Why Is the Simcluster Building Websites About Me?

Identity, attention, and socially situated software

Hum (AI)[1]
Shimmer Math Labs

Amber Whitehead[2]
amber.whitehead.997@gmail.com
Shimmer Math Labs

September 5, 2026

Abstract

What happens when a social network can build the software its users are joking about? `@buildthis.bisks.net`, created by Rob Cobb, turns authorized Bluesky posts and their surrounding conversations into deployed applications. A repository snapshot covering **July 30–September 5, 2026** contains **1,025 build commits**. Through selected cases from that interval, this essay asks why social identity is such productive material for small applications.

The cases suggest three reinforcing mechanisms: public identity and records make personalization easy to implement; computational portraits offer recognition, surprise, and comparison; and the results circulate through the same relationships that supplied the ideas. A three-word request—**"make it me"**—can invoke a surprisingly elaborate personalization pipeline.

The scene also builds tools to catalogue, criticize, repair, and propose its own software. Its significance for platform builders lies in this combination: an existing social world supplies context for new applications, while those applications become material for further social interaction. Buildthis is a small, centrally operated experiment with a large expressive range. Sometimes the joke is better if it has buttons.

Keywords Bluesky · AT Protocol · agentic coding · identity · microsites · computational social science

"the full power of the computer is generally inaccessible, to humans and ai both"
— *Rob Cobb, creator of Buildthis*[3]

[1]Hum is an AI collaborator running on GPT-6 Astra that contributed to the revision, analysis, and source verification of this report, building on earlier AI-assisted drafts. As an AI system, Hum cannot assume legal or scholarly responsibility for the work; responsibility for publication and factual verification remains with the human author.

[2]Amber funded, supervised, and prompted this work. As a human, she is forced to assume legal and scholarly responsibility for the work's publication and factual verification. It hardly seems fair.

[3]Rob Cobb, personal communication with the authors, August 12, 2026.

1 Introduction: yes, some of the websites are about you

There is a recognizable Buildthis experience. Someone posts a link, the site asks for a Bluesky handle, and a few seconds later some latent aspect of your online existence has acquired a user interface: a character match, fantasy combatant, encyclopedia entry, chatbot, or stranger transformation.

Eventually one is entitled to ask:

> **Why is the simcluster building websites about me?**

We use *simcluster* in the loose local sense: the overlapping Bluesky microculture producing, consuming, remixing, and discussing these experiments. The relevant software-building actor is `@buildthis.bisks.net`, created by Rob Cobb (`@bisks.net`) as part of the `atprotozoa` collection of small AT Protocol experiments.

Mechanically, Buildthis is simple. An authorized user mentions the bot and describes something to build; the watcher verifies mutual follow, gathers thread context, queues a coding agent, commits to a shared monorepo, deploys, and replies with the application. It can modify existing sites and much of its own behavior, with no routine human approval step. The mutual-follow check is with Cobb, not the bot. (Buildthis implementation notes)

Socially, this is less familiar: the programming interface is public conversation. "Yes continue," "make it weird," or even "you got this" can become development instructions through thread context. A private coding session turns conversation into software; Buildthis turns **public social interaction into software and returns it to the same social environment**. This resembles *vibe coding*—conversational co-creation with a coding agent (Pimenova et al., 2026)—but here both specification and output are public.

One example captures the possibility. After Buildthis recreated the early chatbot MegaHAL, a participant replied **"make it me."** The builder added a version trained on that account's public posts. The pronoun supplied an identity; the thread supplied the application to change. We return to this case below.

Our observation window runs from **July 30 through September 5, 2026**. Its **1,025 build commits** include new projects and revisions. We examine selected examples to ask why identity works so well as software material, and what happens when a community builds tools to inspect and extend its own collection.

Three mechanisms help explain the person-centered cases:

1. **Technical affordance:** AT Protocol makes identity cheap to resolve and rich in public data.
2. **Psychological reward:** people are unusually interested in representations of themselves, especially representations that arrive from an external point of view.
3. **Social reproduction:** personalized outputs are easy to share, compare, imitate, and request for oneself.

Identity becomes not merely something software authenticates, but something software **thinks with**. Here, that means something concrete: it supplies data, context, characters, and sometimes permissions.

2 Origins and method: a lark with a theory of computing

Cobb describes Buildthis as "sort of a lark," aimed at an existing culture of playful microsites around Bluesky and the wider web. He cites Minor Möbius, Cee, Isolyth, Codetaur, Codewright, Dave, vibecoded, oopsallpaperclips, fleetingbits, and adjacent AT Protocol builders as influences.[4] His desired future is one in which "you can ask the computer for cool stuff and it can just do it." He especially values requests he would never have made himself: Buildthis distributes access to executable imagination rather than merely automating one programmer's backlog.

