Implementation & Ethics Memo

KnowB4YouGo - Eoin O'Loughlin

1. How I Actually Used AI While Building

Throughout the development of KnowB4YouGo, I relied primarily on Google AI Studio as my vibe coding tool, supplemented by Claude for documentation, debugging assistance, and conceptual refinement. The division of labor between human judgment and AI assistance became clearer as the project progressed, revealing both the power and limitations of current generative AI tools for product development.

For code generation, AI Studio proved most valuable for scaffolding, generating initial HTML structure, CSS layouts, and JavaScript event handlers. When I described the deal card grid layout or the search bar component, the tool produced functional starting points roughly 70% of the time. However, the remaining 30% required significant human intervention: fixing CSS specificity conflicts, correcting JavaScript scope issues, and adjusting responsive breakpoints that the AI consistently misjudged. The tool excelled at producing "plausible" code but often missed the nuances that make code production-ready.

Debugging represented a more mixed experience. AI tools could quickly identify syntax errors and suggest fixes for common patterns, but struggled with logic errors specific to my application's state management. When the search results weren't filtering correctly, I spent considerable time explaining the bug to the AI before realizing it would be faster to trace through the code manually. This taught me that AI assistance works best when problems can be isolated and described precisely, vague "it's not working" queries yield vague responses.

For copywriting and UX text, AI assistance was highly effective. Generating placeholder content, refining button labels, and drafting error messages went smoothly. The tool understood tone and context well, producing user-facing text that required minimal editing. Design ideation also benefited from AI input, when I was stuck on how to display deal verification status, prompting the AI for visual approaches yielded several viable options I hadn't considered.

Human judgment proved essential in three areas: architectural decisions (how to structure the codebase, when to split components, how to handle state), user experience trade-offs (what to prioritize when screen space is limited, how to balance information density with clarity), and quality assessment (determining when code was "good enough" versus when it needed refinement). These decisions required context about the product's goals, user needs, and technical constraints that AI tools couldn't fully grasp from prompts alone.

2. Why the AI Feature Looks the Way It Does

The natural language search feature emerged as the centerpiece of AI integration because it directly addresses the core user pain point: the friction of finding relevant deals. Traditional filter-based interfaces require users to make multiple selections (cuisine type, price range, distance, deal category) before seeing results. This works for users who know exactly what they want, but most deal-seekers have fuzzier intentions: "somewhere cheap near campus" or "happy hour that's still going." Natural language search accommodates this ambiguity.

I chose this feature over alternatives like AI-generated deal recommendations or chatbot-style assistance for practical reasons. Recommendations require user history and behavioral data we don't yet collect. A conversational chatbot would need to handle multi-turn interactions, error recovery, and context maintenance, complexity that exceeded what I could build reliably in the prototype timeframe. Natural language search, by contrast, is a single-turn interaction: query in, results out. This simplicity made it feasible to implement robustly.

Significant scoping decisions shaped the final implementation. The original PRD envisioned AI-powered deal aggregation, automatically scraping restaurant websites and social media to extract deal information. I deprioritized this for the prototype because web scraping introduces legal complexity (terms of service compliance), technical fragility (sites change constantly), and data quality challenges (unstructured text is hard to parse reliably). Instead, I used curated sample data to demonstrate the user experience, with aggregation planned for future phases when more development time is available.

The connection to the core value proposition is direct but incomplete. Users can experience the intended workflow, type a natural query, see relevant results, but the deals themselves are illustrative rather than real-time. This creates a gap between the demo and the full vision. I addressed this by being transparent in the UI (labeling data as sample content) and focusing the prototype on proving the interaction model rather than the data pipeline.

3. Risks, Trade-offs, and Integrity

Privacy and Data Use

The prototype collects minimal user data: search queries (sent to the Gemini API for processing) and selected location (stored only in the browser session). No personal identifiers, no accounts, no persistent storage. This was a deliberate choice to minimize privacy risk during the prototype phase. If the product scales, user accounts and preference storage would require a comprehensive privacy policy, data retention limits, and clear consent mechanisms. The original PRD committed to never selling user data to third parties, a commitment I would maintain and formalize in production.

