1. Przegląd specyfikacji i założeń

- Przeczytać „Podsumowanie” w specyfikacji, by zidentyfikować trzy główne moduły: dynamiczny pasek postępu czytania, inteligentny popup „Czytaj więcej” oraz system rekomendacji .
- wszystkie wymagania funkcjonalne:
- Dynamiczne śledzenie pozycji scrolla i obliczanie procentu przeczytanego tekstu .
- Wyzwalacze popupu: osiągnięcie % progresu, czas >60 s, zmiana kierunku scrolla .
- Dobór artykułów w rekomendacjach według kryteriów (popularność, tagi, data) .
- wymagania niefunkcjonalne:
- Kompatybilność z page builderami (Elementor, WP Bakery, Gutenberg, natywny edytor) .
- Optymalizacja wydajności i bezpieczeństwo: lazy loading, cache’owanie, sanitizacja danych, pomiar CLS/LCP .
- wymagania środowiskowe: WordPress ≥5.6, PHP ≥7.4, MySQL ≥5.7, obsługa ES6+ .

2. Przygotowanie środowiska deweloperskiego

- Utworzyć i zainicjować repozytorium Git o nazwie re-progress_bar.
- Skonfigurować wersjonowanie semantyczne (SEMVER) oraz plik .gitignore dla WordPress.
- Zainstalować lokalną instancję WordPress 5.6+ na XAMPP lub Lando; skonfigurować bazę MySQL 5.7+.
- Zainstalować narzędzia:
- PHP 7.4+ oraz Composer (do autoloadingu PSR-4).
- WP CLI (zarządzanie pluginami i bazą).
- ESLint + Prettier dla JS (ES6+) i PHP_CodeSniffer z regułami PSR-12.
- PHPUnit (testy jednostkowe PHP) oraz Jest (testy JS).
- Utworzyć plik konfiguracyjny CI (GitHub Actions) do automatycznego lintingu i uruchamiania testów.

3. Inicjalizacja pluginu

- Stworzyć główny plik re-progress_bar.php z nagłówkiem pluginu: nazwa, wersja, autor, tekst licencji.
- Zarejestrować hook plugins_loaded, w którym załadować autoloader Composer lub własny loader klas.
- Zadeklarować namespace ReProgressBar oraz strukturę podstawowych klas:
- Bootstrap – inicjalizacja modułów.
- Admin\Settings – rejestracja ustawień.
- Frontend\ProgressTracker – moduł śledzenia postępu.
- Dodawać wszystkie dalsze hooki i filtry w metodzie run() klasy Bootstrap, aby separować logikę.

4. Moduł śledzenia postępu (Progress Bar)

- Zarejestrować w wp_enqueue_scripts:
- Skrypt JS z logiką Intersection Observer.
- Styl CSS paska (zmienne dla pozycji, wysokości, kolorów).

- W JS wykorzystać API IntersectionObserver lub scroll listener do:
- Obliczania procentu przeczytanego tekstu (scrollTop ÷ (documentHeight – viewportHeight)).
- Emitowania zdarzeń przekroczenia progu np. reProgressThresholdReached.
- Zaimplementować inteligentne pomijanie elementów: nagłówków, stopek i elementów sticky .
- Udostępnić w PHP funkcję pomocniczą „get_progress_data()” zwracającą obecny procent, umożliwiającą integrację z popupem i rekomendacjami.
- Testować na przykładach stron z różnymi układami (motywy z nagłówkami stałymi, z sidebarami) w celu weryfikacji poprawności działania.

5. Panel administracyjny

- Zarejestrować stronę ustawień w hooku admin_menu i skonfigurować ją przez Settings API.
- Dodać sekcje i pola konfiguracyjne dla:
- **Paska postępu**: pozycja (góra/dół/niestandardowa), wysokość, kolor podstawowy i tła, przezroczystość, wykluczanie elementów po selektorach CSS, obsługa szablonów i typów postów .
- **Popupu „Czytaj Więcej”**: procentowy próg, czas opóźnienia, kierunek scrolla, konfiguracja czasu (>60 s) oraz edytor WYSIWYG do tworzenia treści pop’upa .
- **Rekomendacji**: wybór trybu algorytmu (automatyczny/ręczny), liczba i układ pozycji, opcja integracji z zewnętrznymi API .
- Sanityzować i walidować wszystkie wartości przy zapisie przy użyciu sanitize_text_field, sanitize_hex_color, absint itp.

6. Popup „Czytaj więcej”

- Załadować skrypty i style przez wp_enqueue_script/wp_enqueue_style.
- Implementować wyzwalacze:
- osiągnięcie zdefiniowanego % progresu,
- czas spędzony na stronie (> 60 s),
- zmiana kierunku scrolla po powrocie do góry .
- Wykorzystać wp_localize_script do przekazania do JS parametrów wyzwalaczy i treści pop’upa.
- Udostępnić API JS emitujące zdarzenie re_progress_popup_trigger i umożliwić podpinanie dodatkowych callbacków.
- Zaimplementować moduł personalizacji wyglądu i zawartości (tło, czcionki, przyciski, listę kolejnych artykułów) .