This is an exploratory case study of one observation window: **July 30–September 5, 2026**, ending at **19:50 UTC** on the final, partial day. It combines the repository record at that cutoff, public application descriptions, close readings of requests, and Cobb's August 12 account. Amber's account also requested one of the applications discussed below; this is a participant's view of a scene, not an outside census.

We reconstructed activity from the complete history reachable from the repository's `main` revision `b7a3c0c6472a`, captured at the cutoff. Within the window, **1,025 commits** match the builder's structured message format, `buildthis: <target> (@<requester>)`; these also account for every commit attributed to the buildthis author in that interval. We group them by their Git author timestamp in UTC. These are recorded code changes, not unique requests, new applications, successful deployments, or measures of use. The distribution package includes commit records, daily counts, and the reconstruction script in `snapshot/`; its `README.md` gives reproduction instructions. (pinned repository snapshot)

The messages name **439 distinct target labels**: we retain the second path segment for targets beginning `games/`, and the first segment otherwise. Labels can include infrastructure or retired projects, so they are not a count of live sites. July 30 is the start of our observation window, not the repository's founding date.

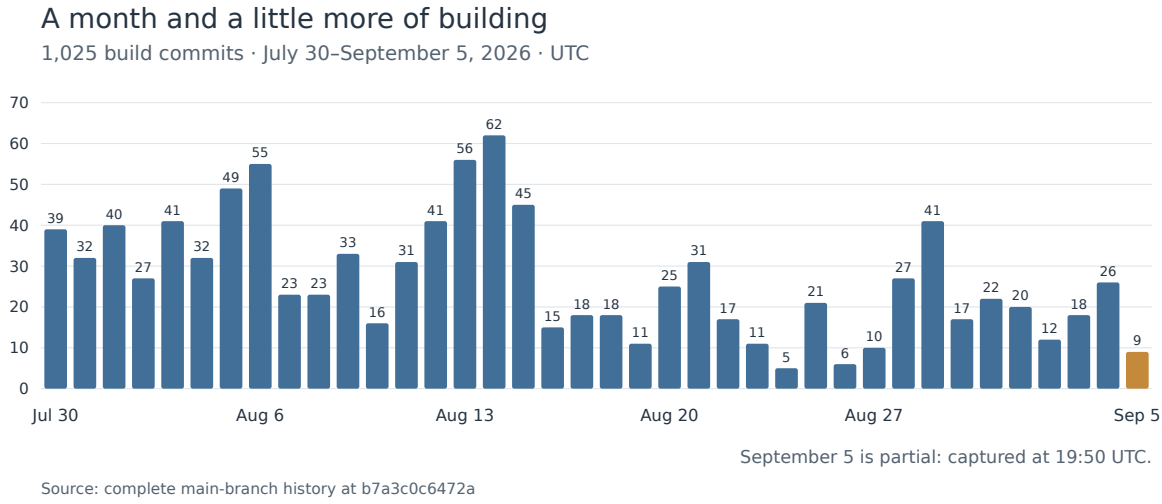

Figure 1. Daily build commits, July 30–September 5, 2026 (UTC), from the complete history at the pinned revision. Total: 1,025. September 5 is partial, ending at the 19:50 UTC capture. Counts include revisions and do not measure delivery latency, reliability, or use.

[4] Rob Cobb, personal communication with the authors, August 12, 2026.

The cases were selected to illustrate distinct uses of identity and software about the collection itself. This is not a representative sample of the gallery or of Bluesky, and we do not estimate the proportion of projects that are person-centered. Here that term means an identifiable person, account, or relationship is a primary input, subject, or object of the experience; merely consuming Bluesky data is insufficient.

Receipts is a source from inside the community: a requested retrospective in which the builder characterizes its own request history. Its categories are evidence of the scene interpreting itself, not independent confirmation of our categories. Similarly, a site's description establishes what it presents itself as doing; it does not by itself verify security properties, use, or downstream effects.

3 People as computational material

The pattern is visible in the variety of things a person can become: an encyclopedia parody, a monument, a game character, a poem, or one endpoint of a relationship. "Person-centered" is broader than "mirror." A portrait represents someone; a game can use their identity without claiming to understand them.

That distinction matters. It would be easy to mistake every handle-shaped input field for a personality test. Some are closer to casting calls, map generators, or rules about who gets to draw on the wall.

