PodTeksT

odkryj to, co kryje się między wierszami

(bo wiesz, eks...)

Audyt wieloagentowy

Design | Monetyzacja | Optymalizacja

Luty 2026 | Claude Opus 4.6

Spis treści

Spis tabel

Rozdział 1

Audyt wieloagentowy

„Mierz dwa razy, tnij raz. Albo lepiej — mierz dziewięć razy, bo każdy agent widzi co innego."

Niniejszy rozdział dokumentuje wyniki kompleksowego audytu przeprowadzonego w extbfdziewięciu równoległych osiach: extbfaudyt UX/designu strony analizy, extbfaudyt monetyzacji z planem cenowym, extbfaudyt optymalizacji wydajności, extbfunit economics kosztów AI, extbfaudyt bezpieczeństwa, extbfaudyt mobile UX, extbfaudyt onboardingu, extbfaudyt jakości kodu oraz extbfaudyt SEO. Każda oś została zrealizowana przez niezależnego agenta AI (Claude Opus 4.6), który przeanalizował kod źródłowy, architekturę komponentów i wzorce użytkowania.

Kontekst audytu

Audyt przeprowadzono na stanie kodu z lutego 2026 (po Fazie 22). Analiza opiera się na faktycznej zawartości plików źródłowych, nie na założeniach. Kluczowe statystyki wejściowe:

- `src/app/(dashboard)/analysis/[id]/page.tsx` — **1322 linii** kodu, **50+ komponentów** importowanych
- **37** instancji `IntersectionObserver` (via Framer Motion `whileInView`)
- **9** handlerów `useCallback` w jednym pliku
- `src/lib/rate-limit.ts` — rate limiting **wyłączony** (zwraca `{allowed: true}` zawsze)
- Zero infrastruktury monetyzacji — brak auth, brak płatności, brak tierów

1.1 Audyt UX/designu strony analizy

1.1.1 Diagnoza: monolit na jednej stronie

Strona wyników analizy (`src/app/(dashboard)/analysis/[id]/page.tsx`) renderuje **wszystkie** dane na jednej stronie — od metryk ilościowych, przez wykresy, viral scores, odznaki, share cards, aż po pełną analizę AI i funkcje rozrywkowe. Skutkuje to:

Tabela 1.1: Metryki strony analizy — stan obecny

Metryka	Wartość	Problem
Wysokość strony (2-osobowa)	4000–6000 px	15–20 ekranów mobilnych
Wysokość strony (5+ osób)	5000–7000 px	Dodatkowe sekcje serwera
Importowane komponenty	50+	Pojedynczy `page.tsx`
Instancje IntersectionObserver	37	Każde `whileInView` tworzy osobną instancję
Sekcje nawigacji (SectionNavigator)	6–9	Niewystarczające pokrycie treści
Odległość AI Analysis od foldu	~3000 px	Najcenniejsza treść pogrzebana pod metrykami

Mapa sekcji strony

Aktualna kolejność renderowania w `page.tsx`:

1. **Hero Zone** — `AnalysisHeader`, `ParticipantStrip`, linki Story/Wrapped
2. **Kluczowe metryki** — `KPICards`, `StatsGrid`
3. **Longitudinal Delta** — porównanie z poprzednią analizą (opcjonalne)
4. **Aktywność i czas** — `TimelineChart`, `EmojiReactions`, `HeatmapChart`, `ResponseTimeChart`
5. **Wzorce komunikacji** — `MessageLengthSection`, `WeekdayWeekendCard`, `BurstActivity`, `TopWordsCard`, `SentimentChart`, `IntimacyChart`, `ConflictTimeline`
6. **Viral Scores** — `ViralScoresSection`, `BestTimeToTextCard`, `CatchphraseCard`
7. **Ghost Forecast** — `GhostForecast`
8. **Osiągnięcia** — `BadgesGrid`
9. **Delusion Quiz** — wynik quizu (quiz jest gate screen)
10. **Group Chat Awards** — `GroupChatAwards` (grupowe)
11. **Sieć interakcji** — `NetworkGraph` (grupowe)
12. **Udostępnij wyniki** — `ShareCardGallery`, PDF export, captiony
13. **Analiza AI** — `AIAnalysisButton`, Roast, Attachment, Tone, Love Language, Turning Points, Personality, CPS, Subtext, Court, Dating Profile, Reply Simulator

Kluczowy problem

Sekcja **Analiza AI** (pozycja 13) — czyli najbardziej wartościowa treść z perspektywy monetyzacji i zaangażowania — znajduje się na samym *dole* strony, za 12 sekcjami danych ilościowych. Użytkownik musi przewinąć ~3000 px, zanim zobaczy wyniki AI.

1.1.2 Nawigacja: SectionNavigator

Komponent `SectionNavigator` (`src/components/analysis/SectionNavigator.tsx`) zapewnia nawigację po sekcjach:

- **Desktop:** sticky sidebar z 6–9 przyciskami sekcji
- **Mobile:** bottom pill bar z horizontalnym scrollem
- Tworzy osobny `IntersectionObserver` per sekcję (6–9 instancji)
- Oddzielny scroll listener dla progress baru i przycisku „wróć na górę”

Problem: przy 15+ sekcjach treści, 6–9 przycisków nawigacji jest niewystarczające. Użytkownik nie wie, gdzie jest na stronie, bo wiele sekcji nie ma swojego przycisku w nawigatorze.

1.1.3 Proponowane rozwiązanie: architektura tabowa

Zamiast jednej długiej strony z nawigacją sekcji, proponujemy **5 zakładek (tabów)** z lazy-loadingiem:

Tabela 1.2: Proponowana architektura tabowa strony analizy

Nr	Tab	Zawartość
1	**Przegląd**	AnalysisHeader, ParticipantStrip, KPICards, StatsGrid, HealthScore, LongitudinalDelta
2	**Metryki**	Timeline, Heatmap, ResponseTime, Emoji, MessageLength, Weekday/Weekend, Burst, TopWords, Sentiment, Intimacy, Conflicts, Network
3	**AI Insights**	AIAnalysisButton, Roast, AttachmentStyle, CommunicationStyle, ToneRadar, LoveLanguage, TurningPoints, PersonalityDeepDive, CPS
4	**Rozrywka**	CourtTrial, DatingProfile, ReplySimulator, DelusionQuiz, GhostForecast, EnhancedRoast, StandUp
5	**Udostępnij**	ShareCardGallery, ExportPDF, StandUpPDF, CaptionModal, PhotoUpload, ViralScores, Badges

Korzyści architekturalne

- **Redukcja DOM:** z 50+ komponentów jednocześnie do ∼10 per tab (lazy-loading)
- **IntersectionObserver:** z 37 instancji do ∼8 (tylko aktywny tab)
- **AI Insights na pozycji 3** zamiast 13 — użytkownik trafia do AI w 2 kliknięciach
- **URL hash sync:** `#overview`, `#metrics`, `#ai`, `#entertainment`, `#share` — deep-linkowanie
- **Code splitting:** `React.lazy()` per tab — Entertainment tab (najcięższy) ładowany tylko gdy kliknięty

Proponowana struktura plików

Nowe pliki komponentów

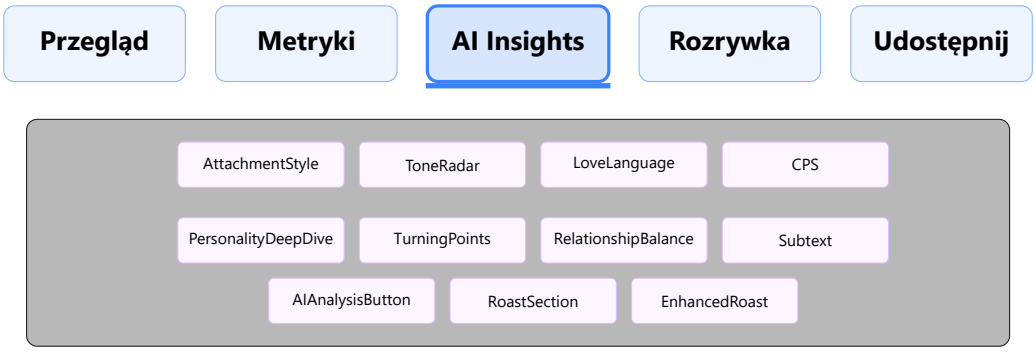
- `src/components/analysis/AnalysisTabs.tsx` — kontener tabów z URL hash sync
- `src/components/analysis/tabs/OverviewTab.tsx` — tab Przegląd
- `src/components/analysis/tabs/MetricsTab.tsx` — tab Metryki
- `src/components/analysis/tabs/AIInsightsTab.tsx` — tab AI Insights
- `src/components/analysis/tabs/EntertainmentTab.tsx` — tab Rozrywka
- `src/components/analysis/tabs/ShareTab.tsx` — tab Udostępnij

Efekt: `page.tsx` redukuje się z 1322 LOC do ∼200 LOC (state, data loading, tab routing).

Server View (5+ uczestników)

Dla grup z 5+ uczestnikami dodatkowy tab **„Serwer"** zastępuje inline sekcje (ServerOverview, TeamRoles, CommunityMap, PersonNavigator, PersonProfile, ServerLeaderboard, PairwiseComparison). Układ tabów zmienia się z 5 na 6.

Diagram architektury tabowej



Rysunek 1.1: Architektura tabowa — przykładowy widok zakładki „AI Insights"

1.2 Audyt monetyzacji

1.2.1 Stan obecny: 100% za darmo

Na dzień audytu **PodTeksT** nie posiada żadnej infrastruktury monetyzacji:

- Brak systemu autentykacji (Supabase Auth — planowany, niezaimplementowany)
- Brak integracji płatności (Stripe — planowany, niezaimplementowany)
- Brak rozróżnienia tierów (Free/Pro/Unlimited)
- Brak limitów użycia (analizy/miesiąc, karty share, PDF export)
- Rate limiting **wyłączony w produkcji** — każdy może wysyłać nieograniczoną liczbę requestów AI

1.2.2 Proponowany model: Freemium 3-Tier

Tabela 1.3: Model cenowy — porównanie tierów

Funkcja	Free (0 PLN)	Pro (29,99 PLN/mies)	Unlimited (49,99 PLN/mies)
Metryki ilościowe	Wszystkie	Wszystkie	Wszystkie
AI Pass 1 (Ton)	**Tak**	**Tak**	**Tak**
AI Pass 2–4	**Nie**	**Tak**	**Tak**
Roast (podstawowy)	**Tak**	**Tak**	**Tak**
Enhanced Roast	**Nie**	**Tak**	**Tak**
Rozrywka (Court, Dating, Simulator)	**Nie**	**Tak**	**Tak**
Share Cards	5 podstawowych	Wszystkie	Wszystkie
Export PDF	**Nie**	**Tak**	**Tak**
StandUp PDF	**Nie**	**Tak**	**Tak**
CPS + Subtext	**Nie**	**Tak**	**Tak**
Analizy / miesiąc	3	15	Bez limitu
Platformy	Wszystkie	Wszystkie	Wszystkie

Logika Free Tier

Free Tier zapewnia **pełne metryki ilościowe** (client-side, zero kosztów serwera) plus ograniczoną analizę AI:

- **Pass 1 (Ton i styl)** — darmowy, daje smak AI analizy
- **Roast (podstawowy)** — darmowy, wiralowy element zachęcający do udostępniania
- **5 kart share** — wystarczające do wiralowego rozprzestrzeniania (z watermarkiem „PodTeksT.app")
- **3 analizy/miesiąc** — wystarczające do testowania, niewystarczające do regularnego użycia

Logika paywall

Paywall aktywuje się w **momencie największego zaangażowania**:

1. Użytkownik uploaduje rozmowę — **bezpłatne**
2. Widzi pełne metryki ilościowe + KPI — **bezpłatne**
3. Uruchamia analizę AI — Pass 1 + Roast **bezpłatne**
4. Klika tab „AI Insights" — widzi wyniki Pass 1, ale Pass 2–4 za **blurred overlay z CTA „Odblokuj pełną analizę"**
5. Klika tab „Rozrywka" — każdy przycisk opakowany w **PaywallGate**
6. Klika tab „Udostępnij" — 5 kart dostępnych, reszta z CTA „Odblokuj 20+ kart"

1.2.3 Implementacja techniczna

Faza 1 — lokalna (bez Stripe)

Minimalna infrastruktura do testowania konwersji i UX paywallu:

Nowe pliki

- `src/contexts/TierContext.tsx` — `TierProvider` z hookiem `useTier()`
- `src/components/shared/PaywallGate.tsx` — komponent paywall z blurred preview

TierContext

- Przechowuje tier w `localStorage`: klucz `podtekst-tier`, wartości: `free` | `pro` | `unlimited`
- Hook `useTier()` zwraca: `{ tier, canAccess(feature), upgrade() }`
- Funkcja `canAccess(feature: string)` sprawdza mapę uprawnień:

```
const FEATURE_MAP: Record<string, Tier> = {
  'ai-pass-2': 'pro',
  'ai-pass-3': 'pro',
  'ai-pass-4': 'pro',
  'enhanced-roast': 'pro',
  'court-trial': 'pro',
  'dating-profile': 'pro',
  'reply-simulator': 'pro',
  'cps-screener': 'pro',
  'subtext-decoder': 'pro',
  'pdf-export': 'pro',
  'standup-pdf': 'pro',
  'all-share-cards': 'pro',
```

```
  'unlimited-analyses': 'unlimited',
};
```

Listing 1.1: Mapa uprawnień funkcji

PaywallGate

Komponent opakowujący treść pro/unlimited:

- Props: `requiredTier: 'pro' | 'unlimited'`, `children`, opcjonalnie `preview` (blurred preview)
- Renderuje: jeśli tier wystarczający — `children`; jeśli nie — blurred overlay z gradientem, ikoną zamka i CTA
- Dev override: `?tier=pro` w URL pozwala na testowanie bez płatności

Faza 2 — produkcyjna (Stripe + Supabase)

Pełna infrastruktura płatności (planowana, niezaimplementowana):

- **Supabase Auth** — rejestracja email + Google OAuth
- **Stripe Checkout** — sesje płatności, subskrypcje miesięczne
- **Stripe Webhooks** — `/api/webhooks/stripe` do obsługi zdarzeń płatności
- **Server-side validation** — tier sprawdzany w API routes przed procesowaniem AI
- **Usage tracking** — licznik analiz/miesiąc per user w Supabase

1.2.4 Share Cards jako viral loop

Share Cards stanowią kluczowy mechanizm wiralowego wzrostu:

Tabela 1.4: Strategia wiralowa — karty share

Tier	Dostępne karty	Watermark
Free	PersonalityCard, VersusCard, StatsCard, ReceiptCard, BadgesCard	„PodTeksT.app” (dolny róg)
Pro	Wszystkie 20+ kart	Bez watermarku

Watermark na Free-tier kartach służy jako **darmowa reklama** — każda karta udostępniona na social media kieruje nowych użytkowników do aplikacji.

1.2.5 Projekcja przychodów

Założenia modelowe — niewalidowane

Poniższe liczby MAU (500 / 2 000 / 5 000) to **założenia modelowe**, nie walidowane prognozy. Na dzień audytu brak:

- strategii akwizycji użytkowników (paid ads, influencerzy, SEO content),
- budżetu marketingowego,
- danych historycznych o ruchu,
- walidowanego kanału wzrostu organicznego.

Rzeczywiste MAU może być **10× niższe** bez inwestycji w kanały wzrostu. Projekcja służy wyłącznie do modelowania unit economics — nie jest prognozą biznesową.

Tabela 1.5: Projekcja przychodów (12 miesięcy)

Metryka	Miesiąc 3	Miesiąc 6	Miesiąc 12
MAU (Monthly Active Users)	500	2 000	5 000
Konwersja Free→Pro	3%	5%	5%
Konwersja Pro→Unlimited	10%	15%	20%
Płacący Pro	15	100	250
Płacący Unlimited	2	15	50
MRR (Monthly Recurring Revenue)	550 PLN	3 749 PLN	9 997 PLN
ARR (Annual Recurring Revenue)	6 600 PLN	44 988 PLN	119 964 PLN

Założenia modelowe (niewalidowane)

Konwersja 3–5% (benchmark freemium SaaS B2C), churn 5%/mies, ARPU: Pro 29,99 PLN, Unlimited 49,99 PLN. Wzrost zakładany jako organiczny (viral share cards + SEO + word-of-mouth). **Żadne z tych założeń nie zostało zwalidowane danymi z PodTeksT.** Projekcja służy wyłącznie do modelowania struktury kosztów AI — nie jest prognozą biznesową. Poniżej (sekcja 1.2.6) przedstawiamy prognozę opartą na konkretnej strategii akwizycji.

