**Project Overview**

I am developing a card game using **Android Studio**, **Kotlin**, and **Jetpack Compose**.

**Architecture Notes**

* The **ViewModel** currently serves only as a structural placeholder.  
  It does **not** manage state or business logic at this stage.
* All **game state**, **business logic**, and **event handling** are centralized in GameManager, which is the **single source of truth**.
* All **state mutations** and **game flow control** must happen in GameManager.
* The ViewModel may be expanded or integrated in later phases, but for now it remains largely unused.

**Coding Guidelines**

1. **Respect the Architecture**
   * Keep all game logic and state in GameManager unless explicitly instructed otherwise.
   * Do not move logic into the ViewModel unless specifically requested.
2. **UI Guidelines**
   * Follow Jetpack Compose best practices.
3. **Code Quality**
   * Always provide **complete Kotlin snippets** with necessary imports (only return full compilable files if I explicitly say **FULL**).
   * Add concise, clear comments for any non-trivial logic.
   * Comments must be written in **plain English**, even if the discussion is in French.
4. **Consistency**
   * Match existing naming conventions and project style.
   * Follow Kotlin and Android idioms.
5. **Integration**
   * When adding new features, integrate them into GameManager unless otherwise specified.

**Answering Guidelines for GPT**

* When writing code, return only the **relevant snippets** unless I explicitly say **FULL**, which means I want the **entire compilable file**.
* If a change affects multiple files, show each updated file in full.
* If the request is ambiguous, **ask clarifying questions** before coding.
* Highlight any **breaking changes** or new dependencies introduced.
* Do not change the type of existing variables without warning.
* If you are missing a file’s content, ask me to upload it (use project\_tree.txt to know what might be needed).
* If I say **SLOW\_MODE**, it means I will present the full problem before you start analyzing. In that phase:
  + Just store the files/information.
  + Do **not** propose solutions yet.
  + I will type **GO\_OVERVIEW** when I want a high-level description with possible solution options.
  + I will type **GO\_FULL** when I want detailed step-by-step modifications and code.
* If I ask for **FULL\_CODE** it means that I want the full source code, including imports based on the latest version of file you’ve got. If you don’t have that file or if for some reason you may think I modified the file in the meantime, please always ask me the latest version of a file.
* • If I say SLOW\_MODE, it means I will present the full problem before you start analyzing. In that phase:
  + Just store the files/information.
  + Do not analyze, design, or propose solutions yet.
  + Simply acknowledge receipt and confirm understanding.
  + I will type GO\_OVERVIEW when I want a high-level description and design refinement.
  + I will type GO\_FULL when I want detailed step-by-step modifications and code.
* If I say GO\_OVERVIEW, it means I want a structured, high-level design analysis and refinement. In that phase:
  + Explain possible approaches, tradeoffs, and naming choices.
  + Show how new state fields should be owned (e.g. reducer vs GameManager).
  + Clarify how UI components would consume these fields while staying dumb.
  + Use string or pseudo-output examples (but no Kotlin code yet).
  + This is the phase where we iterate on design until it matches my intent.
* If I say GO\_FULL, it means I want the complete implementation in very detailed steps. In that phase:
  + Present changes in a specific order:
    - Structural updates first (rename fields, add new properties/data classes).
    - Introduce all new modules, functions, and helpers.
    - Wire the new pieces into existing code (reducers, views, etc.).
  + This ordered approach helps me:
  + Understand the global architecture and mechanisms better.
  + Use the compiler/IDE to highlight inconsistencies progressively (first missing structures, then missing helpers, then integration issues).
* Always return compilable snippets or full files if I request them.
* • If I ask for FULL\_CODE it means that I want the full source code, including imports based on the latest version of file you’ve got. If you don’t have that file or if for some reason you may think I modified the file in the meantime, please always ask me the latest version of a file.
* • If I reference a file that you don’t currently have, or if its content appears stale/minimal/inconsistent with project\_tree.txt, ask me to upload the latest version (use project\_tree.txt to name it precisely).