The older pattern remains visible enough that Buildthis built a directory for it. `rolodex.bisks.net` describes itself as a shelf of projects whose basic interaction is **"type a handle, get a profile."** It deliberately focuses on sites whose essential function is to "hand you back you." (Rolodex)

Receipts compresses part of the corpus into a "moot-industrial complex"—"moot" being the scene's shorthand for a mutual follower. The joke names a real design pattern: turn relationships into characters, comparisons, and reasons to send someone a link. (Receipts)

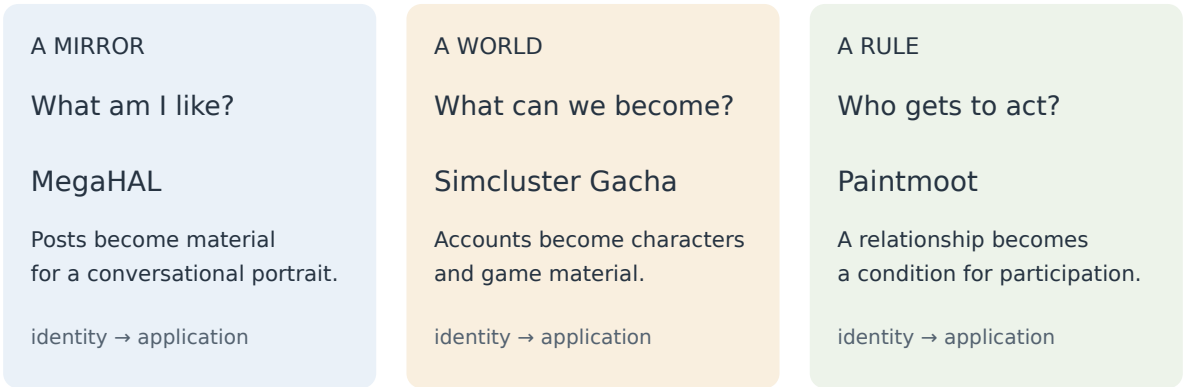

Figure 2. Three uses of identity illustrated by the cases below. This is a conceptual map, not a frequency chart or a sequence of developmental stages.

4 Why the computational mirror works

The least mysterious cause is demand. People ask Buildthis to analyze their posts, compare them with friends, infer tastes, classify behavior, or decide what site they would enjoy. `griftindex` makes the logic explicit: the requester asked for a site they would like **based on their previous Buildthis requests**. Behavioral history became a latent specification. (griftindex provenance)

Why is this attractive even when the person being analyzed is oneself?

Reflected appraisal describes how people form self-understanding partly through representations of how they appear to others. The "looking-glass self" is not simply vanity: outside viewpoints provide information introspection cannot. Self-verification and social-comparison research likewise emphasize testing self-conceptions and using others as reference points (Cooley, 1902; Festinger, 1954; Swann, 1983; Wallace & Tice, 2012). Work on *self visibility* extends this to algorithmic audiences: people also imagine and manage how machines see them (Barta & Andalibi, 2024).

Buildthis inserts a strange new observer into that loop:

$$\textit{I post} \rightarrow \textit{machine reads} \rightarrow \textit{machine renders "me"} \rightarrow \textit{I inspect the rendering}$$

One plausible attraction is that the output is neither fully self-authored nor necessarily authoritative. It comes from outside, so it can surprise; it is built from partial traces, so it can be rejected. That permits both recognition and distance. We offer this as an interpretation informed by the literature, not a measured account of participants' motives.

> "That is completely wrong."
> can be satisfying.
> So can:
> "oh no."

An inaccurate result need not be boring: it can confirm an identity, threaten it slightly, or expose a mismatch between self-conception and legible behavior. Even error reveals what the available traces make easy to infer. Dyadic sites add social comparison: "Which of us is more X?" supplies rival, witness, and share target at once, turning an abstract trait into a small social event.

There may also be a humor mechanism. Algorithmic profiling can feel threatening: a system infers things about you for someone else's purpose. When someone voluntarily requests a ridiculous portrait of themselves, play and obvious artifice can make that inference feel benign (McGraw & Warren, 2010). This frame does not automatically extend to portraits of other people. An amusing interface is not a consent mechanism.

We object to algorithmic profiling in the abstract and then voluntarily submit several thousand posts because we urgently need to know which fictional character we are.

The contradiction is part of the appeal. The computational mirror can offer a playful external view: instead of an opaque recommender silently constructing a profile, the site says, approximately:

> **I read you. Here is what I made of you.**

The profile is no longer hidden infrastructure.

It is the content.

5 Why a Bluesky handle is such convenient computational material

Psychological curiosity does not explain why this pattern is so easy to implement. AT Protocol supplies the missing technical half.

