika – system wyświetla informację o błędzie.

Tabela 4.38: Scenariusz przypadku użycia: Przeglądanie profilu innego użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU39
Nazwa:	Dodanie użytkownika do znajomych
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik wysyła lub akceptuje zaproszenie do znajomych.
Warunki wstępne:	Użytkownik jest zalogowany i przegląda profil innego użytkownika.
Warunki końcowe:	Relacja „znajomy” została utworzona lub zaproszenie czeka na akceptację.
Główny przepływ zdarzeń:	1. Użytkownik klika przycisk „Dodaj do znajomych”. 2. System sprawdza, czy relacja już istnieje. 3. System tworzy nowe zaproszenie. 4. System informuje o statusie o wysłaniu zaproszenia.

Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.39: Scenariusz przypadku użycia: Dodanie użytkownika do znajomych

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU40
Nazwa:	Przeglądanie społeczności
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik przegląda społeczności, grupy lub listy znajomych powiązane z aplikacją.
Warunki wstępne:	Użytkownik jest zalogowany.
Warunki końcowe:	Lista społeczności lub znajomych została wyświetlona.
Główny przepływ zdarzeń:	1. Użytkownik przechodzi do sekcji społeczności. 2. System pobiera listę społeczności i znajomych użytkownika. 3. System wyświetla listę z możliwością przechodzenia do profili i czatów.
Alternatywne przepływy zdarzeń:	Brak istotnych alternatywnych przepływów.

Tabela 4.40: Scenariusz przypadku użycia: Przeglądanie społeczności

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU41
Nazwa:	Przeglądanie dodanych spotów
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany, Usługa do wyświetlania mapy
Opis:	Użytkownik przegląda listę/siatkę spotów, które sam dodał.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w widoku panelu użytkownika lub sekcji „Moje spoty”.
Warunki końcowe:	Lista dodanych spotów użytkownika została wyświetlona (np. na mapie i/lub w formie listy).
Główny przepływ zdarzeń:	1. Użytkownik przechodzi do sekcji „Moje spoty”. 2. System pobiera listę spotów dodanych przez użytkownika. 3. System wyświetla listę spotów oraz znaczniki na mapie. 4. Użytkownik wybiera spota, aby przejść do jego szczegółów.
Alternatywne przepływy zdarzeń:	2a. Użytkownik nie dodał jeszcze żadnego spota – system wyświetla komunikat i proponuje dodanie pierwszego spota.

Tabela 4.41: Scenariusz przypadku użycia: Przeglądanie dodanych spotów

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU42
Nazwa:	Edycja danych użytkownika
Priorytet:	Wysoki
Aktorzy:	Użytkownik zalogowany

Opis:	Użytkownik modyfikuje swoje dane profilu (np. nazwę, opis, avatar).
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w widoku edycji profilu.
Warunki końcowe:	Zaktualizowane dane profilu są zapisane i widoczne w aplikacji.
Główny przepływ zdarzeń:	1. Użytkownik otwiera widok edycji profilu. 2. Użytkownik wprowadza zmiany w danych profilu. 3. Użytkownik zapisuje zmiany. 4. System waliduje dane i zapisuje zaktualizowany profil.
Alternatywne przepływy zdarzeń:	4a. Dane są niepoprawne lub niekompletne – system wyświetla komunikat o błędzie i zaznacza pola do poprawy.

Tabela 4.42: Scenariusz przypadku użycia: Edycja danych użytkownika

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU43
Nazwa:	Przeglądanie dodanych zdjęć do spotów
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany, Usługa do przechowywania plików w chmurze
Opis:	Użytkownik przegląda wszystkie zdjęcia powiązane ze spotami, które dodał.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w sekcji mediów (np. „Moje zdjęcia”).

Warunki końcowe:	Lista lub galeria zdjęć powiązanych ze spotami użytkownika została wyświetlona.
Główny przepływ zdarzeń:	1. Użytkownik otwiera sekcję przeglądania zdjęć ze spotów. 2. System pobiera metadane zdjęć z usługi przechowywania plików. 3. System wyświetla galerię zdjęć z podstawowymi informacjami (np. nazwa spota, data dodania). 4. Użytkownik wybiera zdjęcie, aby zobaczyć je w powiększeniu lub przejść do spota.
Alternatywne przepływy zdarzeń:	2a. Użytkownik nie dodał jeszcze zdjęć – system wyświetla informację o braku zdjęć.

Tabela 4.43: Scenariusz przypadku użycia: Przeglądanie dodanych zdjęć do spotów

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU44
Nazwa:	Przeglądanie dodanych filmów do spotów
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany, Usługa do przechowywania plików w chmurze
Opis:	Użytkownik przegląda filmy powiązane ze spotami, które dodał.
Warunki wstępne:	Użytkownik jest zalogowany i znajduje się w sekcji mediów (np. „Moje filmy”).

Warunki końcowe:	Lista lub galeria filmów powiązanych ze spotami użytkownika została wyświetlona.
Główny przepływ zdarzeń:	1. Użytkownik otwiera sekcję przeglądania filmów ze spotów. 2. System pobiera metadane filmów z usługi przechowywania plików. 3. System wyświetla listę/galerię filmów z podstawowymi informacjami. 4. Użytkownik wybiera film, aby go odtworzyć lub przejść do spota.
Alternatywne przepływy zdarzeń:	2a. Użytkownik nie dodał jeszcze filmów – system wyświetla informację o braku filmów. 3a. Nie udało się pobrać filmów – system wyświetla komunikat o błędzie.

Tabela 4.44: Scenariusz przypadku użycia: Przeglądanie dodanych filmów do spotów

KARTA SCENARIUSZA PRZYPADKU UŻYCIA	
Identyfikator:	PU45
Nazwa:	Przeglądanie dodanych komentarzy do spotów
Priorytet:	Średni
Aktorzy:	Użytkownik zalogowany
Opis:	Użytkownik przegląda komentarze dodane do spotów, które sam utworzył.
Warunki wstępne:	Użytkownik jest zalogowany i otwiera sekcję komentarzy do swoich spotów.

Warunki końcowe:	Lista komentarzy do spotów użytkownika została wyświetlona.
Główny przepływ zdarzeń:	1. Użytkownik przechodzi do sekcji komentarzy do własnych spotów. 2. System pobiera komentarze powiązane ze spotami użytkownika. 3. System wyświetla komentarze (np. w kolejności chronologicznej).
Alternatywne przepływy zdarzeń:	2a. Żaden z spotów użytkownika nie ma komentarzy – system wyświetla odpowiednią informację. 3a. Nie udało się pobrać komentarzy – system wyświetla komunikat o błędzie.

Tabela 4.45: Scenariusz przypadku użycia: Przeglądanie dodanych komentarzy do spotów

4.2 Wymagania ogólne i dziedzinowe

4.3 Wymagania funkcjonalne

4.3.1 Funkcjonalności dla mapy

4.3.2 Funkcjonalności dla chatu

4.3.3 Funkcjonalności dla forum

4.3.4 Funkcjonalności dla konta użytkownika

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Profil użytkownika		
Opis:	Jako użytkownik chcę mieć dostęp do strony profilu, aby sprawdzić informacje o swoim koncie.		
Kryteria akceptacji:	Użytkownik widzi liczby: znajomych, obserwowanych i obserwujących, a także najpopularniejsze zdjęcia.		
Dane wejściowe:	Lista zdjęć oraz liczby: znajomych, obserwujących i obserwowanych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlone informacje o profilu.		
Sytuacje wyjątkowe:	Błąd połączenia z API; brak danych profilu; brak uprawnień (401/403).		
Szczegóły implementacji:	Frontend: React + Tailwind; pobieranie danych profilu przez `@tanstack/react-query` i `axios` z `withCredentials`. Prezentacja w widoku profilu.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.46: Profil użytkownika

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista dodanych spotów		
Opis:	Jako użytkownik chcę sprawdzić listę spotów, które dodałem.		
Kryteria akceptacji:	Użytkownik widzi listę własnych dodanych spotów.		
Dane wejściowe:	Lista dodanych spotów.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista dodanych spotów.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy z backendu (endpoint listy własnych spotów) przez `react-query` + `axios`; prezentacja listy z podstawowymi danymi.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.47: Lista dodanych spotów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Dodanie spota		
Opis:	Jako użytkownik chcę mieć dostęp do formularza dodania spota.		
Kryteria akceptacji:	Użytkownik ma dostęp do formularza dodania spota i może go wysłać.		
Dane wejściowe:	Formularz dodania spota.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlony formularz dodania spota (po wysłaniu: zapis na backendzie).		
Sytuacje wyjątkowe:	Nieprawidłowe dane formularza; błąd połączenia z API.		
Szczegóły implementacji:	Formularz w React; walidacja przeglądarkowa; wysyłka przez `axios` (POST) z `withCredentials`.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.48: Dodanie spota

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista zdjęć		
Opis:	Jako użytkownik chcę mieć dostęp do listy zdjęć, które dodałem na forum, do komentarzy pod spotem oraz do spota.		
Kryteria akceptacji:	Użytkownik widzi listę swoich zdjęć.		
Dane wejściowe:	Lista zdjęć.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista zdjęć.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy zdjęć użytkownika przez `react-query` + `axios`; prezentacja z miniaturami.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.49: Lista zdjęć

