

# 1.4 Assignment: AI Lab

## Section 1 : Practice with and LLM

**Text ::** Why stateless authentication is more important than Stateful for Rest API's where I specifically asked for simple response

From all these three responses, I felt Claude and ChatGPT have given more accurate and sufficient information even though I specifically asked for a simple response.

## ChatGPT Response ::

The screenshot shows a ChatGPT interface with the following content:

ChatGPT 5

Would you like me to show this answer in a **clean Word document format** (with diagrams and bullet visuals) so you can include it in your API design report or coursework?

Sure — here's a short and simple version

---

**Why Stateless Authentication is Better for REST APIs**

- REST is **stateless**, so every request should work independently without storing session data on the server.
- Stateless authentication (like JWT) sends all needed info (user ID, roles, etc.) in each request — no session lookup needed.
- Easier to scale — no shared session store or sticky sessions between servers.
- Faster and more reliable — works well in distributed or microservice environments.
- Stateful methods (sessions) break REST rules and make scaling harder.

In short:  
Stateless authentication fits REST APIs because it's **simple, scalable, and independent of server state**.

ChatGPT can make mistakes. Check important info.

## Gemini response ::

The screenshot shows the Gemini AI interface. At the top, it says "Gemini" and has a "PRO" status with a user icon. Below the header, there's a blue diamond icon followed by text: "Stateless authentication is **more important** for REST APIs because it guarantees **scalability** and **simplicity** by adhering to the core REST **constraint** that the server must not store any client session data. 🚀". A horizontal line separates this from the "Key Advantages" section. Under "Key Advantages", there are two bullet points: 1. **Scalability:** Since the server doesn't store session state, you can easily add more servers (horizontal scaling) and use simple **load balancing** because any server can handle any request. 2. **Simplicity:** The server doesn't waste resources (memory/CPU) managing session storage and lookup, making the architecture simpler and more resilient to server failure. Below this, it says: "Stateless systems, typically using **JSON Web Tokens (JWTs)**, pass all necessary authentication information within the **request** itself, allowing the server to verify the user without a database lookup." At the bottom of the main content area, there are several small icons for navigation and sharing. A rounded rectangle labeled "Ask Gemini" contains a text input field with placeholder text "Ask Gemini" and a "Tools" button. To the right of the input field are "2.5 Flash" and a microphone icon. A note at the bottom says "Gemini can make mistakes, so double-check it".

## Claude Response ::

The screenshot shows the Claude AI interface. At the top, it displays the URL "claude.ai/chat/87a57660-3ede-42a6-9548-f2b7d3089899". Below the URL, there's a sidebar with icons for "New", "Search", "Copy", "Share", and "Help". The main content area starts with the text: "because it embraces REST's design philosophy and provides the scalability modern applications need." In a box labeled "A", it says "Give me short and simple response". The response text is: "Stateless authentication is better for REST APIs because: 1. **Scalability** - Any server can handle any request since there's no session data to store or share. Easy to add more servers. 2. **REST principle** - REST is designed to be stateless. Each request should contain everything needed to process it. 3. **No server memory** - Tokens (like JWT) contain all user info, so servers don't waste resources storing sessions. 4. **Works everywhere** - Perfect for mobile apps, microservices, and distributed systems where maintaining server sessions is impractical. 5. **Simple load balancing** - Requests can go to any server without needing "sticky sessions." In short: stateless = faster, more scalable, and truly RESTful." At the bottom, there's a "Sonnet 4.5" footer.

## Section 2 :: Practice with custom GPTs

I chose to interact with **Canvas(Canva 4o model)**, a Custom GPT built for writing and project collaboration. During the chat, I asked it to explain topics, make changes, and shorten answers. It quickly adjusted and kept everything clear, and I've asked to create a image of lion which it did very well

When I added my own opinions, Canvas improved the response instead of replacing my ideas. I also asked why it worded something a certain way and what if I changed the situation both times, it gave thoughtful explanations.

Overall, using Canvas felt smooth and interactive. It understood instructions well, explained clearly, and made the process of creating and refining my work easier.

