**Subject: Updated Guidance: AI Chatbot Module Documentation - BA Intern Cohort**

**Objective:** This document gives you clear instructions and checklists for the documentation you need to create for your AI Chatbot module. Our standard is Google-level quality: precise, clear, complete. Use these checklists to make sure your work meets that bar. Incomplete or unclear work isn't acceptable.

**Project Context:** We're documenting key parts (modules) of a new AI Chatbot. Users will chat and call with characters via web (React/Node.js) and mobile (Flutter). Your documents are essential for developers, testers, and future team members to understand exactly how each module should work.

**Assigned Modules:**

* **Minh:** Login/Register User Account
* **Trung:** Chat History Management
* **Duy:** Character Management
* **Mui:** Chatting and Voice Call Features

**General Work Rules (Follow These Strictly):**

* [ ] **Be Extremely Clear:** If someone *could* misunderstand you, they *will*. Use exact words. Explain acronyms the first time you use them. Make diagrams easy to read.
* [ ] **Know Your Audience:** Write for developers (who need technical details), testers (who need steps and error cases to test), and product managers (who need to understand the processes).
* [ ] **Be Consistent:** Use the *exact same terms* everywhere. If you call it "User" once, always call it "User" (not "Customer"). Name UI parts, API calls, and database tables the same way in all documents. Use standard diagram symbols consistently.
* [ ] **Use Approved Tools:** Use Google Drawings, Lucidchart (if available), or other tools I specify. Make sure your diagrams can be easily shared and edited.
* [ ] **Be Thorough:** Describe the normal way things work (the "happy path"). But *also* carefully describe other paths, what happens when errors occur, unusual situations (edge cases), and interruptions (like losing network). Think: "What if...?" for every step.
* [ ] **Connect Your Documents:** Make sure your documents logically link together. A step in a Table should match a part of the Flowchart and might show up as messages in a Sequence Diagram. Requirements should link to the Use Cases that satisfy them.
* [ ] **Expect Reviews:** You'll share drafts and get feedback. Review each other’s work *before* sending it to me. Your first draft likely won't be the last. My feedback will be direct to help you meet our high standards.

**Specific Documentation Checklists:**

**(Start Here) 0. Requirements Checklists (What the System Must Do)**

* **Purpose:** To define *what* the user needs and *what* the system must do before we detail *how*. These are the foundation. While you might receive initial requirements, refining and documenting them clearly is crucial.  
  **0a. User Requirements Checklist (What the User Needs to Do/Achieve)**
  + [ ] **User-Focused:** Written from the user's perspective (e.g., "As a Registered User, I want to...")?
  + [ ] **Clear Goal:** Does it state a clear user goal or task (e.g., "...log into my account...")?
  + [ ] **Clear Benefit:** Does it state *why* the user wants this (e.g., "...so I can access my chat history")?
  + [ ] **Unique ID:** Does each requirement have a unique ID (e.g., UR-001)?
  + [ ] **Concise:** Is it short and to the point?
  + [ ] **Testable:** Can you clearly tell if the final feature meets this need?
* **0b. System Requirements Checklist (What the System Must Do / Qualities it Must Have)**
  + [ ] **Unique ID:** Does each requirement have a unique ID (e.g., SR-F-001 for Functional, SR-NF-001 for Non-Functional)?
  + [ ] **Atomic:** Does it describe *one* single capability or quality? (Don't combine multiple requirements into one).
  + [ ] **Clear Verb (Functional):** For functional requirements (what the system *does*), does it use a clear action verb (e.g., "The system *shall* validate user credentials," "The system *shall* display chat history")?
  + [ ] **Measurable (Non-Functional):** For non-functional requirements (system *qualities* like speed, security), is it measurable and testable (e.g., "Login response time *shall* be under 2 seconds," "Password hashing *shall* use bcrypt")?
  + [ ] **Necessary:** Is this requirement truly needed to meet a user need or business goal?
  + [ ] **Unambiguous:** Is the language precise? Can it only be interpreted one way?
  + [ ] **Verifiable/Testable:** Can a tester create test cases to prove the system meets this requirement?

**(Process & Design Documentation)**

**1. Flowchart Checklist (High-Level Process Picture)**

* Purpose: To show the main steps of a process in your module, like a bird's-eye view.
* [ ] **Clear Title:** Does it clearly say which process this flowchart is for (e.g., "User Registration Flow")?
* [ ] **Standard Shapes:** Are you using the correct, standard flowchart shapes (Start/End, Process, Decision, Data)?
* [ ] **Start & End:** Does it have clear Start and End points?
* [ ] **Logical Flow:** Do the steps flow logically? Are arrows clear?
* [ ] **Decisions:** Are all key decision points (like "Password Valid?") shown with clearly labeled paths ("Yes", "No")?
* [ ] **User Steps:** Are major user actions shown (like "User Enters Email")?
* [ ] **Data:** Are important data inputs ("Password") and outputs ("Login Token") shown?
* [ ] **Easy to Read:** Is the diagram clean, not cluttered? Are labels easy to read?
* [ ] **Consistent Terms:** Do the words used match other documents?