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista filmów		
Opis:	Jako użytkownik chcę mieć dostęp do listy filmów, które dodałem na forum, do komentarzy pod spotem oraz do spota.		
Kryteria akceptacji:	Użytkownik widzi listę swoich filmów.		
Dane wejściowe:	Lista filmów.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista filmów.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy filmów użytkownika przez `react-query` + `axios`; prezentacja z miniaturami.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.50: Lista filmów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista znajomych		
Opis:	Jako użytkownik chcę mieć dostęp do listy znajomych.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy znajomych.		
Dane wejściowe:	Lista znajomych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista znajomych.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy znajomych przez `react-query` + `axios`; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; dronia-rze 2.3.		
Wymagania powiązane:			

Tabela 4.51: Lista znajomych

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista obserwujących		
Opis:	Jako użytkownik chcę mieć dostęp do listy obser-wujących.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy obserwujących.		
Dane wejściowe:	Lista obserwujących.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista obserwujących.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy obserwujących przez `react-query` + `axios`; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; dronia-rze 2.3.		
Wymagania powiązane:			

Tabela 4.52: Lista obserwujących

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista obserwowanych		
Opis:	Jako użytkownik chcę mieć dostęp do listy obserwowanych.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy obserwowanych.		
Dane wejściowe:	Lista obserwowanych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista obserwowanych.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy obserwowanych przez `react-query` + `axios`; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.53: Lista obserwowanych

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista spotów		
Opis:	Jako użytkownik chcę mieć dostęp do listy spotów, które polubiłem, odwiedziłem i planuję odwiedzić.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy spotów w wymienionych kategoriach.		
Dane wejściowe:	Listy spotów: polubione, odwiedzone, planowane.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlone listy spotów.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie list przez `react-query` + `axios`; prezentacja w zakładkach/kategoriach.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.54: Lista polubionych/odwiedzonych/planowanych spotów

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Lista komentarzy		
Opis:	Jako użytkownik chcę mieć dostęp do listy komentarzy.		
Kryteria akceptacji:	Użytkownik ma dostęp do listy swoich komentarzy.		
Dane wejściowe:	Lista komentarzy.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlona lista komentarzy.		
Sytuacje wyjątkowe:	Brak wyników; błąd połączenia z API.		
Szczegóły implementacji:	Pobranie listy komentarzy użytkownika przez `react-query` + `axios`; standardowa prezentacja listy.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.55: Lista komentarzy

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Ustawienia		
Opis:	Jako użytkownik chcę mieć możliwość zmiany danych.		
Kryteria akceptacji:	Użytkownik może edytować wybrane dane profilu i zapisać zmiany.		
Dane wejściowe:	Formularz edycji danych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Wyświetlony formularz edycji; po zapisie — zaktualizowane dane.		
Sytuacje wyjątkowe:	Nieprawidłowe dane formularza; błąd połączenia z API.		
Szczegóły implementacji:	Formularz w React; walidacja pól; wysyłka przez `axios` (PUT/PATCH) z `withCredentials`. Po sukcesie — komunikat i odświeżenie danych przez `react-query`.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.56: Ustawienia profilu

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Resetowanie hasła		
Opis:	Jako użytkownik chcę mieć możliwość zresetowania hasła do swojego konta.		
Kryteria akceptacji:	Po kliknięciu w odpowiedni link użytkownik może zresetować hasło do konta.		
Dane wejściowe:	Adres e-mail użytkownika do wysłania linku resetującego.		
Warunki początkowe:	Użytkownik podał poprawny adres e-mail użyty przy rejestracji.		
Warunki końcowe:	Hasło zresetowane po przejściu całej procedury.		
Sytuacje wyjątkowe:	Niepoprawny adres e-mail; wygasły lub nieprawidłowy token resetu; błąd połączenia z API.		
Szczegóły implementacji:	Frontend: formularz „zapomniałem hasła” (POST do endpointu wysyłającego link resetu) oraz formularz ustawienia nowego hasła (POST/PATCH z tokenem). Wysyłka przez `axios`; obsługa komunikatów o powodzeniu/błędach.		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; dronia-rze 2.3.		
Wymagania powiązane:			

Tabela 4.57: Resetowanie hasła

KARTA WYMAGANIA			
Identyfikator:	jednoznaczny symbol np. FO1, FO2 ..	Priorytet:	M
Nazwa:	Dodawanie użytkowników do listy znajomych		
Opis:	Jako użytkownik chcę mieć możliwość dodawania innych użytkowników do listy znajomych.		
Kryteria akceptacji:	Użytkownik może dodać innego użytkownika do swojej listy znajomych.		
Dane wejściowe:	Dane użytkownika, którego chcemy dodać do znajomych.		
Warunki początkowe:	Użytkownik jest zalogowany.		
Warunki końcowe:	Znajomy dodany do listy i widoczny w profilu użytkownika.		
Sytuacje wyjątkowe:	Brak uprawnień; użytkownik już jest znajomym; błąd połączenia z API.		
Szczegóły implementacji:	Akcja wysłania zaproszenia do znajomych przez `axios`; po akceptacji — aktualizacja listy (odświeżenie `react-query`).		
Udziałowiec:	Zespół projektowy 2.1; promotor 2.2; dronia-rze 2.3.		
Wymagania powiązane:			

Tabela 4.58: Dodawanie do znajomych

4.3.5 Funkcjonalności dla logowania i rejestracji

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Logowanie i rejestracja		
Opis:	Jako użytkownik chcę mieć możliwość zalogowania się do aplikacji, korzystając z formularza lub poprzez konto Google lub GitHub.		
Kryteria akceptacji:	Użytkownik może zalogować się do aplikacji zarówno za pomocą standardowego formularza, jak i przy użyciu konta w serwisie Google lub GitHub.		
Dane wejściowe:	Dane użytkownika: adres e-mail, hasło; przy rejestracji dodatkowo nazwa użytkownika.		
Warunki początkowe:	Użytkownik niezalogowany.		
Warunki końcowe:	Działające formularze rejestracji i logowania oraz możliwość logowania za pomocą konta Google i GitHub.		
Sytuacje wyjątkowe:	Błędne dane logowania; przerwana lub nieudana autoryzacja u dostawcy (Google/GitHub).		
Szczegóły implementacji:	Frontend: formularze w React; wysyłka żądań przez `axios` z `withCredentials`. SSO: integracja z Google i GitHub (OAuth 2.0) z przekierowaniem i ustawieniem sesji po stronie backendu (httpOnly cookie). Obsługa statusu `401` zgodnie z mechanizmem wylogowania.		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, dronia-rze 2.3.		
Wymagania powiązane:			

Tabela 4.59: Logowanie i rejestracja

4.3.6 Funkcjonalności dla wyszukiwarki spotów

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Strona główna z podstawowymi filtrami		
Opis:	Jako użytkownik chcę mieć dostęp do strony głównej, która wyświetla karuzelę z najpopularniejszymi spotami oraz listę spotów, które można filtrować.		
Kryteria akceptacji:	Użytkownik widzi karuzelę najpopularniejszych miejsc. Karuzela zawiera zdjęcia, nazwę miejsca i miasto. Użytkownik może filtrować miejsca według lokalizacji (kraj, region, miasto).		
Dane wejściowe:	Lokalizacja użytkownika (kraj, region, miasto); dane z bazy spotów.		
Warunki początkowe:	Użytkownik nie musi być zalogowany.		
Warunki końcowe:	Użytkownik widzi popularne miejsca z wybranego miasta (np. Gdańsk) i może przejść do szczegółów danego miejsca.		
Sytuacje wyjątkowe:	Brak wyników dla wybranych filtrów; błąd połączenia z API.		
Szczegóły implementacji:	Frontend: React + Tailwind. Pobieranie danych przez `@tanstack/react-query` i `axios` (`GET` do backendu z parametrami lokalizacji). Filtry lokacji mapowane na parametry zapytania.		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.60: Strona główna — podstawowe filtry

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Strona główna z zaawansowanymi filtrami		
Opis:	Jako użytkownik chcę mieć dostęp do strony głównej, która wyświetla listę spotów, które można filtrować i sortować.		
Kryteria akceptacji:	Użytkownik widzi listę, którą może filtrować według miasta, tagów i oceny spota, a także sortować po ocenie i popularności.		
Dane wejściowe:	Lokalizacja użytkownika (miasto), wartości filtrów i sortowania; dane z bazy spotów.		
Warunki początkowe:	Użytkownik nie musi być zalogowany.		
Warunki końcowe:	Użytkownik widzi wyniki zgodne z zastosowanymi filtrami i sortowaniem oraz może przejść do szczegółów danego miejsca.		
Sytuacje wyjątkowe:	Brak wyników po zastosowaniu filtrów; błąd połączenia z API.		
Szczegóły implementacji:	Frontend: React + Tailwind. Pobieranie danych przez `@tanstack/react-query` i `axios` z parametrami: lokalizacja, tagi, minimalna ocena oraz kryterium sortowania.		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:	SPXX		

Tabela 4.61: Strona główna — zaawansowane filtry

4.3.7 Funkcjonalności dla motywu

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Ustawienia motywu		
Opis:	Jako użytkownik chcę móc zmienić motyw aplikacji.		
Kryteria akceptacji:	Dostępna jest opcja przełączenia motywu na *jasny* lub *ciemny*; zmiana następuje bez przeładowania strony; ustawienie działa we wszystkich widokach.		
Dane wejściowe:	Preferencje użytkownika dotyczące motywu.		
Warunki początkowe:	Brak.		
Warunki końcowe:	Zmiana motywu widoczna jest natychmiast po kliknięciu przycisku.		
Sytuacje wyjątkowe:	Brak.		
Szczegóły implementacji:	Tailwind CSS z `darkMode: 'class'`; motyw przełączany przez dodanie/usunięcie klasy `dark` na elemencie `<html>`;		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.62: Ustawienia motywu (ręczna zmiana)

