**Perfect — here's your full milestone document (v3) rewritten in your preferred detailed format with:**

* **Clear milestone goals**
* **What you’ll learn in each**
* **Concept-based mini-goals (what to understand + implement)**
* **Commit message**
* **Estimated duration**

**✅ Crispit Telegram Bot — Milestone Plan (v3: Functional First)**

**🚀 First focus: Full working features.  
🛠️ Second focus: Scaling, fallback, and enhancements.  
🧠 Learn as you build — each milestone includes concept-based mini-goals.**

**✅ Milestone 1 — Setup & Hello Bot**

**Goal: Run your bot locally and connect it to Telegram.**

**You’ll Learn:**

* **What BotFather is and how to create a bot**
* **How Telegram bots are event-driven**
* **Installing and initializing python-telegram-bot**
* **Token and secret management with .env**
* **Concept of polling vs. webhooks (no config needed yet)**

**Mini-goals:**

* **Create your bot on BotFather**
* **Install python-telegram-bot**
* **Set up .env with your token**
* **Create main() function that runs your bot**
* **Handle /start and /help commands**

**Definition of Done:  
Your bot starts with python main.py and replies to /start.**

**Git commit: milestone-1: bot setup + hello world  
ETA: 0.5–1 day**

**✅ Milestone 2 — Routing, Logging & Home UI**

**Goal: Handle different types of messages and set up a simple button-based interface.**

**You’ll Learn:**

* **Telegram’s Update, Context, and Application models**
* **CommandHandler, MessageHandler, CallbackQueryHandler**
* **Global logging of all user interactions**
* **InlineKeyboardButton and building simple menus**

**Mini-goals:**

* **Create button layout for: Get Transcript, Get Notes, Help**
* **Route buttons using callback\_data and CallbackQueryHandler**
* **Set up global logging for messages, commands, callbacks**
* **Add simple navigation (e.g. back/home button)**

**Definition of Done:  
Bot shows a navigable menu and logs all actions.**

**Git commit: milestone-2: routing + handlers + home ui  
ETA: 1 day**

**✅ Milestone 3 — Transcript Pipeline**

**Goal: Extract and return YouTube transcript from user link.**

**You’ll Learn:**

* **Extracting video IDs from YouTube URLs**
* **Using youtube\_transcript\_api to fetch auto/manual transcripts**
* **Supporting multiple languages via label selection**
* **Sending files (send\_document) on Telegram**
* **Tracking state using context.chat\_data**

**Mini-goals:**

* **Accept and validate YouTube links**
* **Parse and fetch available transcript options**
* **Let user choose transcript version (e.g., “English (auto)”)**
* **Generate .txt and send it to the user**
* **Store the transcript file path in session**

**Definition of Done:  
User sends a video link → bot returns .txt transcript file.**

**Git commit: milestone-3: working transcript route  
ETA: 1–1.5 days**

**🔄 Milestone 4 — Basic Notes Generation + PDF Return**

**Goal: Generate notes from transcript using Gemini and send them as a PDF.**

**You’ll Learn:**

* **Calling Gemini API with prompt + transcript text**
* **Basic API authentication and error handling**
* **Creating a PDF using reportlab or fpdf**
* **Deciding between chat message vs. file return**

**Mini-goals:**

* **Write generate\_notes(transcript\_text) to call Gemini**
* **Format prompt (use fixed style for now)**
* **Create a function to generate a simple .pdf from plain text**
* **Send the PDF using send\_document**
* **Optional: Show a text preview if short enough**

**Definition of Done:  
User receives a clean notes PDF based on the transcript.**

**Git commit: milestone-4: basic notes generation + pdf return  
ETA: 1.5–2 days**

**⏭️ Milestone 5 — Style Selection (Presets + Custom Prompt)**

**Goal: Let users choose from preset note styles or define their own.**

**You’ll Learn:**

* **Building preset option buttons with callback\_data**
* **Capturing free text input from users**
* **Temporarily storing selected style in session**
* **Updating the prompt dynamically based on style**

**Mini-goals:**

* **Add style selection panel with buttons (e.g., Bullet Points, Teacher Mode)**
* **Add “Custom Style” option → user sends text → confirm it**
* **Apply style to prompt before calling Gemini**
* **Show selected style name in note output**

**Definition of Done:  
Bot uses the user’s selected/custom style while generating notes.**

**Git commit: milestone-5: style selection + custom prompts  
ETA: 1–1.5 days**

**⏭️ Milestone 6 — SQLite Presets & User Memory**

**Goal: Store user-defined styles and past session data persistently.**

**You’ll Learn:**

* **How to use SQLite in Python with sqlite3 or SQLAlchemy**
* **Designing simple tables for users, presets, and sessions**
* **Writing reusable CRUD helper functions**
* **Mapping styles to Telegram user\_id**