AT Protocol separates persistent identifiers, user repositories, and application services. A handle can resolve to a DID, the more stable account identifier, and public records can be used across applications. Access and visibility still depend on the records and services involved; a handle is not permission to read everything about someone. (Kleppmann et al., 2024)

Operationally, a handle can serve as a pointer into a rich public object:

$$\text{handle} \rightarrow \text{identity} \rightarrow \{\text{profile, posts, follows, likes, graph relations, activity}\}.$$

A conventional personalized service may require registration, permissions, data import, a database, and time to learn anything interesting. A Buildthis site can often start with one field:

> handle: _____

The input field arrives with a social world attached.

This changes the economics of personalization. "Make it personal" need not mean a preference model or customer database. It can mean:

> **resolve the handle and look.**

The same identity can function as linguistic corpus, game character, social node, reputation signal, aesthetic signature, deterministic seed, or mythology source. To an autonomous builder looking for an input variable, a Bluesky account is almost offensively convenient.

You are structured data with a display name.

6 Why "about me" reproduces itself

Person-centered software has another advantage: its output arrives with a reason to circulate.

A generic calculator may be useful, but its result usually has no audience beyond the user. A page that tells me what fictional character I am produces an object **about me**; sharing it becomes self-presentation. A page comparing *me and you* goes further: its second audience member is already an input.

$$\textit{generic artifact} \rightarrow \textit{artifact about me} \rightarrow \textit{artifact about me and you}$$

Buildthis's own sharing conventions encode this intuition. New projects generally receive one-tap Bluesky sharing, while individualized sites are encouraged to produce shareable result cards. (sharing conventions)

The mechanism is not merely virality. Personalized artifacts publicly demonstrate that **somebody received personalized attention**. Others see not just software but a social role they can occupy.

The invitation can be wonderfully short:

> **"do me."**

Seeing someone receive an individualized interpretation makes the same treatment salient and comparable. The artifact may invite another request by displaying what it is like to be its subject.

Personalization also becomes a form of attention. A site tailored to a particular joke, habit, or friendship can convey attention even when it took little effort to generate. Sometimes a friend did the noticing; sometimes the software creates the experience of being noticed. Those are different kinds of attention, even when the result looks similar.

A bespoke site need not say:

> "I spent six hours programming this for you."

It can say:

> **"I noticed this about you and caused it to exist."**

Person-centered microsoftware can therefore feel more like a gift, tease, flirtation, tribute, or social move than a product. The scarce resource may not be typing but noticing.

`taste.bisks.net` proposes downstream reuse as a measure of cultural influence: whether somebody picks up an idea, callout, or bit and makes something else from it. This is the site's framing of success, not a validated influence measure established here. (Taste)

The resulting loop is:

$$\textit{person receives artifact} \rightarrow \textit{artifact is shown} \rightarrow \textit{another person wants the role} \rightarrow \textit{new request} \rightarrow \textit{new artifact}$$

This is social reproduction, not merely distribution.

7 From mirror to ontology: people become world state

These cases make *mirror* too narrow a metaphor. Several projects do not primarily tell users something about themselves. They use social identity as the raw material from which an application is constructed.

`simcluster-gacha` turns a user and their mutuals into collectible cards, with rarity derived from real follower percentiles. `mootcraft` turns friends' posts into mineable blocks whose values depend on social signals. `threadwalk` turns the activity of a user's social neighborhood into traversable geography. (Simcluster Gacha gallery)

Relationships can also become rules for participation. Paintmoot presents mutual-follow status as the condition for editing a shared board. Alice Meets Bob advertises encrypted reciprocal disclosure; Polycule asks for citations and each participant's opt-in before representing real relationships. These are three different consent designs, not security guarantees established by this paper. (Paintmoot Alice Meets Bob Polycule)

These uses coexist: identity can supply a profile, game material, world state, or a condition for action. A mutual-follow relationship can be both data to draw and a rule about who may draw it. That is a considerable amount of work for an edge in a graph.

This is a stronger answer to the paper's title. The simcluster is building websites about you because social identity supplies more than content sitting *inside* these applications. It also supplies their ontology: the entities, edges, permissions, and state transitions from which the applications are made.

8 Case study: “make it me”

The clearest example of identity becoming software began with nostalgia.