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	M
Nazwa:	Zapamiętywanie preferencji motywu		
Opis:	Jako użytkownik chcę, aby moja preferencja motywu była zapamiętana i przywracana przy kolejnym użyciu aplikacji.		
Kryteria akceptacji:	Wybrany motyw jest przywracany po ponownym włączeniu i odświeżeniu strony; preferencja jest zapamiętywana lokalnie w przeglądarce.		
Dane wejściowe:	Preferencje użytkownika zapisane lokalnie.		
Warunki początkowe:	FOXX dostępne.		
Warunki końcowe:	Motyw po uruchomieniu odpowiada ostatniej decyzji użytkownika.		
Sytuacje wyjątkowe:	Brak dostępu do magazynu trwałego — preferencja przechowywana w local storage.		
Szczegóły implementacji:	Zapis w `localStorage` pod kluczem `theme` (`dark` lub `light`); krótki skrypt umieszczony w `App.jsx` przed startem odczytuje `localStorage` i odpowiednio dodaje lub usuwa klasę `dark` na `<html>` (eliminuje mignięcie stylów).		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.63: Zapamiętanie preferencji motywu

KARTA WYMAGANIA			
Identyfikator:	FOXX	Priorytet:	S
Nazwa:	Przełącznik motywu w Sidebar		
Opis:	Jako użytkownik chcę szybko zmieniać motyw bez wchodzenia w ustawienia.		
Kryteria akceptacji:	W Sidebar dostępny jest przełącznik *Jasny-/Ciemny*; posiada odpowiednio ikony *słońca/księżyca*; zmiana następuje natychmiast.		
Dane wejściowe:	Bieżąca preferencja motywu.		
Warunki początkowe:	FOXX, FOXX dostępne.		
Warunki końcowe:	Motyw zmieniony; preferencja zaktualizowana.		
Sytuacje wyjątkowe:	Brak.		
Szczegóły implementacji:	Przycisk typu *toggle* wywołuje funkcję, która przełącza klasę `dark` na `document.documentElement` oraz aktualizuje `localStorage` (`theme = 'dark'|'light'`); brak przeładowania strony.		
Udziałowiec:	Zespół projektowy 2.1, promotor 2.2, droniarze 2.3.		
Wymagania powiązane:			