7. Silnik rekomendacji

- Opracować klasę odpowiedzialną za pobieranie propozycji na podstawie:
- popularności (liczba wyświetleń/komentarzy),
- tagów i kategorii,
- daty publikacji .
- Wdrążyć cache’owanie wyników przez Transients API lub zewnętrzny store (Redis/Memcached) z TTL konfigurowalnym w panelu .
- Udostępnić filtry i hooki (apply_filters('re_progress_recommendations'), do_action('re_progress_after_load')) dla rozszerzalności algorytmu.
- Zapewnić fallback do prostego sortowania po dacie, gdy API zewnętrzne jest niedostępne.

8. Kompatybilność z Page Builderami

- Przeanalizować API każdego buildera: Elementor, WPBakery, Gutenberg, zweryfikować dostępne hooki i klasy .
- **Elementor**:
- Zarejestrować widget przez \Elementor\Plugin::instance()->widgets_manager->register_widget_type()
- Utworzyć klasę Elementor\Widget_Progress_Bar z metodami get_name(), get_title(), get_icon(), register_controls(), render()
- Dodać wsparcie dla Dynamic Tags (metadane progresu)
- **WPBakery**:
- Zdefiniować shortcode w callbacku vc_map() ze wszystkimi parametrami paska, popupu i rekomendacji
- Utworzyć funkcję renderującą HTML/CSS/JS paska i popupu, odwołującą się do ReProgressBar\Frontend\Assets
- **Gutenberg**:
- Napisać blok w React z registerBlockType('re-progress-bar/block', { ... }), uwzględnić atrybuty dla pozycji, kolorów, progów popupu
- Enqueue edytorowe i front-endowe skrypty/styles przez enqueue_block_editor_assets i enqueue_block_assets
- **Fallback**:
- Zarejestrować globalny shortcode re_progress_bar i podłączyć go do the_content w add_filter('the_content', ...) dla stron bez buildera

9. Bezpieczeństwo i optymalizacja

- **Sanityzacja i eskalacja**: każdą wartość z ustawień admina filtrować przez sanitize_text_field, sanitize_hex_color, absint; każdą wartość wyjściową owijać w esc_attr, esc_html, wp_kses_post .
- **Lazy loading zasobów**:
- Rejestracja skryptów z wp_register_script(..., [], false, true) i ładowanie tylko na stronach z progres barem
- Użycie loading="lazy" dla obrazów w rekomendacjach
- **Minimalizacja i bundling**: zbudować produkcyjną paczkę CSS/JS przez webpack z trybem produkcyjnym (mode: 'production')
- **Core Web Vitals**: wstawić w CI zadanie uruchamiające Lighthouse CI dla metryk CLS i LCP; analizować raport i optymalizować krytyczne CSS/JS

10. Testy i kontrola jakości

- **Testy jednostkowe PHP**:
- Utworzyć klasy testowe w tests/php/ rozszerzające WP_UnitTestCase
- Testować rejestrację hooków, poprawność obliczeń procentu, sanityzację danych
- **Testy JavaScript**:
- W tests/js/ napisać testy funkcji obliczających procent, wyzwalaczy popupu i hooków eventowych za pomocą Jest
- **Testy manualne**:
- Sporządzić matrycę testów w różnych motywach (TwentyTwentyFive, Astra, OceanWP) i builderach (Elementor, WPBakery, Gutenberg)

- Sprawdzać poprawność paska, popupu, rekomendacji oraz fallback w natywnym edytorze
- **CI na GitHub Actions**:
- Workflow uruchamiający: PHP_CodeSniffer (PSR-12), ESLint, PHPUnit, Jest przy każdym pushu i pull requeście

11.Dokumentacja i przygotowanie do wydania

- **README.md**:
- Sekcje: Wprowadzenie, Wymagania, Instalacja, Konfiguracja panelu, Użycie shortcodów/bloku, Przykłady, FAQ
- **Changelog**:
- Stosować format Keep a Changelog: wersja, data, dodane, zmienione, usunięte, naprawione
- **readme.txt**:
- Krótkie streszczenie, instrukcja instalacji, link do dokumentacji, sekcja „Upgrade Notice”
- **Pakowanie ZIP**:
- Zawrzeć główny plik pluginu, includes/, assets/, languages/, readme.txt
- Wykluczyć pliki rozwojowe (.gitignore, .editorconfig, tests/, vendor/, build/)

12.Wskazówki do dalszego rozwoju

- **Wersjonowanie REST API**: wprowadzić stałą REPB_API_VERSION, trasy rejestrować pod /re-progress-bar/v1/... i przygotować migracje pod v2
- **A/B testing popupów**: zintegrować feature flags (np. LaunchDarkly), dodać warunki losowego przydziału użytkowników do wariantów A/B
- **Analityka zaangażowania**:
- Emitować eventy JS do dataLayer dla Google Analytics (próg progressu, odsłony popupu, kliknięcia rekomendacji)
- Zbierać metryki CTR rekomendacji i średni czas pozostały do scrolla końca artykułu
- Przeglądać raporty i optymalizować algorytm rekomendacji na podstawie zebranych danych