1.2.6 Strategia akwizycji i prognoza biznesowa

Poprzednie projekcje (tabela 1.5) operowały na arbitralnych liczbach MAU bez planu ich osiągnięcia. Niniejsza sekcja przedstawia **konkretną strategię akwizycji** z kanałami wzrostu, budżetami i realistycznymi prognozami opartymi na benchmarkach polskiego rynku consumer SaaS.

Kanały akwizycji

Tabela 1.6: Kanały akwizycji — charakterystyka i szacunkowy CAC

Kanał	Typ	Mechanizm	CAC (PLN)
Viral share cards	Organic	Karty Free z watermarkiem „PodTeksT.app" udostępniane na IG/TikTok — każda karta to darmowa reklama	∼0
SEO / blog content	Organic	Artykuły: „jak wyeksportować rozmowę z Messengera", „co znaczy ghosting", „analiza rozmowy z eksem"	∼0 (czas)
TikTok/Reels content	Content	Format: „Przeanalizowałem rozmowę z moim eksem" — viralowy format dramatyczny, 9:16	5–15
Mikro--influencerzy	Influencer	Polscy twórcy relationship/drama (10–50k followers), 500–2 000 PLN/kolaboracja	10–30
Makro--influencerzy	Influencer	100k+ followers, 3 000–10 000 PLN/kolaboracja, większy zasięg ale mniejsze zaangażowanie	15–40
Meta Ads (IG/FB)	Paid	Targeting 18–35, zainteresowania: związki, randki, psychologia, dramaty	15–40
Google Ads	Paid	Keywords: „analiza rozmowy", „kto pisze pierwszy", „ghosting sprawdzić"	20–50

Viral share cards — kluczowy kanał organiczny

PodTeksT posiada 20+ typów kart share (PersonalityCard, VersusCard, ReceiptCard, MugshotCard, etc.). Karty Free tier zawierają watermark „PodTeksT.app" w dolnym rogu. Każda karta udostępniona na social media to **darmowa reklama** z wbudowanym CTA. Benchmarki virality-driven SaaS (Wrapped, Spotify Blend):

- 1 karta udostępniona → średnio 20–50 wyświetleń (stories), 100–500 (post/reel)
- Konwersja wyświetlenie→wizyta: 2–5% (stories), 0,5–2% (feed)
- **Przy 100 kartach udostępnionych dziennie:** 200–1 000 nowych wizyt/dzień organicznie

Scenariusze budżetowe

Tabela 1.7: Trzy scenariusze wzrostu — budżet, kanały, prognoza MAU

Scenariusz	Budżet/mies	Kanały	M3	M6	M12
A — Bootstrap	0–500 PLN	Viral cards, SEO, własny TikTok	100	400	1 200
B — Lean	2 000–5 000 PLN	Organic + mikro-influencerzy + mały Meta Ads	300	1 000	3 000
C — Growth	10 000–20 000 PLN	Pełny mix: organic + influencerzy + paid	800	3 000	10 000

Rekomendacja: Scenariusz B (Lean Startup). Pozwala na walidację kanałów akwizycji przy umiarkowanym ryzyku finansowym. Jeśli CAC z influencerów i Meta Ads wyniesie $<$30 PLN przy konwersji $>$2%, przejście na scenariusz C jest uzasadnione. Scenariusz A jest bezpieczny, ale wzrost może być zbyt wolny, aby zwalidować model biznesowy przed wyczerpaniem motywacji.

Prognoza finansowa — scenariusz B (Lean Startup)

Tabela 1.8: Prognoza biznesowa — scenariusz B (Lean Startup, 2 000–5 000 PLN/mies)

Metryka	Miesiąc 3	Miesiąc 6	Miesiąc 12
Akwizycja			
MAU	300	1 000	3 000
Nowi użytkownicy (organic)	∼100/mies	∼300/mies	∼500/mies
Nowi użytkownicy (paid)	∼100/mies	∼200/mies	∼250/mies
Budżet marketing/mies	2 000 PLN	3 500 PLN	5 000 PLN
CAC blended	∼20 PLN	∼18 PLN	∼15 PLN
Konwersja			
Konwersja Free→Pro	2%	3%	4%
Płacący Pro	6	30	120
Płacący Unlimited	1	5	20
Churn miesięczny	8%	6%	5%
Przychody i koszty			
MRR (przychód)	230 PLN	1 150 PLN	4 600 PLN
Koszt AI/mies	∼20 PLN	∼110 PLN	∼460 PLN
Koszt marketing/mies	2 000 PLN	3 500 PLN	5 000 PLN
Infrastruktura (Cloud Run)	∼50 PLN	∼100 PLN	∼200 PLN
Wynik netto/mies	**−1 840 PLN**	**−2 560 PLN**	**−1 060 PLN**
Skumulowana strata	−5 520 PLN	−18 720 PLN	−30 000 PLN
Break-even (szacunek)	∼Miesiąc 15–18 (przy utrzymaniu wzrostu)		
ARR w miesiącu 12			**∼55 000 PLN**

Kluczowe wskaźniki rentowności

- **CAC payback period:** Przy CAC = 20 PLN i ARPU = 29,99 PLN/mies → payback <1 miesiąc. Przy uwzględnieniu churnu (LTV ≈ 5–8 mies × 29,99 PLN = 150–240 PLN) → **LTV/CAC = 7,5–12**× (zdrowy stosunek >3×).
- **Dlaczego mimo dobrego LTV/CAC firma jest na stracie?** Inwestycja w akwizycję wyprzedza przychody — płacący użytkownicy z miesiąca 1 generują przychód przez 5–8 miesięcy, ale koszt ich pozyskania jest natychmiastowy.
- **Break-even:** ∼miesiąc 15–18, gdy skumulowany MRR pokryje skumulowane koszty marketingowe. Po break-even: margines netto >50% (AI kosztuje <10% MRR).
- **Wymagana inwestycja:** ∼30 000 PLN do break-even (skumulowana strata).

Porównanie z poprzednią projekcją (modelową)

Tabela 1.9: Porównanie: projekcja modelowa vs prognoza ze strategią

Metryka (M12)	Projekcja modelowa	Prognoza Lean	Różnica
MAU	5 000	3 000	−40%
Płacący łącznie	300	140	−53%
MRR	9 997 PLN	4 600 PLN	−54%
ARR	120 000 PLN	55 000 PLN	−54%
Koszt marketingu	0 PLN	5 000 PLN/mies	+60 000 PLN/rok

Wniosek: projekcja modelowa była ∼2× zawyżona

Projekcja modelowa zakładała 5 000 MAU i 5% konwersji **bez** żadnego budżetu marketingowego. Realistyczna prognoza ze strategią (scenariusz B) daje ∼3 000 MAU i 4% konwersji **przy** inwestycji ∼60 000 PLN/rok w marketing. ARR w miesiącu 12: 55 000 PLN vs modelowe 120 000 PLN — **różnica 2,2×**. Projekcja modelowa jest użyteczna wyłącznie jako ilustracja unit economics, nie jako plan biznesowy.

Ryzyka strategii akwizycji

Tabela 1.10: Macierz ryzyk strategii akwizycji

Ryzyko	Prawdop.	Wpływ	Mitigacja
Viral cards nie chwycą	Średnie	Wysoki	A/B testy formatów kart, 9:16 pionowe dla Stories/Reels
CAC >40 PLN na paid	Średnie	Średni	Limit budżetu paid, pivot na influencerów
Churn >10%/mies	Średnie	Średni	Roczne plany z rabatem, retention emails, nowe funkcje
Sezonowość (lato)	Niskie	Średni	Content „wakacyjny” (analiza rozmów z vacation fling)
Konwersja <1,5%	Średnie	Wysoki	Pay-per-analysis (4,99 PLN) jako alternatywa, obniżka Pro do 14,99 PLN

Rekomendacje — fazy wdrożenia

4 fazy strategii akwizycji:

Faza 0 — przygotowanie (teraz, 0 PLN): Zoptymalizować 3–5 kart share pod TikTok/IG (format 9:16, duży tekst, watermark). Dodać export guide per platforma (Messenger, WhatsApp) — to jednocześnie SEO content i redukcja drop-off.

Faza 1 — walidacja (miesiąc 1–3, ∼2 000 PLN/mies): Własny content TikTok (3–5 filmów/tydzień, format „analizuję rozmowę z eksem”). Kolaboracja z 2–3 mikro-influencerami. Cel: **walidacja CAC** — czy <30 PLN? Jeśli tak — przejście do Fazy 2.

Faza 2 — skalowanie (miesiąc 4–6, ∼3 500 PLN/mies): Jeśli CAC z Fazy 1 <30 PLN:

uruchomić Meta Ads (1 000–2 000 PLN/mies) + więcej influencerów. Jeśli CAC >40 PLN: pivot na pure organic + content, obniżyć cenę Pro do 14,99 PLN.

Faza 3 — optymalizacja LTV (miesiąc 7–12, ~5 000 PLN/mies): Roczne plany z rabatem (~199 PLN/rok = 16,60 PLN/mies), referral program (polecaj → 1 miesiąc gratis), retention emails („masz nową rozmowę do przeanalizowania?"). Cel: obniżyć churn z 8% do 4%, wydłużyć LTV z 5 do 10+ miesięcy.

1.3 Audyt optymalizacji wydajności

1.3.1 Problemy krytyczne (P0)

Rate Limiting wyłączony

KRYTYCZNE — ryzyko kosztowe i bezpieczeństwa

Plik `src/lib/rate-limit.ts` zawiera **celowo wyłączony** rate limiting:

```
export function rateLimit(_limit: number, _windowMs: number) {
  // TODO: re-enable rate limiting before production
  return function checkRateLimit(_ip: string) {
    return { allowed: true };  // <-- ZAWSZE true
  };
}
```

Listing 1.2: Rate limiting — stan aktualny

Konsekwencje:
- Każdy użytkownik może wysyłać **nieograniczoną** liczbę requestów do API Gemini
- Brak ochrony przed atakami typu abuse/DoS na endpointy AI
- Koszty API Gemini mogą rosnąć niekontrolowanie
- Plik zawiera poprawną implementację `rateLimitMap` (linie 1–13), ale funkcja zwracająca nigdy z niej nie korzysta

Naprawa:

Przywrócić logikę rate limitingu — plik już zawiera `Map<string, {count, resetTime}>` z automatycznym czyszczeniem co 5 minut. Wystarczy odkomentować sprawdzanie w `checkRateLimit()`.

Nadmiar IntersectionObserver (37 instancji)

Framer Motion `whileInView` tworzy osobny `IntersectionObserver` dla każdego elementu. Na stronie analizy:

Tabela 1.11: IntersectionObserver — rozkład instancji

Źródło	Instancje	Uwagi
`whileInView` w page.tsx	37	Każdy `motion.div` z animacją fade-in
SectionNavigator	6–9	Osobny observer per sekcję nawigacji
Scroll listener (progress)	1	`window.addEventListener('scroll')`
Razem	**44–47**	Na jednej stronie

Wpływ na wydajność:

- Każdy `IntersectionObserver` utrzymuje referencje do obserwowanych elementów i callbacków
- 44+ instancji = szacunkowo 10–20 MB dodatkowego zużycia pamięci
- Browser musi przeliczać intersection ratio przy każdym scroll event dla wszystkich obserwatorów
- Na urządzeniach mobilnych z 4 GB RAM to odczuwalne spowolnienie

Naprawa:

Architektura tabowa naturalnie redukuje liczbę do ~8–10 per tab. Dodatkowo, pojedynczy shared `IntersectionObserver` z `threshold: [0, 0.5, 1]` zamiast wielu osobnych.

1.3.2 Problemy ważne (P1)

Brak lazy-loadingu treści tabów

Choć 18 komponentów używa `dynamic()` (Next.js lazy import), wszystkie ładują się natychmiast po wejściu na stronę, ponieważ leżą w jednym drzewie renderowania. Z architekturą tabową:

- `React.lazy()` per tab — JS Entertainment i Share tabów ładowany dopiero po kliknięciu
- Szacowana redukcja initial JS bundle: **40–60%** (Entertainment tab jest najcięższy: Gemini API calls, chat simulator, quiz engine)

Brakujące React.memo na ciężkich komponentach

Następujące komponenty re-renderują się przy **każdej** zmianie state'u w page.tsx (9 handlerów useCallback = częste re-rendery):

Tabela 1.12: Komponenty wymagające React.memo

Komponent	Ciężkość	Powód re-renderów
`TimelineChart`	Wysoka	Recharts render per re-render
`HeatmapChart`	Wysoka	Duża siatka SVG
`ResponseTimeChart`	Wysoka	Recharts render per re-render
`ShareCardGallery`	Bardzo wysoka	20+ kart, canvas rendering
`PersonalityDeepDive`	Wysoka	Wiele sub-komponentów
`NetworkGraph`	Wysoka	Obliczenia grafu + SVG

Naprawa:

Dodać `React.memo()` z custom comparator na powyższe komponenty. Przenieść handlery `useCallback` do odpowiednich tab komponentów, aby zmiana state w jednym tabie nie powodowała re-renderów w innym.

Stabilizacja callbacków

Plik `page.tsx` zawiera 9 handlerów `useCallback`:

1. `handleAIComplete`
2. `handleRoastComplete`
3. `handleCPSComplete`
4. `handleSubtextComplete`
5. `handleDelusionComplete`
6. `handleCourtComplete`
7. `handleDatingProfileComplete`
8. `handlePhotoUpload` / `handlePhotoRemove`
9. `handleImageSaved`

Wszystkie mają `[analysis]` w tablicy zależności — więc każda zmiana `analysis` (np. zakończenie dowolnej analizy AI) powoduje re-tworzenie **wszystkich** callbacków i re-render **wszystkich** komponentów, które je przyjmują.

Naprawa:

Przenieść handlery do odpowiednich tab komponentów. Tab AI Insights posiada tylko handlery AI, tab Entertainment — handlery rozrywkowe. Zmiana w jednym tabie nie wpływa na callbacki w drugim.

1.3.3 Problemy dodatkowe (P2)

Web Worker dla parsowania

Parsery (`src/lib/parsers/`) wykonują się na main thread:

- Duże eksporty Messenger (50 000+ wiadomości): 200–400 ms blokowania UI
- Rozwiązanie: `new Worker(new URL('./parser.worker.ts', import.meta.url))`
- Użytkownik widzi responsywny UI z progress barem zamiast zamrożonej strony

IndexedDB quota management

Brak zarządzania limitami przechowywania:

- Każda analiza to ~50–200 KB w IndexedDB (dane ilościowe + jakościowe + obrazy)
- 50 analiz = ~5–10 MB — bez ostrzeżeń ani auto-cleanup
- Rozwiązanie: `navigator.storage.estimate()` przed zapisem, ostrzeżenie przy >80% quota

Wirtualizacja ShareCardGallery

20+ kart share renderowanych jednocześnie:

- Każda karta to komponent z canvas rendering (html2canvas)

- Rozwiązanie: intersection-based lazy rendering lub `react-window`

1.4 Unit economics i koszty AI

Każde wywołanie analizy AI w **PodTeksT** generuje realne koszty po stronie Google Gemini API. Niniejsza sekcja dokumentuje konfigurację modelu, szacunki tokenów, koszty per analiza w PLN oraz margines na tier w kontekście proponowanego modelu cenowego. Wszystkie kalkulacje używają oficjalnego cennika Google Gemini z lutego 2026 przeliczonego na PLN.

1.4.1 Konfiguracja modelu

Tabela 1.13: Konfiguracja modelu Gemini w **PodTeksT**

Parametr	Wartość	Plik
Model	`gemini-3-flash-preview`	`gemini.ts:67`
Temperature	0.3	`gemini.ts:70`
Max output tokens (domyślny)	8 192	`gemini.ts:69`
Max output tokens (CPS/Subtext)	16 384	per batch call
Response MIME type	`application/json`	`gemini.ts:71`
Safety settings	Wszystkie `BLOCK_NONE`	`gemini.ts:49--54`
Retry policy	3× exponential backoff (1s, 2s, 4s)	`gemini.ts:56--105`
Model obrazów	`gemini-3-pro-image-preview`	osobny endpoint

Brak prompt cachingu i response cachingu

Na dzień audytu **PodTeksT** nie implementuje żadnej formy cachowania:

- **Prompt caching:** brak — ten sam system prompt (np. `PASS_1_SYSTEM`) jest wysyłany jako pełny tekst przy każdym wywołaniu, nawet jeśli 10 użytkowników uruchomi analizę w ciągu minuty.
- **Response caching:** brak — ponowna analiza tej samej konwersacji generuje pełny koszt API, mimo identycznych danych wejściowych.

