Project Plan: APSS Search Overlay (WordPress Plugin)

1. Executive Summary

The goal is to build a high-performance, secure, and professional live search overlay for the All Pumps portfolio website. The UI/UX will mimic the full-page overlay seen on gplplus.com, where the header remains visible while the search content covers the rest of the page. The technical architecture will prioritize the WordPress REST API for speed and portability, ensuring compatibility with Oxygen Builder and future theme changes.

2. Technical Architecture & Pillars

2.1. Security First

- **Input Validation:** Every search query must be sanitized using `sanitize_text_field()` before database interaction.
- **Output Escaping:** All dynamic data (titles, links, excerpts) must be escaped using `esc_html()` and `esc_url()` before rendering to prevent XSS.
- **REST API Nonce:** Implement `X-WP-Nonce` header validation to ensure requests are authentic and originating from the site.
- **Permission Callbacks:** Set up `permission_callback` in `register_rest_route` to `__return_true` for public accessibility while strictly controlling query parameters.

2.2. High-Performance Logic

- **Debouncing:** JavaScript will wait 300ms after the last keystroke before firing the API request to prevent server hammering.
- **Request Cancellation:** Utilize the `AbortController` API to cancel pending fetch requests if the user continues typing.
- **Server-Side Caching:** Use the WordPress Transients API to cache search results for 1 hour, significantly reducing database load for common queries.
- **Asset Optimization:** Use `wp_enqueue_script` with `strategy => 'defer'` and only load assets when the search shortcode is present on the page.

2.3. Oxygen Builder Integration

- **Z-Index Management:** The search overlay will use `z-index: 999`, while the Oxygen Header template will be assigned `z-index: 1000` to remain visible on top.

- **Shortcode Implementation:** An `[apss_search_trigger]` shortcode will be created for easy placement of the search icon within Oxygen's header.

3. Detailed Development Phases

Phase 1: Core Plugin Infrastructure

- **Task 1.1:** Setup plugin directory structure:
  - `/apss-search/`
  - `/apss-search/assets/css/apss-search.css`
  - `/apss-search/assets/js/apss-search.js`
  - `/apss-search/apss-search.php`
- **Task 1.2:** Write the main plugin file with asset registration and shortcode definitions using the `APSS_` namespace.

Phase 2: Secure Backend (REST API)

- **Task 2.1:** Register custom route `wp-json/apss/v1/search`.
- **Task 2.2:** Build a `WP_Query` function that handles:
  - Post types: `post`, `page`, `portfolio`, `product`.
  - Results count: Limit to top 8 items for speed.
  - Response data: ID, Title, Permalink, Featured Image URL.
- **Task 2.3:** Implement Transient caching based on an MD5 hash of the search term.

Phase 3: The Overlay UI (HTML/CSS)

- **Task 3.1:** Create the HTML template for the full-screen overlay (Close button, Input field, Results grid, Loading spinner).
- **Task 3.2:** Apply CSS for the full-screen effect:
  - `position: fixed; top: 0; left: 0; width: 100%; height: 100vh;`
  - Backdrop filter: `blur(8px)` with a high-opacity background.
  - Responsive Grid: 1-column mobile, 3-column desktop layout.

Phase 4: Dynamic Frontend Logic (JavaScript)

- **Task 4.1:** Write the toggle logic for opening/closing the overlay via the `.apss-trigger` class.
- **Task 4.2:** Implement the `fetch` logic using `async/await` and the `AbortController`.

- **Task 4.3:** Add keyboard support (closing on `Escape` key and focusing input on open).
- **Task 4.4:** Handle "Empty" states with a professional "No results found" message.

4. Gemini CLI Curriculum (Prompts)

Prompt 1: The Foundation

> "Act as a senior WordPress developer. Generate a production-ready plugin boilerplate named 'APSS Search'. Create the main PHP file that registers a shortcode `[apss_search_trigger]` and enqueues `assets/css/apss-search.css` and `assets/js/apss-search.js`. Ensure the assets are only loaded if the shortcode is present. Use the namespace 'APSS_Search' for all functions and include security checks to prevent direct file access."

Prompt 2: The Secure API

> "Add a REST API endpoint to the 'APSS Search' plugin at `apss/v1/search`. It should accept a 'term' parameter. Use `WP_Query` to search 'post', 'page', and 'product' types. **Requirements:** Sanitize input with `sanitize_text_field`, escape JSON output, and implement a 60-minute cache using WordPress Transients based on the search term hash."

Prompt 3: The Full-Screen UI

> "Generate the HTML and CSS for a full-screen search overlay for the 'APSS Search' plugin. The overlay must have a `z-index` of 999. CSS requirements: use `position: fixed`, a white background with 98% opacity, and a large centered search input. Results should display in a responsive grid. Design it to be minimalist and premium for a B2B portfolio site."

Prompt 4: Performance JS

> "Write a high-performance Vanilla JS file for 'APSS Search'. **Requirements:** Use a 300ms debounce for the search input. Use `AbortController` to cancel previous fetch requests. Update the results container with JSON data from the `apss/v1/search` endpoint. Include a CSS loading spinner. Ensure the `ESC` key closes the overlay and the search input is focused automatically on open."

5. QA & Launch Checklist

- [ ] Verify `z-index` hierarchy (Header at 1000, Overlay at 999).
- [ ] Test mobile responsiveness and virtual keyboard clearance.

- [ ] Run a security audit on the REST API endpoint.
- [ ] Check page speed impact (ensure assets only load on necessary pages).
- [ ] Verify the "Close" button and "Escape" key functionality.