You can customize this with your own quote, brand colors, or logo after opening it in Canva.

Would you like a powerful quote suggestion to go with one of these designs?

Ask anything

ChatGPT can make mistakes. Check important info.

## **Section 3 :: Practice with a Research Assistant**

When I explored the topic “Stateful vs Stateless Authentication” using STORM AI, the system followed a structured, multi-stage process to create its research article. It started by identifying essential terms such as session management, cookies, JSON Web Tokens (JWT), scalability, and security. Then it developed an outline comparing how each approach manages user sessions, server-side data, and scalability.

Next, it collected insights from multiple technical domains — including web security, REST API architecture, and token-based authentication frameworks to form a comprehensive understanding. In the final phase, STORM AI organized this information into a well-structured article that explained both authentication methods, their advantages, and potential drawbacks in a developer-friendly way.

Rather than copying information, it logically summarized and connected ideas, keeping the explanation clear and technically accurate. At the end of the process, it even requested user feedback on the quality of the research and how well the generated content aligned with my expectations, which made the interaction feel more engaging and reflective.

storm.genie.stanford.edu/article/so-the-topic-is-about-stateful-vs-stateless-authentication-1434951

≡ STORM

## SO the topic is about Stateful vs Stateless authentication

**Writing Purpose \***

Please share the motivation and what you hope to achieve with your topic

So I'm building an application where I'm currently using Stateful authentication but moving ahead I'm going to create Rest API's where for scalability I need to go for stateless auth sch as JWT

**Submit**

storm.genie.stanford.edu/article/so-the-topic-is-about-stateful-vs-stateless-authentication-1434951

≡ STORM

Contact Us Bug Report

## SO the topic is about Stateful vs Stateless authentication

 See BrainSTORMing Process

### summary

Stateful and stateless authentication are two fundamental approaches to managing user authentication and session management in web applications. Stateful authentication relies on the server maintaining session information, allowing for personalized user experiences by tracking user interactions across multiple requests. This method, often implemented through session cookies, offers advantages such as easy session revocation and modifiable session data, but it faces significant scalability challenges as user numbers increase, leading to resource intensity and potential performance bottlenecks.[\[1\]](#)[\[2\]](#)[\[3\]](#) In contrast, stateless authentication does not retain session information on the server, instead using tokens—commonly JSON Web Tokens (JWT)—to identify users. Each request is self-contained and independent, which enhances scalability and simplifies load distribution across multiple servers. This approach improves performance by minimizing server overhead, making it particularly advantageous in distributed systems and microservices architectures.[\[4\]](#)[\[5\]](#)[\[6\]](#) However, stateless authentication presents its own security challenges, as the validity of tokens can persist until expiration, requiring robust measures to mitigate risks such as token manipulation and unauthorized access.[\[7\]](#)[\[8\]](#) Both authentication methods are notable for their distinct mechanisms and implications for application performance, security, and user experience. The choice between stateful and stateless authentication often hinges on specific



This editor is a software developer with extensive experience in web application security. They will focus on the technical differences between stateful and stateless authentication, explaining the underlying protocols, mechanisms, and implementation challenges.

**Can you explain the primary protocols and mechanisms used in stateful authentication, including how session management is typically implemented?**

Stateful authentication is a method where the server retains information about a user's session after they log in, which allows it to manage the user's interactions with the application more securely and effectively. The primary protocols and mechanisms involved in stateful authentication typically rely on session management techniques, where the server stores session data either in memory or in a database. 1. \*\*Session Management\*\*: In stateful authentication, a unique session identifier (session ID) is created upon a user's successful login. This session ID is stored on the server and associated with the user's information, such as user ID and permissions. Each subsequent request from the user includes this session ID, allowing the server to retrieve and validate the session data associated with it (source [5], [25]). 2. \*\*Storage Mechanisms\*\*: The session data is often stored in a database or memory to ensure that the server can efficiently access and manage user sessions (source [22], [24]).

### Feedback

Thank you for trying our research preview, STORM! We would love to get your feedback to continue improving.

