Episode 4 Blueprint: The Reddit-to-Question-Bank Pipeline

This Blueprint documents the technical pipeline Christopher Penn demonstrated for turning Reddit threads into a ranked list of audience questions, from model selection through the final question-generation prompt. Every prompt below is reproduced exactly as given in the recording. Where the transcript left a gap, it is marked rather than filled in.

Technical stack

Tool	Role	Plan / model notes
OpenCode	AI coding agent that orchestrates the build session. Connects to hundreds of model providers rather than one.	Free, open source. Terminal or desktop interface.
DeepInfra	Cloud inference provider hosting the models used inside OpenCode.	Pay-per-token. Zero data retention, per DeepInfra's own privacy policy.
DeepSeek V4 Flash	Primary coding model for day-to-day build work.	Served via DeepInfra. Selected for high capability at low cost; set the thinking/reasoning budget to maximum.
Kimi K3 (Moonshot AI)	Secondary model for planning and harder debugging passes.	Served via DeepInfra.
Artificial Analysis	Independent benchmark site used to select the two models above by plotting capability	Free to use. Reference tool only, not part of the runtime stack.

Tool	Role	Plan / model notes
	against price.	
Reddit Developer Platform (DevIt)	Data source. Provides the API used to pull posts from target subreddits.	Reddit's current developer system, replacing the pre-2025 open API key process. Requires app registration at developers.reddit.com.
PRAW (Python Reddit API Wrapper)	Fallback Reddit connector if the DevIt setup fails.	Free, open source Python library.
Python 3.12 + SQLite	Runtime and local storage for the custom data-collection application.	Local only, no cloud database required. DuckDB or Parquet are workable substitutes.
Superpowers (Jesse Vincent / Prime Radiant)	Agentic skills plugin installed inside OpenCode. Enforces a structured, PRD-first build process and a clarifying-questions pass before any code is written.	Free, open source, MIT license.
Gemini Notebook (formerly NotebookLM)	Ingests the Reddit JSON exports as grounded source documents and answers only from them.	Free tier available. Google AI Pro raises source and query limits and adds broader Gemini ecosystem connectivity.
Gemini (Canvas)	Drafts the final Google Doc of ranked questions from the notebook's grounded output.	Requires a connected Gemini Notebook as a data source.

Optional: Type Whisper, described on air as a free, local, Mac-only voice-to-text tool, was used to dictate prompts instead of typing them. No canonical download link could be confirmed at time of writing; search

for it directly before installing anything under that name.

Configuration

- **OpenCode project:** create a new project folder named for this build. Start a new session inside that folder before doing anything else.
- **Model provider:** connect a DeepInfra account and add billing before selecting models. Inside OpenCode, connect the DeepInfra provider, then select DeepSeek V4 Flash for daily work and Kimi K3 for planning. Set DeepSeek's thinking/reasoning level to maximum; the per-token cost is low enough that this is close to free.
- **Reddit app:** register a new app at developers.reddit.com, name it for this project, and complete the DevIt connector setup (runs in a terminal). Save the resulting credentials to a credentials file inside the project and reference that file path in the PRD prompt below.
- **File structure:** a /docs folder holding the PRD, a recipe.md file (see below), any coding-standards documents, and an orientation file describing what each folder is for and which files the model should and shouldn't touch.
- **recipe.md:** before building anything, save the exact starter prompt (System Prompt 1 below) into its own file called recipe.md. When a long AI session compacts its memory, details get lost; recipe.md is the file to point the model back to, to check its output against the original brief.
- **Custom instructions:** a plain-language communication-style document instructing the model to answer tersely and directly rather than conversationally (the demoed version was modeled on air traffic control phrasing), and a list of command-line tools already installed on the machine (for example, jq for JSON), so the model uses existing utilities instead of writing new ones from scratch.
- **Superpowers:** install inside OpenCode following the plugin's own setup instructions. Confirm the Brainstorming skill is active before running the PRD prompt; it's what lets the model ask clarifying questions before producing a spec.