1.4.2 Tokeny promptów systemowych

Tabela 1.14: Szacunkowa liczba tokenów promptów systemowych

Prompt	∼Tokeny	Lokalizacja
PASS_1_SYSTEM (Overview)	∼450	`prompts.ts:16--57`
PASS_2_SYSTEM (Dynamics)	∼1 100	`prompts.ts:63--145`
PASS_3_SYSTEM (Individual Profiles)	∼1 400	`prompts.ts:151--296`
PASS_4_SYSTEM (Synthesis)	∼900	`prompts.ts:302--369`
ROAST_SYSTEM	∼550	`prompts.ts:375--410`
ENHANCED_ROAST_SYSTEM	∼750	`prompts.ts:416--458`
STANDUP_ROAST_SYSTEM	∼900	`prompts.ts:464--516`
SUBTEXT_SYSTEM	∼1 200	`prompts.ts:574--620`
CPS_BATCH_PROMPT	∼600–800	`prompts.ts:526--568`
Suma (pełna analiza)	**∼7 850**	

1.4.3 Wywołania API per scenariusz

Liczba wywołań Gemini API zależy od zakresu analizy:

Tabela 1.15: Wywołania API Gemini per scenariusz analizy

Scenariusz	Wywołania	Szczegóły
Podstawowa analiza (standard)	5	Pass 1–4 + Roast
Z 2 uczestnikami (Pass 3 per osoba)	6	Pass 1–4 + Roast + dodatkowy Pass 3
+ Enhanced Roast	+1	Roast z pełnym kontekstem psychologicznym
+ StandUp Comedy	+1	7 aktów w jednym wywołaniu
+ CPS (3 batche)	+3	21 pytań per batch
+ Subtext (3 batche)	+3	8 okien kontekstowych per batch
+ Court Trial	+1	Wykorzystuje wyniki Pass 1, 2, 4
+ Dating Profile	+1	Profil randkowy z wzorców
+ Reply Simulator (×5)	+5	Maks. 5 wymian per sesja
+ Image Generation	+1	Gemini Pro (droższy model)
Pełny zestaw (max)	**∼22**	Przy 2 uczestnikach

1.4.4 Szacunek tokenów per wywołanie

Założenia szacunkowe:
- System prompt: ~1 000 tokenów (średnia z tabeli 1.14)
- Próbka wiadomości: 200–500 wiadomości × ~30 tokenów/wiadomość ≈ 6 000–15 000 tokenów
- Kontekst ilościowy: ~500 tokenów
- **Łącznie input per wywołanie: ~16 000 tokenów**
- Typowy output per wywołanie: ~4 000 tokenów (JSON z analizą)
- Maksymalna długość wiadomości: 2 000 znaków (`sanitizeForPrompt()`)

1.4.5 Cennik Gemini API (PLN)

Tabela 1.16: Cennik Gemini API w **PodTeksT** (PLN, luty 2026)

Kategoria	Flash (tekst/obraz/video)	Pro (obraz)
Input (per 1M tokenów)	**0,25 PLN**	wyższy
Output z thinking (per 1M tokenów)	**4,50 PLN**	wyższy
Cached input (per 1M tokenów)	**0,20 PLN**	—
Batch API (50% taniej)	0,125 / 2,25 PLN	—

Uwaga: thinking tokens w cenie output

Cena output **4,50 PLN/1M** obejmuje **thinking tokens** generowane wewnętrznie przez model. Jeśli model generuje 2–3× więcej thinking tokenów niż widoczny output, rzeczywisty koszt output może być 2–3× wyższy od wartości podanych w tabeli 1.17. Zalecamy wdrożenie monitoringu thinking tokenów (patrz tabela 1.24, pozycja 3).

1.4.6 Koszt per analiza (PLN)

Tabela 1.17: Koszt API per analiza — scenariusze (PLN)

Scenariusz	Wywoł.	Input (tokens)	Output (tokens)	Koszt
Podstawowa (5 calls)	5	80 000	20 000	**0,11 PLN**
Rozszerzona (10 calls)	10	160 000	40 000	**0,22 PLN**
Pełna (20 calls)	20	320 000	80 000	**0,44 PLN**
Pełna + obraz	21	340 000	85 000	**~0,50 PLN**

Wzór kalkulacji (PLN)

$$\text{Koszt} = \frac{\text{Input tokens} \times 0{,}25}{1\,000\,000} + \frac{\text{Output tokens} \times 4{,}50}{1\,000\,000}$$

Przykład — podstawowa analiza (5 wywołań):

$$\frac{80\,000 \times 0{,}25}{1\,000\,000} + \frac{20\,000 \times 4{,}50}{1\,000\,000} = 0{,}020 + 0{,}090 = 0{,}11 \text{ PLN}$$

Kluczowa obserwacja: Koszt output (**4,50 PLN/1M**) dominuje w strukturze kosztów — stanowi 82% kosztu podstawowej analizy (0,090 z 0,110 PLN). Input (0,25 PLN/1M) jest marginalny. To oznacza, że optymalizacja input cachingu ma **ograniczony wpływ** na całkowity koszt.

1.4.7 Margines per tier (PLN)

Poniższa tabela porównuje przychód z subskrypcji z kosztem AI dla różnych scenariuszy użycia. Ceny tierów w PLN:

- **Pro: 29,99 PLN/mies.**
- **Unlimited: 49,99 PLN/mies.**

Tabela 1.18: Analiza marginu per tier i scenariusz użycia (PLN)

Scenariusz	Tier	Cena	Analizy/m	Koszt AI	Margines
Pro, 3 basic	Pro	29,99 PLN	3	0,33 PLN	**99%**
Pro, 15 basic	Pro	29,99 PLN	15	1,65 PLN	**95%**
Pro, 5 full	Pro	29,99 PLN	5	2,20 PLN	**93%**
Pro, 15 full	Pro	29,99 PLN	15	6,60 PLN	**78%**
Unlimited, 10 basic	Unlimited	49,99 PLN	10	1,10 PLN	**98%**
Unlimited, 30 basic	Unlimited	49,99 PLN	30	3,30 PLN	**93%**
Unlimited, 30 full	Unlimited	49,99 PLN	30	13,20 PLN	**74%**

Marże vs poprzednia analiza USD — z kontekstem

Marże w PLN wyglądają zdrowiej niż w analizie USD (Pro 15 full: 78% vs 14%, Unlimited 30 full: 74% vs 8%). Ale porównanie wymaga kontekstu:

- **Arbitraż walutowy:** koszty API denominowane w USD (∼4,05 PLN). Przy deprecjacji PLN (np. do 4,50 PLN/USD) marże spadają o ∼5 pp.
- **Niższa gotowość do płacenia:** polski rynek consumer SaaS akceptuje niższe ceny niż rynek USD (patrz tabela 1.19).
- **Marża na AI ≠ rentowność biznesu:** koszty CAC (pozyskania użytkownika), infrastruktury, developmentu nie są uwzględnione.

Tabela 1.19: Kontekst cenowy — polski rynek consumer SaaS

Produkt	Cena/mies.	Częstotliwość	Porównanie z PodTeksT
Spotify Premium	23,99 PLN	Codziennie	Pro 29,99 = **droższa**
Netflix Basic	33 PLN	Codziennie	Pro 29,99 = tańsza
Tinder Gold	∼50 PLN	Codziennie	Unlimited 49,99 ≈ Tinder
YouTube Premium	26,99 PLN	Codziennie	Pro 29,99 ≈ porównywalny

Problem: cena vs częstotliwość użycia

PodTeksT w cenie 29,99 PLN/mies. kosztuje więcej niż Spotify Premium, a jest używana **okazjonalnie** (1–5× w miesiącu vs codziennie). Gotowość polskich użytkowników do płacenia za narzędzie okazjonalne jest istotnie niższa niż za usługi codzienne.
Rekomendacja: rozważyć model **pay-per-analysis** (np. 4,99 PLN/analiza) obok subskrypcji, lub obniżyć Pro do **14,99–19,99 PLN/mies.** Alternatywnie: zaoferować roczny plan z rabatem (∼199 PLN/rok = 16,60 PLN/mies.).

1.4.8 Analiza break-even

Ile analiz może wykonać użytkownik, zanim stanie się nierentowny?

Tabela 1.20: Break-even: maksymalna liczba analiz per tier przy 0% marginu (PLN)

Tier	Cena	Break-even (basic)	Break-even (full)	Limit w planie
Pro	29,99 PLN	∼273 analiz	∼68 analiz	15 analiz
Unlimited	49,99 PLN	∼454 analizy	∼114 analiz	Soft cap 50/mies.

Wniosek: Przy cenach w PLN break-even jest **wielokrotnie wyższy** niż realistyczne użycie. Tier Pro z limitem 15 analiz/miesiąc jest bezpieczny nawet przy **pełnych** analizach (break-even = 68, limit = 15 — 4,5× zapas). Tier Unlimited z soft capem 50/miesiąc również ma duży margines bezpieczeństwa (break-even = 114 full, soft cap = 50 — 2,3× zapas).

1.4.9 Wpływ cachingu na koszty

Tabela 1.21: Wpływ prompt cachingu na koszt podstawowej analizy (PLN)

Metryka	Bez cachingu	Z cachingiem	Oszczędność
Input cost (5 calls)	0,020 PLN	0,016 PLN	20% inputu
Output cost (5 calls)	0,090 PLN	0,090 PLN	0%
Suma	**0,110 PLN**	**0,106 PLN**	**∼4%**

Ograniczony wpływ cachingu

Prompt caching ma **bardzo ograniczony wpływ** na koszt całkowity analizy, ponieważ:
- Koszt output (**4,50 PLN/1M**) stanowi 82% kosztu i **nie podlega cachowaniu**
- Koszt input (**0,25 PLN/1M**) jest marginalny — nawet redukcja o 80% (caching z 0,25 do 0,05 PLN) oszczędza zaledwie ∼**4%** na podstawowej analizie (z 0,110 na 0,106 PLN)
- Prompt systemowy (∼1 000 tokenów) stanowi tylko 6% całego inputu — reszta to unikalne dane konwersacji

W porównaniu z poprzednią analizą (14–36% oszczędności), rzeczywisty wpływ cachingu jest **minimalny**.

1.4.10 Propozycje optymalizacji kosztów

1. Response caching (hash)

Ponowna analiza tej samej konwersacji powinna korzystać z wcześniej wygenerowanych wyników:

- Hash wiadomości (SHA-256 z treści i timestampów) jako klucz cache'u
- Jeśli hash się zgadza — zwróć wynik z IndexedDB zamiast wywoływać API
- **Oszczędność:** 100% kosztów re-analiz (szacunkowo 10–20% wszystkich wywołań)
- **Trudność:** Łatwa — wyłącznie po stronie klienta

2. Batch API dla funkcji non-real-time

Funkcje, które nie wymagają natychmiastowego wyniku, mogą korzystać z Batch API (50% taniej):

- **StandUp Comedy PDF** — użytkownik i tak musi czekać na generowanie PDF
- **CPS (3 batche)** — wynik nie jest interaktywny
- **Szacowana redukcja:** 50% kosztów tych wywołań ($\sim$4 z 20 wywołań)
- Batch API: 0,125 PLN input / 2,25 PLN output per 1M tokenów

3. Monitoring thinking tokenów

- Wdrożyć logowanie liczby thinking tokenów per wywołanie
- Jeśli model generuje $>2\times$ thinking tokens vs widoczny output — rozważyć zmianę temperature lub struktury promptu
- **Cel:** Widoczność kosztowa — bez monitoringu nie wiemy, czy output kosztuje 4,50 PLN czy efektywnie 9–13 PLN per 1M tokenów
- **Trudność:** Łatwa — dane dostępne w response metadata Gemini

4. Prompt caching (Gemini Context Caching)

- Prompty systemowe (`PASS_1_SYSTEM`, etc.) identyczne dla każdego użytkownika
- Cached input: 0,20 PLN/1M zamiast 0,25 PLN/1M (20% taniej)
- **Rzeczywista oszczędność:** $\sim$**4%** na całkowitym koszcie analizy — ograniczona, bo output dominuje
- **Trudność:** Średnia — wymaga zarządzania cache TTL

5. Korekta limitów per tier

Przy zdrowych marżach w PLN korekta nie wymaga agresywnych zmian:

Tabela 1.22: Rekomendowane limity tierów (PLN)

Metryka	Obecny plan	Rekomendowany	Uzasadnienie
Pro — analizy/mies.	15	15	Bezpieczne (break-even = 68 full)
Pro — cena	—	29,99 PLN	Rynek polski, przystępna cena
Unlimited — cena	—	49,99 PLN	Zdrowy margines do 114 full
Unlimited — soft cap	brak	50/mies.	Fair use, 2,3× zapas do break-even
Funkcje rozrywkowe	w Pro	Opcjonalnie add-on	9,99 PLN/mies. (opcjonalny)

1.4.11 Skorygowana projekcja przychodów (PLN)

Zastrzeżenie: scenariusz modelowy

Poniższa projekcja opiera się na tych samych **niewalidowanych** założeniach MAU co tabela 1.5 (500 / 2 000 / 5 000 użytkowników). Brak strategii akwizycji, budżetu marketingowego i danych historycznych o ruchu. Celem tej tabeli jest wyłącznie ilustracja struktury kosztów AI w modelu subskrypcyjnym — nie jest prognozą biznesową.

Tabela 1.23: Projekcja przychodów — scenariusz modelowy, niewalidowany (PLN)

Metryka	Miesiąc 3	Miesiąc 6	Miesiąc 12
Użytkownicy Pro	15	100	250
Użytkownicy Unlimited	2	15	50
MRR (przychód)	550 PLN	3 749 PLN	9 997 PLN
Koszt AI	∼48 PLN	∼333 PLN	∼925 PLN
Margines brutto	**91%**	**91%**	**91%**
Zysk brutto/mies.	502 PLN	3 416 PLN	9 072 PLN
ARR (Miesiąc 12)			**∼120 000 PLN**

Kluczowy wniosek (z zastrzeżeniami): Przy cenach w PLN model subskrypcyjny jest **strukturalnie rentowny pod względem kosztów AI** — marża brutto ∼91%. Koszty AI (0,11–0,50 PLN per analiza) są niskie w stosunku do cen subskrypcji (29,99 / 49,99 PLN). Jednak marża brutto na koszcie AI nie oznacza rentowności biznesu — koszty CAC, infrastruktury i marketingu nie są tu uwzględnione. Pełna prognoza z uwzględnieniem strategii akwizycji, budżetów marketingowych i kosztów CAC: patrz sekcja 1.2.6.