**2. Use Case Checklist (Detailed User Interaction Story)**

* Purpose: To describe step-by-step how a user (or system) uses the module to achieve a goal, including errors.
* [ ] **ID and Name:** Does it have a unique ID (like UC-001) and a clear, action-focused name (like "Log In Successfully")?
* [ ] **Actor(s):** Is it clear who is performing the actions (e.g., Registered User, System)?
* [ ] **Goal:** Is the actor's goal clearly stated?
* [ ] **Conditions Before:** What must be true *before* this starts (e.g., "User is on Login screen")? (Preconditions)
* [ ] **Conditions After (Success):** What must be true *after* it finishes successfully (e.g., "User is logged in")? (Postconditions)
* [ ] **Main Success Steps:** Are there numbered steps showing the normal, successful path? Are steps clear actions (e.g., "1. User enters email...")?
* [ ] **Alternative Paths:** Are other valid ways to do this described (e.g., "Login with Google")? Does it say where these paths branch off the main steps?
* [ ] **Error/Exception Paths:** Is it meticulously clear what happens when things go wrong? For each error:
  + [ ] What triggers the error (e.g., Wrong password, Network down)?
  + [ ] How does the system react (e.g., Show error message "Invalid login")?
  + [ ] What is the final state for this error path (e.g., User stays on login page)?
* [ ] **How Often (Optional):** Is there an estimate of how often this happens (High, Medium, Low)?
* [ ] **Assumptions:** Are any assumptions listed?

**3. Tabular Processing Flow Checklist (Step-by-Step Logic for Developers)**

* Purpose: To break down a process into tiny steps in a table, easy for developers to follow.
* [ ] **Link to Process:** Does it clearly say which Flowchart or Use Case scenario this table details?
* [ ] **Correct Columns:** Does it use these columns: Step #, Step Description, Inputs, Outputs/Results, Systems/Components Involved?
* [ ] **Logical Steps:** Are steps numbered in order? Do they make sense?
* [ ] **Clear Descriptions:** Is each Step Description a clear action (e.g., "Check password format," "Get user data from DB")?
* [ ] **Accurate Inputs:** Does it list *all* the specific data or triggers needed for the step?
* [ ] **Expected Outputs:** Does it define the specific results or system changes for the step? Includes errors if handled in the step.
* [ ] **Systems/Components:** Does it name the exact parts involved (e.g., React UI, Node API, Auth Service, User DB, Flutter UI)? Be specific.
* [ ] **Right Level of Detail:** Are steps detailed enough for a developer, but not *too* tiny?

**4. Sequence Diagram Checklist (Timing of Component Interactions)**

* Purpose: To show how different parts (user, UI, API, database) talk to each other over time for one specific scenario.
* [ ] **Scenario Link:** Does it clearly state which Use Case or interaction this shows (e.g., "Sequence for Successful Login")?
* [ ] **Participants (Lifelines):** Are all the parts involved shown as vertical lines (e.g., Actor, Web UI, Backend API, DB)? Are they labeled clearly?
* [ ] **Messages:** Are messages between parts shown with arrows? Are messages labeled clearly (e.g., login(email, pass), getUser(), userData)?
* [ ] **Time Order:** Do interactions go down the page, showing the order they happen in?
* [ ] **Active Time (Activation Bars):** Are there narrow boxes on lifelines showing when a part is busy/processing?
* [ ] **Clear and Simple:** Does it focus *only* on the interactions for this scenario? Is it uncluttered? Use notes for extra explanation if needed.
* [ ] **Logic Blocks (Optional):** Are blocks like alt (alternatives), opt (optional), loop used correctly if needed for complex logic?
* [ ] **Actor Shown:** Is the user or system starting the interaction included?

**5. Activity Diagram Checklist (Workflow and Decisions)**

* Purpose: To show the flow of work, including choices, parallel tasks, and loops within your module.
* [ ] **Clear Title:** Does it clearly state what activity is being shown (e.g., "Character Creation Flow")?
* [ ] **Start/End:** Does it have one Start point (filled circle) and one or more End points (circled filled circle)?
* [ ] **Actions:** Are tasks shown as rounded rectangles? Are they labeled with clear actions (e.g., "Display Characters," "Validate Input")?
* [ ] **Decisions & Conditions:** Are choices shown as diamonds? Do the paths leading out have clear conditions that don't overlap (e.g., [Data OK], [Data Bad])?
* [ ] **Merge Points:** Are diamonds used to bring paths back together after a decision?
* [ ] **Parallel Tasks (Fork/Join):** If tasks happen at the same time, are black bars used to split (fork) and rejoin (join) the flow? Use only if truly parallel.
* [ ] **Flow Arrows:** Do arrows clearly show the direction of work? Does every path lead somewhere logically and eventually reach an end?
* [ ] **Lanes (Optional):** Are swimlanes used if it helps show *who* or *what system* does each action (e.g., User Lane, Backend Lane)?
* [ ] **Easy to Follow:** Is the diagram logical and easy to understand?

**Final Instructions:**

* Start by understanding and refining the **User and System Requirements** for your module. They guide everything else.
* Then, create **all five** process/design document types (Flowchart, Use Case, Tabular Flow, Sequence Diagram, Activity Diagram) for your assigned module.
* Save your work in the shared Google Drive folder I specified. Use clear file names: [Module Name]\_[Document Type]\_[Your Initials]\_[Version].ext (e.g., LoginRegister\_UserReqs\_AA\_v1.gdoc, ChatHistory\_Flowchart\_BB\_v1.png).
* Be ready to explain your documents and decisions during reviews.
* If you find unclear requirements or get stuck, try to figure it out first, then ask for clarification.

Let's get started. I want to see initial drafts of the **Requirements** first by End of Day **Friday Apr 11th**, and then the other documents by End of Day **Saturday Apr 12th**.