* **Project Name:** opportunityknocks.ai - Wizard's Toolkit
* **Version:** 1.0
* **Date:** August 9, 2025
* **Author:** Software Entrepreneur Agent for Jim Younkin

**1. Overview & Goals**

**1.1. Problem Statement**

Aspiring entrepreneurs and product managers spend dozens of hours on unstructured, manual research to find and validate new business ideas. This process is inefficient, prone to noise, and a significant barrier to starting a new venture.

**1.2. Proposed Solution**

An internal, command-line toolkit that automates the discovery and synthesis of business ideas from high-value online sources (YouTube and academic papers). This tool will empower a "human wizard" (the founder) to rapidly generate high-quality "Opportunity Brief" reports for early customers.

**1.3. Strategic Goals**

* **Market Validation:** To validate customer demand for the Opportunity Brief service before investing in a full-fledged, customer-facing SaaS platform.
* **Enable Manual Fulfillment:** To equip the founder with a tool that makes the "Wizard of Oz" fulfillment process efficient and repeatable.
* **IP Development:** To create the core data processing and AI synthesis logic that will serve as the foundation for the future automated product.

**2. Scope**

**2.1. In Scope for MVP ✅**

* A Python script executed from the command line (CLI).
* A module to fetch and analyze YouTube video transcripts for a given topic.
* A module to fetch and analyze academic papers from arXiv.org for a given topic.
* Integration with a Large Language Model (e.g., OpenAI, Anthropic) for text synthesis.
* Configuration managed via a .env file for API keys and a config.py file for parameters.
* Output of structured data to local JSON or text files.
* Basic error handling for common issues (e.g., API failures, missing transcripts).

**2.2. Out of Scope for MVP ❌**

* **Any customer-facing UI (No Vite/React frontend).**
* A web server or API (No FastAPI). The tool is a local script.
* User accounts, authentication, or databases.
* Automated payment processing integration.
* Deployment to a cloud environment. The script will run on a local machine.

**3. User Persona & Use Case**

* **Persona:** "The Wizard" (the Founder, Jim). A technically proficient operator who needs to fulfill customer research orders quickly and to a high standard.
* **Key Use Case:**
  1. The Wizard receives a customer order with a specified topic (e.g., "AI in preventative healthcare").
  2. He opens his terminal and runs the toolkit script, passing the topic as an argument.
  3. The script runs and generates two structured output files: [topic]\_youtube.json and [topic]\_arxiv.json.
  4. The Wizard opens these files, reviews the AI-generated insights, and uses them as the primary building blocks to write the final, polished PDF Opportunity Brief for the customer.

**4. Functional Requirements**

**FR-1: Core Application**

1.1. The application must be a Python script run from the command line.

1.2. The application must manage all dependencies via a requirements.txt file.

1.3. The application must load sensitive API keys (e.g., OPENAI\_API\_KEY) from a .env file, which will be excluded from version control.

1.4. The application must provide a command-line interface (e.g., using argparse) that accepts the following arguments:

\* --topic: A required string defining the research subject.

\* --source: An optional argument to specify running only one module (youtube or arxiv). Defaults to running both.

**FR-2: YouTube Analysis Module**

2.1. The module must use the youtube-transcript-api library to fetch video transcripts.

2.2. Given a topic, the module must programmatically find the Top N (configurable, default=10) relevant videos.

2.3. The module must aggregate the text from all retrieved transcripts.

2.4. The module must pass the aggregated text to an LLM for synthesis, handling text that exceeds the model's context window by chunking.

2.5. The module must save its output as a structured JSON file named [topic]\_youtube.json.

**FR-3: arXiv Analysis Module**

3.1. The module must use the arxiv library to search for papers and pypdf to extract text.

3.2. Given a topic, the module must download the PDFs of the Top N (configurable, default=5) relevant papers.

3.3. The module must extract the full text content from each downloaded PDF.

3.4. The module must implement a two-step AI synthesis process:

\* First, summarize each paper individually.

\* Second, synthesize the summaries to identify potential business applications.

3.5. The module must save its output as a structured JSON file named [topic]\_arxiv.json.

**5. Non-Functional Requirements**

* **NFR-1 (Configurability):** Key parameters like the number of videos/papers to fetch and the specific LLM model to use (gpt-4-turbo, claude-3-sonnet-20240229, etc.) **must** be easily editable in a central config.py file.
* **NFR-2 (Error Handling):** The script **must not** crash on common, recoverable errors. It should log a warning and continue (e.g., a specific video transcript is unavailable, a PDF is corrupted/unreadable).
* **NFR-3 (Security):** The script **must not** contain any hardcoded API keys or other secrets.

**6. Data Flow**

The data will flow through the system as follows:

CLI Command (Topic) -> [YouTube Module | arXiv Module] -> Fetch Raw Content (Transcripts, PDFs) -> Extract Text -> Chunk Text -> LLM API for Synthesis -> Structured JSON File -> Wizard uses JSON to write Final PDF Report

**7. Success Criteria & Timeline**

* **Success Criteria:** The project will be considered complete when the Wizard can successfully execute a single command (python main.py --topic "wearable tech for seniors") and receive two well-structured JSON output files containing relevant, AI-synthesized ideas, enabling the efficient creation of a customer-ready report.
* **Timeline:** This PRD is designed to be fully implemented within the **1-Week "First Dollar" Sprint** previously outlined.