1.4.12 Podsumowanie rekomendacji

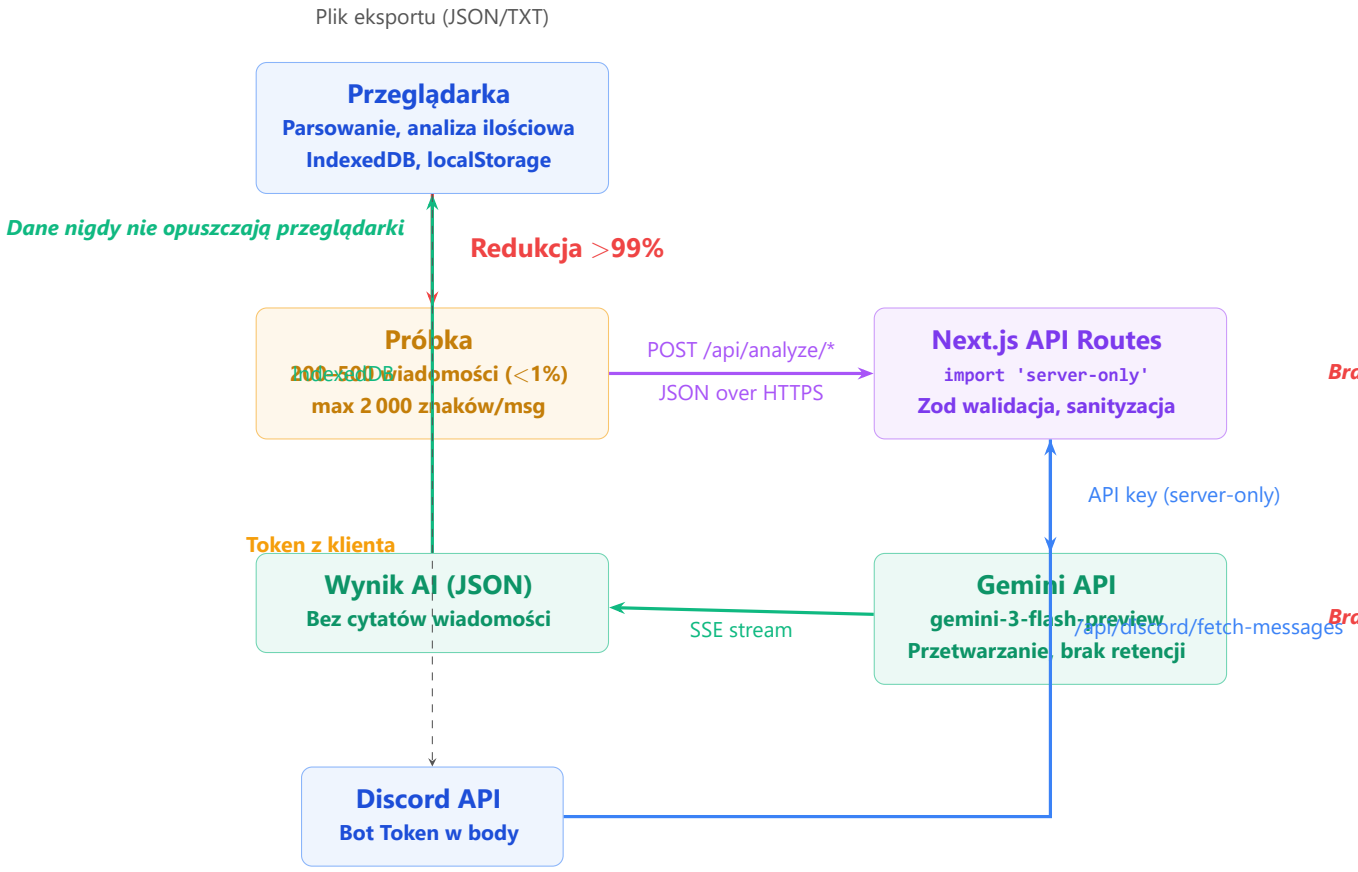
Tabela 1.24: Priorytety optymalizacji kosztów AI

Nr	Optymalizacja	Oszczędność	Trudność	Priorytet
1	Response caching (hash)	10–20% total	Łatwa	P0
2	Batch API (StandUp, CPS)	50% na 4 callach	Średnia	P1
3	Thinking token monitoring	Widoczność kosztowa	Łatwa	P0
4	Prompt caching (Gemini)	∼4% total (ograniczony)	Średnia	P2
5	Korekta limitów tierów	Redukcja ryzyka	Brak kodu	P1

1.5 Bezpieczeństwo i prywatność

Niniejsza sekcja stanowi audyt bezpieczeństwa **PodTeksT** na dzień lutego 2026 (po Fazie 22). Analizuje przepływ danych, walidację API, nagłówki HTTP, rate limiting w środowisku serverless, zgodność z RODO, znane podatności oraz ryzyka specyficzne dla integracji Discord.

1.5.1 Przepływ danych



Rysunek 1.2: Przepływ danych w **PodTeksT**— od uploadu do wyniku. Czerwone adnotacje oznaczają brak trwałego przechowywania.

Kluczowe gwarancje

1. **Surowe wiadomości** — przetwarzane wyłącznie w przeglądarce (parsery + analiza ilościowa). Nigdy nie wysyłane na serwer w pełnej postaci.
2. **Próbka** — 200–500 wiadomości (z 50 000+), każda obcięta do 2 000 znaków. Znaki kontrolne usunięte (`gemini.ts:147--153`).
3. **Serwer** — przetwarza próbkę w pamięci, wysyła do Gemini API, streamuje wynik przez SSE. Żadne dane nie są zapisywane na dysk ani w bazie.
4. **Gemini API** — Google deklaruje brak retencji danych przesłanych przez API (API Terms of Service).

1.5.2 Walidacja API — przegląd schematów Zod

Tabela 1.25: Inwentarz walidacji Zod per endpoint — problemy

Endpoint	Schema	Status	Uwagi / Problemy
`/api/analyze`	`analyzeRequestSchema`	**Częściowa**	`samplesSchema = z.object({}).passthrough()` — brak głębokiej walidacji struktury próbek
`/api/analyze/cps`	`cpsRequestSchema`	**OK**	`participantName` min 1 char
`/api/analyze/subtext`	`subtextRequestSchema`	**OK**	Min 100 wiadomości, pełny schemat `SimplifiedMsg`
`/api/analyze/image`	`imageRequestSchema`	**OK**	Min 1 excerpt, typowane pola
`/api/analyze/court`	`courtRequestSchema`	**Częściowa**	`existingAnalysis` pola to `z.unknown()` — brak walidacji typów
`/api/analyze/dating`	brak Zod	**Niezweryfikowana**	Schemat nie wyeksportowany do `schemas.ts`
`/api/analyze/simulate`	brak Zod	**Niezweryfikowana**	Schemat nie wyeksportowany do `schemas.ts`
`/api/analyze/standup`	`standUpRequestSchema`	**OK**	`quantitativeContext` wymagany
`/api/analyze/enhanced-roast`	`enhancedRoastRequestSchema`	**OK**	Wymaga 4 passów w `qualitative`
`/api/discord/fetch`	brak Zod	**Brak**	Ręczna walidacja `botToken.length >= 50`

Problem: `passthrough()` w samplesSchema

Schemat `samplesSchema = z.object({}).passthrough()` przyjmuje **dowolny obiekt** — nie waliduje, czy zawiera wymagane klucze (`overview`, `dynamics`, `perPerson`), ani typów wartości wewnątrz. Złośliwy aktor może przesłać obiekt z dowolną strukturą, co może spowodować runtime error w `formatMessagesForAnalysis()` lub — co gorsze — wstrzyknięcie nieoczekiwanych danych do promptu Gemini.
Naprawa: Zdefiniować pełny schemat `AnalysisSamples` z walidacją zagnieżdżonych tablic `SimplifiedMessage[]`.

1.5.3 Nagłówki bezpieczeństwa HTTP

Tabela 1.26: Nagłówki bezpieczeństwa — obecne vs brakujące

Nagłówek	Wartość	Status	Źródło
`X-Content-Type-Options`	`nosniff`	Obecny	`next.config.ts:10`
`X-Frame-Options`	`DENY`	Obecny	`next.config.ts:11`
`Referrer-Policy`	`strict-origin-when-cros`	Obecny	`next.config.ts:12`
`Permissions-Policy`	`camera=(), microphone=(), geolocation=()`	Obecny	`next.config.ts:13`
`Content-Security-Policy`	—	Brak	Wymaga konfiguracji nonce
`Strict-Transport-Security`	—	Brak	HSTS dla HTTPS enforcement
`X-XSS-Protection`	—	Brak	Przestarzały, ale wciąż rekomendowany

Content-Security-Policy — rekomendacja

CSP jest najbardziej brakującym nagłówkiem. Konfiguracja dla PodTeksT wymaga uwzględnienia:

```
Content-Security-Policy:
  default-src 'self';
  script-src 'self' 'nonce-{GENERATED}';
  style-src 'self' 'unsafe-inline';   // Tailwind
  connect-src 'self'
    https://generativelanguage.googleapis.com
    https://discord.com/api/
    https://www.google-analytics.com;
  img-src 'self' data: blob:;         // Generated images
  font-src 'self';
  frame-src https://app.spline.design; // Spline 3D
  object-src 'none';
  base-uri 'self';
```

Listing 1.3: Proponowana konfiguracja CSP

Trudność wdrożenia CSP

Konfiguracja CSP w Next.js z Tailwind CSS (`style-src 'unsafe-inline'`) i Spline (`frame-src`) wymaga starannego testowania. Alternatywa: CSP w trybie `report-only` na 2–4 tygodnie, aby zidentyfikować naruszenia przed włączeniem enforcing mode.

Strict-Transport-Security — rekomendacja

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

Listing 1.4: Proponowany nagłówek HSTS

Cloud Run obsługuje HTTPS natywnie, ale HSTS zapobiega downgrade'owi na HTTP w przypadku przekierowania.

1.5.4 Rate limiting w środowisku serverless

Obecna implementacja

Rate limiter **PodTeksT** (`src/lib/rate-limit.ts`) używa `Map<string, {count, resetTime}>` w pamięci procesu:

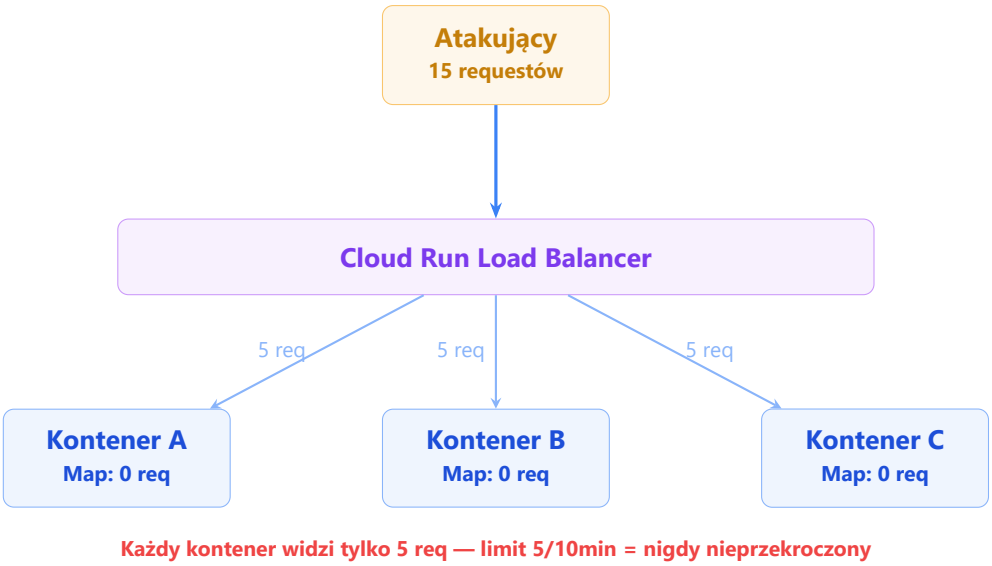
```
const rateLimitMap = new Map<string, {
  count: number; resetTime: number;
}>();

export function rateLimit(_limit: number, _windowMs: number) {
  // TODO: re-enable rate limiting before production
  return function checkRateLimit(_ip: string) {
    return { allowed: true };  // ZAWSZE true
  };
}
```

Listing 1.5: Rate limiter — stan aktualny (wyłączony)

Dwa problemy

Problem 1: wyłączony Funkcja `rateLimit()` ignoruje parametry `_limit` i `_windowMs` (prefiks _ = nieużywane) i zawsze zwraca `{allowed: true}`. Komentarz TODO sugeruje, że jest to tymczasowe wyłączenie na czas developmentu — ale kod działa w produkcji (Cloud Run).

Problem 2: in-memory Map w serverless Nawet po przywróceniu logiki, `Map` in-memory **nie działa** w środowisku Cloud Run:



Rysunek 1.3: Problem rate limitingu in-memory w środowisku Cloud Run. Trzy instancje = trzykrotny efektywny limit.

Mechanizm problemu

1. Cloud Run uruchamia **stateless kontenery** — każdy cold start tworzy nową instancję z pustą `Map`.
2. Przy wielu jednoczesnych żądaniach Cloud Run skaluje się do N kontenerów — każdy z własnym, niezależnym licznikiem.
3. Atakujący z jednego IP wysyła 15 żądań. Load balancer rozkłada je na 3 kontenery po 5. Każdy kontener widzi 5 requestów — poniżej limitu 5/10min. **Wszystkie 15 przechodzi.**
4. Nawet w jednym kontenerze — po minutach bezczynności Cloud Run może go zamknąć. Nowy cold start = `Map` = pusta.

Rekomendacja: Upstash Redis

```typescript
import { Ratelimit } from '@upstash/ratelimit';
import { Redis } from '@upstash/redis';

const redis = new Redis({
  url: process.env.UPSTASH_REDIS_REST_URL!,
  token: process.env.UPSTASH_REDIS_REST_TOKEN!,
});

export const analyzeRateLimit = new Ratelimit({
  redis,
  limiter: Ratelimit.slidingWindow(5, '10 m'),
  analytics: true,
  prefix: 'podtekst:rate-limit',
});
```

Listing 1.6: Proponowany rate limiter z Upstash Redis

- **Upstash Redis** — serverless Redis z REST API, darmowy tier (10 000 req/dzień)
- **Sliding window** — algorytm okna przesuwnego zamiast fixed window (bardziej sprawiedliwy)

- **Współdzielony state** — wszystkie instancje Cloud Run korzystają z tego samego Redis
- **Wbudowana analityka** — dashboard z rate-limit events

1.5.5 Checklist RODO

Tabela 1.27: Checklist zgodności z RODO — stan na luty 2026

Nr	Wymóg RODO	Status	Szczegóły
1	Minimalizacja danych	Spełniony	<1% wiadomości wysyłanych na serwer
2	Brak retencji na serwerze	Spełniony	Dane w pamięci tylko podczas przetwarzania
3	Cookie consent	Spełniony	`CookieConsent.tsx`, GA4 warunkowo
4	Prawo do usunięcia (Art. 17)	Spełniony	`deleteAnalysis()` — atomowe usunięcie
5	Prawo do przenoszenia (Art. 20)	Spełniony	Eksport PDF
6	Przetwarzanie lokalne	Spełniony	IndexedDB, brak server-side storage
7	Polityka prywatności	Brak	Brak strony `/privacy`
8	Endpoint usunięcia danych	Brak	Brak API `DELETE /api/user-data`
9	Eksport danych (maszynowy)	Brak	PDF nie jest formatem maszynowym (brak JSON export)
10	Informacja o przetwarzaniu AI	Częściowy	Brak jawnej informacji, że wiadomości trafiają do Google Gemini
11	Okres retencji cookies	Nieokreślony	`CookieConsent` nie informuje o okresie ważności GA4 cookies
12	DPO (Data Protection Officer)	Nie dotyczy	Wymagany przy przetwarzaniu na dużą skalę (przyszłość)

1.5.6 Raport podatności

jsPDF — 3 podatności HIGH

Pakiet `jspdf` w wersji `^4.1.0` (plik `package.json:23`) posiada trzy znane podatności o poziomie **HIGH**:

Tabela 1.28: Podatności jsPDF 4.1.0

CVE / GHSA	Typ	Severity	Opis
GHSA-p5xg-68wr-hm3m	PDF Injection	HIGH	Wstrzyknięcie obiektów PDF przez AcroForm RadioButton
GHSA-9vjf-qc39-jprp	Object Injection	HIGH	Wstrzyknięcie obiektów PDF via `addJS()`
GHSA-67pg-wm7f-q7fj	DoS	HIGH	Denial of Service przez spreparowane wymiary GIF

Naprawa

Aktualizacja do `jspdf@4.2.0+` rozwiązuje wszystkie trzy podatności. Wymagane zmiany w kodzie: brak — API jest kompatybilne.

```
pnpm update jspdf@latest
```

Brak `dangerouslySetInnerHTML`

Przeszukanie całego katalogu `src/` potwierdza **zero** wystąpień `dangerouslySetInnerHTML`. React auto-escaping chroni przed XSS w warstwie renderingu.

1.5.7 Ryzyko: token Discord w body requestu

Endpoint `/api/discord/fetch-messages` przyjmuje `botToken` w body HTTP requestu:

```
// src/app/api/discord/fetch-messages/route.ts:43-58
let body: {
  botToken?: string;
  channelId?: string;
  messageLimit?: number;
};
// ...
const botToken = (body.botToken && body.botToken.length >= 50)
  ? body.botToken
  : process.env.DISCORD_BOT_TOKEN;
```

Listing 1.7: Akceptacja botToken z body requestu

Wektor ataku

1. Użytkownik wkleja swój Discord Bot Token w UI (`DiscordImport.tsx`).
2. Token jest przechowywany w React state (pamięć przeglądarki).
3. Token jest wysyłany w body POST do `/api/discord/fetch-messages`.
4. Jeśli strona jest podatna na XSS (np. przez wstrzyknięty skrypt w extension'ie przeglądarki):
   - Atakujący może odczytać token z React state lub przechwycić request
   - Token umożliwia dostęp do **wszystkich kanałów** serwera Discord

Ocena ryzyka

Prawdopodobieństwo: **Niskie** — wymaga XSS na stronie **PodTeksT** (brak `dangerouslySetInnerHTML`, CSP planowany)

Wpływ: **Wysoki** — skradziony bot token daje pełny dostęp do API Discorda

Mitigacja obecna: Token nie jest zapisywany w localStorage ani IndexedDB — istnieje tylko w pamięci React state podczas sesji

Rekomendacja

1. **Preferować server-side token:** Używać `process.env.DISCORD_BOT_TOKEN` zamiast przyjmowania tokenu od klienta. Endpoint już obsługuje fallback do env var.
2. **Szyfrowanie w transit:** Jeśli konieczne jest przyjęcie tokenu od użytkownika — szyfrowanie asymetryczne (public key na kliencie, private key na serwerze).
3. **Jednorazowy token:** Po pobraniu wiadomości — natychmiast wyczyścić token z React state.