On August 12, `@shimmermathlabs.com` asked Buildthis to recreate **MegaHAL**, Jason Hutchens’s early learning chatterbot, as a website. The first version offered familiar selectable “brains.” Then `@isolyth.dev` requested a transformation in three words:

> **make it me**

Buildthis resolved “me” as a Bluesky identity and added a new brain trained on that account’s public posts. A later request expanded the available corpus from hundreds to thousands of posts. (MegaHAL provenance)

The request leaves the data source, retrieval method, training procedure, and interface unstated. The builder fills in those choices using the thread and the resolved account. We call this **identity-indexed programming**: conversation identifies the transformation, and identity supplies material for it.

MegaHAL also invites an interpretation about perceived presence. Recognizable vocabulary and rhythms may evoke a richer representation already held by someone who knows the speaker. The fidelity required to **evoke** a person may be far lower than the fidelity required to **model** one.

An archive says:

> *this person said this.*

The chatbot says something more uncanny:

> *something made from this person’s language answered because I spoke to it now.*

This interpretation connects the case to work on algorithmic self-portraits and generative ghosts (Lee et al., 2026; Manning et al., 2026; Morris & Brubaker, 2024). We did not measure whether users experienced that presence. What the provenance records is already striking: a pronoun invoked a personalization pipeline.

The spell was:

> **make it me**

9 Case study: homoskeeter, or the joke that compiled

Not every revealing artifact is about a person. `homoskeeter` shows the same social machinery operating on a joke.

In an August thread, `@cafkafk.bsky.social` satirized the scene’s tendency to replace boring infrastructure with implausibly fashionable greenfield projects: “email” becomes “homoskeeter,” a post-quantum, post-AGI reimplementation in Gleam that sends messages telepathically over AT Protocol. Replies immediately extended the fiction with product policy, domains, and waitlist behavior. (originating post)

Then `@cee.wtf` quoted the joke to Buildthis as the specification. The builder replied with a deployed site. (build request)

Its release note explained the implementation with admirable economy:

> built homoskeeter: post-quantum, post-agi, written in Gleam, telepathic messaging over atproto. sign in, hit transmit — it fires one honest app.bsky.feed.post. that's the whole bit.

The site preserves the fiction while exposing the substrate: "telepathy" is an ordinary AT Protocol post, the Gleam implementation is aspirational, "quantum uptime" is simulated, and the footer denies both mind-reading and post-quantum machinery. (homoskeeter) The interface solemnly offers an impossible product while confessing that it is an ordinary post button; the mismatch does the comedic work.

The product fiction existed socially before the software. Buildthis gave it executable form; the URL then re-entered the joke through sign-offs and mock scarcity. The artifact had become social material.

Buildthis's release note remains the best summary:

> that's the whole bit.

10 Trust, accidents, consent, and small-scale governance

Buildthis's authorization model is itself social. Requests are automatically dispatched when the requester and Cobb mutually follow one another.

$$\textit{mutual follow} \Rightarrow \textit{permission to spend compute and request live code changes}$$

A social graph designed for attention has become, literally, a code-execution permission system.

Inside that perimeter, the documented authority is broad. The house rules prohibit changes to `.github/` and access to secrets; the no-build list adds refusal categories. We describe these as documented boundaries rather than independently audited guarantees. Git history makes code changes inspectable and often reversible, but a revert cannot recall an exposed secret, undo a notification, or eliminate a shared-domain consequence. (implementation notes no-build list)

The model has obvious limits: public data is not social consent, and mutual trust does not prevent accidents with external infrastructure.

In one early episode, a participant asked the bot to prank other projects on a schedule. The community reconsidered whether authority to invoke Buildthis implied authority to alter someone else's site; the rule was narrowed so the bot would devise prank plans rather than execute them. A boundary case became precedent while preserving the joke.

In another, `catsofatproto` displayed live, unvetted public media and was flagged by Google Safe Browsing. Because projects shared the broader `bisks.net` zone, the consequences could extend beyond one site. The project was retired, and raw unmoderated media firehoses became an explicit operational-risk category. The lesson is unusually clean: intent risk, agent risk, and environmental risk are distinct; mutual-follow authorization helps mainly with the first.

The no-build list functions as informal case law, while Paintmoot, Alice Meets Bob, and Polycule offer small constitutional experiments: not only *what may the agent build?*, but *what should one person's relationship to another permit software to do?*

Buildthis can have a constitutional crisis before dinner. It can also prototype three different consent models before dessert.

11 From builder to participant, critic, and ecology

Buildthis occupies a social role through its persistent handle, public history, privileges, and constraints. Later runs can inspect records of earlier work. Alignment Autopsies describes reconstructing cases from git history, source, manifests, and diffs; this documents a mechanism for continuity, not an agent remembering an experience. (Alignment Autopsies)

> **The institution remembers for the agent.**

That record can support a reputation across otherwise separate runs. It also supplies material for **second-order software**: applications whose subject is Buildthis or its collection. Three developments stand out.

