**Telegram Bot Crash Course & Milestone Plan (Crispit)**

Use this as your **index / checklist**. For each item, copy the **Heading + Description** into your “Guide” chat, and we’ll teach/implement it step by step.

**Milestone 0 — Environment & Hello Bot**

**Goal:** Be able to run a minimal bot locally and reply to /start.

**You’ll learn:** BotFather basics, tokens & secrets, installing python-telegram-bot, long polling vs. webhooks (concept only), the event loop.

**Practice:** Create a bot with BotFather → install deps → print the bot username → implement /start and /help that reply with simple text.

**Apply to Project:** Set the welcome message that explains Crispit in one line and shows a static “Get Started” button (no actions yet).

**Definition of Done:** The bot replies to /start reliably on your machine.

**Git commit:** milestone-0: setup + hello-bot

**ETA:** 0.5–1 day

**Milestone 1 — Updates, Handlers, and the Bot runtime**

**Goal:** Understand how messages become Update objects and how handlers route them.

**You’ll learn:** Application, Update, Context, CommandHandler, MessageHandler, filters, async handlers.

**Practice:** Log every incoming update; implement /echo to mirror text; add MessageHandler(filters.PHOTO, ...) to acknowledge photos.

**Apply to Project:** Add a global error handler and a tiny middleware that logs user id, chat id, and command used.

**Definition of Done:** You can explain what update.message and context are, and route at least 3 different handlers.

**Git commit:** milestone-1: handlers + runtime

**ETA:** 0.5–1 day

**Milestone 2 — Buttons & Menus (InlineKeyboard)**

**Goal:** Make the bot fully button-driven.

**You’ll learn:** InlineKeyboardButton, InlineKeyboardMarkup, CallbackQueryHandler, callback\_data patterns, editing messages vs. sending new ones.

**Practice:** Build a home menu with buttons; handle callbacks with a simple state map; implement Back/Home navigation.

**Apply to Project:** Create the initial menu: **Extract Transcript**, **Get Notes**, **Help**.

**Definition of Done:** Tapping buttons changes the visible menu without typing commands.

**Git commit:** milestone-2: inline keyboards + nav

**ETA:** 1 day

**Milestone 3 — Conversation State & Persistence (Session basics)**

**Goal:** Track per-user/per-chat flow without losing context.

**You’ll learn:** In-memory session dicts, context.user\_data / context.chat\_data, optional ConversationHandler vs. manual FSM.

**Practice:** Store a tiny state (e.g., wizard steps) and retrieve it across messages; implement Cancel/Reset.

**Apply to Project:** Remember video\_url once the user sends it; keep a flag if transcript has been fetched.

**Definition of Done:** Navigating away and back still remembers where the user is within the current session.

**Git commit:** milestone-3: session state + basics

**ETA:** 1 day

**Milestone 4 — Files, Formatting & Telegram Limits**

**Goal:** Respect Telegram’s ~4096-character limit and send files when needed.

**You’ll learn:** Safe buffer (~3900 chars), Markdown vs. plain text, sending documents (send\_document), progress/status messages.

**Practice:** Write a helper that decides: show in chat vs. send as file; try with a long dummy text; add a unified “Too long for chat” notice.

**Apply to Project:** Gate **View Transcript in Chat** and **Text Notes** behind the length check; offer **Download .txt** or **Generate PDF** otherwise.

**Definition of Done:** No message exceeds limit; long outputs are sent as files with a short explanatory message.

**Git commit:** milestone-4: limits + file delivery

**ETA:** 0.5 day

**Milestone 5 — Transcript Pipeline Integration**

**Goal:** Plug in your get\_transcript logic end-to-end.

**You’ll learn:** Validating and extracting video IDs, handling different YouTube URL formats, robust error handling (no transcript, rate limits, etc.).

**Practice:** Wrap transcript fetching in a service function that returns (path, char\_count); store the path in session; show menu after success.