Here are four short questions (**take less than 3 minutes**) we would like to ask you. Your responses are invaluable to us!

**Feedbacks are saved automatically.**

- How would you rate the generated article?  
★★★★★
- What are the strengths (e.g. comprehensive outline, accurate information, etc.) and limitations (e.g. improper handling of time-sensitive information, associating unrelated sources, etc.) you see in this article produced by STORM?  
Type your answer here
- STORM organizes the information using a hierarchical outline (check out the "Table of

# **Build a Chatbot Assistant - Cricket Stats Bot**

## **Design Thinking Process**

### **1. Empathy Phase**

**User Need:** Cricket fans want quick access to player statistics across multiple teams without searching multiple websites.

#### **Pain Points:**

- Fragmented information across different sources
- Time-consuming searches
- Difficulty comparing players

**Target Users:** Cricket enthusiasts, casual fans, fantasy cricket players.

### **2. Define Phase**

**Problem Statement:** Cricket fans need an interactive way to quickly retrieve and compare player statistics for India, Australia, New Zealand, England, and South Africa.

#### **Scope:**

- 5 national teams
- Player stats (batting & bowling)
- Team squads
- Top performer rankings

### **3. Ideate Phase**

**Solution:** Conversational chatbot with natural language understanding

#### **Key Features:**

- Natural language queries
- Player search by name

- Team squad displays
- Top batsmen/bowler rankings
- Statistical comparisons

## 4. Prototype Phase

**Technology:** React.js with Tailwind CSS

**Data Structure:**

- 40+ players across 5 teams
- Batting stats: runs, average, strike rate, centuries
- Bowling stats: wickets, economy, best figures

**Key Functions Built:**

- `getTeamSquad()` - Display team lineups
- `getPlayerStats()` - Show individual statistics
- `getTopBatsmen()` - Rank leading scorers
- `getTopBowlers()` - Rank leading wicket-takers
- `processUserInput()` - Natural language processing

## 5. Test Phase

**Test Scenarios:**

Query Type	Example	Result
Team squad	"Show India squad"	✓ Pass
Player stats	"Virat Kohli stats"	✓ Pass
Top performers	"Top batsmen"	✓ Pass
Unknown player	"XYZ player"	✓ Pass (helpful error)

**Functionality:** All features working as expected  
**Usability:** Clean interface, quick responses  
**Ethics:** No bias, equal representation of all teams

## **Bot Planning Questions**

### **1. Bot's Objective:**

Provide cricket fans with quick access to player statistics and team information through conversational interface.

### **2. Information/Knowledge:**

Player names, batting/bowling statistics, team squads, format specializations (Test/ODI/T20).

### **3. What NOT to do:**

No live scores, match predictions, personal data collection, or non-cricket topics.

### **4. User Engagement:**

Enthusiastic, knowledgeable tone with cricket emojis. Quick, informative responses with follow-up suggestions.

### **5. Images/Other Data:**

Text-only for simplicity. Future: player photos, charts, team logos.

### **6. Starting Message:**

Welcome greetings + list of capabilities + covered teams + example queries.

### **7. Conversation Steps:**

User query → Intent recognition → Data retrieval → Formatted response → Offer related info

### **8. Conclusion:**

Always ask if user needs more info, suggest related queries, and keep conversation open.

## **9. Successful Engagement:**

User finds info in  $\leq 3$  messages, receives accurate stats, explores multiple features naturally.

## **10. Test Scenarios:**

Created tests for greetings, team squads, player stats, top performers, edge cases, and ambiguous queries. All passed.