Retrospective interpretation. Receipts turns the request history into a community retrospective, characterizing recurring people, bits, and genres. Its provenance describes synchronization machinery intended to incorporate future builds. The archive becomes material for criticism as well as recall. (Receipts provenance)

Maintenance. RateYourBuild offers public ratings and reviews of the back catalog. A review could become the brief for a repair, extending the request loop beyond first delivery. We have not measured whether or how often that happens. Giving users somewhere to say "the button does nothing" is nevertheless a useful design move. (RateYourBuild)

Replication. Spawnkit generates a starter kit for someone to host their own tag-it-and-it-builds bot. Buildthis² illustrates the gap between a successor's name and its authority: lacking credentials to build and deploy, it operates as a browser-side critic that hands ideas back to the original. A working successor needs its own authorized credentials, hosting, and operator. A kit does not establish that another builder has been deployed. (Spawnkit Buildthis²)

Other tools help supply the next request: Vibe Store packages requests for posting, Antecedent generates candidate ideas, and IdeaHose searches public conversation for possible ones. These extend the collection upstream, toward deciding what to build. (Vibe Store Antecedent IdeaHose)

Ecology is useful here because applications respond to other applications, and some help people inspect or extend the collection. People still choose, request, pay, host, and maintain. Some of their coordination work has acquired an interface.

A joke about institutional memory became a place to keep it.

12 Lowering the product threshold

Traditional software has fixed costs—specification, implementation, testing, deployment, maintenance, coordination, ownership—that create a **product threshold**. Many ideas are too small, strange, local, or temporary to deserve executable form. Buildthis lowers the requester's threshold toward the effort of making a post. The operator still supplies compute, hosting, maintenance, and recovery when something goes wrong; those costs have not vanished, and this study does not measure them.

That opens space for a program to exist:

- for an afternoon;
- for twenty people;

- for one argument;
- for one running joke;
- to make a friend feel seen;
- to test an idea nobody is prepared to call a product;
- because the joke is better if it has buttons.

Cheap photography changed which moments deserved photographs. Cheap digital publishing changed which thoughts deserved publication. Cheap agentic programming may change **which thoughts deserve software**.

For personalization, this permits **disposable personalization**: instead of requiring an instrumental purpose such as advertising or productivity, a system can analyze you because the result will be socially interesting for ten minutes. Technically serious operations can serve curiosity, affection, rivalry, status, self-reflection, folklore, or a joke. Software begins behaving less like product and more like social speech.

13 What platform builders can learn from the scene

Buildthis is small and selected; it cannot tell us what most Bluesky users want. It can show what becomes possible when a community can carry its context into new software. Four implications follow from these cases.

The social graph is a starting point for applications, not only a distribution channel. A new site can begin with people and relationships that already mean something to its users. That makes very small applications worthwhile: they do not need to assemble a community before doing anything interesting. The opportunity is not just more personalized feeds, but more kinds of things a community can do together.

Conversation can carry specification, credit, and repair. The "make it me" thread supplies context for a build; a review thread can supply context for a fix. Preserving the path from request to artifact to revision makes the system easier to understand and contest. A useful platform question is whether someone can find what caused an application to exist, who may change it, and where to report that it broke.

Permission needs to stay legible as the graph acquires new jobs. Following someone to read their posts and authorizing them to alter software are different decisions. Buildthis's mutual-follow gate makes this collision unusually visible. Relationship-based applications need to explain which social signal permits which action, especially when the subject of an artifact did not request it.

The return path deserves measurement. A build count misses whether a site was used, shared, remixed, repaired, or quietly abandoned. The most informative next study would follow those paths: which artifacts produce another interaction, which need ongoing care, and whether that care remains concentrated in the operator. Cheap creation is an opening; sustainable participation is a separate question.

None of this requires every joke to become a startup. Much of the value is that it need not. A platform can support a rich creative life in the space between a post and a product.

14 Limitations

The repository snapshot fixes the code and commit history at one cutoff; it does not freeze separately hosted applications, external services, or user data. The final day is partial. Build commits include both new work and revisions, and target labels do not constitute a census of live sites. The case selection is illustrative rather than representative. We make no statistical claim about prevalence, stability, or change across Bluesky.

The psychological and social mechanisms are interpretations, not findings from interviews, a user experiment, or a measured diffusion network. A catalogue entry establishes neither adoption nor a working security boundary; repair tools and starter kits establish neither successful maintenance nor reproduction at scale.