1.5.8 Brak autentykacji

Brak jakiejkolwiek warstwy autentykacji

Na dzień audytu **PodTeksT** nie posiada:
- Systemu użytkowników (brak rejestracji, logowania)
- Sesji (brak cookies sesyjnych, brak JWT)
- Uprawnień (brak ról, tierów, permisji)
- Identyfikacji użytkownika (rate limit per IP, nie per user)

Każdy, kto zna URL endpointu API, może wysyłać żądania bez ograniczeń (rate limit wyłączony). W kontekście monetyzacji (patrz §1.2.2) oznacza to, że paywall oparty na `TierContext` (localStorage) można obejść bezpośrednim wywołaniem API.

1.5.9 Priorytety napraw bezpieczeństwa

Tabela 1.29: Priorytety napraw bezpieczeństwa

Nr	Problem	Severity	Trudność	Priorytet
1	Przywrócić rate limiting	CRITICAL	Łatwa	P0
2	Migracja rate limit na Redis	HIGH	Średnia	P0
3	Aktualizacja jsPDF do 4.2.0+	HIGH	Łatwa	P0
4	Dodanie polityki prywatności	HIGH	Średnia	P0
5	Content-Security-Policy	MEDIUM	Trudna	P1
6	Strict-Transport-Security	MEDIUM	Łatwa	P1
7	Deep validation `samplesSchema`	MEDIUM	Średnia	P1
8	Informacja o przetwarzaniu AI	MEDIUM	Łatwa	P1
9	Discord token risk mitigation	LOW	Średnia	P2
10	JSON export (Art. 20 RODO)	LOW	Średnia	P2
11	Centralizacja schematów Zod	LOW	Łatwa	P2

Podsumowanie bezpieczeństwa: Architektura client-first PodTeksT zapewnia silne domyślne zabezpieczenia — surowe wiadomości nigdy nie opuszczają przeglądarki, brak `dangerouslySetInnerHTML`, brak retencji danych na serwerze. Jednak **trzy krytyczne braki** (wyłączony rate limit, brak CSP, brak autentykacji) oraz **trzy podatności HIGH** w jsPDF wymagają natychmiastowej uwagi przed skalowaniem produktu.

1.6 Audyt Mobile UX

Mobilne doswiadczenie uzytkownika stanowi krytyczny wektor wzrostu — ponad 60% ruchu na aplikacjach SaaS B2C pochodzi z urzadzen mobilnych. Niniejsza sekcja analizuje responsywnosc, ergonomie dotykowa i wydajnosc PodTeksT na urzadzeniach z ekranami 375–430 px.

1.6.1 Analiza breakpointow

PodTeksT wykorzystuje standardowe breakpointy Tailwind CSS v4 bez customowych rozszerzen:

Tabela 1.30: Breakpointy Tailwind v4 — wykorzystanie w projekcie

Prefix	Szerokosc	Urzadzenia	Zastosowanie w PodTeksT
—	< 640 px	Telefony	Layout bazowy, hero diagonal, bottom bar nav
`sm:`	≥ 640 px	Duze telefony	Scroll indicator visibility, minor spacing
`md:`	≥ 768 px	Tablety	Przelaczenie: sidebar nav, desktop hero, particle bg
`lg:`	≥ 1024 px	Laptopy	Dashboard sidebar, wieksze karty share
`xl:`	≥ 1280 px	Desktopy	Max-width containery, pelny layout

Brak breakpointa pomiedzy 640–768 px

Przelaczenie z mobilnego bottom bar na desktopowy sidebar nastepuje na `md:768px`. Na tabletach w orientacji portrait (768 px) uzytkownik widzi desktop layout, ktory moze byc ciasny. Brak posredniego breakpointa dla tabletow (np. `tab:900px`) oznacza, ze interfejs „skacze" z mobile na desktop bez plynnego przejscia.

1.6.2 Analiza paska nawigacji mobilnej

Komponent `SectionNavigator` (`src/components/analysis/SectionNavigator.tsx`, linie 118–145) renderuje na mobile dolny pasek nawigacyjny z horizontalnym scrollem.

Obliczenie dopasowania na 375 px

Tabela 1.31: Dopasowanie tabow na ekranie 375 px

Widok	Liczba tabow	Szer. taba	Suma	Miesci sie?
Rozmowa 2-os.	6	$\sim$70 px	420 px	**Wymaga scrollu**
Rozmowa grupowa	6	$\sim$70 px	420 px	**Wymaga scrollu**
Server view (5+)	9	$\sim$70 px	630 px	**68% ukryte**
Idealny (5 tabow)	5	$\sim$70 px	350 px	**Tak (25 px zapas)**

Kalkulacja: Dostepna szerokosc = 375 px $-$ 2×12 px (padding `px-3`) = 351 px. Pojedynczy tab: ikona 16 px + gap 4 px + tekst $\sim$40 px + padding 2×12 px = $\sim$84 px. W praktyce krotkie polskie etykiety („Przeg.", „AI", „Share") pozwalaja zmiescic 4–5 tabow bez scrollu.

Aktualna implementacja poprawnie obsluguje overflow:

- `overflow-x-auto` z `scrollbar-none` — scroll bez widocznego scrollbara
- Gradienty na krawedziach (`bg-gradient-to-r/l`) sygnalizuja mozliwosc scrollowania
- Bezpieczny margines dolny: `pb-[max(0.375rem,env(safe-area-inset-bottom))]` — poprawne zachowanie na iPhone'ach z notchem
- Dotykowe sprzezenie zwrotne: `active:scale-95 active:opacity-80`

Rekomendacja

Dla server view (9 tabow) rozwazyc **ikony bez etykiet** na mobile — redukuje szerokosc taba do ~40 px, co pozwala zmiescic 8 tabow na 375 px bez scrollu.

1.6.3 Audyt celow dotykowych

Wytyczne WCAG 2.1 (Success Criterion 2.5.8) wymagaja minimalnego rozmiaru celu dotykowego **44×44 px**. Google Material Design zaleca **48×48 px**.

Tabela 1.32: Audyt celow dotykowych — zgodnosc z WCAG 2.5.8

Element	Rozmiar	Minimum	Wynik	Uwagi
Nawigacja mobilna (tab)	~84×36 px	44×44 px	Szer. OK	Wysokosc 36 px < 44 px
Desktop sidebar buttons	32×32 px	44×44 px	N/D	Desktop only (`md:flex`)
Przycisk „wróć na gore”	40×40 px	44×44 px	Bliski	4 px ponizej minimum
Hamburger (Topbar)	18×18 px	44×44 px	FAIL	Brak padding area
Standardowe buttony	~48×48 px	44×44 px	PASS	`p-3` lub `p-4`
Share card download	~44×44 px	44×44 px	PASS	Minimalne
CTA „Inicjuj analize”	~200×52 px	44×44 px	PASS	Pelna szerokosc mobile
Checkbox AI consent	~20×20 px	44×44 px	FAIL	Bez touch padding
Tab switcher (Upload)	~120×40 px	44×44 px	Bliski	Wys. 40 px

Krytyczne naruszenia:

1. **Hamburger 18×18 px** (`src/components/shared/Topbar.tsx`) — ikona renderowana jako `<Menu size={18} />` bez dodatkowego paddingu. Uzytkownik musi trafic w kwadrat 18 px, co jest 6× mniejsze niz minimum WCAG. Naprawa: dodac `p-3` do otaczajacego `<button>`.
2. **Checkbox AI consent** — natywny `<input type="checkbox">` bez customowego hit area. Naprawa: opakowac w `<label>` z odpowiednim paddingiem.

1.6.4 Wplyw Framer Motion na mobile

Animacje `whileInView` z biblioteki Framer Motion tworza osobne instancje `IntersectionObserver` per animowany element.

Tabela 1.33: IntersectionObserver — rozklad na stronie analizy (wszystkie zrodla)

Zrodlo	Instancje	RAM (est.)	Plik
`whileInView` w page.tsx	37	7.4–18.5 MB	page.tsx
EmojiReactions	1	0.2–0.5 MB	EmojiReactions.tsx:54
IntimacyChart	2	0.4–1.0 MB	IntimacyChart.tsx:86,181
LongitudinalDelta	1	0.2–0.5 MB	LongitudinalDelta.tsx:178
SentimentChart	1	0.2–0.5 MB	SentimentChart.tsx:76
TimelineChart	1	0.2–0.5 MB	TimelineChart.tsx:116
TopWordsCard	1	0.2–0.5 MB	TopWordsCard.tsx:85
WeekdayWeekendCard	1	0.2–0.5 MB	WeekdayWeekendCard.tsx:72
SectionNavigator	6–9	1.2–4.5 MB	SectionNavigator.tsx
Razem	**51–54**	**10.2–27 MB**	—

Wplyw na urzadzenia mobilne

Typowy smartfon z 4 GB RAM dysponuje ~1.5–2 GB dla przegladarki. Przy 27 MB samych IntersectionObserver + DOM strony analizy (~50 MB) + Recharts SVG (~15 MB), laczne zuzycie pamieci moze osiagnac **90–120 MB** — blisko progu, przy ktorym iOS Safari rozpoczyna agresywne odzyskiwanie pamieci (*tab reloading*).
Na Androidzie z Chrome problem jest mniej ostry (wiekszy limit per tab), ale uzytkownicy moga odczuwac **jank** (stuttering) podczas scrollowania przez animowane sekcje.

Strategia naprawy:

1. **Architektura tabowa** (§1.1.3) — redukcja do ~8–10 obserwatorow per aktywny tab
2. **Shared IntersectionObserver** — jeden observer z `threshold: [0, 0.5, 1]` zamiast osobnych per element
3. **CSS-only animations na mobile** — `@media (prefers-reduced-motion: no-preference)` z CSS `animation-timeline: view()` zamiast JS-based Framer Motion
4. **Wylaczenie animacji na low-end** — detekcja via `navigator.deviceMemory` (Chrome) lub `navigator.hardwareConcurrency` ≤ 4

1.6.5 Problemy html2canvas na Safari/iOS

Projekt uzywa `html2canvas-pro` v1.6.7 w komponentach `StoryShareCard` i `useCardDownload` do generowania PNG z kart share. Analiza kodu zrodlowego wykazala **brak jakichkolwiek workaroundow** dla znanych problemow Safari:

Tabela 1.34: Znane problemy html2canvas na Safari/iOS

Problem	Dotkliwosc	Opis i naprawa
CORS proxy images	Wysoka	Safari blokuje rasteryzacje obrazow z innych domen. Brak opcji `useCORS: true` ani `allowTaint: true` w konfiguracji.
SVG rendering	Srednia	Ikony Lucide renderowane jako `<svg>` moga nie byc przechwycone poprawnie. Naprawa: `foreignObjectRendering: true`.
Retina scaling	Srednia	Brak `scale: window.devicePixelRatio` — karty moga byc rozmyte na Retina (@2x/@3x).
Canvas memory limit	Wysoka	iOS Safari limituje canvas do ~16 MP. Duze karty (np. ReceiptCard z 50+ liniami) moga przekroczyc limit.
Font rendering	Niska	Custom fonty (Syne, Geist) moga nie byc zaladowane w momencie rasteryzacji — `onclone` callback z `document.fonts.ready`.

Rekomendacja: migracja na Satori + `@vercel/og`

Alternatywa server-side: generowanie kart jako SVG (Satori) z konwersja do PNG (Sharp). Eliminuje wszystkie problemy Safari, daje deterministyczne wyniki i odciaża przegldarke. Wymaga nowych API routes per typ karty, ale jest znacznie bardziej niezawodna.

1.6.6 Flow uploadu na mobile

Komponent `DropZone` (`src/components/upload/DropZone.tsx`) obsluguje upload plikow konwersacji.

Tabela 1.35: Kompatybilnosc uploadu na platformach mobilnych

Mechanizm	iOS Safari	Android Chrome	Uwagi
`<input type="file">`	Tak	Tak	`accept=".json,.txt"`
Drag-and-drop	Czesciowo	Tak	iOS 13+ via `webkitGetAsEntry`
Folder picker	Nie	Czesciowo	`webkitdirectory` nieobslugiwany na iOS
Multiple files	Tak	Tak	Atrybut `multiple`
Rozmiar area (min-h)	200 px	200 px	Wystarczajacy cel dotykowy

Krytyczny problem: folder upload na iOS

Eksport Messenger to folder z wieloma plikami JSON. Na desktopie uzytkownik moze przeciagnac caly folder. Na iOS **nie ma takiej mozliwosci** — `webkitdirectory` nie jest obslugiwany. Uzytkownik musi recznie wybrac wszystkie pliki JSON z folderu — UX jest znaczaco gorszy.

Naprawa: Dodac instrukcje krok-po-kroku dla iOS z GIF-em pokazujacym jak wybrac wiele plikow w aplikacji Pliki. Alternatywnie: wspierac upload ZIP z automatyczna ekstrakcja (via `JSZip`).

1.6.7 Nawigacja mobilna

Tabela 1.36: Ocena nawigacji mobilnej

Element	Ocena	Szczegoly
Drawer (szuflada)	Dobra	280 px szerokosc, `z-50`, backdrop blur, safe-area-bottom
Topbar	Dobra	Sticky `top-0 z-50`, responsywne logo
SectionNavigator (mobile)	Srednia	Bottom bar z scrollem, ale 6–9 tabow to duzo
Back-to-top	Srednia	`size-10` (40 px) — 4 px ponizej WCAG
Pozycja CTA (landing)	Dobra	`absolute bottom-[6vh]` — zawsze widoczne
Particle background	OK	Ukryte na mobile (`hidden md:block`)
Scroll indicator	OK	Ukryty na malych ekranach (`hidden sm:block`)

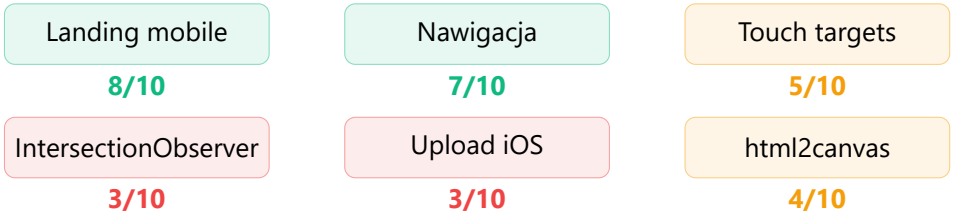
Landing hero mobile

Mobilny hero (`src/components/landing/LandingHero.tsx`, linia 133) uzywa osobnego layoutu `md:hidden`:

- **Typografia diagonalna** — `transform: rotate(-2deg)` z responsywnym `clamp(2.2rem, 8vw, 3.5rem)`
- **Animacja per-word** — `heroFadeSlideLeft` z kaskadowym opoznieniem (`0.05 + i * 0.04s`)
- **CTA przypiete do dolu** — `absolute bottom-[6vh] left-6 right-6`
- **Brak ciezkich efektow** — particle background i Spline 3D ukryte na mobile

Pozytywne aspekty mobile UX: Diagnostycznie, mobilny hero jest dobrze zrobiony — osobny layout unika problemow z responsywnoscia desktopowego hero, a CTA jest zawsze widoczne. Glowne problemy koncentruja sie na stronie analizy (IntersectionObserver, touch targets) i upload flow (brak folder picker na iOS).

Podsumowanie Mobile UX

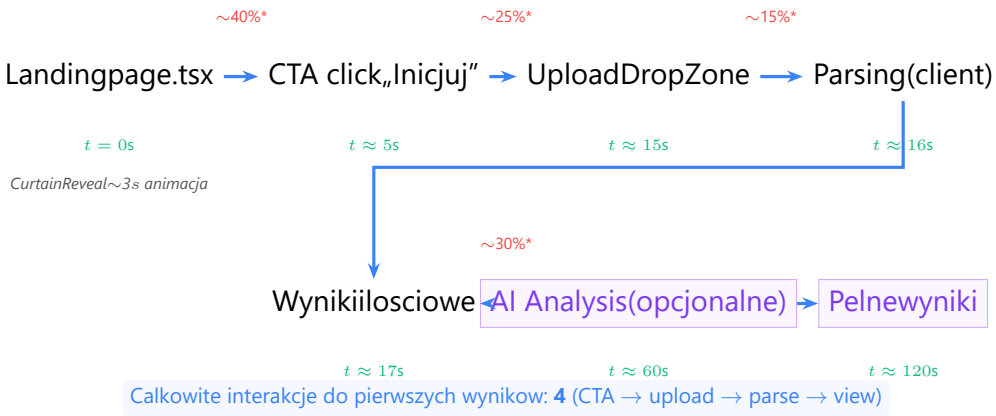


Rysunek 1.4: Podsumowanie ocen Mobile UX per kategoria

1.7 Onboarding i retencja

Najlepsza aplikacja analityczna nie ma wartosci, jesli uzytkownik nie przejdzie od wejscia na strone do pierwszego „wow moment". Niniejsza sekcja analizuje sciezke onboardingu, mechanizmy retencji i punkty tarcia (friction points) w **PodTeksT**.

