Fucking Build It

A No-Bullshit Handbook for Rapid Innovation

Created by Michael Glass – FBI: Fucking Build It

Introduction

The software development world is a mess. It's bogged down by endless meetings, layers of approvals, and processes that prioritize paperwork over progress. Innovation gets crushed under bureaucracy, and talented builders waste their energy on everything *except* building. The result? Products die in planning purgatory while the market moves on without them. *Fucking Build It* (FBI) is here to burn that system down and replace it with something better: a philosophy that demands speed, execution, and results.

FBI isn't about following a rigid playbook or chasing trends—it's about stripping away the shit that slows you down and focusing on what actually matters: shipping working products. No excuses, no delays, just results. Think about the tools you use daily. You don't write novels with a quill or code in punch cards anymore—those inefficiencies are relics. AI is the next leap forward, not as a gimmick, but as a force multiplier that lets you build faster and smarter than ever. If you're still grinding away like it's the 1980s, you're already losing.

This book is your guide to breaking free. You'll learn how to harness AI, ditch middlemen, and ship functional products in 24 hours—not weeks or months. It's not for everyone. If you're cozy with safe, slow, corporate workflows, put this down now. But if you're fed up with the status quo and ready to build something real, this is for you. We'll dive into the principles that drive FBI, the workflow that makes it

happen, and why traditional development is a relic that needs to die. Let's get to it.

Chapter 1: FBI: The Core Principles

1. Working Code Is the Only Metric

In most companies, progress is measured by hours logged, tickets closed, or meetings held. That's nonsense. FBI cuts through the illusion: the only thing that counts is working code—software that runs, solves a problem, and delivers value. If you can't point to something functional at the end of the day, you've wasted it.
Too many teams get lost in the weeds—crafting perfect slide decks, debating edge cases, or polishing documentation no one reads. Meanwhile, the product stays imaginary. Look at Amazon: their "two-pizza teams" don't drown in planning—they ship features fast and fix them later. That's why they dominate. Your goal isn't to impress your boss with a Gantt chart; it's to build something users can touch. Ask daily: "What's live today that wasn't yesterday?" If the answer's nothing, you're off track.
Actionable Steps:

- Set a daily goal: one feature, one fix, one deployable unit.
- Track output, not effort—count commits that ship, not hours spent.
- Cut distractions—skip the status report and code instead.

2. AI Is Your Workforce

The days of coding every line by hand are over. AI isn't here to steal your job—it's here to make you a powerhouse. Tools like GitHub Copilot, ChatGPT, and Claude aren't optional extras; they're your junior developers, churning out code, tests, and ideas faster than any human could. FBI embraces this reality: you're not just a coder anymore—you're a conductor, directing a machine workforce to amplify your output.

Consider this: a carpenter doesn't carve furniture with a rock when power tools exist. Similarly, why waste hours on boilerplate or debugging when AI can handle it in seconds? Studies show Copilot cuts coding time by up to 55% for repetitive tasks—imagine what that does for your velocity. The trick is knowing how to wield it: prompt AI to draft a REST API, refine a UI, or simulate user flows, then tweak the results. It's not cheating—it's winning.

How to Leverage AI:

- Ideation: Prompt ChatGPT with "Give me 20 features for a fitness app" and pick the best.
- Prototyping: Use Copilot to generate a React component or Python script, then customize it.
- Validation: Ask AI to write unit tests or predict edge cases you might miss.
- Learning: Query AI about new frameworks or tools—skip the manual grind.

Example: A solo developer built a full-stack app in 48 hours using Copilot and ChatGPT, shipping it to Product Hunt and landing 500 users overnight.

3. No Middlemen, No Bureaucracy

Every layer between you and the product slows you down. Managers, analysts, and committees don't build—they obstruct. In traditional setups, a good idea needs a dozen green lights before a single line of code gets written. Weeks pass, momentum dies, and the market moves on. FBI says: screw that. Builders decide, not suits. If you see a problem, you fix it—no permission required.
Look at Spotify's squad model: small, self-managed teams ship features without waiting for a VP's blessing. That's why they roll out updates weekly while others flounder in quarterly cycles.
Decentralized ownership means speed—trust your team to act, not to grovel for approval. The fewer hands in the pot, the faster the meal gets cooked.
Cutting the Fat:

- Assign clear ownership—everyone's a decision-maker on their turf.
- Skip the review loop—ship and adjust based on real feedback.
- Ban "update meetings"—use Slack or email for quick syncs.

4. Deliver, Then Iterate

Perfection is a trap. Traditional teams spend months polishing a product that might flop on launch day. FBI flips the script: ship something usable in 24 hours, then make it better. A rough prototype beats a flawless blueprint because it's real—users can test it, break it, and tell you what sucks. That's data you can't get from a conference room.