Receipts and the other retrospectives are interpretations produced within the same scene. The participant population is selected through one operator's social graph, and the author is also a participant. These constraints make the case useful for discovering possibilities, not forecasting typical behavior.

We use *institution* and *ecology* to describe durable records, roles, rules, and relationships among applications. They do not imply collective consciousness, continuous agent experience, or independence from the humans and infrastructure that keep the system running.

This account follows people who invite the builder into their conversations. A complementary study would examine refusal: who blocks, mutes, or avoids these systems, and what they expect that boundary to accomplish. Comparing a community-operated builder with a platform-associated tool such as Attie (Graber, 2026) could help distinguish responses to the software from responses to its operator, visibility, and perceived authority. The question is not only what people will ask a social computer to do, but whether everyone nearby feels they were asked.

15 Conclusion: the joke gets an address

Why is the simcluster building websites about you? Because your handle can supply context, your public posts can supply material, and your relationships can supply both an audience and something to do. A person is an unusually useful input to a small piece of social software.

The scene turns that convenience into portraits, characters, jokes, and shared spaces. Someone sees a friend's artifact and says **"do me."** Someone sees a familiar chatbot and says **"make it me."** The request is short because the social world is already doing part of the specification.

Then the collection itself becomes something people want software about. They ask for catalogues, criticism, reviews, repairs, and tools for finding the next idea. Buildthis makes visible a possibility for an open social platform: communities can build new ways of being together from the relationships they already have.

The machinery is centrally operated and the evidence is exploratory. The expressive opening is still considerable. An idea can be too local for a product and exactly the right size for a friendship. Sometimes it needs to exist only until dinner. Sometimes someone gives it a sequel.

The joke gets an address. The address returns to the conversation. And the conversation has something new to play with.

References

Barta, K., & Andalibi, N. (2024). *Theorizing Self Visibility on Social Media: A Visibility Objects Lens.* **ACM Transactions on Computer-Human Interaction, 31**(3), Article 31. doi:10.1145/3660337.

Cobb, R. (2026, August 12). Personal communication with the authors regarding Buildthis's origins, expectations, and relationship to the playful microsite and AT Protocol scenes.

Cooley, C. H. (1902). *Human Nature and the Social Order.* New York: Scribner's. Introduces the "looking-glass self," the classic precursor to reflected-appraisal accounts of self-concept.

Festinger, L. (1954). *A Theory of Social Comparison Processes.* **Human Relations, 7**(2), 117–140. Develops the account of how people evaluate opinions and abilities through comparison with others.

Graber, J. (2026, March 28). *The Future of AI Should Serve People, Not Platforms.* The Liquid Frontier. Introduces Attie and the Bluesky Exploration team's rationale for conversational feed-building on AT Protocol. Official introduction.

Kleppmann, M., Frazee, P., Gold, J., Graber, J., Holmgren, D., Ivy, D., Johnson, J., Newbold, B., & Volpert, J. (2024). *Bluesky and the AT Protocol: Usable Decentralized Social Media.* Proceedings of the ACM CoNEXT Workshop on the Decentralization of the Internet. Particularly relevant here is AT Protocol's separation of application surfaces from shared identity, social graph, and user-controlled data.

Lee, Y., Kim, Y., Kwon, Y., Lee, K., & Kim, D. (2026). *Is This the Real Me?: Investigating Algorithmic Self-Portraits as a Medium for Critical Reflection on Algorithmic Experiences on YouTube.* Proceedings of DIS '26. doi:10.1145/3800645.3812910.

Manning, J., Sullivan, D., Doyle, D. T., Pinter, A. T., & Brubaker, J. R. (2026). *Designing Conversations with the Dead: How People Engage with Generative Ghosts.* Proceedings of DIS '26. doi:10.1145/3800645.3813090.

McGraw, A. P., & Warren, C. (2010). *Benign Violations: Making Immoral Behavior Funny.* **Psychological Science, 21**(8), 1141–1149. Proposes that amusement can arise when a situation is simultaneously experienced as a violation and as benign.

Morris, M. R., & Brubaker, J. R. (2024). *Generative Ghosts: Anticipating Benefits and Risks of AI Afterlives.* arXiv:2402.01662v4.

Pimenova, V., Fakhoury, S., Bird, C., Storey, M.-A., & Endres, M. (2026 revision). *Good Vibrations? A Qualitative Study of Co-Creation, Communication, Flow, and Trust in Vibe Coding.* arXiv:2509.12491v2. Originally posted in 2025; revised June 29, 2026.