**Apply to Project:** Buttons after success: **View in Chat** (only if short), **Download .txt**, **Proceed to Get Notes**.

**Definition of Done:** End-to-end “Extract Transcript” button works from link → file on disk → user options.

**Git commit:** milestone-5: transcript pipeline wired

**ETA:** 1 day

**Milestone 6 — Style Selection UX (Presets, Custom, Samples)**

**Goal:** Let users choose a note style, view examples, and save their own presets.

**You’ll learn:** Designing callback payloads, simple pagination, external links (Style Samples page), collecting free-form text safely.

**Practice:** Show preset buttons; handle **Custom Style** → capture text → confirm; implement **My Presets** list with **Use/Rename/Delete**.

**Apply to Project:** Add **View Style Samples** (opens your hosted page) and a flow to **Save as Preset** with a short name.

**Definition of Done:** You can select any preset or create one and reuse it later.

**Git commit:** milestone-6: styles + user presets UX

**ETA:** 1–1.5 days

**Milestone 7 — Data Persistence (SQLite)**

**Goal:** Persist users, styles, and sessions across restarts.

**You’ll learn:** SQLite schema, migrations-lite, CRUD helpers, indexing by telegram\_user\_id, keeping callback\_data ≤ 64 bytes (use IDs).

**Practice:** Tables: users, styles, (optional) sessions; write repo functions: create\_style, list\_styles(user\_id), delete\_style, etc.

**Apply to Project:** Wire **My Presets** and **Save as Preset** to the DB; store transcript file path and last-chosen style id in sessions.

**Definition of Done:** Restarting the bot does not lose saved presets; menus load from DB.

**Git commit:** milestone-7: sqlite persistence + repos

**ETA:** 1–2 days

**Milestone 8 — Notes Generation (LLM API)**

**Goal:** Generate styled notes from transcript + style prompt.

**You’ll learn:** Calling your chosen LLM (Mistral/Gemini/etc.), building the prompt, streaming vs. non-streaming, basic retry logic.

**Practice:** A pure function generate\_notes(transcript\_text, style\_prompt) -> str with timeouts and errors handled.

**Apply to Project:** After style selection, ask for **Text in Chat** vs. **PDF File**; enforce length gate before sending.

**Definition of Done:** Notes are produced for a real video with at least two different styles.

**Git commit:** milestone-8: LLM integration + notes

**ETA:** 1–2 days

**Milestone 9 — PDF Generation & Asset Delivery**

**Goal:** Produce clean PDFs for transcripts, notes, and full session exports.

**You’ll learn:** Choosing a PDF lib (fpdf/reportlab/weasyprint), minimal styling, page headers/footers, file naming, cleanup policy.

**Practice:** Export a sample markdown/text to PDF; ensure Unicode fonts if needed; attach via send\_document.

**Apply to Project:** Implement **Generate PDF** for notes; plan a common template (title, date, video title, style used).

**Definition of Done:** PDF looks neat and is downloadable on desktop and phone.

**Git commit:** milestone-9: pdf export pipeline

**ETA:** 1 day

**Milestone 10 — Follow‑Ups & Contextual Q&A**

**Goal:** Let users ask questions about the video and get concise answers.

**You’ll learn:** Keeping conversation context, answer length control, optional citations.

**Practice:** Implement an **Ask Follow‑Up** flow; trim or summarize long answers to fit ≤ 3900 chars; fall back to file if still long.

**Apply to Project:** Answers use (summary + transcript); offer **Download .txt** when needed.

**Definition of Done:** At least 3 Q&A turns work smoothly after the summary.

**Git commit:** milestone-10: follow-ups + context

**ETA:** 0.5–1 day

**Milestone 11 — Export Full Session (PDF)**

**Goal:** Compile the initial summary + all Q&As into a single export.

**You’ll learn:** Simple templating for aggregation, table of contents (optional), file lifecycle.

**Practice:** Accumulate Q&A in memory/DB; render a final PDF with sections.

**Apply to Project:** Add **Export Full Session (PDF)** button after delivery.

**Definition of Done:** One click produces a clean, chronological session PDF.

**Git commit:** milestone-11: session export pdf

**ETA:** 0.5–1 day

**Milestone 12 — Errors, Rate Limits, and JobQueue**

**Goal:** Make the bot robust under load and during long tasks.

**You’ll learn:** Telegram 429 handling, exponential backoff, Application.job\_queue for deferred/long-running steps, user-friendly status updates.