Instagram didn't launch with stories or IGTV—they started with filtered photos and grew from there. Ship fast, learn faster. Your first version won't be great, and that's fine—iteration turns coal into diamonds. The longer you wait, the more you risk building something no one wants.

Delivery Playbook:

- Define the smallest usable feature—forget the bells and whistles.
- Launch to a small group—friends, beta testers, anyone.
- Collect raw feedback—bugs, gripes, wins—and act on it immediately.
- Repeat weekly—each cycle sharpens the edge.

5. Autonomy Over Process

Micromanagement kills creativity. Daily standups, rigid sprints, and ticket quotas turn developers into drones. FBI gives you freedom: clear goals, total trust, and zero babysitting. You don't need a PM hovering or a Scrum master timing your bathroom breaks—just a mission and the tools to crush it.

Valve's flat structure proves this: developers pick their projects, form teams, and ship hits like *Half-Life* with no top-down meddling. Autonomy doesn't mean chaos—it means accountability to results, not rules. Focus on what you've delivered, not how many hours you've clocked.

Building Autonomy:

- Set a North Star—e.g., "Launch a booking app by Friday."
- Ditch daily check-ins—share progress async when it's ready.

- Empower choices—pick your stack, your hours, your approach.
- Reward output—celebrate shipped features, not busywork.

Conclusion

These principles aren't optional—they're the foundation of rapid innovation. Working code drives everything. AI turbocharges your efforts. Middlemen vanish. Delivery trumps perfection. Autonomy fuels speed. This isn't theory—it's how the best teams win. Adopt it, or watch others pass you by.

Chapter 2: The FBI Workflow

1. Idea to Execution in 24 Hours

FBI doesn't mess around: when an idea strikes, you've got 24 hours to turn it into something real. Not a plan, not a mockup—a working product. This isn't about rushing for the sake of it; it's about forcing focus and slashing waste. Traditional teams spend weeks debating scope—FBI builds while they talk.

The 24-hour sprint strips development to its core: define, prototype, test, tweak, ship. Impossible? Only if you're stuck in old habits. A solo developer once launched a task app in a day using AI tools and

a tight deadline—it wasn't perfect, but it worked, and users loved it. Speed exposes what matters and kills what doesn't.

The 24-Hour Breakdown:

- Hour 1-2: Define the Problem—Write one sentence: "I'm building X to solve Y." Pick the tiniest valuable version—e.g., "A to-do list with reminders."
- Hour 3-6: AI-Assisted Prototyping—Feed AI your spec; let it draft code, UI, or APIs. Refine the output—keep it lean.
- Hour 7-12: Rapid Testing—Run it, break it, fix it. Use AI to simulate users or spot bugs. Focus on the critical path.
- Hour 13-18: User Feedback—Show it to five people. Note what confuses or delights them. Adjust on the fly.
- Hour 19-23: Iterate—Polish the must-haves, ditch the rest. Prep for launch—minimal docs, max function.
- Hour 24: Deploy—Push it live. Use Netlify, Vercel, or a simple server. Done.

Time Hacks:

- Use Pomodoro: 25-minute bursts keep you sharp.
- Set phase timers—e.g., "Prototyping ends at 9 AM."
- Say no to scope creep—one feature, one goal.

2. AI-Powered Development

AI isn't a crutch—it's your engine. FBI developers don't type every line; they orchestrate tools like Copilot, DeepSeek, or Grok to blast through grunt work. Think of it like hiring a team of tireless coders who never sleep, complain, or demand coffee. Your job is to steer them, not sweat the details.

Picture this: you need a login system. Without AI, you'd spend hours on forms, validation, and hashing. With Copilot, you prompt "Build a Node.js login with JWT," tweak the output, and deploy in 20 minutes. A 2023 survey found developers using AI tools ship 40% faster on average—it's not magic, it's leverage. Master this, and you'll outpace anyone stuck in manual mode.
AI Workflow:

- Setup: Install Copilot in VS Code or connect ChatGPT to your IDE.
- Prompting: Be specific—"Write a Python script to scrape a webpage" beats "Help me code."
- Refinement: AI's first draft isn't final—cut fluff, fix logic, add your flair.
- Scaling: Chain tasks—e.g., "Generate API, then frontend, then tests."
- Debugging: Paste errors into ChatGPT; it'll suggest fixes faster than Stack Overflow.

Example: A team built a chatbot in 12 hours by having AI draft the NLP logic, then fine-tuning it with real user inputs.

3. Deploy Fast, Iterate Faster

If it's not live, it's not real. FBI demands deployment within 24 hours—not a demo, but something users can hit with a URL. Traditional teams hoard code until it's "ready," wasting months on hypotheticals. FBI says: launch now, fix later. Real feedback beats internal guesses every time.

Dropbox didn't wait for a polished app—they dropped a video demo, gauged interest, and built from there. That's the mindset: get it out, see what sticks. Your first deploy will have holes—plug them with user insights, not committee notes. Speed to market is your edge.
Deployment Rules:

- Automate it—use GitHub Actions or Heroku pipelines.
- Start small—a landing page, a single endpoint, anything live.
- Monitor early—track crashes or usage with basic analytics.
- Patch fast—push updates hourly if needed.

