



# Mobile Technical assignment

The goal of this assignment is to build a simple mobile application using **Expo (React Native)** that runs seamlessly on both **Android and iOS**. The app should allow a user to enter a question, send it to a streaming API, and display the AI's response in a clear and structured way.

**We will pay particular attention to how you handle real-time streaming and markdown rendering as the AI's response arrives.**

---

## Requirements

### API Configuration

Please use the following endpoint for your app:

```
https://vera-assignment-api.vercel.app/api/stream?prompt=your-question-here
```

The endpoint returns **Server-Sent Events (SSE)** with the following characteristics:

- **Method:** GET
- **Header:** `Content-Type: text/event-stream`
- **Streamed Data Example:**

```
data: {"type":"NodeChunk","content":{"nodeName":"STREAM","content":"Partial text here..."}}
```

Your app should connect to this endpoint, handle the stream, and render the response incrementally as chunks arrive.

## Structured Content and Collapsible Sections

The streamed text may include structured tagged sections representing different content types.

These tags can appear multiple times or one after another, for example:

```
<guideline>Some markdown-formatted text here...</guideline> <drug>Drug A: 10mg daily</drug> <guideline>Another section...</guideline>
```

### Your app should:

- **Detect** when a tag (like `<guideline>` or `<drug>` ) starts and ends in the streamed text.
- **Render each tag as a collapsible section** with a title based on the tag name (for example, "Guideline" or "Drug").
- The **content inside the tag** should be displayed as **markdown-formatted text**.
- Any text **outside of tags** should still be rendered normally in markdown.

Here's an example of how these collapsible sections might look:

Drug Information

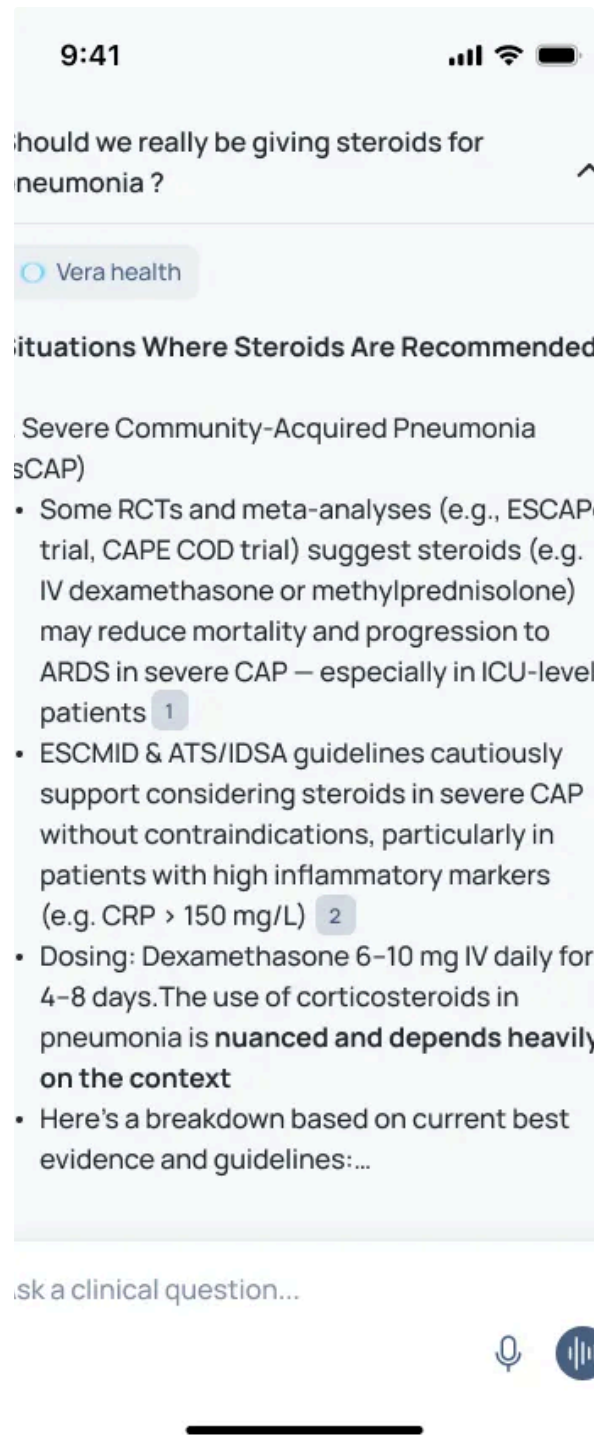
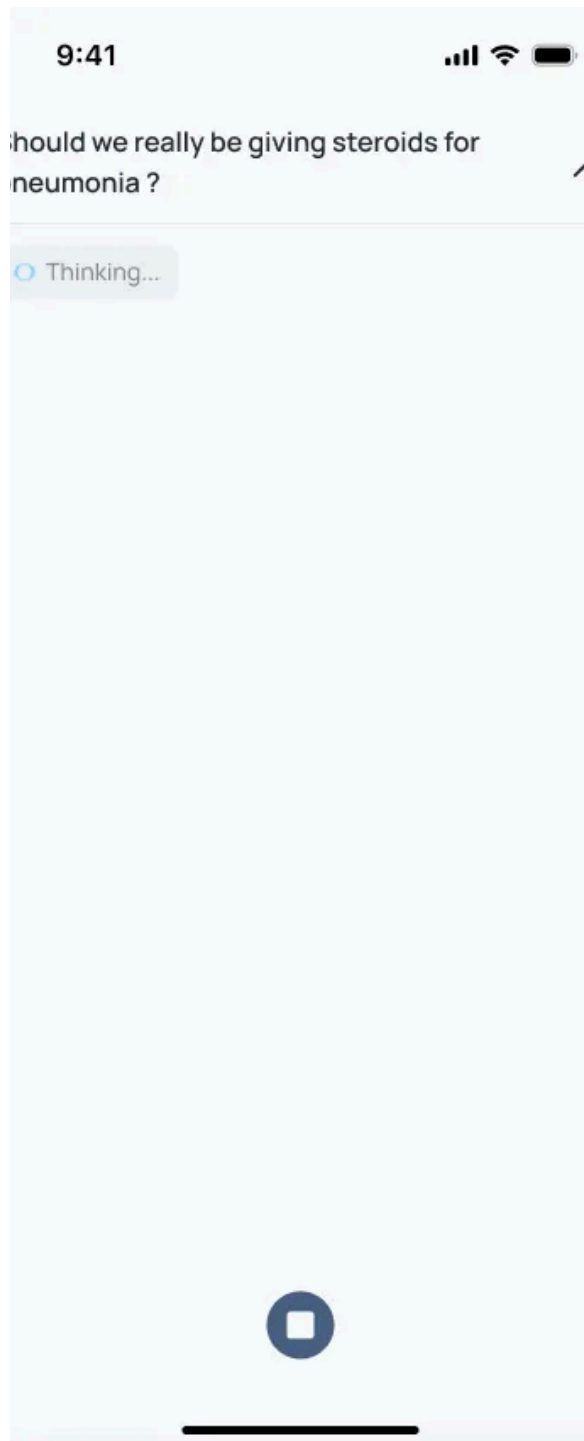
ancomycin must be **administered via slow IV infusion over ≥60 minutes** (maximum 1 mg/min) to prevent **Red Man Syndrome** and **hypotension**. Avoid IM route; verify compatibility before Y-site use; and monitor **renal function and serum drug levels** throughout therapy (*ASHP*).

## App Presentation

The application should open with a single screen where the user can type a question and submit it. Once the question is sent, the app will connect to the **Server-Sent Events (SSE)** API endpoint and start receiving the streamed response.

While the stream is active, the app must display the AI's answer in real time as chunks arrive. The user's question should appear at the top of the screen, and the corresponding answer should be displayed below it inside a **collapsible section**. The final answer will be returned in **markdown format**, and the app must render this markdown properly as formatted text.

The overall layout and design should follow the screenshot below.



## Evaluation Criteria

The review of this project will focus primarily on **code quality** and **performance**. The code should be clear, well-structured, and easy to follow. The app should perform efficiently, with smooth rendering and responsive interactions on both platforms. A well-designed user interface will be considered a plus, but the emphasis will be on maintainability and execution rather than visual polish.

---

## After Submission

After the project is submitted, we will schedule a **technical review call**. During this call, we'll go through the code and discuss the key decisions you made — such as how you handled streaming, how you structured the app, and what trade-offs or optimizations you considered. This discussion will help us understand your thought process, problem-solving skills, and ability to work in a real-world production environment.