1. Executive Summary

Jiu-Jitsu Genie is a generative AI application designed to provide grappling athletes with real-time, context-aware strategy support. The project addresses a key limitation in current martial arts education: most instructional content is static, lacking responsiveness to positional dynamics, athlete attributes, or opponent behavior. This is especially problematic in solo training environments, where access to coaching is limited.

The app explores whether large language models (LLMs) can simulate expert-level decision-making in Brazilian Jiu-Jitsu (BJJ) and mixed martial arts (MMA). It integrates structured natural language prompts with AI-driven modules that generate position-specific flowcharts, analyze still frames and video segments, and simulate coaching dialogue with historical figures like Rickson or Helio Gracie.

Key innovations include the use of Mermaid.js for flowchart generation, session-based state management for strategic continuity, and prompt-based adversarial logic to account for opponent responses. The application architecture is modular and built on Streamlit, with Python-based helper functions and FFmpeg integration for video trimming.

This project serves as both a software prototype and a technical investigation into the applicability of generative AI for real-time tactical modeling in combat sports.

2. Problem Statement

In the realm of Brazilian Jiu-Jitsu (BJJ) and mixed martial arts (MMA), technical mastery alone is no longer sufficient for competitive success. High-level grappling is increasingly defined by adaptability, positional intelligence, and the ability to make rapid decisions under dynamic, often adversarial conditions. Yet most instructional tools available to practitioners remain static: videos, books, and even structured curriculums tend to present technique in linear formats, devoid of opponent interaction, individual context, or evolving strategy. There is a growing disconnect between the complex cognitive demands of live grappling and the limited interactivity of traditional training resources.

This gap is particularly apparent in solo learning contexts, where practitioners lack immediate feedback or coaching. While video breakdowns and online seminars can deliver high-quality content, they do not offer real-time scenario analysis, tactical branching, or visual representations of decision trees in a match. This creates a critical bottleneck in the development of grappling intelligence—especially for remote athletes, hobbyists without regular coaching, or competitors trying to simulate strategic problem-solving outside the gym.

3. The Hypothesis

This project investigates whether generative AI—specifically large language models (LLMs)—can close this gap by offering context-aware, user-specific grappling feedback. The core hypothesis is that modern LLMs, trained on a combination of general knowledge and technical martial arts content, may now possess enough latent understanding of BJJ and MMA to:

- Analyze still frames or match segments
- Anticipate tactical decision points
- Generate personalized grappling flowcharts based on user input
- Simulate adversarial strategy and counters
- Serve as conversational coaching agents mimicking the philosophy and tone of elite instructors

A central goal of this project is to determine whether LLMs can not only describe what is happening in a grappling exchange, but also suggest what should happen next, given the athlete's physical attributes, positional status, and opponent behavior.

3.A. Methodological Note: Using Claude for AI-Augmented Coding

The technical scaffolding for this project was significantly accelerated through the use of Claude, the generative AI assistant from Anthropic. Claude was used extensively for:

- Drafting and debugging the modular Python architecture
- Writing prompt logic for flowchart generation, video analysis, and coaching simulation
- Cleaning and formatting Mermaid.js syntax outputs
- Designing state-handling logic in the Streamlit app interface
- Refactoring backend code to ensure modularity, error handling, and iterative logic

Claude was especially valuable for translating conceptual problems into working code blocks in an iterative, conversational format—speeding up development and allowing the project to remain agile across multiple design pivots.

4. Exploratory Breakthroughs

The development of *Jiu-Jitsu Genie* uncovered several key breakthroughs that redefined what AI-powered grappling coaching could achieve. These breakthroughs occurred not just at the level of code, but at the intersection of conceptual design, technical implementation, and philosophical alignment with how high-level jiu-jitsu is practiced and understood.

4.A. Flowchart-Based Grappling Logic as Tactical Cartography

One of the most novel contributions of this project is the use of **Mermaid.js** to represent grappling strategy as a living, branching map. Traditional coaching often relies on isolated technique chains or set drills. In contrast, *Jiu-Jitsu Genie* uses Mermaid to **visualize the flow of decision-making itself**, constructing diagrams where each node

represents a grappling position or key transition, and each edge is labeled with a biomechanical or tactical action.

This architecture allows grappling logic to be encoded **not as linear sequences, but as a decision tree**—a map of possibilities that adapts to the user's context (e.g., physical attributes, preferred positions, MMA vs. gi rule set). The AI-generated Mermaid object outputs not only the moves but the logic between them, capturing the "why" behind each branch. This is especially important for developing strategic awareness in novice-to-intermediate practitioners.