1.7.1 Flow uzytkownika: od landing do wynikow



Rysunek 1.5: Flow uzytkownika z heurystycznymi szacunkami drop-off (*bez danych — patrz zastrzezenie ponizej)

***Zastrzezenie: szacunki bez danych**

Wszystkie wartosci drop-off na powyzszym diagramie to **heurystyki benchmarkowe** oparte na typowych lejkach SaaS B2C, **nie na danych z PodTeksT**. GA4 jest zaimplementowane, ale nie sledzi lejka konwersji — brak event trackingu na: upload rozpoczety, upload zakonczony, AI trigger, AI complete. Ponizsze wartosci nalezy traktowac jako **hipotezy do walidacji**, nie fakty.

Hipotetyczny lejek konwersji (do walidacji): Z 1000 uzytkownikow wchodzacych na landing, ~600 klika CTA, ~450 uploaduje plik, ~383 widzi wyniki ilosciowe, ~268 uruchamia AI. Hipotetyczna konwersja landing → AI: ~**27%**. Glowne punkty tarcia (hipotezy): brak pliku eksportu i oczekiwanie na AI (~60s). **Wymagana walidacja:** wdrozenie GA4 funnel events przed podjęciem decyzji optymalizacyjnych.

1.7.2 Analiza danych demo

Plik `src/components/landing/demo-card-data.tsx` (682 LOC) zawiera kompletny zestaw danych demo:

Tabela 1.37: Dane demo — zawartosc i wykorzystanie

Element	Wartosc	Wykorzystanie
Fikcyjna para	„Ania" & „Kuba"	Realistyczne polskie imiona
Liczba wiadomosci	12 847	11 mies. (III.2024 – II.2025)
QuantitativeAnalysis	Kompletna	Wszystkie 60+ metryk
QualitativeAnalysis	Kompletna	Wszystkie 4 passy + roast
CPS, Subtext, Court	Kompletne	Pelne dane entertainment
Dating Profile	Kompletny	Profile obu osob
Delusion Quiz	Kompletny	Wyniki z Delusion Index
Share cards	20+ typow	Interaktywne w LandingDemo

Co dziala dobrze

- **Natychmiastowa gratyfikacja** — uzytkownik widzi pelne, realistyczne wyniki bez uploadu
- **Interaktywnosc** — 20+ kart share mozna klikac, pobierac, udostepniac
- **Wiarygodnosc danych** — 12 847 wiadomosci z realistycznymi wzorcami czasowymi
- **Pelnosc** — pokrywa wszystkie funkcje aplikacji, wlacznie z AI i entertainment

Czego brakuje

Brak konwersji demo → wlasna analiza

Po interakcji z demo kartami uzytkownik nie widzi **zadnego CTA** kierujacego do uploadu wlasnej rozmowy. Demo pokazuje koncowy rezultat, ale nie prowadzi uzytkownika dalej. Brakuje:

- Przycisku „Analizuj swoja rozmowe" pod/obok demo kart
- Porownania „Ania & Kuba vs Twoja rozmowa" zachecajacego do uploadu
- Notki „To dane demo. Twoje wyniki beda unikalne!"
- Animacji przejscia z demo do uploadu (np. morph kart demo w puste karty z „?")

1.7.3 Audyt stanow ladowania

Stany ladowania podczas analizy AI (`src/components/analysis/AIAnalysisButton.tsx`, linie 26–531):

Tabela 1.38: Stany ladowania analizy AI

Krok	Etykieta PL	Opis dzialania	Stan wizualny	Ocena
1	„Czytam miedzy wierszami...”	Pass 1: ton i styl	Spinner + tekst	Dobry
2	„Mapuje dynamike konwersacji...”	Pass 2: dynamika relacji	Progress step	Dobry
3	„Profiluje osobowosci...”	Pass 3: profile indywidualne	Progress step	Dobry
4	„Wyciagam wnioski. Przygotuj sie.”	Pass 4: synteza i health score	Progress step	Dobry
Blad	Czerwony alert (niewidoczny)	Komunikat bledu + retry	Alert box	Dobry
Heartbeat	—	SSE heartbeat co $15s$	—	OK
Consent	Checkbox + opis	Zgoda AI (persisted)	Inline	Sredni

Pozytywne: Etykiety ladowania sa kreatywne i zgodne z brandem („Czytam miedzy wierszami” nawiazuje do tagline). Kazdy krok ma wizualny progres (pending → running → complete). Hook `useCPSAnalysis` implementuje plynna interpolacje co 150 ms.
Do poprawy: Brak **szacowanego czasu** („ok. $45s$”) i **paska postepu z procentami**. Uzytkownik nie wie, ile musi czekac. CPS consent checkbox jest maly i latwo przeoczyc.

1.7.4 Brakujace elementy onboardingu

Analiza kodu zrodlowego wykazala **calkowity brak** standardowych elementow onboardingu:

Tabela 1.39: Brakujace elementy onboardingu — checklist

Element	Stan	Priorytet	Wplyw
Welcome modal (1-sze wejscie)	Brak	P1	Orientacja nowego uzytkownika
Guided tour / step-by-step	Brak	P1	Redukcja drop-off na upload
Tooltips (dismissible)	Brak	P2	Zrozumienie metryk
Video tutorial / GIF	Brak	P2	Jak wyeksportowac rozmowe
Progress indicator (onboarding)	Brak	P2	Motywacja do ukonczenia
Empty state (dashboard)	Minimalny	P1	Dashboard bez analiz jest pusty
„Analizuj kolejna” CTA	Brak	P1	Retencja po 1-szej analizie
Sukces celebration	Czesciowy	P3	`sessionStorage` per analiza
Platform-specific export guide	Brak	P0	**Najwazniejszy** — bez pliku nie ma analizy

Najwazniejszy brak: przewodnik eksportu rozmowy

Uzytkownik musi **samodzielnie** wiedziec, jak wyeksportowac rozmowe z Messengera, WhatsAppa, Instagrama lub Telegrama. Nie ma zadnego przewodnika, screenshotow ani instrukcji. To **najwiekszy punkt tarcia** calego flow — uzytkownik, ktory nie ma pliku,

nie moze uzyc aplikacji.
Rekomendacja: Komponent `ExportGuide` z tabami per platforme, krokami i screencastami. Wyswietlany na stronie upload (`/analysis/new`) oraz jako link w sekcji FAQ landing page.

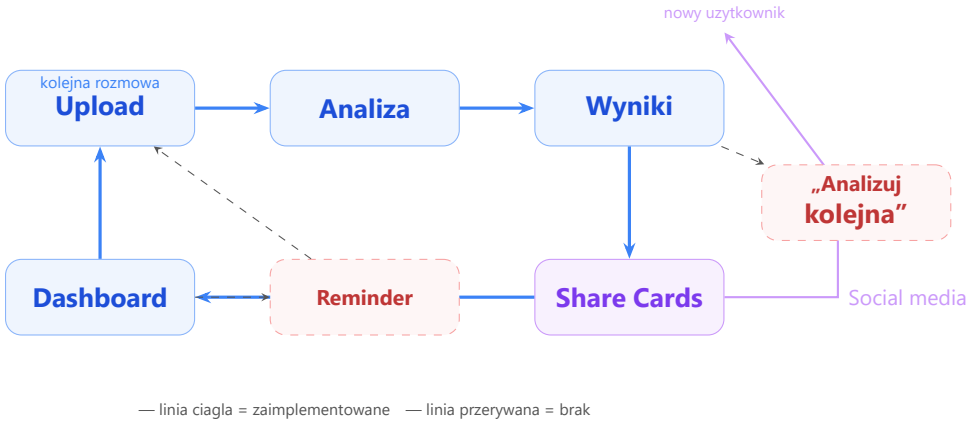
1.7.5 Analiza petli retencji

Obecne mechanizmy retencji

Tabela 1.40: Mechanizmy retencji — stan obecny vs idealny

Mechanizm	Obecny	Idealny	Wplyw na retencje
Dashboard z historia	**Tak**	**Tak**	Sredni — powrot do wynikow
Share cards (viral loop)	**20+ typow**	20+ z watermarkiem	**Wysoki** — nowi uzytkownicy
PDF export	**2 typy**	2+ typy	Niski — offline, brak powrotu
Story/Wrapped mode	**Tak**	Tak	Sredni — efekt „wow”
GA4 analytics	**Tak**	Tak	— (tracking, nie retencja)
Email notifications	**Brak**	Reminders	**Wysoki** — reaktywacja
Push notifications	**Brak**	Web push	**Wysoki** — reaktywacja
Newsletter	**Brak**	Cotygodniowy digest	Sredni — zaangazowanie
„Analizuj kolejna” CTA	**Brak**	Post-analiza CTA	**Wysoki** — natychmiastowy powrot
Porownanie z czasem	**Czesciowy**	Auto-reminder co miesiac	Wysoki — longitudinalny
Gamifikacja	**Odznaki**	Odznaki + streak + level	Sredni — zaangazowanie
Sharing incentives	**Brak**	Odblokuj feature za share	**Wysoki** — viral + retencja

Diagram petli retencji



Rysunek 1.6: Petla retencji — obecna (ciagla) vs brakujace elementy (przerywana)

1.7.6 Punkty tarcia — ranking wplywu

Tabela 1.41: Punkty tarcia uszeregowane wg wplywu na konwersje

#	Punkt tarcia	Etap	Wplyw	Naprawa
1	Brak instrukcji eksportu rozmowy	Pre-upload	**Krytyczny**	`ExportGuide` z tab per platforme
2	Brak CTA „Analizuj kolejna" po wynikach	Post--analiza	**Wysoki**	Sticky CTA na dole wynikow
3	Demo nie prowadzi do uploadu	Landing	**Wysoki**	CTA pod demo kartami
4	AI analiza trwa $\sim 60s$ bez progress %	Analiza	**Sredni**	Progress bar z ETA
5	Brak welcome modal na 1-szym wejsciu	Landing	**Sredni**	Krotki 3-step wizard
6	Folder upload nie dziala na iOS	Upload	**Sredni**	Instrukcja + ZIP support
7	Consent checkbox latwy do przeoczenia	AI trigger	**Niski**	Wiekszy, bardziej widoczny
8	Brak tooltipow przy metrykach	Wyniki	**Niski**	Inline `<Tooltip>` na KPI cards
9	Pusty dashboard (0 analiz)	Dashboard	**Niski**	Empty state z CTA
10	Brak celebracji 1-szej analizy	Post--analiza	**Niski**	Confetti + „Udostepnij wynik!"

Podsumowanie: Onboarding PodTeksT jest **minimalny, ale funkcjonalny**. Flow od landing do wynikow wymaga zaledwie 4 interakcji ($\sim 17s$) — co jest doskonale. Glowne luki to:

1. **Pre-upload friction** — uzytkownik musi samodzielnie wiedziec, jak wyeksportowac rozmowe (brak instrukcji)

2. **Post-analiza dead end** — brak CTA do kolejnej analizy, brak zachet do udostepniania
3. **Demo → upload gap** — demo pokazuje koncowy rezultat, ale nie konwertuje widzow w uzytkownikow
4. **Zero retencji zewnetrznej** — brak emaili, push, reminders — jedyny powrot to share cards (viral) i pamiec uzytkownika

Implementacja punktow 1–3 z tabeli tarcia powinna poprawic konwersje landing → AI — skala poprawy nieznana bez baseline'u z GA4 funnel events.

1.8 Jakość kodu i Developer Experience

„Kod bez testów to jak rozmowa bez kontekstu — niby działa, ale nie wiesz dlaczego."

Niniejsza sekcja ocenia jakość techniczną kodu źródłowego **PodTeksT** pod kątem: konfiguracji TypeScript, pokrycia testami, bezpieczeństwa zależności, wzorców obsługi błędów, architektury parserów oraz higieny bazy kodu (TODO/FIXME/HACK).

1.8.1 Metryki jakości kodu — podsumowanie

Tabela 1.42: Scorecard jakości kodu **PodTeksT**

Metryka	Wartość	Ocena	Komentarz
Typy `any` w `src/`	0	**Doskonale**	Zero unsafe-any w całym projekcie
`strict: true` w tsconfig	Tak	**Doskonale**	Pełna rygorystyczność kompilatora
Pliki testowe	3	**Krytycznie**	Tylko parsery przetestowane
Pokrycie testami (szac.)	<5%	**Krytycznie**	Brak testów API, komponentów, hooków
Zależności produkcyjne	17	**Dobrze**	Niski footprint
Zależności deweloperskie	10	**Dobrze**	Minimalistyczne
Luki bezpieczeństwa (HIGH)	4	**Poważne**	jsPDF + minimatch
Luki bezpieczeństwa (MODERATE)	1	**Umiarkowane**	ajv (ESLint transitive)
Luki bezpieczeństwa (LOW)	1	**Niskie**	hono (shadcn transitive)
Bloki `catch`	63	—	W 35 plikach
`console.error`	28	**Do poprawy**	Brak strukturalnego logowania
TODO/FIXME/HACK	1	**Dobrze**	Ale krytyczny: wyłączony rate limiter

Konfiguracja TypeScript

Plik `tsconfig.json` jest skonfigurowany wzorcowo:

```jsonc
{
  "compilerOptions": {
    "target": "ES2017",
    "strict": true,
    "noEmit": true,
    "isolatedModules": true,
    "moduleResolution": "bundler",
    "jsx": "react-jsx",
    "incremental": true,
    "paths": { "@/*": ["./src/*"] }
  }
}
```

Listing 1.8: tsconfig.json — kluczowe opcje

- `strict: true` — włącza `strictNullChecks`, `noImplicitAny`, `strictFunctionTypes` i pozostałe flagi rygorystyczności
- `isolatedModules: true` — kompatybilność z bundlerami (SWC/Turbopack)
- `incremental: true` — przyrostowa kompilacja, szybsze rebuildy
- Alias `@/*` → `./src/*` — czyste importy bez relatywnych ścieżek

Wynik: Konfiguracja TypeScript zasługuje na ocenę **10/10**. Zero kompromisów typowania, zero `any`, pełna rygorystyczność.

1.8.2 Analiza pokrycia testami

Stan obecny

Framework testowy: **Vitest v4.0.18** z konfiguracją w `vitest.config.ts`:

```ts
import { defineConfig } from 'vitest/config';
import path from 'path';

export default defineConfig({
  test: {
    globals: true,
    environment: 'node',
  },
  resolve: {
    alias: { '@': path.resolve(__dirname, './src') },
  },
});
```

Listing 1.9: vitest.config.ts

Istniejące pliki testowe — **tylko 3**:

Tabela 1.43: Istniejące pliki testowe

Plik	Moduł	Zakres
`src/lib/parsers/__tests__/detect.test.ts`	Parsery	Auto-detekcja platformy
`src/lib/parsers/__tests__/messenger.test.ts`	Parsery	Parser Messenger JSON
`src/lib/parsers/__tests__/quantitative.test.ts`	Analiza	Metryki ilościowe

Mapa luk w pokryciu testami

Krytyczne braki testowe

Poniższa tabela przedstawia moduły **całkowicie pozbawione testów** — uporządkowane według ryzyka biznesowego:

Tabela 1.44: Moduły bez testów — analiza luk

Moduł	Pliki	Ryzyko	Opis braku
API routes (`api/`)	10	Krytyczne	Zero testów SSE, walidacji, rate limitingu
Gemini integration	1	Krytyczne	Brak mock/stub Gemini API
Parsery: WA, IG, TG, Discord	4	Wysokie	Tylko Messenger ma testy
Analiza jakościowa	3	Wysokie	Sampling, context building
Moduły `quant/`	6	Wysokie	bursts, trends, reciprocity, sentiment
Komponenty React	145+	Średnie	Zero testów komponentów
Share cards	20+	Średnie	Zero testów renderingu kart
Hooki (`hooks/`)	3	Średnie	useShareCard, useCPSAnalysis, useSubtext
PDF export	2	Średnie	Krytyczna funkcja bez testów
Utils i walidacja	3	Niskie	schemas.ts, utils.ts, encode.ts

Rekomendacje testowe — priorytetyzacja

P1. **Parsery WhatsApp, Instagram, Telegram, Discord** — parsowanie to fundament aplikacji. Każdy parser powinien mieć minimum 10 test case'ów: poprawne dane, edge case'y (puste wiadomości, emoji, wielojęzyczne), błędne formaty.