**Mini-goals:**

* **Create DB schema for styles + users**
* **Store presets with a name (e.g., “My Exam Style”)**
* **Load, edit, or delete saved styles via bot menu**
* **Store last transcript path + style ID in sessions table**

**Definition of Done:  
Restarting the bot does not erase user styles or recent state.**

**Git commit: milestone-6: sqlite persistence for presets  
ETA: 1–2 days**

**⏭️ Milestone 7 — Follow-Up Q&A**

**Goal: Let users ask questions about the video or notes and get meaningful answers.**

**You’ll Learn:**

* **How to keep context between turns**
* **Prompt chaining with Gemini or fallback LLMs**
* **Trimming responses to fit Telegram’s message limit**

**Mini-goals:**

* **Ask follow-up → pass summary + transcript + question**
* **Limit responses to 3900 characters or less**
* **Show answers inline or as downloadable .txt if long**
* **Store Q&A turns in memory for session export**

**Definition of Done:  
Bot handles 2–3 follow-up questions after note generation.**

**Git commit: milestone-7: follow-up question flow  
ETA: 1 day**

**⏭️ Milestone 8 — Session PDF Export**

**Goal: Compile the full session (summary + Q&A) into a single exportable PDF.**

**You’ll Learn:**

* **PDF templating with headers, sections, and TOC (optional)**
* **Combining notes + multiple follow-ups into one file**
* **Organizing file cleanup and naming logic**

**Mini-goals:**

* **Store all Q&A turns during the session**
* **Create make\_full\_session\_pdf() from stored data**
* **Send one clean file when user taps "Export Session"**

**Definition of Done:  
Bot returns a well-structured PDF with summary and all Q&As.**

**Git commit: milestone-8: session pdf export  
ETA: 1 day**

**⏭️ Milestone 9 — Gemini API Key Rotation**

**Goal: Automatically rotate API keys if one hits its quota.**

**You’ll Learn:**

* **Reading API keys from .env or .json**
* **Cycling keys on 429 / quota exceeded errors**
* **Logging which key was used**
* **Optional: daily rate tracking**

**Mini-goals:**

* **Store multiple Gemini keys**
* **Wrap API call with retry logic and rotation**
* **Log key usage and failures**

**Definition of Done:  
Bot switches to the next available key if one fails.**

**Git commit: milestone-9: gemini key rotation logic  
ETA: 1 day**

**⏭️ Milestone 10 — Fallback LLM Integration (Mistral / Command-R)**

**Goal: Generate notes using another model if Gemini completely fails.**

**You’ll Learn:**

* **Calling Hugging Face Inference API or Cohere API**
* **Writing standardized prompts across models**
* **Fallback chaining logic in your code**

**Mini-goals:**

* **Add generate\_notes\_fallback() function**
* **Handle both short and long outputs from fallback model**
* **Show which model was used in PDF**

**Definition of Done:  
Bot produces notes with Mistral/Command-R if Gemini fails.**

**Git commit: milestone-10: fallback llm integration  
ETA: 1.5 days**

**⏭️ Milestone 11 — Multilingual Transcript Translation**

**Goal: Detect and translate non-English transcripts before summarization.**

**You’ll Learn:**

* **Detecting language using langdetect**
* **Using Gemini or Google Translate API for conversion**
* **Handling errors in long transcript translation**

**Mini-goals:**

* **Detect language from transcript text**
* **Translate non-English transcripts to English**
* **Then call summarization on translated text**

**Definition of Done:  
Hindi (or other) videos now return English notes.**

**Git commit: milestone-11: multilingual translation support  
ETA: 1–1.5 days**

**⏭️ Milestone 12 — Robustness & JobQueue**

**Goal: Add reliability to avoid crashes and lag.**

**You’ll Learn:**

* **Handling Telegram 429 rate limits**
* **Using Application.job\_queue for long tasks**
* **Locking to prevent duplicate processing**
* **Editing messages to show “Working…” updates**

**Mini-goals:**

* **Handle rate-limited API calls without crashing**
* **Show interim feedback during long tasks**
* **Prevent repeated callbacks if user double-clicks**

**Definition of Done:  
Bot responds smoothly under pressure with no freezes or crashes.**

**Git commit: milestone-12: jobqueue + retries + user feedback  
ETA: 1 day**

**Let me know when you're ready to begin Milestone 4, and I’ll walk you through Gemini API + PDF generation step by step in your guide chat.**