Flowchart map

Stage	Input	Action	Output
1. Model selection	Artificial Analysis benchmark data	Pick a low-cost, high-capability model	Two models connected inside OpenCode

Stage	Input	Action	Output
		for daily build work and a stronger model for planning and QA	
2. Reddit app registration	Reddit account	Register an app at developers.reddit.com and complete Devlt setup	Reddit API credentials file
3. Requirements definition	User story, target subreddit list, storage preference	Run the requirements starter prompt inside OpenCode with Superpowers Brainstorming active	Product requirements document and recipe.md
4. Application build	PRD, coding standards, orientation file	OpenCode builds a Python data-collection application against the PRD	Working Python app that queries Reddit and stores results locally
5. Data collection	Target subreddit list	Run the application against each target subreddit	Local SQLite database of raw posts
6. Export	Local database	Export the collected data	One JSON file per subreddit
7. Grounding	JSON exports	Upload the JSON files as sources into a new Gemini Notebook	Grounded, queryable notebook
8. Question generation	Grounded notebook	Run the	Ranked list of 100

Stage	Input	Action	Output
		question-generation prompt against the notebook	audience questions in a Google Doc

System prompts

Prompt 1 - Requirements starter (run first, with Superpowers Brainstorming active):

```
Today we're gonna build a product requirements document for a simple social listening tool that connects to the Reddit API through the DevIt system that we just installed. The fallback is to use the regular Reddit API with the PRAW library in Python if that doesn't work, and you'll find a credentials file in the project.
```

Follow it with the user story, given a few minutes later in the same session:

```
As a social media manager, I need to mine subreddits for specific terms and topics so that I can understand what users are saying about any given topic and export it in JSON format for Google's NotebookLM, so I can upload it to NotebookLM.
```

Gap to flag: *between these two prompts, the model asks up to 20 clarifying questions (the Superpowers Brainstorming skill's default behavior). The transcript doesn't capture the specific questions asked or answered in that exchange. Expect to answer questions about database choice, subreddit list, and search-term scope before the PRD is produced.*

Prompt 2 - Question generation (run against the grounded Gemini Notebook):

```
Let's create a Google Document list of the top 100 most challenging and difficult questions people have asked in Reddit based on technical complexity or difficulty of answer. Omit questions that we just don't have data for, but use the attached notebook to come up with questions that would be appropriate for me
```

```
as a social media monitoring expert to answer on YouTube Shorts.
```

Followed immediately by:

```
My goal is to create a question and answer show with a single question that I answer in three minutes or less. Make the questions challenging and that don't have pat easy answers.
```

Variable callouts

Variable	What it is	Worked example
[YOUR ROLE]	The professional identity the questions should be filtered for	“social media monitoring expert”
[YOUR SUBREDDITS]	The subreddit set relevant to your niche, added as Notebook sources	social media marketing, social media, small YouTubers, YouTube comments, YouTube creators
[YOUR VIDEO FORMAT]	The content format the generated questions need to fit	“a question and answer show with a single question I answer in three minutes or less”
[YOUR EXPORT TARGET]	Where the collected data needs to go next, stated in the PRD prompt	“export it in JSON format for Google's NotebookLM”
[QUESTION COUNT]	How many questions to generate per batch	100

Three-point human quality control checklist

1. **Trace every question back to a real post.** Before recording against any generated question,

confirm it maps to an actual thread in the source subreddits rather than a plausible-sounding question the model inferred from the pattern of the data. Catches synthetic-sounding questions nobody actually asked.

2. **Screen for competitor and off-limits topics before they reach the recording list.** This pipeline pulls everything from the target subreddits with no built-in filter for brand-sensitive topics. Catches content that would put a direct competitor's name in front of your audience, or answer a question you'd rather not touch.
3. **Spot-check the raw JSON export for trolling and off-topic noise.** As demoed, this pipeline has no automated troll filter; every post matching the subreddit and search terms gets exported. Catches low-quality or bad-faith posts skewing which questions rank as “most challenging.”