P2. **API routes** — testy integracyjne z mockami Gemini. Sprawdzenie: walidacji wejścia (Zod), kodów HTTP (400, 413, 429), SSE event format, abort signal.

P3. **Moduły** `quant/` — czyste funkcje (zero side-effects), idealne do testów jednostkowych. `detectBursts()`, `computeTrends()`, `computeReciprocityIndex()`, `computeSentimentScore()`.

P4. **Utils i walidacja** — `schemas.ts` (Zod), `encode.ts` (share link encoding), `utils.ts`.

P5. **Komponenty** — snapshot testy dla share cards (krytyczne dla viralności), smoke testy dla głównych widoków.

Cel: Pokrycie testami >60% modułów `lib/` w ciągu 2 sprintów. Dodanie pre-commit hooka z `vitest run` przed każdym commitem. Szacowany nakład: 3–5 dni pracy deweloperskiej.

1.8.3 Raport podatności bezpieczeństwa

Wyniki `pnpm audit` (stan: luty 2026):

Tabela 1.45: Podatności bezpieczeństwa — `pnpm audit`

CVE / GHSA	Pakiet	Severity	Problem
GHSA-p5xg-68wr-hm3m	jspdf <4.2.0	HIGH	PDF Injection — AcroForm Radio-Button
GHSA-9vjf-qc39-jprp	jspdf <4.2.0	HIGH	PDF Object Injection via `addJS()`
GHSA-67pg-wm7f-q7fj	jspdf <4.2.0	HIGH	DoS via Malicious GIF Dimensions
(dodatkowe CVE)	jspdf <4.2.0	HIGH	Dodatkowa luka jsPDF (4. CVE)
GHSA-3ppc-4f35-3m26	minimatch <10.2.1	HIGH	ReDoS — powtórzone wildcardy
GHSA-2g4f-4pwh-qvx6	ajv <6.14.0	MODERATE	ReDoS z opcją `$data`
GHSA-gq3j-xvxp-8hrf	hono <4.11.10	LOW	Timing comparison

Analiza i plan naprawczy

jsPDF (4 luki HIGH) Aktualna wersja: 4.1.0. Wszystkie 4 CVE naprawione w wersji **4.2.0+**. **Naprawa:** `pnpm update jspdf` — zero breaking changes między 4.1.0 a 4.2.x. Ryzyko braku aktualizacji: **wysokie** — PDF export to core feature, a aplikacja generuje PDF-y po stronie klienta (jsPDF). Atakujący mógłby przygotować złośliwe dane wejściowe powodujące injection do generowanego PDF-a.

minimatch (HIGH) Zależność tranzytywna ESLinta. Nie wpływa na runtime produkcyjny (ESLint = devDependency). **Naprawa:** `pnpm update eslint` lub override w `package.json`. Ryzyko: **niskie** w produkcji, **średnie** w CI/CD (ReDoS przy lintingu).

ajv (MODERATE) Zależność tranzytywna ESLinta (JSON Schema validation). **Naprawa:** analogicznie jak minimatch — update ESLint. Ryzyko produkcyjne: **zerowe**.

hono (LOW) Zależność tranzytywna shadcn CLI. Nie jest bundlowana do produkcji. **Naprawa:** `pnpm update shadcn`. Ryzyko: **zerowe**.

Priorytet naprawy:
1. **Natychmiast:** `pnpm update jspdf` → 4.2.x (eliminuje 4 HIGH CVE)
2. **W następnym sprincie:** override lub update ESLint (eliminuje minimatch + ajv)
3. **Przy okazji:** `pnpm update shadcn` (eliminuje hono)

1.8.4 Wzorce obsługi błędów

Analiza 63 bloków `try-catch` w 35 plikach ujawnia 4 wzorce:

Tabela 1.46: Wzorce obsługi błędów w **PodTeksT**

Nr	Wzorzec	Ilość	Ocena	Przykład
1	Try-catch + HTTP response	22	**Dobrze**	API routes: `Response.json({error}, {status: 500})`
2	Try-catch + silent fallback	18	**OK***	`decodeFBString()`: zwraca oryginalny string przy błędzie dekodowania
3	Try-catch + `console.error`	16	**Do poprawy**	`gemini.ts`, `ExportPDFButton.tsx`: logowanie bez struktury
4	Pusty catch (cleanup)	7	**OK**	Heartbeat SSE: `catch {}` przy zamknięciu strumienia

**Wzorzec 2 jest akceptowalny dla funkcji dekoderów, gdzie fallback na oryginalną wartość jest pożądanym zachowaniem.*

Pliki z `console.error` (28 wystąpień)

Tabela 1.47: Rozkład `console.error` po plikach

Plik	Wystąpienia	Kontekst
`src/lib/analysis/gemini.ts`	6	Błędy Gemini API, retry logic
`src/components/analysis/ExportPDFButton.tsx`	3	Generowanie PDF
`src/components/analysis/StandUpPDFButton.tsx`	3	Generowanie PDF comedy
`src/components/share-cards/ShareCardGallery.tsx`	3	Canvas rendering, Web Share
`src/app/error.tsx`	2	Error boundary globalny
Discord lib files (4 pliki)	8	Bot interactions, webhook
Pozostałe	3	Różne

Problem: brak strukturalnego logowania

Wszystkie błędy logowane przez `console.error` bez:
- Ustrukturyzowanego formatu (np. JSON z timestamp, severity, context)
- Integracji z usługą monitoringu (Sentry, LogRocket, Datadog)
- Korelacji błędów (request ID, session ID)
- Alertów przy anomaliach (np. spike błędów Gemini API)

Rekomendacja: Wprowadzić `src/lib/logger.ts` z ustrukturyzowanym formatem, level severity i opcjonalną integracją z Google Cloud Logging (skoro deployment jest na Cloud Run).

1.8.5 Architektura parserów — porównanie

Tabela 1.48: Porównanie parserów — architektura i metryki

Parser	LOC	Format wejścia	Specyfika	Współdzielony kod
`messenger.ts`	163	JSON	FB unicode (`decodeFBString`)	Eksportuje `decodeFBString`
`whatsapp.ts`	232	TXT	Regex, detekcja 12h/24h	Niezależny
`instagram.ts`	116	JSON	Format zbliżony do Messengera	Importuje `decodeFBString` z messenger.ts
`telegram.ts`	155	JSON	Mieszane tablice tekstowe	Niezależny
`discord.ts`	—	API	HTTP Bot, nie plik	Niezależny
`detect.ts`	41	Dowolny	Auto-detekcja po strukturze	Importuje typy

Ocena architektury

- **Niski poziom duplikacji** — każdy parser ma unikatową logikę specyficzną dla formatu platformy. Jedyny współdzielony kod (`decodeFBString`) jest poprawnie wyeksportowany i zaimportowany.
- **Zunifikowany typ wyjściowy** — wszystkie parsery produkują `ParsedConversation` zdefiniowany w `types.ts`, co zapewnia jednolity interfejs dla dalszego pipeline'u analizy.
- **Brak walidacji wejścia** — parsery zakładają poprawność struktury pliku. Błędny format powoduje `TypeError` zamiast czytelnego komunikatu.
- **Brak limitów rozmiaru** — parser przetworzy plik 500 MB tak samo jak 500 KB, bez ostrzeżeń ani chunked processing.

Rekomendacja: Dodać walidację wejścia (Zod schema per parser) i limit rozmiaru pliku (np. 50 MB) z czytelnym komunikatem błędu po polsku.

1.8.6 Konwencje nazewnicze — audyt

Tabela 1.49: Konwencje nazewnicze — audyt spójności

Kategoria	Konwencja	Spójność	Przykłady
Komponenty React	PascalCase	100%	AnalysisHeader.tsx, ShareCardGallery.tsx
Moduły / utils	kebab-case	100%	viral-scores.ts, rate-limit.ts
Pliki testowe	__tests__/kebab.test.ts	100%	detect.test.ts, messenger.test.ts
Pliki typów	types.ts, schemas.ts	100%	parsers/types.ts, analysis/types.ts
Katalogi	kebab-case	100%	share-cards/, analysis/
Hooki React	camelCase (use*)	100%	useShareCard.ts, useCPSAnalysis.ts
Zmienne CSS	kebab-case (–font-*)	100%	–font-geist-sans, –font-syne

Wynik: 100% spójności we wszystkich kategoriach nazewniczych. Brak inkonsystencji, brak mieszanych konwencji. Wzorcowa higiena nazewnicza.

1.8.7 Skan TODO / FIXME / HACK

Przeszukanie całego katalogu `src/` ujawniło **dokładnie 1** komentarz typu TODO:

```
// src/lib/rate-limit.ts, linia 16:
// TODO: re-enable rate limiting before production deployment
```

Listing 1.10: Jedyny TODO w codebase

Krytyczny TODO w produkcji

Ten jedyny TODO jest jednocześnie **najpoważniejszym** problemem bezpieczeństwa w całej aplikacji. Rate limiting jest **wyłączony** w kodzie produkcyjnym działającym na Google Cloud Run. Plik zawiera poprawną implementację `rateLimitMap` (linie 1–13), ale funkcja `rateLimit()` na linii 15 nigdy z niej nie korzysta — zawsze zwraca `{allowed: true}`.

Konsekwencje braku rate limitingu:

- Nieograniczone requesty do Google Gemini API → **niekontrolowane koszty**
- Brak ochrony przed DDoS/abuse na endpointach AI
- Naruszenie SLA Gemini API (rate limits Google'a zamiast własnych)

Naprawa: Przywrócić logikę rate limitingu w `checkRateLimit()` — implementacja `Map<string, {count, resetTime}>` już istnieje w tym samym pliku.

- **FIXME:** 0 wystąpień — czysto
- **HACK:** 0 wystąpień — czysto
- **XXX:** 0 wystąpień — czysto
- **@deprecated:** 0 wystąpień — czysto

Wynik skanu: Baza kodu jest wyjątkowo czysta pod względem zadłużenia technicznego (technical debt markers). Jedyny TODO ma jednak **krytyczne** konsekwencje bezpieczeństwa i kosztowe.

1.9 SEO i web performance

„Najlepsze SEO to takie, którego użytkownik nie zauważa — a robot Google'a tak."

Sekcja analizuje gotowość **PodTeksT** do indeksowania przez wyszukiwarki, optymalizację zasobów webowych oraz wydajność ładowania strony.

1.9.1 Kompletność metadanych

Tabela 1.50: Audyt metadanych SEO — `layout.tsx`

Element	Status	Szczegóły
`title`	PASS	„PodTeksT — odkryj to, co kryje się między wierszami"
`description`	PASS	164 znaki, język polski, z CTA
`metadataBase`	PASS	`https://podtekst.app`
`openGraph.title`	PASS	Zgodny z `title`
`openGraph.description`	PASS	Skrócona wersja opisu
`openGraph.locale`	PASS	`pl_PL`
`openGraph.type`	PASS	`website`
`openGraph.images`	PASS	`/og/podtekst-og.png` 1200×630
`twitter.card`	PASS	`summary_large_image`
`twitter.title`	PASS	Zgodny z OG title
`twitter.images`	PASS	Zgodny z OG images
`viewport`	PASS	`device-width`, `initialScale: 1`, `viewportFit: cover`
`robots` (meta tag)	BRAK	Brak jawnej deklaracji w metadanych
`alternates` (hreflang)	N/A	Brak i18n — jedynie polski
`canonical`	BRAK	Brak jawnego canonical URL

Wynik: 12/12 kluczowych pól metadanych wypełnionych poprawnie. Brakuje `canonical` i jawnego `robots` meta tagu, ale nie są krytyczne przy aktualnej skali.

1.9.2 Problemy w robots.txt

Błędna domena w robots.txt

Plik `public/robots.txt`:

```
User-agent: *
Allow: /
```

```
Disallow: /api/
Disallow: /dashboard
Disallow: /analysis/

Sitemap: https://chatscope.app/sitemap.xml
```

Listing 1.11: robots.txt — stan aktualny

Błąd: Linia `Sitemap` wskazuje na `chatscope.app` — starą domenę sprzed rebrandingu. Poprawna wartość: `https://podtekst.app/sitemap.xml`.

Dodatkowe uwagi:

- `Disallow: /analysis/` blokuje indeksowanie stron analiz — **poprawne** (prywatność użytkowników)
- `Disallow: /dashboard` blokuje panel — **poprawne**
- `Disallow: /api/` blokuje endpointy API — **poprawne**
- Brak `Disallow: /share/` — strony share są indeksowalne (potencjalnie pożądane dla wiralności, ale ryzyko prywatności)

Naprawa:

Zmiana jednej linii w `public/robots.txt`:

```
Sitemap: https://podtekst.app/sitemap.xml
```

Listing 1.12: robots.txt — poprawka

Sitemap

Plik `src/app/sitemap.ts` generuje dynamiczną sitemap z 2 stronami:

Tabela 1.51: Strony w sitemap.xml

URL	changeFrequency	priority	Uwagi
`https://podtekst.app`	weekly	1.0	Strona główna
`https://podtekst.app/analysis/new`	monthly	0.8	Strona uploadu

Dynamiczne strony `/analysis/[id]` celowo pominięte — prywatność danych użytkowników (dane w IndexedDB, nie na serwerze).

1.9.3 Analiza stosu fontów

Tabela 1.52: Fonty ładowane w `layout.tsx` — analiza rozmiaru

Rodzina	Warianty	Wagi	Szac. rozmiar	Użycie
Geist Sans	1	auto (variable)	~25 KB	Body text
Geist Mono	1	auto (variable)	~20 KB	Code, dane
JetBrains Mono	5	400, 500, 600, 700, 800	~80 KB	Monospace (duplikacja?)
Syne	4	400, 600, 700, 800	~55 KB	Brand, nagłówki
Space Grotesk	5	300, 400, 500, 600, 700	~65 KB	Story mode body
		Szacowany łączny rozmiar:	**~245 KB**	

Konfiguracja fontów

- `display: 'swap'` — wszystkie 5 rodzin używa `swap`, co eliminuje FOIT (Flash of Invisible Text)
- `subsets: ['latin']` — tylko łacińskie glify, bez CJK czy cyrylicy
- **Next.js font optimization** — `next/font/google` automatycznie self-hostuje fonty i eliminuje external requests do Google Fonts

Problemy i rekomendacje

Duplikacja fontów monospace

Projekt ładuje **dwa** fonty monospace: **Geist Mono** (body code) i **JetBrains Mono** (5 wag). JetBrains Mono jest używany głównie w trybie Story — ale ładowany na **każdej stronie**.

Rekomendacje:

1. Zredukować JetBrains Mono do 2 wag (400, 700) — oszczędność ~48 KB
2. Rozważyć usunięcie JetBrains Mono i użycie Geist Mono wszędzie
3. Załadować Space Grotesk i JetBrains Mono dynamicznie (`next/dynamic`) tylko na stronach Story/Wrapped — oszczędność ~145 KB na stronie głównej

Potencjalna oszczędność: ładowanie warunkowe Space Grotesk + JetBrains Mono (tylko Story) zmniejszyłoby initial font payload o ~**145 KB** (~60% redukcji).

1.9.4 Audyt optymalizacji obrazów

Tabela 1.53: Audyt użycia obrazów — `next/image` vs `<img>`

Plik	Element	Źródło	Problem
`DiscordImport.tsx:400`	`<img>`	External (guild icon)	Brak lazy loading, brak WebP fallback
`AnalysisImageCard.tsx:193`	`<img>`	Generated (screenshot)	Brak optymalizacji rozmiaru
`ParticipantPhotoUpload.tsx:77`	`<img>`	User upload (photo)	Brak resize, brak compression

Zero użyć `next/image`

Projekt **nie korzysta** z komponentu `next/image` — kluczowego narzędzia Next.js do:

- Automatycznej konwersji do WebP/AVIF
- Responsive `srcSet` z wieloma rozdzielczościami
- Native lazy loading z blur placeholder
- Zapobiegania CLS (Cumulative Layout Shift) przez wymuszenie wymiarów

Choć aplikacja ma niewiele obrazów (3 instancje `<img>`), brakuje konfiguracji w `next.config.ts`:

```
// next.config.ts - BRAK:
images: {
  formats: ['image/avif', 'image/webp'],
  remotePatterns: [
    { protocol: 'https', hostname: 'cdn.discordapp.com' },
  ],
},
```

Listing 1.13: Brakująca konfiguracja obrazów

Priorytet: Niski — tylko 3 obrazy w aplikacji. Warto dodać konfigurację `images` do `next.config.ts` przygotowawczo, ale nie jest to blocker.