Bias and Fairness

Search result ranking introduces potential bias. If the AI consistently interprets "good food" as certain cuisines or price points, it could disadvantage restaurants that don't fit those patterns. The sample data is too small to surface these issues, but at scale, systematic evaluation would be essential. I would implement logging of search queries and results, then analyze whether certain venue types are systematically under-ranked. Corrective measures might include diversifying training examples or adding explicit fairness constraints to ranking logic.

Over-reliance and User Trust

The "AI verified" badges in the prototype represent a design tension. They communicate trustworthiness to users, but without actual verification infrastructure, they risk being misleading. I chose to include them to demonstrate the intended UX, but clearly labeled them as illustrative. In production, verification badges should only appear when backed by genuine cross-reference checking. Users making plans based on deal information deserve accuracy; a badge that implies verification without delivering it would erode trust and potentially cause real harm (wasted trips, disappointed expectations).

Academic Integrity

I used AI tools throughout this project in ways consistent with course guidelines: as coding assistants, documentation aids, and idea generators. All AI-generated content was reviewed, edited, and integrated by me, nothing was submitted as-is without human oversight. This memo and the PRD were drafted with AI assistance for structure and language refinement, but the ideas, decisions, and evaluations are my own. I've documented AI usage in appendices where relevant, maintaining transparency about which elements benefited from generative AI input.

Intentional Limitations

Several explicit constraints limit what the AI does in this product. Search queries are capped at 200 characters to prevent prompt injection attacks or resource abuse. Rate limiting (20 searches per session) prevents automated scraping of our AI integration. The AI interprets queries but doesn't generate deal content, all deal information comes from our database, preventing hallucinated specials that venues don't actually offer. These guardrails prioritize reliability over capability, a trade-off I believe is appropriate for a product people rely on for real-world decisions.

4. What I Learned About Building with GenAI

The biggest surprise was how much iteration prompt engineering requires. I initially assumed that clear instructions would yield consistent results, but discovered that small phrasing changes dramatically affected output quality. The search intent extraction prompt went through at least eight revisions before producing reliable JSON formatting. This isn't a flaw in the technology, it's a fundamental characteristic of working with probabilistic language models. Prompt design is a skill that rewards practice and systematic experimentation.

The hardest challenge was maintaining context across development sessions. AI tools don't remember previous conversations (unless explicitly designed to), so each session required re-establishing context about the project, codebase, and current issues. This created friction that slowed development. Future projects would benefit from better documentation practices, maintaining a "context document" that could be fed to AI tools at the start of each session.

If I were advising another founder on using GenAI tools effectively, I would emphasize three principles. First, use AI for acceleration, not replacement. It's faster to edit AI-generated code than to write from scratch, but reviewing that code is non-negotiable. Second, break complex tasks into smaller pieces; AI handles focused prompts better than sprawling ones. Third, develop a testing habit immediately; AI-generated code often works in isolation but fails when integrated, so continuous testing catches issues early.

This project has meaningfully shaped how I think about AI in my capstone and future ventures. For my Lenovo diagnostics work, I now see clearer parallels: AI can interpret natural language descriptions of hardware problems just as it interprets deal search queries. The pattern of "fuzzy human input → structured machine output" applies broadly. I'm also more realistic about what AI can deliver in a prototype timeframe versus what requires longer-term infrastructure investment. The gap between demo and production is real, and planning for it from the start leads to more honest pitches and more achievable roadmaps.

Building KnowB4YouGo reinforced that AI is a powerful tool but not a shortcut. The best results came from treating AI as a collaborator that needed clear direction, consistent feedback, and human judgment to produce genuinely useful output. That collaboration model, human intent plus machine capability, feels like the right framework for building responsible, effective AI-enhanced products.