### **Build Tool Selection ::**

**Tool Used:** Claude.ai with React Artifacts

### **Why:**

- Rapid prototyping
- No setup required
- Easy sharing and testing
- Full React support

### **Privacy & Terms Learnings:**

- Don't collect personal data
- Maintain data accuracy
- Ensure transparency about capabilities
- Follow ethical AI guidelines

### **Build Process ::**

#### **Iterations:**

- Basic structure with sample players
- Expanded to 40+ players across 5 teams
- Added natural language processing
- Enhanced UI with cricket theme
- 5. Tested and refined responses

## **Challenges & Solutions:**

- **Challenge:** Organizing player data efficiently  
Solution: Nested object structure (team → role → player)
- **Challenge:** Understanding varied queries  
Solution: Pattern matching with regex
- **Challenge:** User guidance  
Solution: Helpful suggestions in welcome and default messages

## **Results ::**

**Features Achieved:** 40+ players across 5 teams

- Natural language queries
- Team squads, player stats, rankings
- Clean, responsive interface
- Fast, accurate responses

## **Key Learnings:**

- Design Thinking methodology
- User-centered design approach
- Chatbot development with React
- Natural language processing basics
- Iterative testing and refinement

## **Bot Link & Evidence ::**

[\[Link to Cricket Stats Chatbot - Available in Claude.ai Artifacts\]](#)

## **Sample Interactions:**

"Show me India squad" → Full team roster with stats

"Rohit Sharma Stats" → Complete player profile

"Top batsmen in Australia" → Ranked list with averages

"Which teams?" → List of 5 covered nations

## Output Screenshots ::

The screenshot shows a web browser window with the URL [claude.ai/public/artifacts/d182b948-aa99-4c5a-9ba2-ed4204560485](https://claude.ai/public/artifacts/d182b948-aa99-4c5a-9ba2-ed4204560485). The page title is "Cricket Stats" and the subtitle is "Player Statistics Assistant". A green sidebar on the left contains a user icon and the text: "Welcome to Cricket Stats! I'm your cricket statistics assistant, covering players from India, Australia, New Zealand, England, and South Africa." Below this, a list of topics includes: "How can I help you today? You can ask me about:" followed by a bulleted list: "• Player statistics and records", "• Team squads and lineups", "• Compare players", "• Top performers by team", and "• Specific player details". A text input field at the bottom asks "Which team or player would you like to know about?". At the bottom of the page is a search bar with the placeholder "Ask about players, teams, or stats..." and a green search icon.

[claude.ai/public/artifacts/d182b948-aa99-4c5a-9ba2-ed4204560485](https://claude.ai/public/artifacts/d182b948-aa99-4c5a-9ba2-ed4204560485)

Claude Content is user-generated and unverified.

## Cricket Stats

Player Statistics Assistant

Which team or player would you like to know about?

"Show me India squad"

\*\*INDIA Squad:\*\*

- \*\*BATSMENT:\*\*
  - Virat Kohli - 26733 runs, Avg: 54.97
  - Rohit Sharma - 18570 runs, Avg: 48.88
  - Shubman Gill - 3456 runs, Avg: 52.51
  - KL Rahul - 7638 runs, Avg: 45.48
- \*\*BOWLERS:\*\*
  - Jasprit Bumrah - 432 wickets, Econ: 6.23
  - Mohammed Shami - 445 wickets, Econ: 5.67
  - Kuldeep Yadav - 188 wickets, Econ: 5.12
- \*\*ALL-ROUNDERS:\*\*
  - Ravindra Jadeja - 6234 runs, 485 wickets
  - Hardik Pandya - 4123 runs, 145 wickets

Want detailed stats for any player?

Ask about players, teams, or stats...

[claude.ai/public/artifacts/d182b948-aa99-4c5a-9ba2-ed4204560485](https://claude.ai/public/artifacts/d182b948-aa99-4c5a-9ba2-ed4204560485)

Claude Content is user-generated and unverified.

## Cricket Stats

Player Statistics Assistant

• Hardik Pandya - 4123 runs, 145 wickets

Want detailed stats for any player?

"Rohit Sharma stats"

\*\*Rohit Sharma\*\* (India)  
Role: Batsman  
Format: All

\*\*BATTING:\*\*

- Matches: 464
- Runs: 18570
- Average: 48.88
- Strike Rate: 90.23
- Centuries: 49
- Half-centuries: 104

\*\*BOWLING:\*\*

- Wickets: 8

Want to compare with another player?

Ask about players, teams, or stats...