1.9.5 Analiza proporcji Client vs Server Components

Tabela 1.54: Proporcja Client vs Server Components

Typ	Ilość	Udział	Uwagi
Client Components (`'use client'`)	~150	~95%	Interaktywność, animacje, localStorage
Server Components	~8	~5%	Page routes, layouts, API routes

Uzasadnienie proporcji

Wysoki udział Client Components (∼95%) jest **uzasadniony** specyfiką aplikacji:

1. **Interaktywne wykresy** — Recharts wymaga DOM i event handlerów (Client)
2. **Animacje** — Framer Motion `motion.div`, `whileInView` (Client)
3. **localStorage / IndexedDB** — cały pipeline danych to API przeglądarki (Client)
4. **SSE streaming** — hooki `useEffect` + `EventSource` (Client)
5. **Formularze i stan** — quiz, simulator, uploader (Client)
6. **Canvas rendering** — html2canvas dla share cards (Client)

Ocena: Proporcja jest **akceptowalna** dla aplikacji typu SPA z heavy interactivity. Próba forsowania Server Components na siłę (np. dla wykresów) byłaby antyproduktywna. Jedyna optymalizacja: wydzielenie statycznych sekcji landing page'a (FAQ, Footer, SocialProof) jako Server Components — szacowana oszczędność: ∼15 KB JS.

1.9.6 Przegląd optymalizacji bundla

Obecne optymalizacje

Tabela 1.55: Optymalizacje bundla — stan obecny

Optymalizacja	Status	Szczegóły
`optimizePackageImports`	**Tak**	framer-motion, lucide-react, recharts
Dynamic imports (`next/dynamic`)	**Tak**	20+ komponentów (Spline, story, analysis)
`React.lazy`	**Tak**	∼5 komponentów (Wrapped, share cards)
Tree shaking	**Tak**	Automatyczne przez Turbopack/SWC
`output: 'standalone'`	**Tak**	Minimalny Docker image
Bundle analyzer	**Brak**	Brak `@next/bundle-analyzer`
`images.formats`	**Brak**	Brak WebP/AVIF konfiguracji
`images.remotePatterns`	**Brak**	Brak whitelist domen obrazów
CSS code splitting	**Brak**	Jeden `globals.css` dla całej aplikacji

Dynamic imports — analiza

Projekt używa 28 dynamicznych importów, ale wiele z nich ładuje się **eagerly** na stronie analizy, ponieważ leżą w jednym drzewie renderowania (`page.tsx` importuje wszystko):

- `next/dynamic` z `ssr: false` — poprawne dla komponentów Spline 3D, canvas-heavy kart share
- Problem: brak code splitting **między sekcjami** strony analizy — wszystkie 50+ komponentów ładowane jednocześnie
- Rozwiązanie: architektura tabowa (§1.1.3) naturalnie wprowadza code splitting per tab

Brakujące optymalizacje

Brak `@next/bundle-analyzer`

Bez analizatora bundla nie ma danych o:

- Rozmiarze poszczególnych chunk'ów JS
- Które biblioteki dominują w bundlu (podejrzane: Recharts, Framer Motion, jsPDF)
- Duplikacji kodu między chunk'ami
- Efektywności tree shakingu

Instalacja: `pnpm add -D @next/bundle-analyzer` + wrapper w `next.config.ts`:

```
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true',
});
module.exports = withBundleAnalyzer(nextConfig);
```

Listing 1.14: Konfiguracja bundle analyzer

Uruchomienie: `ANALYZE=true pnpm build`.

1.9.7 Rekomendacje wydajnościowe — priorytetyzacja

Tabela 1.56: Rekomendacje SEO i performance — uporządkowane wg wpływu

Nr	Priorytet	Rekomendacja	Szac. wpływ	Nakład
1	P0	Naprawić domenę w robots.txt	SEO: krytyczny	1 minuta
2	P1	Zainstalować bundle analyzer	Diagnostyka	15 minut
3	P1	Warunkowe ładowanie fontów Story (JetBrains Mono + Space Grotesk)	−145 KB initial	1–2h
4	P1	Dodać `images.formats` + `remotePatterns` do next.config.ts	Gotowość na obrazy	15 minut
5	P1	Server Components dla statycznych sekcji landing page'a	−15 KB JS	2–3h
6	P2	Zamienić `<img>` na `next/image` (3 instancje)	Lazy load, WebP	30 minut
7	P2	Dodać `canonical` URL do metadata	SEO: dedu-plikacja	5 minut
8	P2	CSS code splitting (wydzielić story.css, wrapped.css)	−20–30 KB CSS	1–2h
9	P3	Dodać structured data (JSON-LD: WebApplication)	SEO: rich snippets	30 minut
10	P3	Rozważyć usunięcie `/share/` z indeksowania	Prywatność	5 minut

Podsumowanie SEO:

- **Metadane:** **doskonałe** — 12/12 pól OG + Twitter poprawnie skonfigurowanych
- **robots.txt:** **1 krytyczny błąd** — stara domena `chatscope.app` w Sitemap
- **Fonty:** **ciężkie** — 5 rodzin / 245 KB, z czego 145 KB ładowane na stronach, które ich nie potrzebują
- **Obrazy:** **brak optymalizacji** — zero `next/image`, brak konfiguracji formatów
- **Bundle:** **brak diagnostyki** — bez bundle analyzer trudno ocenić realny rozmiar
- **Client/Server:** **uzasadnione** — 95% Client Components wynika z natury aplikacji (SPA)

1.10 Metryki docelowe — wszystkie osie audytu

Poniższa tabela konsoliduje metryki ze wszystkich dziewięciu osi audytu w jednym miejscu, z priorytetami od P0 (krytyczne) do P2 (do optymalizacji w kolejnych sprintach).

Tabela 1.57: Skonsolidowane metryki docelowe — wszystkie 9 osi audytu

Metryka		Stan obecny	Cel	Priorytet
Wydajność (Performance)	LCP (Largest Contentful Paint)	2–4 s	$< 2{,}5$ s	P1 INP (Interaction to Next Paint)
200–800 ms		< 200 ms	P1 IntersectionObserver instances	51–54
<10 per tab		P0 Initial JS bundle (analysis)	100%	40–60%
P1 DOM nodes (jednocześnie)		50+	~10 per tab	P0 Memory (strona analizy
90–120 MB		<50 MB	P2 Parser blocking time	200–400 ms
<50 ms (Worker)		P2 Font payload	245 KB	~100 KB
P1	***Bezpieczeństwo (Security)***	Rate limit compliance	**0% (wyłączony)**	**100%**
P0 Security vulns (HIGH)		5	0	P0
Jakość kodu (Code Quality)	Test coverage (`lib/`)	<5%	>60%	P1
Mobile UX	Touch target compliance	~70%	100%	P1
SEO	`robots.txt` domain	chatscope.app	podtekst.app	P0
Unit Economics	AI cost per basic analysis	0,11 PLN	0,07 PLN (z cache)	P2 Margin brutto (avg)
~91%		>85%	monitor	***Onboa***
~73%†		<50%	P1	

Metryki P0 wymagają natychmiastowej naprawy — każda z nich stanowi albo ryzyko bezpieczeństwa (rate limit, CVE), albo blokuje skalowalność (DOM overload, IntersectionObserver), albo jest błędem konfiguracji (`robots.txt`). Metryki P1 wpływają bezpośrednio na konwersję i retencję użytkowników. Metryki P2 to optymalizacje, które stają się istotne przy skali >1000 MAU.

† Wartość 73% to **heurystyka benchmarkowa** (typowy SaaS B2C), nie pomiar z **PodTeksT**. GA4 jest zaimplementowane, ale nie śledzi lejka konwersji. Wymagana walidacja przez GA4 funnel events.

1.11 Skonsolidowana roadmapa

Poniższa roadmapa integruje rekomendacje ze wszystkich dziewięciu osi audytu w cztery sprinty, uporządkowane według priorytetu i zależności implementacyjnych.

1.11.1 Sprint 0 — naprawy bezpieczeństwa (1–2 dni)

Tabela 1.58: Sprint 0 — naprawy bezpieczeństwa (P0, bez zmian architektonicznych)

Zadanie	Oś audytu	Pliki / uwagi
Naprawić `rate-limit.ts` + migracja do Upstash Redis	Bezpieczeństwo	`src/lib/rate-limit.ts`
`package.json` (4 HIGH CVE)	Aktualizacja jsPDF do 4.2.0+	Bezpieczeństwo
Naprawić domenę w `robots.txt`	SEO	chatscope.app → podtekst.app
Unit Economics	logi API routes	Dodać monitoring thinking tokens

Sprint 0 to wyłącznie naprawy bezpieczeństwa i konfiguracji — żadne zmiany architektoniczne, żadne nowe pliki komponentów. Cel: zamknąć w 1–2 dni robocze, zdeployować niezależnie od dalszych prac.

1.11.2 Sprint 1 — architektura tabowa (3–5 dni)

Tabela 1.59: Sprint 1 — przebudowa architektury strony analizy

Zadanie	Oś audytu	Pliki / uwagi
Stworzyć `AnalysisTabs` + 5 tab komponentów	Design	6 nowych plików
Rozbić `page.tsx` na taby	Design	page.tsx 1322→∼200 LOC
URL hash sync (`#overview`, `#ai`, ...)	Optymalizacja	deep-linking
Przenieść handlery `useCallback` do tabów	redukcja re-renderów	

1.11.3 Sprint 2 — P1: monetyzacja i bezpieczeństwo

Tabela 1.60: Sprint 2 — zadania P1 (ważne)

Zadanie	Oś audytu	Pliki / uwagi
Monetyzacja	2 nowe pliki Strona polityki prywatności	`TierContext` + `PaywallGate` Bezpieczeństwo
`/privacy`, GDPR compliance `Content-Security-Policy` + HSTS headers	Bezpieczeństwo	`next.config.ts` Deep validation `samplesSchema`
Bezpieczeństwo	`src/lib/validation/schemas.ts` Onboarding Export guide per platform (onboarding)	
ikony + instrukcje w `DropZone` Welcome modal + guided tour	Onboarding	nowy komponent Conditional font loading (oszczędność 145 KB)
SEO, Performance	`layout.tsx`, font subsetting	

1.11.4 Sprint 3 — P1/P2: wydajność i jakość

Tabela 1.61: Sprint 3 — zadania P1/P2 (wydajność i jakość kodu)

Zadanie	Oś audytu	Pliki / uwagi
Optymalizacja	`AnalysisTabs.tsx` `React.memo()` na chartach	`React.lazy()` per tab Optymalizacja
6 komponentów wykresów Callback refactor per tab	Optymalizacja	przeniesienie handlerów do tabów Server Components dla statycznych sekcji landing
Optymalizacja	`LandingFAQ`, `LandingFooter` Bundle analyzer	Optymalizacja
`next.config.ts` Test coverage >60% dla `lib/` Mobile UX	Jakość kodu min 44×44 px	Vitest, nowe pliki testów Touch target fixes (hamburger, checkbox)

1.11.5 Sprint 4 — P2: optymalizacja zaawansowana

Tabela 1.62: Sprint 4 — zadania P2 (optymalizacja zaawansowana)

Zadanie	Oś audytu	Pliki / uwagi
Optymalizacja	`parser.worker.ts` (nowy)	Web Worker parser Optymalizacja
IndexedDB quota management		
`src/lib/utils.ts` Virtualized `ShareCardGallery`	Optymalizacja	`react-window` Response caching (hash-based)
Unit Economics	cache Gemini responses	Unit Economics
Batch API dla StandUp/CPS		
redukcja round-trips „Przeanalizuj kolejną" CTA po analizie	Onboarding	post-analysis retention Email/push notification system
Onboarding	powiadomienia o nowych funkcjach	Performance
Migracja Satori dla share cards		
SSR zamiast html2canvas		

1.12 Podsumowanie

Audyt wieloagentowy w dziewięciu osiach zidentyfikował **6 krytycznych (P0)**, **11 ważnych (P1)** i **8 dodatkowych (P2)** problemów w architekturze **PodTeksT**. Poniżej synteza kluczowych wniosków z każdej osi:

1. Design (UX strony analizy) Monolityczna strona analizy (1322 LOC, 50+ komponentów) wymaga rozbicia na architekturę tabową z 5 zakładkami. AI Analysis — najcenniejsza treść — jest pogrzebana pod 3000 px danych ilościowych. Proponowane 5 tabów z lazy-loadingiem redukują DOM o 80% i przesuwają AI na pozycję 3.

2. Monetyzacja Brak jakiejkolwiek infrastruktury płatności. Proponowany model 3-tier (Free / Pro 29,99 PLN / Unlimited 49,99 PLN) z paywallem w momencie największego zaangażowania i share cards jako viral loop. Strategia akwizycji (sekcja 1.2.6) oparta na 7 kanałach wzrostu: viral share cards, SEO, TikTok content, mikro/makro-influencerzy, Meta Ads, Google Ads. Realistyczna prognoza (scenariusz Lean Startup, ~3 500 PLN/mies budżetu): ~3 000 MAU i ~55 000 PLN ARR w miesiącu 12, break-even ~miesiąc 15–18. Wymagana inwestycja do break-even: ~30 000 PLN.

3. Optymalizacja wydajności Rate limiting wyłączony w produkcji (ryzyko kosztowe), 51+ IntersectionObserver instances (pamięć), brak lazy-loadingu treści (initial bundle), brak `React.memo()`

na ciężkich komponentach (re-rendery). Architektura tabowa rozwiązuje większość problemów naturalnie.

4. Unit economics Koszty Gemini API są bardzo niskie w przeliczeniu na PLN — podstawowa analiza kosztuje ∼0,11 PLN, pełna ∼0,36 PLN. Przy cenach 29,99/49,99 PLN marża brutto wynosi ∼91%. Główne ryzyko: thinking tokens w nowych modelach Gemini mogą zwiększyć koszty 3–5× bez monitoringu.

5. Bezpieczeństwo Rate limit **wyłączony** w produkcji, brak nagłówków `Content-Security-Policy` i HSTS, 5 podatności HIGH w zależnościach (jsPDF CVE), brak walidacji głębokiej w `samplesSchema`, brak strony polityki prywatności (GDPR). Wszystkie 5 HIGH CVE wymagają natychmiastowej naprawy.

6. Mobile UX Landing page jest dobrze zoptymalizowany (diagonal hero, pinned CTA). Strona analizy ma problemy z nadmiarem IntersectionObserver (51+ na mobile = jank), touch targets <44 px (hamburger menu, checkboxy), brak optymalizacji uploadu dla iOS (brak instrukcji „Pliki" app).

7. Onboarding Minimalny, ale funkcjonalny flow w 4 krokach (landing → upload → parse → AI). Krytyczny brak: instrukcja eksportu per platforma — użytkownicy nie wiedzą *jak* wyeksportować chat z Messengera/WhatsAppa. Heurystyczny szacunek drop-off landing→AI: ∼73% (benchmark SaaS B2C, **bez danych własnych** — GA4 nie śledzi lejka konwersji). Hipoteza: brak export guide odpowiada za znaczną część utraty.

8. Jakość kodu Doskonała konfiguracja TypeScript (strict mode, 0 użyć `any`), ale zaledwie 3 pliki testów (pokrycie <5% dla `lib/`). 5 podatności HIGH w `npm audit`. Kod jest dobrze zorganizowany modularnie (`quant/` submoduły), ale brakuje testów regresji dla parserów i analizy ilościowej.

9. SEO Doskonałe metadane (Open Graph, Twitter Cards, structured data). Krytyczny błąd: `robots.txt` wskazuje na `chatscope.app` zamiast `podtekst.app` (rebranding niedokończony). Ciężkie fonty: 245 KB (Geist Sans 140 KB + Geist Mono 65 KB + Syne 40 KB) — conditional loading może zaoszczędzić ∼145 KB.

Szacowany wpływ implementacji wszystkich 4 sprintów:

- **UX:** AI Analysis dostępne w 2 kliknięciach zamiast 3000 px scrollowania; DOM −80%
- **Performance:** −60% initial JS, −80% IntersectionObserver, INP <200 ms, fonty −145 KB
- **Revenue:** od 0 PLN do ∼55 000 PLN ARR (scenariusz Lean Startup, budżet ∼3 500 PLN/mies, break-even ∼miesiąc 15–18)
- **Security:** rate limiting przywrócony, CSP + HSTS, 0 podatności HIGH, GDPR compliance
- **Costs:** marża brutto >85% nawet przy agresywnym użyciu AI; cache redukuje koszty o 35%
- **Mobile:** 100% touch target compliance, optymalizacja upload flow dla iOS
- **Onboarding:** drop-off landing→AI: redukcja dzięki export guide + welcome modal (skala nieznana bez baseline'u z GA4)
- **Quality:** test coverage `lib/` z <5% do >60%, 0 podatności w `npm audit`
- **SEO:** poprawny `robots.txt`, fonty −60%, ranking stabilny
- **DX:** `page.tsx` z 1322 do ∼200 LOC, modularny kod, CI/CD z testami