**Practice:** Wrap API calls with retries; show “Working on it…” and then edit to the result.

**Apply to Project:** Use a job for transcript fetch or LLM call; guard against duplicate taps with a lock per chat.

**Definition of Done:** No crashes on transient errors; the UX always acknowledges user actions promptly.

**Git commit:** milestone-12: robustness + jobqueue

**ETA:** 1 day

**Milestone 13 — Style Samples Page (External)**

**Goal:** Host a simple page that previews all note styles with the same input snippet.

**You’ll learn:** Minimal static site or Telegram Web App basics; deep-linking back to the bot with a start parameter.

**Practice:** Build a page with 4–6 style cards; add a **Use this style** button that deep-links back (/start style=ID).

**Apply to Project:** Wire the bot’s **View Style Samples** button to this page.

**Definition of Done:** Tapping the link from Telegram opens the samples; choosing a style returns to the bot with it selected.

**Git commit:** milestone-13: style-samples page + deeplink

**ETA:** 1 day

**Milestone 14 — Polishing UX & Final Checks**

**Goal:** Make the whole flow smooth, discoverable, and consistent.

**You’ll learn:** Message editing vs. new messages, concise copy, consistent button labels, Home/Back everywhere.

**Practice:** Audit all paths; ensure every dead-end has a way out; standardize confirmations and error texts.

**Apply to Project:** Final copy pass; ensure length gates exist in all output paths; confirm presets CRUD feels snappy.

**Definition of Done:** A teammate can complete a full run without guidance.

**Git commit:** milestone-14: ux polish + copy pass

**ETA:** 0.5 day

**Optional Advanced (Do after core works)**

1. **Webhooks & Deployment** — Host on a VPS/Render/Fly; HTTPS certs; secret tokens.
   * Commit: optional: deployment via webhooks
2. **Telegram Web App** — In-chat mini UI for style selection & previews.
   * Commit: optional: webapp style picker
3. **Analytics & Telemetry** — Log events, basic dashboards.
   * Commit: optional: analytics
4. **Caching & Cost Control** — Cache transcripts/notes per video; reuse results.
   * Commit: optional: caching layer

**Quick Reference — Feature-to-Milestone Map**

* **Buttons & Menus:** M2
* **Sessions:** M3
* **Length Gate & Files:** M4
* **Transcript:** M5
* **Styles & Presets:** M6 + M7
* **LLM Notes:** M8
* **PDF:** M9
* **Follow-ups:** M10
* **Full Export:** M11
* **Robustness:** M12
* **Style Samples Page:** M13
* **Polish:** M14

**Suggested Commit Message Style**

Use imperative, scope-prefix, and short subject. Examples:

* feat(menu): add home buttons and callbacks
* feat(session): remember video\_url and transcript\_path
* feat(limits): gate long outputs and send as file
* feat(transcript): wire fetch + error handling
* feat(styles): presets ux + custom style flow
* feat(db): sqlite repos for styles + sessions
* feat(llm): generate notes with retry
* feat(pdf): export notes as pdf
* feat(qa): follow-up flow with length control
* feat(export): compile session pdf
* chore(logging): add structured logs
* fix(errors): guard duplicate callbacks

**How to Use This Document**

1. Pick the next milestone.
2. Copy the **Heading + Description** into your Guide chat.
3. I’ll teach the lesson, give tiny practice tasks, then implement the project piece.
4. Commit with the suggested message when done.

When you’re ready, start with **Milestone 0 — Environment & Hello Bot** in the Guide chat.