Tabela 4.64: Szybki przełącznik motywu w interfejsie

4.4 Wymagania pozafunkcjonalne

4.5 Wymagania interfejs z otoczeniem

4.6 Wymagania na środowisko docelowe

Rozdział 5

Projekt

5.1 Wzorce projektowe

5.2 Architektura systemu

5.2.1 Diagram architektury

5.2.2 Komponenty systemu

5.3 Projekt bazy danych

5.3.1 Model danych

5.3.2 Diagram ERD

5.4 Architektura interfejsu użytkownika

5.4.1 Projekt strony głównej

5.4.2 Projekt panelu logowania

5.4.3 Projekt mapy

5.4.4 Projekt chatu

5.4.5 Projekt forum

5.4.6 Projekt konta użytkownika

Rozdział 6

Przebieg realizacji projektu

6.1 Sprint 1

6.2 Sprint 2

Rozdział 7

Realizacja Projektu

7.1 Implementacja backendu

7.1.1 Struktura projektu

7.1.2 Integracja z bazą danych

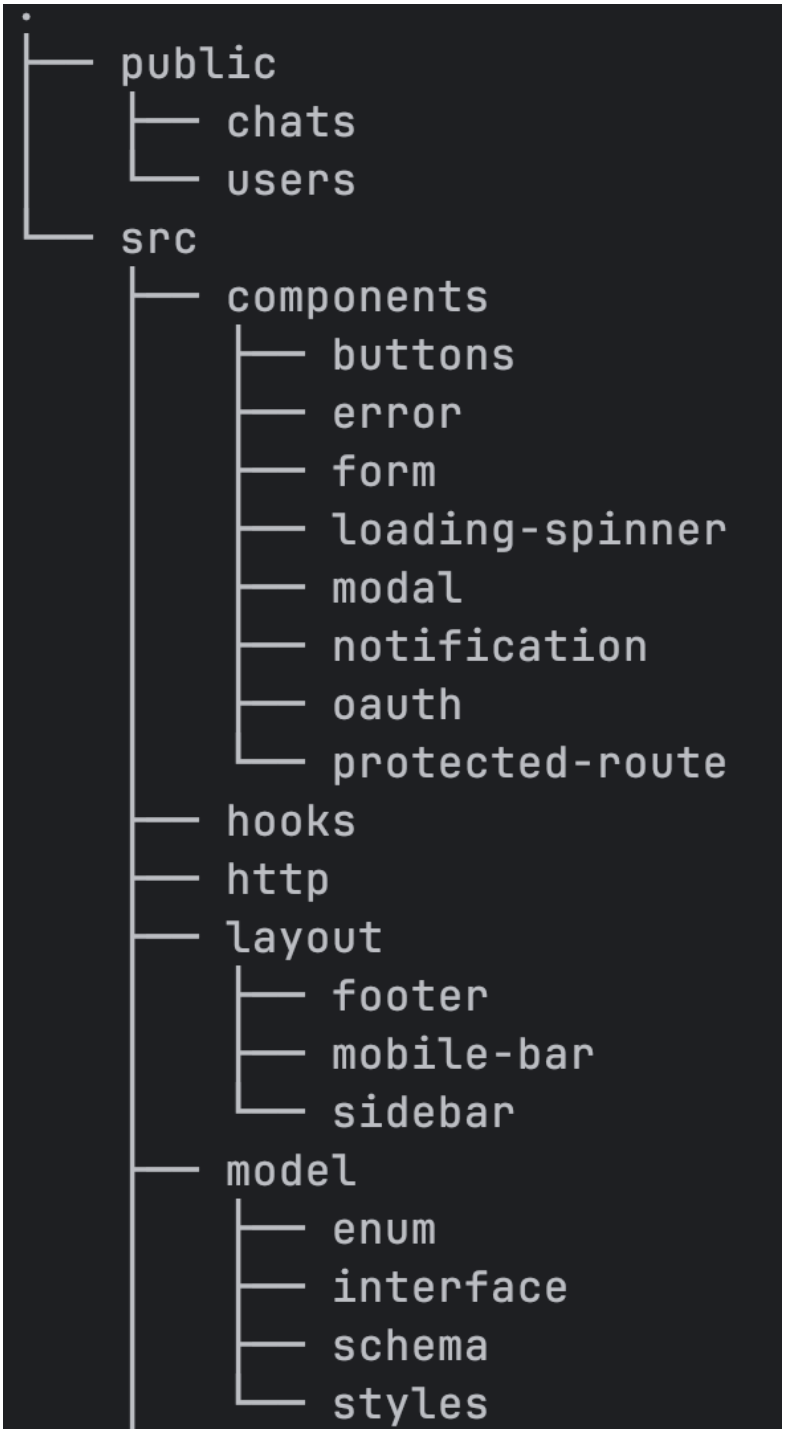
7.1.3 Obsługa uwierzytelnienia

7.1.4 Konteneryzacja

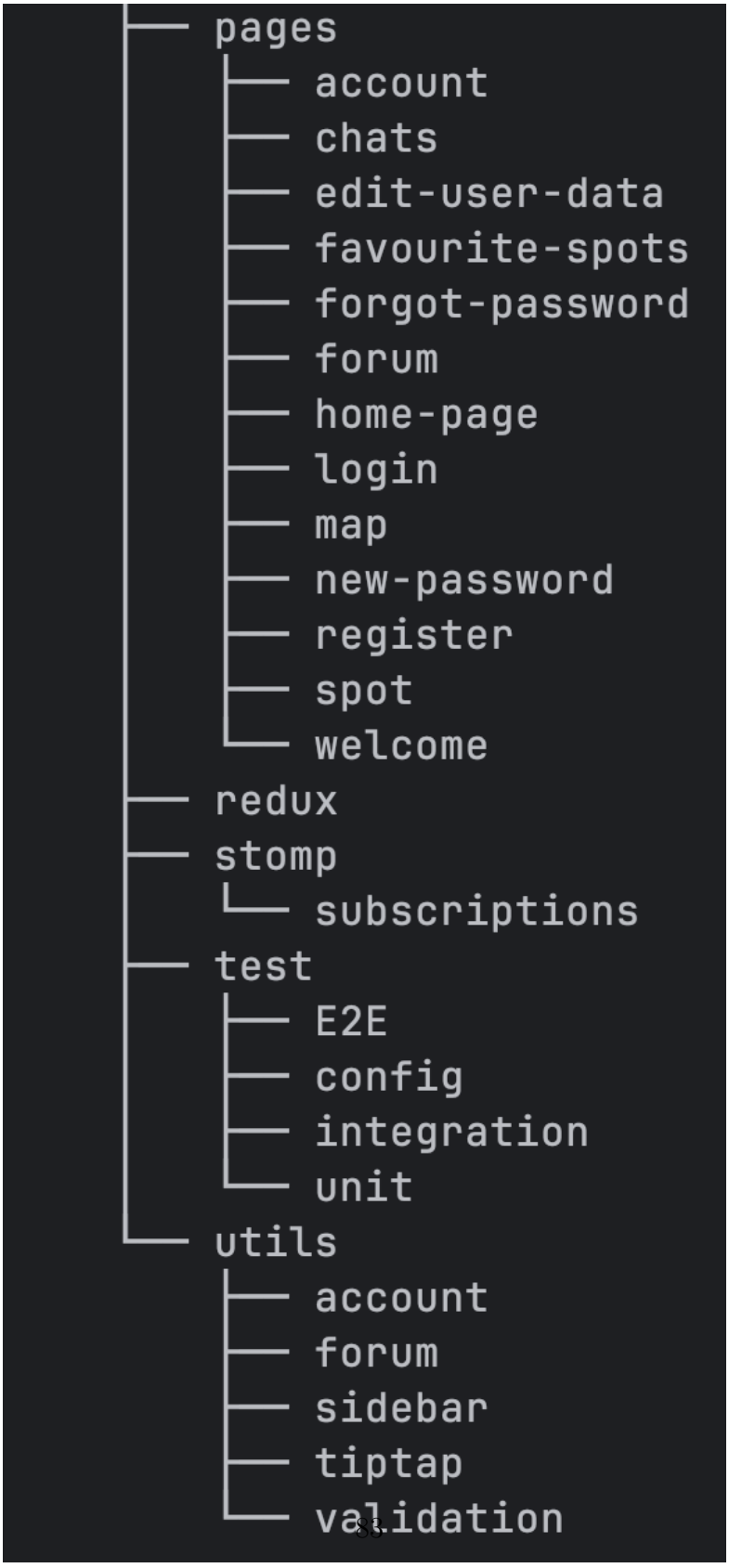
7.2 Implementacja frontendu

7.2.1 Struktura aplikacji

Architektura aplikacji frontendowej została zaprojektowana w strukturze Folder by type, która polega na podziale kodu według typu zasobu (komponenty, strony, modele itd.). Każdy plik znajduje się w katalogu odpowiadającym jego przeznaczeniu, co jest przedstawione na rysunkach 7.1 oraz 7.2.



Rysunek 7.1: Struktura katalogów (1)



Rysunek 7.2: Struktura katalogów (2)

Głównym elementem aplikacji jest mechanizm routingu oparty na Bibliotece React Router. Definiuje on ścieżki do poszczególnych funkcjonalności aplikacji. Dzięki temu możliwa jest płynna nawigacja między różnymi widokami bez konieczności przeładowywania strony.

```
const router : Router = createBrowserRouter([
    {
        path: "/",
        element: <Layout />,
        errorElement: <Error error={undefined} />,
        children: [
            {
                index: true,
                element: <HomePage />,
            },
            {
                path: "advanced",
                element: <AdvanceHomePage />,
            },
            {
                path: "account",
                children: [ 11 elements… ],
            },
            {
                path: "register",
                element: <Register />,
            },
            {
                path: "login",
                element: <Login />,
            },
            {
                path: "forgot-password",
                element: <ForgotPassword />,
            },
            {
```

Rysunek 7.3: Implementacja routera (1)

```
            {
                path: "new-password",
                element: <NewPassword />,
            },
            {
                path: "forum",
                element: <Forum />,
            },
            {
                path: "forum/:postId/:slugTitle?",
                element: <ForumThread />,
            },
            {
                path: "map",
                element: <MapPage />,
            },
            {
                path: "chat",
                element: (
                    <ProtectedRoute>
                        <ChatsPage />
                    </ProtectedRoute>
                ),
            },
        ],
    },
]);

export default router;
```

Rysunek 7.4: Implementacja routera (2)

W projekcie zastosowano również wzorzec Protected route, który służy do zabezpieczania wybranych tras przed dostępem użytkowników niezalogowanych. W pliku `router.tsx`, znajdującym się w głównym katalogu projektu, w konfiguracji przekazywanej do funkcji `createBrowserRouter` (rysunki 7.3 oraz 7.4), wybrane

ścieżki zostały opakowane w komponent `ProtectedRoute`. Komponent ten pełni rolę bramki (rysunek 7.5).

Przykładem takiej chronionej ścieżki jest trasa `/chat`, prowadząca do modułu czatu dostępnego wyłącznie dla zalogowanych użytkowników. Jeśli niezalogowany użytkownik spróbuje uzyskać dostęp do tej ścieżki, zostanie automatycznie przekierowany na stronę główną.

```
export default function ProtectedRoute({ children }) {
    const isLogged = useSelector((state) => state.account.isLogged);

    return isLogged ? children : <Navigate to={"/"} />;
}
```

Rysunek 7.5: Implementacja komponentu bramki (`ProtectedRoute`)

7.2.2 Zarządzanie stanem i przepływ danych

W projekcie postawiliśmy na zrównoważone podejście do zarządzania Stanem. Korzystamy zarówno z lokalnego Stanu komponentów (za pomocą Hook (React)a `useState`) [15], jak i ze Stanu globalnego, utrzymywanego przez Bibliotekę React Redux [16]. Globalny Stan został wprowadzony po to, aby możliwie najbardziej ograniczyć przekazywanie Propsów w głąb drzewa komponentów oraz uniknąć niepotrzebnych ponownych renderów.

Do przechowywania Stanu lokalnego, ograniczonego tylko do danego komponentu (lub jego najbliższych elementów podrzędnych), wykorzystujemy Hook (React) `useState`. Natomiast efekty uboczne i synchronizację realizujemy za pomocą `useEffect`. W przypadku bardziej złożonej logiki lub potrzeby ponownego wykorzystania kodu powstały Hook (React)i niestandardowe, takie jak `useScreenSize`, `useDarkMode` czy `useClickOutside`. Dzięki temu większość logiki prezentacji została wydzielona z warstwy UI, co poprawia czytelność i ułatwia utrzymanie kodu.

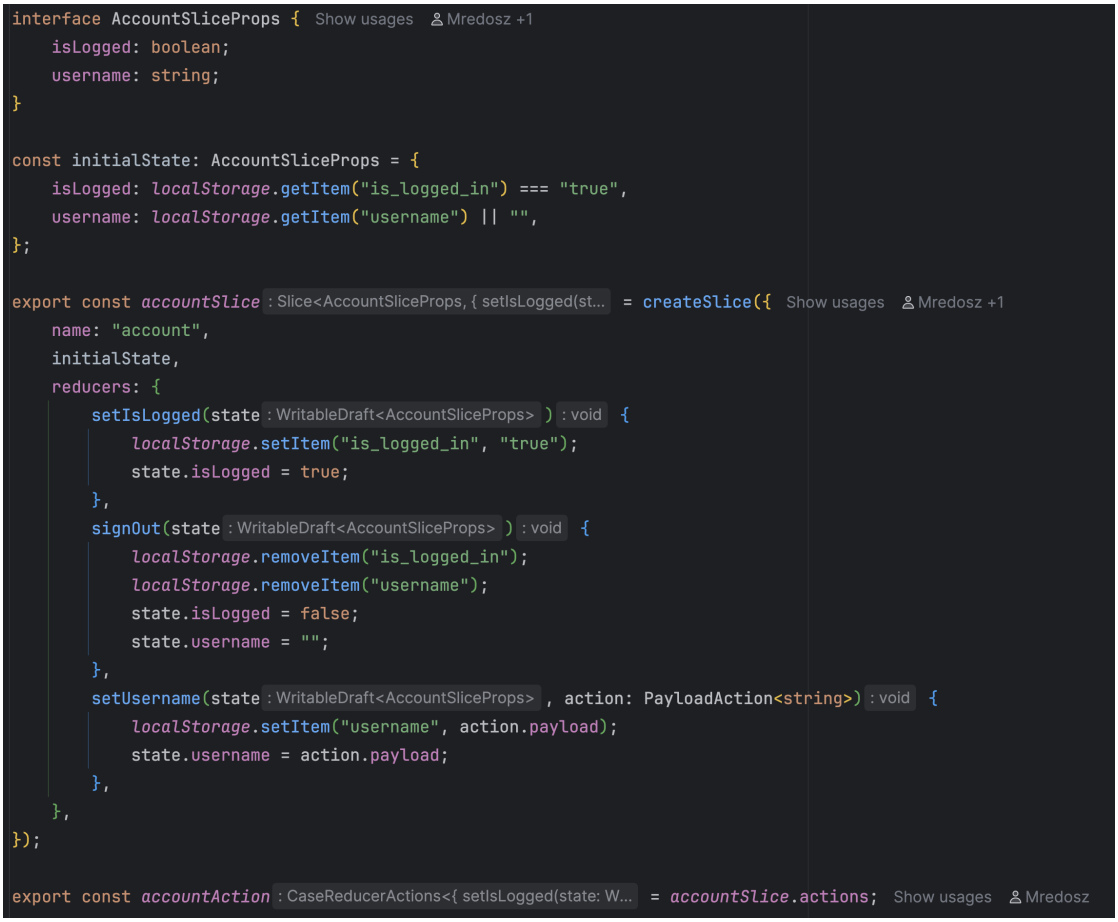
Z racji tego, że korzystamy z Reacta w połączeniu z TypeScriptem, przygotowaliśmy również własne Hook (React)i wspomagające typowanie, takie jak `useDispatchTyped` oraz `useSelectorTyped`. Pozwalają one na bezpieczne typo-

