

The Discord AI Bot Template — Production-Ready Community Automation

Version 1.0 · 2026 · Includes: Blueprint PDF + Working Bot Template +
Config Guide

1. The 30-Second Version

A complete, production-ready Discord bot template: welcomes members, answers FAQ from your facts, keeps moderation rules enforced 24/7, and quietly captures leads into a file you own — running on your own hardware with your own model. Customize three files, run one command, done.

Deliverables - `blueprint.pdf` — this guide - `bot/bot.py` — the working template (`stdlib` + `discord.py` + `requests`) - `bot/config.json` — your channel IDs, FAQ list, rules - `bot/README.txt` — the one-page run card - Single-business commercial license

2. What the Bot Does (and how each switch turns)

Feature	Default	Turn it on/off
Welcome + role prompt	on	<code>config.json</code> → <code>welcome.enabled</code>
FAQ auto-answer	on, strict-facts mode	append to <code>faqs[]</code>
FAQ → AI fallback	off at start	enable after a clean first week
Lead capture	on	every DM/question lands in <code>leads.jsonl</code>
Moderation: link policy	links-off in #general	per-channel rules
Moderation: invites	blocked (common raid vector)	per-channel rules
Escalation alerts	on	pings your staff channel

Design rule baked into the template: deterministic first, AI second. Keyword matches for known questions always beat model output. The model only handles what the facts can't. That's how you get a bot that never embarrasses the brand.

3. Setup Walk-Through (30 minutes)

Step 1 — Create the application (5 min).

`discord.com/developers/applications` → New Application → Bot tab → paste the token into `config.json`. Never commit that token anywhere.

Step 2 — Invite the bot (5 min). OAuth2 → URL Generator → scopes `bot` + `applications.commands`; permissions: View Channels, Send Messages, Manage Messages, Read Message History, Manage Roles. Open the generated URL in your server.

Step 3 — Wire your facts (10 min). Edit `config.json`: server ID, welcome channel, staff channel, your FAQ entries. The shipped config has 6 example facts — replace with yours.

Step 4 — Run (2 min).

```
pip3 install -U discord.py
python3 bot.py
```

The console prints `READY`. Drop a question in `#general`. Watch it answer from facts.

Step 5 — Harden (8 min). Run the 8-point test battery (below), then let it run in “answer-only” mode for a week before enabling moderation actions.

4. The 8-Point Test Battery

☐ welcome message lands with correct pings ☐ every FAQ question in config answers instantly + correctly ☐ unknown question → clean handoff line (no invention) ☐ FAQ miss logged (check the console counter — that's your content backlog) ☐ link-posting member gets the delete + educational message, not a ban ☐ invite link gets scrubbed; staff alert fires ☐ lead question in DMs lands in `leads.jsonl` ☐ `Ctrl-C` stops the bot cleanly (your kill switch)

8/8 = ship. Anything less = fix before customers find it.

5. Growth Notes From Operating Community Bots

- **The first 2 weeks, run answer-only.** Add moderation switches after trust is earned.
- **Every FAQ miss is a content gift** — one new fact per day compounds your bot's IQ.
- **Never let the bot argue.** Handoff is a feature, not a failure.
- **One bot, one job.** Don't bolt a shop onto your support bot; ship a second one.

6. Troubleshooting

- **PrivilegedIntents error** → Bot tab → enable Message Content Intent.
- **Bot online, silent** → wrong channel IDs in config; copy IDs with Developer Mode (right-click).
- **Slow FAQ answers** → you enabled AI fallback too early; facts are instant by design.
- **Rate-limit warning** → welcome bursts; space your joins naturally, don't mass-invite.

7. File Map

```
discord-ai-bot-template/  
├─ blueprint.pdf      ← this guide  
├─ bot/  
│   ├─ bot.py        ← working template  
│   ├─ config.json   ← your IDs, facts, rules  
│   └─ README.txt    ← one-page run card  
└─ LICENSE.txt
```

Ship it quiet, let it prove itself, then switch on the next feature.

— *All Is One. Build fast.*