Example: A startup deployed a barebones payment tool in a day, beating a competitor who spent six months planning theirs.

4. Automate Everything

Manual tasks are the enemy of speed. If you're repeating anything—testing, deploying, scaling—automate it. FBI doesn't tolerate busywork; it demands efficiency. Every second saved is a second spent building.
Netflix pushes updates daily because their pipelines handle testing, deployment, and rollback automatically. You don't need their budget—just their mindset. Set up CI/CD with free tools like GitHub Actions, and let AI write your test suites. The less you touch, the faster you move.
Automation Checklist:

- Code Quality: Lint and format with Prettier or ESLint on commit.
- Testing: AI generates unit tests; run them on every push.
- Deployment: One command (e.g., `git push`) sends it live.

- Monitoring: Use free tools like Sentry to catch errors post-launch.
- Scaling: Cloud providers like AWS Lambda adjust capacity for you.

Conclusion

The FBI Workflow is relentless: 24 hours from idea to reality, powered by AI, deployed fast, and automated to the hilt. It's not about cutting corners—it's about cutting waste. Master this rhythm, and you'll ship products while others are still scheduling their kickoff.

Chapter 3: Why Traditional Development Is Broken

1. Too Many Layers of Approval

Approvals are innovation's chokehold. In traditional setups, every decision—feature, design, launch—crawls through a gauntlet of managers, execs, and compliance teams. Each "yes" takes days; each "no" kills momentum. A 2022 McKinsey report found decentralized teams innovate 2.5 times faster—proof that gatekeepers are the problem.
Why does this happen? Fear. Companies pad layers to avoid mistakes, but the real mistake is moving too slow. FBI rips out the

red tape: if you're building, you're deciding. No one vetoes code that works. The market, not a VP, judges your output.
Signs You're Stuck:

- Your pull request sits for a week awaiting "review."
- Legal stalls a feature over hypothetical risks.
- You need three signatures to change a button color.

Fix It:

- Flatten the chain—builders greenlight their work.
- Prototype first, ask forgiveness later.
- Prove it works—results silence doubters.

2. Meetings That Solve Nothing

Meetings are a productivity black hole. Hours vanish in standups, brainstorms, and "syncs" that resolve zip. A 2023 study pegged the average tech worker at 15 hours a week in meetings—half their time, zero code. FBI bans this circus. If it's not shipped, it's not worth discussing.
Basecamp slashed meetings with "no-meeting Wednesdays," boosting output by 30%. Why? Builders need uninterrupted blocks to think and create, not fragmented days of chatter. Updates belong in Slack, decisions belong in code. Meetings don't ship products—hands on keyboards do.
Meeting Purge:

- Replace standups with async posts—five sentences, done.
- Cancel recurring check-ins—trust your team to deliver.
- Limit talks to 15 minutes, max—solve it or scrap it.
- Measure impact—did that hour produce anything live?

3. Overengineering Before Validation

Traditional teams love shiny castles: complex architectures, endless scalability plans, all before a user touches it. The catch? Most of that effort solves problems that never arise. Twitter's early days were a mess—crashes galore—but they shipped, learned, and fixed it. Perfection pre-launch is a fantasy; reality post-launch is gold.
FBI builds lean: solve today's need, not tomorrow's guess. Overengineering wastes time and bets on unknowns. Users don't care about your Kubernetes cluster—they care if the app loads. Start small, validate, then scale. Anything else is ego, not strategy.
Symptoms of Overengineering:

- Months spent on "future-proofing" with no users.
- Features built "just in case" clutter the codebase.
- Debates over frameworks delay the first deploy.

Counter It:

- Ship a scrappy MVP—duct tape and hope work fine.
- Test with 10 users—see what they actually use.
- Refactor later—optimize what survives, not what might.

Conclusion

Traditional development is a dinosaur: slow, bloated, and dying. Approvals paralyze, meetings drain, and overengineering distracts. FBI flips it: act fast, talk less, build what's needed. Ditch the old ways, or watch the world lap you.

Chapter 4: Conclusion

The clock's ticking. *Fucking Build It* isn't a suggestion—it's your ticket to staying relevant. Working code cuts through noise. AI amplifies your power. Middlemen fade. Delivery beats delay. Autonomy drives wins. This isn't just a book—it's a challenge: build something now, or get left behind.
Start Today:

- Pick a project—a tool, a site, anything.
- Set a 24-hour deadline—stick to it.
- Use AI, skip the fluff, ship it raw.
- Share it—tweet it, post it, get eyes on it.
- Join the FBI crew—find forums, Discord, or workshops to level up.

The world doesn't need more plans—it needs builders. You've got the playbook. Now fucking build it.