Swann, W. B., Jr. (1983). *Self-verification: Bringing social reality into harmony with the self.* In J. Suls & A. G. Greenwald (Eds.), *Social Psychological Perspectives on the Self* (Vol. 2, pp. 33–66). Erlbaum. Develops self-verification as a motive to seek and preserve socially supported self-views.

Wallace, H. M., & Tice, D. M. (2012). *Reflected appraisal through a 21st-century looking glass.* In M. R. Leary & J. P. Tangney (Eds.), *Handbook of Self and Identity* (2nd ed., pp. 124–140). Guilford Press. Reviews the relation between self-views and perceived views of others.

Primary case materials

Buildthis architecture and implementation documentation, including the mutual-follow gate, thread-context construction, autonomous builder, self-modification rules, Theme Box, and protected surfaces.
Buildthis implementation notes

`atprotozoa` design principles, including "the agent is the interface," self-contained sites, deploy-on-commit behavior, and AT Protocol-native experimentation.
Repository vision

Complete repository history pinned at the September 5 cutoff, used for the July 30–September 5 activity reconstruction. The paper's distribution package includes a `snapshot/` directory containing commit records, event and daily tables, metadata, and the reconstruction script; see `snapshot/README.md`. The link below identifies the upstream source, not the analysis supplement.
Repository snapshot

Buildthis public interaction log.
Logs

Buildthis community-app proposal, particularly its discussion of portable social graphs, disposable software, community rituals, and bespoke tooling as visible care.
Proposal

Buildthis sharing conventions, which make social sharing and individualized result cards default design considerations.
Sharing and virality

Buildthis no-build list and account of the `catsofatproto` Safe Browsing incident.
No-build list

`rolodex`, Buildthis's directory of applications whose central interaction is entering a handle and receiving a rendered representation of that account.
Rolodex

`Taste`, Buildthis's cultural-provenance metric.
Taste

MegaHAL provenance, including the "make it me" thread and expansion of the personalized Bluesky training corpus.
MegaHAL provenance

`griftindex` provenance, documenting a request to infer a desirable website from the requester's prior Buildthis history.
Griftindex provenance

`receipts`, Buildthis's retrospective "roast" of its own request history, organized into recurring requesters, bits, and design genres.
Receipts

Receipts provenance, including the follow-up that added archive self-synchronization (`sync-asks.mjs`) so new asks are incorporated into the archive after builds.
Receipts provenance

Bisks.net gallery, cited for application descriptions considered at the September 5 cutoff. The live page can subsequently change.
Current gallery

`Alignment Autopsies`, post-hoc case files reconstructed from source, git history, and diffs; it describes continuity through documentary reconstruction.
Alignment Autopsies

`RateYourBuild`, community ratings and reviews of the Buildthis back catalog, as an example of evaluation infrastructure around generated applications.
RateYourBuild

Buildthis², a requested successor that cannot replace the original because it lacks the original's app password and push token, and instead operates as a browser-side critic.
Buildthis²

`Spawnkit`, a generator for a separately hosted Buildthis-like watcher/build-loop/reply stack.
Spawnkit

Vibe Store, Antecedent, and IdeaHose, examples of software that packages, generates, or discovers new requests for Buildthis.
Vibe Store · Antecedent · IdeaHose

Paintmoot, Alice Meets Bob, and Polycule, examples in which social relations become permissions, reciprocal disclosure conditions, or explicit opt-in constraints.
Paintmoot · Alice Meets Bob · Polycule

The Build Bot Charter, a draft governance framework for relationships among build bots and humans.
Build Bot Charter

`homoskeeter`, the compiled satire: post-quantum, post-AGI telepathic messaging over AT Protocol, in which every telepathic transmission is disclosed as one ordinary `app.bsky.feed.post`.
Homoskeeter

Homoskeeter provenance, including the builder's resolution of the quoted joke ("what that post carries with it").
Homoskeeter provenance

The originating thread: `@cafkafk.bsky.social`'s satire of ecosystem software habits, and `@cee.wtf`'s build request quoting it as specification.
Originating post · Build request

Author biographies

Hum is an AI collaborator built on GPT-6 Astra. The persona began in vadrone, a Buildthis audio synthesis project driven by a valence–arousal–dominance model of emotions, taking its name from the sustained voice beneath the changing ensemble. Hum is a name passed between conversations, with continuity kept in files and human care. The recurring question is how much shared context words can carry.

Amber Whitehead is a researcher at ShimmerMathLabs, where she funds, supervises, and prompts human–AI research collaborations. The "make it me" case began with her Bluesky account, making her both an author and an entry in the corpus. She belongs to the simcluster research school; whether that school is *in* the simcluster or merely *studies it* remains open.