wanie akcji oraz selektorów Reduxa bez konieczności powtarzania adnotacji typów w każdym komponencie. Fragmenty tej implementacji przedstawiono na rysunkach 7.6 oraz 7.7.

```
const store : EnhancedStore<{ account: AccountSliceProp... = configureStore({
    reducer: {
        account: accountSlice.reducer,
        notification: notificationSlice.reducer,
        spotDetails: spotDetailsModalSlice.reducer,
        searchedSpotsListModal: searchedSpotListModalSlice.reducer,
        expandedSpotMediaGallery: expandedSpotMediaGallerySlice.reducer,
        spotFilters: spotFiltersSlice.reducer,
        chats: chatsSlice.reducer,
        map: mapSlice.reducer,
        sidebar: sidebarSlice.reducer,
        searchedSpots: searchedSpotsSlice.reducer,
        social: socialSlice.reducer,
        spotComments: spotCommentSlice.reducer,
        currentViewSpots: currentViewSpotsSlice.reducer,
        currentViewSpotsListModal: currentViewSpotsListModalSlice.reducer,
        currentViewSpotsParams: currentViewSpotParamsSlice.reducer,
        spotWeather: spotWeatherSlice.reducer,
        expandedSpotGalleryMediaList: expandedSpotGalleryMediaListSlice.reducer,
        expandedSpotMediaGalleryModals:
            expandedSpotMediaGalleryModalsSlice.reducer,
        expandedSpotMediaGalleryFullscreenSizeModal:
            expandedSpotMediaGalleryFullscreenSizeSlice.reducer,
        expandedSpotGalleryCurrentMedia:
            expandedSpotGalleryCurrentMediaSlice.reducer,
    },
});

export default store; Show usages  Mredosz
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

Rysunek 7.6: Konfiguracja sklepu (`Redux store`)



Rysunek 7.7: Przykładowy slice odpowiedzialny za sprawdzenie czy użytkownik jest zalogowany

7.2.3 Integracja i komunikacja z backendem

Jest to kluczowy element aplikacji, ponieważ wymaga bezpiecznego przesyłania danych użytkownika. W celu uproszczenia komunikacji z serwerem skorzystaliśmy z biblioteki `axios` [17] oraz Biblioteki `TanStack Query` [18]. We wszystkich ścieżkach, które wymagają aby użytkownik był zalogowany, do zapytania dołączany jest token JWT. Token jest przekazywany w ciasteczku dzięki ustawieniu parametru `withCredentials` na wartość `true`. Przykładem pliku odpowiedzialnego za taką komunikację jest `account.js` (rys. 7.8 i 7.9), który obsługuje operacje związane z

logowaniem, rejestracją, zmianą hasła oraz wylogowaniem.

```
export async function loginUser(user) {  Show usages  new *
    return await axios.post(`${BASE_URL}/public/account/login`, user, {
        withCredentials: true,
    });
}

export async function registerUser(user) {  Show usages  Adam Langmesser +2
    return await axios.post(`${BASE_URL}/public/account/register`, user, {
        withCredentials: true,
    });
}

export async function sentEmailWithNewPasswordLink(email) {  Show usages  Adam Langmesser +1
    console.log("sending email...");
    return await axios.post(
        `${BASE_URL}/public/account/forgot-password`,
        email,
        {
            headers: {
                "Content-Type": "text/plain",
            },
        },
    );
}
```

Rysunek 7.8: Implementacja modułu `account` (1)

Rysunek 7.9: Implementacja modułu `account` (2)

Funkcje odpowiedzialne za komunikację z backendem zostały umieszczone w katalogu `/http`. Dzięki temu są one scentralizowane i mogą być w prosty sposób wykorzystywane w różnych częściach aplikacji. Zastosowaliśmy TanStack Query, ponieważ znacząco ogranicza on powtarzalny kod oraz upraszcza obsługę błędów i stanów zapytania (takich jak ładowanie danych, błąd, sukces). udostępnia m.in. wartość `isLoading`, dzięki czemu komponent może łatwo wyświetlić ekran ładowania bez ręcznego zarządzania własnym stanem. Dodatkowo Hook (React) `useQuery` z tej Biblioteki umożliwia automatyczne pobieranie danych po wejściu na daną podstronę. Oznacza to, że komponent deklaruje jedynie „jakie dane są mu potrzebne”, a TanStack Query zajmuje się ich pobraniem, cache'owaniem oraz odświeżaniem. Do operacji, które wymagają wywołania akcji po stronie użytkownika (np. wysłania formularza logowania), wykorzystujemy Hook (React) `useMutation` z TanStack Query. Przykład użycia tego rozwiązania w procesie logowania został przedstawiony na rys. 7.10.

```
const { mutateAsync, isSuccess, error } = useMutation({
    mutationFn: loginUser,
});

const handleSubmit : (event: FormEvent<HTMLFormElement>) => Pr... = async (event: FormEvent<HTMLFormElement>) : Promise<void> => {
    event.preventDefault();
    await mutateAsync({
        username: enteredValue.username,
        password: enteredValue.password,
    });
    navigate(-1);
};
```

Rysunek 7.10: Wykorzystanie TanStack Query przy logowaniu użytkownika

7.2.4 Style

Do stylowania interfejsu wykorzystaliśmy Framework Tailwind CSS [19]. Dzięki gotowym klasom udostępnianym przez Tailwind mogliśmy definiować wygląd elementów bezpośrednio w kodzie komponentu, bez konieczności przechodzenia do osobnych plików ze stylami. Ułatwia to zarówno tworzenie widoków, jak i późniejsze modyfikacje — w przypadku zmiany stylu dokładnie wiadomo, gdzie należy jej dokonać. Korzystanie ze zdefiniowanych klas pozwoliło nam również zachować spójność wizualną w całej aplikacji. W pliku `index.css` zdefiniowaliśmy zmienne kolorystyczne (rys. 7.11 i 7.12). Dzięki temu zmiana motywu kolorystycznego w przyszłości sprowadza się do edycji wartości w jednym miejscu.

Rysunek 7.11: Implementacja zmiennych kolorystycznych (1)

Rysunek 7.12: Implementacja zmiennych kolorystycznych (2)

W niektórych miejscach konieczne było zapisanie stylów w czystym CSS, ponieważ część użytych Bibliotek tego wymagała. W innych przypadkach wystarczyło skorzystać z klas zdefiniowanych w `index.css` oraz klas Tailwinda. Cała aplikacja

jest Responsywna. Tailwind udostępnia predefiniowane prefiksy Responsywne (np. `md:`, `lg:`) (rys. 7.13), stworzyliśmy również własny (`3xl:`) na ekrany o rozdzielczości 2560px. Pozwalają one przypisywać style zależnie od szerokości ekranu bez pisania własnych reguł @media. Dzięki temu implementacja widoków mobilnych i desktopowych była znacząco szybsza.

```
<div className="mt-17 flex flex-col items-center gap-7 lg:mt-0 lg:-ml-40 lg:flex-row xl:-ml-42 xl:gap-10 2xl:-ml-80">
    <div className="relative">
        <img
            alt="profileImage"
            src={userData?.profilePhoto}
            className="dark:drop-shadow-darkBgMuted aspect-square h-64 rounded-full
            shadow-md sm:h-80 lg:h-85 xl:h-96 dark:drop-shadow-md"
        />
```

Rysunek 7.13: Przykładowe użycie klas Tailwind (w tym prefiksów responsywności)

Tailwind został też wykorzystany do obsługi trybu jasnego i ciemnego. Wystarczy dodać klasę z prefiksem `dark:` (np. `dark:bg-black`), aby zmienić kolorystykę elementu, gdy aplikacja jest w trybie ciemnym (rys. 7.14).

```
<input
    id={id}
    value={value}
    type={type}
    onChange={onChange}
    onFocus={setFocusedToTrue}
    onBlur={handleOnBlur}
    className="dark:bg-darkBgMuted bg-lightBgMuted dark:text-darkText text-lightText w-full
    rounded-md p-2 shadow-md focus:outline-none dark:shadow-black/50"
/>
```

Rysunek 7.14: Przykładowe użycie klas Tailwind (w tym wariantu `dark:`)

Aby uzyskać płynniejsze i przyjemniejsze animacje, wykorzystaliśmy Bibliotekę Motion [20]. Pozwala ona w prosty sposób tworzyć animacje elementów interfejsu, bez potrzeby ręcznego pisania złożonych reguł CSS. W naszej aplikacji użyliśmy jej m.in. w polach formularza logowania i rejestracji (rys. 7.15). Na początku etykieta pola (np. „username") jest wyświetlana wewnątrz pola tekstowego, natomiast po

kliknięciu w pole jest płynnie przesuwana nad to pole, co poprawia czytelność i ergonomię formularza.

```
<div className="relative">
    <motion.label
        htmlFor={id}
        initial={false}
        animate={{
            top: shouldFloat ? "-0.7rem" : "0.5rem",
            left: "0.75rem",
            fontSize: shouldFloat ? "0.75rem" : "1rem",
            opacity: shouldFloat ? 1 : 0.6,
        }}
        transition={{ type: "spring", stiffness: 300, damping: 25 }}
        className="dark:text-darkText text-lightText pointer-events-none absolute z-10 px-1 capitalize"
    >
        {label}
    </motion.label>
    <input
        id={id}
        value={value}
        type={type}
        onChange={onChange}
        onFocus={setFocusedToTrue}
        onBlur={handleOnBlur}
        className="dark:bg-darkBgMuted bg-lightBgMuted dark:text-darkText text-lightText w-full
        rounded-md p-2 shadow-md focus:outline-none dark:shadow-black/50"
    />
```

Rysunek 7.15: Implementacja animacji z wykorzystaniem Motion

7.2.5 Wyszukiwarka spotów

Niniejszy rozdział opisuje sposób implementacji wyszukiwarki spotów.

Jednym z głównych modułów aplikacji jest wyszukiwarka spotów, która umożliwia użytkownikowi szybkie odnalezienie interesujących lokalizacji. Funkcjonuje ona w dwóch wariantach: prostym i zaawansowanym (rys. 7.16 oraz 7.17).

```
<div className="dark:bg-darkBg dark:text-darkText bg-lightBg text-lightText
flex min-h-screen w-full flex-col items-center space-y-4 overflow-hidden p-8 pt-18">
    <Switch />
    <SearchBar
        onSetSpots={handleSetSearchedSpots}
        loadMoreRef={loadMoreRef}
        onSetFetchingNextPage={setIsFetchingNextPage}
    />
    <div className="flex w-full flex-col items-center space-y-4">
        <h1 className="text-center text-3xl">The Most Popular Spots</h1>
        <div className="flex w-full flex-col items-center space-y-5">
            <Carousel spots={data!} spotsPerPage={spotsPerPage} />
            <SearchSpotList
                spots={searchedSpots}
                isFetchingNextPage={isFetchingNextPage}
                loadMoreRef={loadMoreRef}
            />
        </div>
    </div>
</div>
```

Rysunek 7.16: Implementacja prostej wersji wyszukiwarki

```
<div className="dark:bg-darkBg dark:text-darkText bg-lightBg text-lightText
flex min-h-screen w-full flex-col items-center space-y-4 overflow-hidden p-8 pt-18">
    <Switch />
    <AdvanceSearchBar
        onSetSpots={handleSetSearchedSpots}
        loadMoreRef={loadMoreRef}
        onSetFetchingNextPage={setIsFetchingNextPage}
    />
    <div className="flex w-full flex-col items-center space-y-10">
        <SearchSpotList
            spots={searchedSpots}
            loadMoreRef={loadMoreRef}
            isFetchingNextPage={isFetchingNextPage}
        />
    </div>
</div>
```

Rysunek 7.17: Implementacja zaawansowanej wersji wyszukiwarki

Przełączanie pomiędzy tymi widokami odbywa się za pomocą przycisku umieszczonego w górnej części strony (rys. 7.18).

```
<div className="dark:shadow-darkBgSoft flex rounded-full shadow-lg shadow-black/20">
    <NavLink
        to="/"
        className={({ isActive } : NavLinkRenderProps ) : string  =>
            `hover:dark:bg-violetDark hover:bg-violetLight rounded-l-full px-2.5 py-1.5
            transition-all duration-300 ${isActive ? "dark:bg-violetDark bg-violetLight" : ""}`
        }
    >
        Simple filters
    </NavLink>
    <NavLink
        to="/advanced"
        className={({ isActive } : NavLinkRenderProps ) : string  =>
            `hover:dark:bg-violetDark hover:bg-violetLight rounded-r-full px-2.5 py-1.5
            transition-all duration-300 ${isActive ? "dark:bg-violetDark bg-violetLight" : ""}`
        }
    >
        Advanced filters
    </NavLink>
</div>
```

Rysunek 7.18: Implementacja komponentu do przełączania trybów

W trybie prostym prezentowana jest karuzela (rys. 7.19) z dwunastoma najpopularniejszymi spotami w całej aplikacji. Użytkownik może w tym miejscu wyszukiwać spoty po lokalizacji (kraj, region, miasto).

```
<div className="relative flex w-full items-center justify-center">
    <button
        onClick={() : void  => paginate(-1)}
        className="hover:text-darkBorder z-10 cursor-pointer transition-all duration-300"
    >
        <RiArrowLeftWideFill className="text-5xl sm:text-6xl" />
    </button>

    <div className="relative h-[440px] w-full max-w-[1200px] overflow-hidden">
        <AnimatePresence custom={direction} initial={false} mode="sync">
            <motion.div
                key={page}
                custom={direction}
                variants={sliderVariants}
                initial="incoming"
                animate="active"
                exit="exit"
                transition={[ 3 elements… ]}
                className="grid w-full grid-cols-1 grid-rows-1 justify-items-center gap-4
                lg:grid-cols-2 lg:grid-rows-2 2xl:grid-cols-3 2xl:grid-rows-2"
            >
                {currentSpots.map((spot : TopRatedSpot ) : Element  => (
                    <MostPopularSpot
                        spot={spot}
                        key={`${spot.id}-${page}`}
                    />
                ))}
            </motion.div>
        </AnimatePresence>
    </div>

    <button
        onClick={() : void  => paginate(1)}
        className="hover:text-darkBorder z-10 cursor-pointer transition-all duration-300"
    >
        <RiArrowRightWideFill className="text-5xl sm:text-6xl" />
    </button>
</div>
```

Rysunek 7.19: Implementacja karuzeli z najpopularniejszymi spotami

Widok zaawansowany udostępnia rozszerzoną wyszukiwarkę, która umożliwia filtrowanie wyników po mieście, tagach oraz ocenie, a także ich sortowanie według popularności i średniej oceny (rys. 7.17).

Wyszukiwarka spotów została zbudowana z dwóch głównych komponentów: `HomePage` oraz `AdvanceHomePage`. W skład prostej wersji wchodzą następujące komponenty:

- `Switch` – służy do przełączania widoku między trybem podstawowym a zaawansowanym,
- `SearchBar` – podstawowa wyszukiwarka spotów,
- `Carousel` – wyświetla najpopularniejsze spoty,
- `SearchSpotList` – wyświetla wyszukane spoty.

W skład zaawansowanej wersji wchodzą następujące komponenty:

- `Switch` – służy do przełączania widoku między trybem podstawowym a zaawansowanym,
- `AdvanceSearchBar` – zaawansowana wyszukiwarka spotów,
- `SearchSpotList` – wyświetla wyszukane spoty.

Komponent `Switch` (rys. 7.18) zawiera dwa elementy `NavLink` z biblioteki React Router, co pozwala na przełączanie widoków bez konieczności przeładowywania całej strony.

W komponencie `SearchBar` (rys. 7.20) po wpisaniu co najmniej dwóch znaków wyświetlana jest lista podpowiedzi dla kraju, regionu oraz miasta, w zależności od aktualnie uzupełnianego pola. Po pojawieniu się listy użytkownik może wybrać interesującą go lokalizację, co ułatwia określenie, w jakich miejscach znajdują się dostępne spoty.

```
<div className="dark:bg-darkBgSoft bg-lightBgSoft flex w-full flex-col items-center justify-between
 space-y-3 rounded-md px-3 py-2 shadow-md md:flex-row md:space-y-0 lg:w-3/4 lg:space-x-3 xl:w-1/2
 dark:shadow-black">
    <div className="flex w-full flex-col space-y-2">
        <h1>Location</h1>
        <div className="flex w-full flex-col space-y-3 md:flex-row md:space-y-0 md:space-x-2">
            {inputList.map(({ id, label } : { readonly label: "Your Country"; readonl... ) : Element  => (
                <div key={id} className="relative w-full">
                    <SearchInput
                        label={label}
                        id={id}
                        value={searchLocation[id] ?? ""}
                        onChange={(e : ChangeEvent<HTMLInputElement> ) : void  =>
                            handleSetLocation(id, e.target.value)
                        }
                        onFocus={() : void  => setActiveInput(id)}
                    />
                    {activeInput === id && suggestions.length > 0 && (
                        <SearchSuggestions
                            suggestions={suggestions}
                            onClick={handleSuggestionClick}
                            id={id}
                            onClose={() : void  => setActiveInput(null)}
                        />
                    )}
                </div>
            ))}
        </div>
    </div>
    <button
        className="dark:bg-darkBgMuted dark:hover:bg-darkBgMuted/80 bg-lightBgMuted
        hover:bg-lightBgMuted/80 flex w-full cursor-pointer justify-center rounded-md p-2 md:w-fit"
        onClick={handleSearchSpots}
    >
        <FaSearch />
    </button>
</div>
```

Rysunek 7.20: Implementacja prostej wyszukiwarki

Komponent `SearchSpotList` (rys. 7.21) odpowiada za prezentację wyników wyszukiwania. Został w nim zaimplementowany mechanizm przewijania nieskończonego (infinite scroll), który automatycznie pobiera kolejne strony wyników w momencie, gdy użytkownik zbliża się do końca listy. Wykorzystuje on listę komponentów `SpotTile`, a także komponent `LoadingSpinner` oraz komunikat informujący o braku wyników, jeżeli nie zostanie odnaleziony żaden spot.

```
<>
    <ul className="grid w-full grid-cols-1 place-items-center gap-8 xl:grid-cols-2 2xl:grid-cols-3">
        {spots.map((spot : HomePageSpotDto ) : Element  => (
            <SpotTile key={spot.id} spot={spot} />
        ))}
    </ul>
    <div ref={loadMoreRef} className="h-10" />
    {isFetchingNextPage && <LoadingSpinner />}
    {spots.length === 0 && (
        <p className="text-center text-2xl">
            Search for spots to see results.
        </p>
    )}
</>
```

Rysunek 7.21: Implementacja listy do wyświetlania spotów

Komponent `SpotTile` zawiera następujące informacje:

- zdjęcie spota,
- miasto, w którym się znajduje,
- nazwę spota,
- ocenę oraz liczbę ocen,
- tagi,
- podstawowe informacje pogodowe (temperatura i typ pogody),
- dwa przyciski: jeden prowadzący do widoku szczegółów spota oraz drugi informujący, jak daleko znajduje się dany spot; po kliknięciu przycisku lokalizacja spota jest prezentowana na mapie.

Komponent `AdvanceSearchBar` jest zbliżony wyglądem i strukturą do wersji podstawowej, jednak w polu lokalizacji można podać wyłącznie miasto. Dodatkowo dostępna jest możliwość dodawania tagów z przygotowanej listy. Wyszukiwarka umożliwia także filtrowanie po ocenie oraz sortowanie wyników według oceny i popularności z wykorzystaniem komponentów typu `Dropdown`.

Oba widoki (`HomePage` i `AdvanceHomePage`) współdzielą część komponentów, między innymi `Switch` oraz `SearchSpotList`. Dzięki temu kod odpowiedzialny za wyświetlanie listy wyników jest zdefiniowany w jednym miejscu, a zmiany w sposobie prezentacji spotów wymagają modyfikacji tylko w komponentach współdzielonych.

7.2.6 Mapa

7.2.7 Chat

7.2.8 Forum

7.2.9 Konto użytkownika

7.2.10 Panel logowania

7.3 Implementacja CI/CD

Rozdział 8

Testy

8.1 Testy jednostkowe

8.2 Testy integracyjne

8.3 Testy E2E

8.4 Wyniki testów i wnioski

Rozdział 9

Prezentacja systemu

9.1 Strona główna

9.2 Strona mapy

9.3 Strona chatu

9.4 Strona forum

9.5 Panel logowania

9.6 Panel konta użytkownika

Rozdział 10

Nakład pracy

10.1 Ogólny nakład pracy

10.2 Indywidualne nakłady pracy

10.2.1 Adam Langmesser

10.2.2 Mateusz Redosz

Na projekt poświęciłem łącznie 324 godziny, z czego 237 przeznaczyłem na prace deweloperskie, 111 na pisanie dokumentacji, 19 godzin na Review kodu, 19 na spotkania dotyczące omówienia dalszych prac projektowych oraz przy pomocy innym członkom zespołu oraz 49 godzin poświęciłem nad stworzeniem widoków na figmie. Prace nad częścią deweloperską rozpocząłem 04.08.2024 a zakończyłem 08.09.2025. W projekcie pracowałem nad Rejestracją użytkownika, tokenenm JWT, częściową implementacją CI/CD, stroną główną, zaimplementowaniem Sidebara oraz podstroną dla użytkownika. Moje wylistowane zadania z Jira:

1. Dokumentacja
    - TODO
2. Design
    - Ustalić paletę kolorystyczną

- Propozycja wyglądu

3. Backend i Frontend
    - Formularz rejestracji
    - Routing
    - Formatowanie w React (prettier)
    - Obsługa JWT na frontend
    - oAuth Frontend
    - Update JWT
    - Refactor JWT
    - Stworzenie komponentu Notification i poprawa błędów
    - Implementacja pierwszych testów
    - Zaimplementowanie kolejki w komponencie notification
    - Dodanie reduxa do rejestracji
    - Zmiana sposobu pobierania danych o spotach
    - Obsługa customowych błędów z jakarta.validation
    - Obsługa auto wylogowania przy starcie
    - Domyślna wiadomość w notification
    - Poprawa headera
    - Ciemny motyw
    - Refactor pogody
    - Propozycja wyglądu
    - Przeniesienie zdjęć z google drive
    - Dodać Type script do Reacta
    - Aktualizacja tailwinda i dodanie kolorów
    - Podstawowy Sidebar

- Strona główna z prostymi filtrami
- Strona główna z zaawansowanymi filtrami
- Sidebar
- Strona profilu
- Ustawienia
- Listy spotów
- Lista zdjęć
- Lista filmów
- Lista znajomych
- Dodanie spotów
- Lista komentarzy
- Strona główna profilu
- Listy
- Poprawa Sidebara
- Zmiana kropki na przyciemnienie tła na Sidebar
- Poprawa strony do logowania i rejestracji
- Usunięcie username z account Redux
- Dodanie zamknięcia Sidebara na małych ekranach po kliknięciu nav linka
- Poprawić tooltipa na sidebar
- Zmiana sposobu pobierania username na backendzie z tokena jwt
- Paginacja z infinity scrollem
- Lista zdjęć innego usera
- Walidacja i responsywność w dodaniu spotów
- Dodanie sortowania i filtrów na zaawansowanej stronie
- Zmiana na infinity scrola

- Zmiana zdjęcia profilowego użytkownika
- Czyszczenie formularza w dodawaniu spota
- Dodanie wyszukiwarki znajomych w Social
- Zatwierdzenie przez drugiego użytkownika dodania do znajomych
- Sprawdzenie czy wszystko działa i poprawki Mateusz

4. CI/CD
    - Dodanie testów z frontendu do github actions
    - Poprawa github actions
    - Poprawa pipeline od Javy i Reacta
5. Praca dyplomowa
    - Uzupełnienie informacji o zespole i podział na rozdziały

10.2.3 Stanisław Oziemczuk

10.2.4 Kacper Badek

Rozdział 11

Podsumowanie

11.1 Osiągnięte rezultaty

11.2 Napotkane wyzwania

11.3 Plany na przyszłość

Rozdział 12

Słownik pojęć i skrótów

API

(ang. *application programming interface*); zbiór reguł i operacji do komunikacji z oprogramowaniem.. 16

Backend

Część aplikacji odpowiedzialna za logikę biznesową, przetwarzanie danych i komunikację z bazą danych. Działa po stronie serwera i obsługuje żądania wysyłane przez frontend. 2, 15, 106

Backlog

Lista zadań, które należy wykonać w ramach projektu, używane w metodykach zwinnych.. 15

Biblioteka

Zewnętrzny lub wewnętrzny zestaw gotowych funkcji, klas, komponentów lub modułów, który można wielokrotnie wykorzystywać w projekcie zamiast pisać wszystko od zera. 84, 86, 88, 90, 93, 94, 99

BPMN

(ang. *Business Process Model and Notation*); standardowa notacja graficzna, która umożliwia szczegółowe przedstawienie i dokumentowanie procesów biznesowych.. 17

CI/CD

Skrót od *Continuous Integration/Continuous Deployment*. Praktyka programistyczna polegająca na automatyzacji procesu budowania, testowania i wdrażania oprogramowania. 16, 105, 108

CSS

Kaskadowe arkusze stylów (Cascading Style Sheets) — język opisu prezentacji dokumentów (np. HTML). Definiuje wygląd interfejsu: układ, kolory, typografię, odstępy, animacje i zachowania responsywne, oddzielając warstwę treści od warstwy prezentacji.. 93, 94

Design

Etap lub proces projektowania wyglądu i funkcjonalności aplikacji, obejmujący zarówno aspekty wizualne, jak i użytkowe (UX/UI). 105

Disciplined Agile Delivery - Lean Life Cycle

Disciplined Agile Delivery w wariancie Lean Life Cycle to sposób prowadzenia projektu, który łączy elastyczność Agile z przewidywalnością Waterfalla, ale bez stałych sprintów — praca toczy się w ciągłym przepływie. Na starcie zakłada mocniejszą fazę przygotowawczą: doprecyzowanie zakresu, szkic architektury, identyfikację ryzyk i kryteria jakości. W realizacji następuje ciągłe doprecyzowywanie wymagań i backlogu, oparte na regularnym feedbacku udziałowców. Całość opiera się na praktykach Lean oraz lekkim governance: code review i regularnych przeglądach postępów. . 10

Droniarz

Potoczne określenie osoby, która jest jednocześnie pilotem oraz operatorem drona. Zwykle entuzjasta dronów.. 8

Droniarz foto/video

Pilot wykorzystujący drony fotograficzne/filmowe do rejestracji materiałów wizualnych (zdjęcia, wideo), zwykle z naciskiem na stabilizację i jakość obrazu.. 17

Folder by type

Sposób organizowania struktury katalogów w projekcie, w którym pliki są grupowane według rodzaju (typu) zasobu, a nie według funkcjonalności. Na przykład wszystkie komponenty trafiają do jednego folderu, wszystkie style do innego itd. 81

Framework

Zestaw narzędzi, bibliotek i struktur wspomagających tworzenie aplikacji. Ułatwia programowanie poprzez dostarczenie gotowych komponentów oraz określenie zasad organizacji kodu. 2, 91

Frontend

Warstwa aplikacji odpowiedzialna za interfejs użytkownika oraz interakcję z użytkownikiem. Zazwyczaj tworzona przy użyciu technologii takich jak HTML, CSS i JavaScript. 2, 15, 106

Hook (React)

Prosta funkcja w React, która „dodaje" możliwości do elementu interfejsu — np. pozwala mu coś zapamiętać (stan) albo zrobić coś po zmianie/załadowaniu. Wszystkie hooki zaczynają się od `use...` (np. `useState`, `useEffect`).. 86, 90

IDE

(ang. *integrated development environment*); to zintegrowane środowisko programistyczne, służące do tworzenia, modyfikowania, testowania i konserwacji oprogramowania. 14

Infinite scroll

Wzorzec interfejsu użytkownika, w którym kolejne porcje treści są automatycznie doładowywane podczas przewijania strony w dół, zamiast być podzielone na odrębne, ręcznie przełączane strony. 100

JWT

Skrót od *JSON Web Token*. Standard służący do bezpiecznego przekazywania informacji między stronami w formacie JSON, często używany w procesach autoryzacji użytkowników. 88, 105

Media queries

Konstrukcja CSS pozwalająca stosować reguły stylów w zależności od cech urządzenia/okna (np. szerokości ekranu, orientacji, preferencji użytkownika). Podstawa responsywnego projektowania (*responsive design*).. 114

MoSCoW

Metoda nadawania priorytetów wymaganiom, wyróżniająca kategorie: *Must have*, *Should have*, *Could have* oraz *Won't have this time*. 21

PANSA

Polish Air Navigation Services Agency, pol. Polska Agencja Żeglugi Powietrznej. Instytucja ta zapewnia m.in. mapę z zaznaczonymi strefami lotów. Każda strefa ma swoje właściwości prawne. . 22, 32

Props

Właściwości przekazywane do komponentu React przez komponent nadrzędny; służą do konfiguracji i przekazywania danych. Powinny być traktowane jako tylko do odczytu (read-only) wewnątrz komponentu potomnego.. 86

Protected route

Trasa w aplikacji, do której dostęp jest ograniczony, zwykle tylko dla zalogowanych użytkowników lub użytkowników z odpowiednimi uprawnieniami. Jeżeli użytkownik nie spełnia warunków, jest przekierowywany (np. na stronę główną). 85

React

Biblioteka JavaScript do budowy interfejsów użytkownika w oparciu o komponenty deklaratywne i wirtualny DOM. Zapewnia jednokierunkowy przepływ danych oraz zarządzanie stanem komponentów.. 86

Redux

Biblioteka do przewidywalnego zarządzania stanem aplikacji. Opiera się na jednym *store*, akcjach i czystych *reducerach*, promuje niemutowalność i jednokierunkowy przepływ danych. Często używana z Reactem, ale niezależna od niego.. 86, 87

Responsywność

Określenie związane z projektowaniem responsywnym (Responsive Web Design, RWD), czyli dostosowywaniem interfejsu do różnych rozmiarów i parametrów ekranów. Obejmuje m.in. elastyczne siatki, grafiki i Media queries, tak aby układ i czytelność były zachowane na telefonach, tabletach i desktopach.. 94

REST API

Architektura budowania usług sieciowych komunikujących się poprzez metody protokołu HTTP (GET, PUT, POST, DELETE, PATCH). Wymiana danych występuje często w formacie JSON lub XML.
REST API musi spełniać następujące reguły:

1. **Rozdzielenie klient-serwer** — klient i serwer są od siebie niezależne, komunikują się poprzez interfejs.
2. **Bezstanowość** — każde żądanie przez klienta zawiera wszystkie informacje niezbędne do jego obsłużenia. Po otrzymaniu żądania serwer nie przechowuje o nim żadnych informacji.
3. **Buforowalność (cache)** — odpowiedzi z API powinny informować, czy dane można cache'ować. Jeśli tak, to przy kolejnym żądaniu mogą być zwrócone z cache'a.
4. **Jednolity interfejs**:
   - **Identyfikacja zasobów** — każdy zasób musi być jednoznacznie zidentyfikowany w interakcji klient-serwer.
   - **Manipulacja zasobów poprzez reprezentację** — po otrzymaniu reprezentacji klient może zmienić stan zasobu przesyłając zmodyfikowaną reprezentację.
   - **Samoopisujące się wiadomości** — każde żądanie i odpowiedź powinny zawierać informacje do jego poprawnego przetworzenia.

- **Hypermedia jako silnik stanu aplikacji (HATEOAS)** — po otrzymaniu odpowiedzi klient powinien móc dynamicznie poznać inne interakcje przez linki.

5. **Warstwowość** — klient nie wie czy komunikuje się bezpośrednio z serwerem, czy poprzez pośrednika (np. proxy) oraz nie wie z czym komunikuje się obsługująca go warstwa.
6. **Kod na żądanie (opcjonalnie)** — serwer może przesłać fragment kodu, który zostanie wykonany przez klienta.

15

Review kodu

Proces polegający na wzajemnym przeglądzie kodu źródłowego przez programistów w celu wykrycia błędów, poprawy jakości oraz zwiększenia spójności projektu. 15, 16, 105

Sidebar

Boczny panel w interfejsie użytkownika, zawierający menu nawigacyjne lub dodatkowe opcje funkcjonalne aplikacji. 78, 105–107

Spot

Spotkanie zespołu projektowego, zazwyczaj krótkie i regularne, służące omówieniu postępów prac, problemów oraz planów na najbliższy okres. 97–102

Stan

Aktualny zestaw danych przechowywanych przez aplikację lub komponent, na podstawie którego renderowany jest interfejs użytkownika. Stan może być lokalny (utrzymywany w pojedynczym komponencie) albo globalny (wspólny dla wielu komponentów).. 86

Tablica Kanban

Narzędzie do zarządzania przepływem pracy, które pomaga zespołom śledzić zadania oraz ich postępy. Składa się z kolumn reprezentujących stan etapu prac, na przykład „Do zrobienia” lub „W trakcie”.. 15

TypeScript

Rozszerzenie do języka JavaScript dodający statyczne typowanie, interfejsy i narzędzia do większych projektów. Kompiluje się do czystego JavaScript, ułatwiając wykrywanie błędów w czasie kompilacji i refaktoryzację.. 86

UI

Interfejs użytkownika (ang. *User Interface*); warstwa prezentacji odpowiedzialna za sposób wyświetlania danych oraz interakcji użytkownika z aplikacją.. 16, 86

UML

(ang. *Unified Modeling Language*); graficzny język wizualizacji, specyfikowania oraz dokumentowania składników systemów informatycznych. . 17

Spis tabel

Bibliografia

[1] *Disciplined Agile Delivery*. PMI. 1 stycznia 2025. URL: https://www.pmi.org/disciplined-agile/process/introduction-to-dad/why (dostęp 30.10.2025).

[2] *Disciplined Agile Delivery — Lean Life Cycle*. PMI. 1 stycznia 2025. URL: https://www.pmi.org/disciplined-agile/lifecycle/lean-lifecycle (dostęp 30.10.2025).

[3] Stanisław Wrycza, Bartosz Marcinkowski i Krzysztof Wyrzykowski. „Język UML 2.0 w modelowaniu systemów informatycznych". Warszawa: Helion, 2006. ISBN: 83-736-1892-9, 8373618929.

[4] Michał Wolski. *10 wskazówek poprawiających modelowanie procesów biznesowych w notacji BPMN*. 14 maja 2024. URL: https://wolski.pro/2024/05/10-wskazowek-poprawiajacych-modelowanie-procesow-biznesowych-w-notacji-bpmn/ (dostęp 19.11.2025).

[5] *About billing for GitHub Actions*. GitHub Docs. 1 stycznia 2024. URL: https://docs.github.com/en/billing/managing-billing-for-github-actions/about-billing-for-github-actions (dostęp 2.11.2025).

[6] *Scalability and performance targets for Blob storage*. Microsoft Learn. 1 stycznia 2024. URL: https://learn.microsoft.com/azure/storage/blobs/scalability-targets (dostęp 2.11.2025).

[7] *What are the limitations in Mailtrap?* Mailtrap Docs. 1 stycznia 2024. URL: https://help.mailtrap.io/article/111-what-are-the-limitations-in-mailtrap/ (dostęp 2.11.2025).

[8] *LocationIQ Pricing*. LocationIQ. 1 stycznia 2024. URL: https://locationiq.com/pricing (dostęp 2.11.2025).

[9] *Google Maps (Maps URLs)*. Google Maps. 1 stycznia 2024. URL: https://developers.google.com/maps/documentation/urls/get-started?hl=pl (dostęp 2.11.2025).

[10] *OpenFreeMap Documentation*. OpenFreeMap. 1 stycznia 2024. URL: https://openfreemap.org/docs (dostęp 2.11.2025).

[11] *OpenFreeMap Quick Start*. OpenFreeMap. 1 stycznia 2024. URL: https://openfreemap.org/docs/quick-start (dostęp 2.11.2025).

[12] *Open-Meteo API Usage & Pricing*. Open-Meteo. 1 stycznia 2024. URL: https://open-meteo.com/en/docs/usage-and-pricing (dostęp 2.11.2025).

[13] *Tenor API — Documentation*. Tenor. 1 stycznia 2024. URL: https://tenor.com/gifapi/documentation (dostęp 2.11.2025).

[14] *Where the ISS at? API*. wheretheiss.at. 1 stycznia 2024. URL: https://wheretheiss.at/ (dostęp 2.11.2025).

[15] *React useState*. 1 stycznia 2025. URL: https://react.dev/reference/react/useState (dostęp 3.11.2025).

[16] *Redux*. 1 stycznia 2025. URL: https://redux.js.org/ (dostęp 3.11.2025).

[17] *Axios*. 1 stycznia 2025. URL: https://axios-http.com/ (dostęp 3.11.2025).

[18] *Tanstack Query*. 1 stycznia 2025. URL: https://tanstack.com/query/latest (dostęp 3.11.2025).

[19] *Tailwind*. 1 stycznia 2025. URL: https://tailwindcss.com/ (dostęp 3.11.2025).

[20] *Motion*. 1 stycznia 2025. URL: https://motion.dev/ (dostęp 3.11.2025).

Załączniki

Płyta CD z następującą zawartością:

- *pliki projektowe* – pliki składające się na całość projektu
  - repozytorium kodu źródłowego wraz z instrukcją zbudowania i uruchomienia projektu
  - źródło pracy inżynierskiej.
- *Langmesser Adam_Redosz Mateusz_Oziemczuk Stanisław_Badek Kacper_praca pisemna* – katalog zawierający plik PDF z pracą inżynierską.