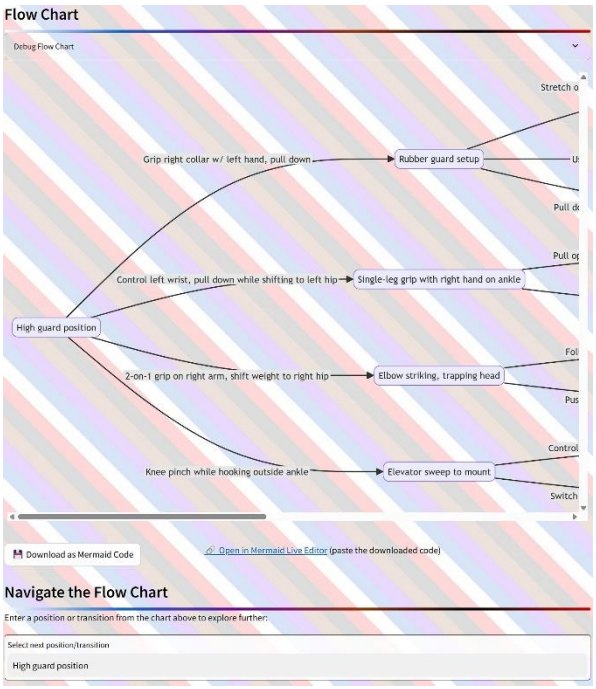

Figure 2: UI Screenshot of a Mermaid Flowchart

4.B. Iterative Traversal: Flowchart-as-Interface

A second major breakthrough was enabling **user-driven traversal of the flowchart network**. After the initial chart is generated from a user-defined position, the user can select any downstream node (e.g., "triangle choke setup", "elevated butterfly sweep", etc.), and re-center the chart on that position by calling next_move()—which regenerates a new flowchart using that node as the new base.
This mimics how **elite grapplers mentally re-anchor their strategy** after a failed sequence, changing plans mid-match. In software terms, this required parsing Mermaid objects, isolating nodes, and rerouting prompts in a closed feedback loop.
Philosophically, it advances the idea that learning jiu-jitsu is not about memorizing technique—it's about **practicing transitions between decision states**.

4.C. Adversarial Thinking Embedded Throughout

Traditional grappling instruction often presents idealized flows: "if you do X, then Y will happen." But real opponents resist, adapt, and counter. *Jiu-Jitsu Genie* embraces this reality by embedding **adversarial logic into every major function**. The flowchart prompts explicitly request the inclusion of branches based on common opponent reactions (e.g., "if opponent posts the arm, switch to sweep"). The analyse_grappling_match function analyzes match footage not only for the user's actions, but for what went wrong due to **opponent interference or tactical mismatch**.

This mindset of **continuous counterplay** culminates in the adversarial gameplan visualizer, which uses side-by-side flowchart views to simulate a matchup between contrasting strategies. This allows athletes to test their approach against hypothetical

counter-scenarios—a form of visual sparring that reflects the **OODA loop (Observe, Orient, Decide, Act)** used in high-level combat training and military strategy.

4.D. Visual Grammar of Combat

Finally, the project's design unlocked an aesthetic and pedagogical insight: **jiu-jitsu has a visual language**, and Mermaid allows that language to be rendered in clean, interpretable form. By enforcing radial layouts, concise labels, and clear arrows of intent, the app turns abstract grappling theory into **readable tactical grammar**. Practitioners are not just told what to do—they are shown how the game itself is structured.

This visual clarity is especially valuable for remote learners, neurodivergent athletes, or anyone who processes movement best through schematics. It also lays the groundwork for future integrations—augmented reality overlays, haptic training devices, or opponent modeling engines.

Collectively, these breakthroughs establish *Jiu-Jitsu Genie* not merely as a tool, but as a **framework for exploratory martial cognition**. It reconceives grappling not as rote technique, but as an adaptive problem space—one that generative AI is now capable of modeling, enhancing, and co-navigating.

5. App Architecture and Functionality

The *Jiu-Jitsu Genie* application was developed using a modular architecture that integrates generative AI into an intuitive user interface for grappling analysis and strategic planning. This section outlines the technical design and core components of the application, with emphasis on backend architecture, session-state logic, and user experience features that enable persistent, dynamic interaction.

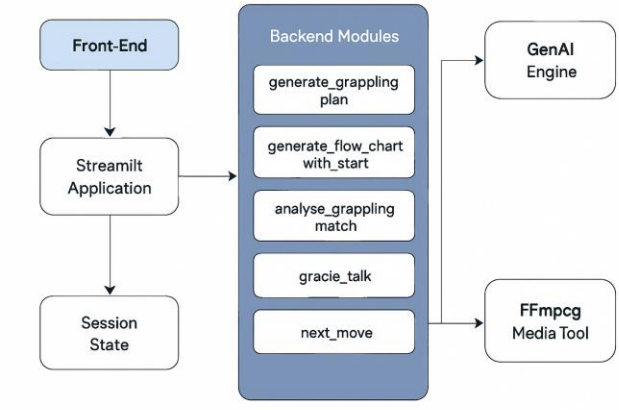

Figure 2: a visual depiction of the application architecture

The primary user interface is built in **Streamlit**, chosen for its rapid development capabilities, ease of deployment, and seamless integration with Python-based AI tools. Streamlit's layout engine and sidebar functionality allowed for the creation of **distinct**

user modes—such as "Position Image Recommendations", "Master Talk", and "FLOW Chart Generator"—each routed through an application state variable for persistent context management. CSS customizations were embedded directly into the flowstate.py file to create a unique **belt-rank inspired visual hierarchy**, using alternating gradients and color-coded panels (white to red belt) to symbolize user progression and strategy complexity.

Application logic resides in jiu_jitsu_functions.py, which contains a suite of **modular top-level functions** that encapsulate AI service calls and domain-specific logic. These functions are written in a semi-object-oriented style and are invoked directly by the Streamlit front end. Each function adheres to a clean input/output signature and separates error handling, prompt construction, and AI response parsing into modular layers. This modularity ensures that new grappling logic, AI endpoints, or flowchart formats can be integrated with minimal refactoring.

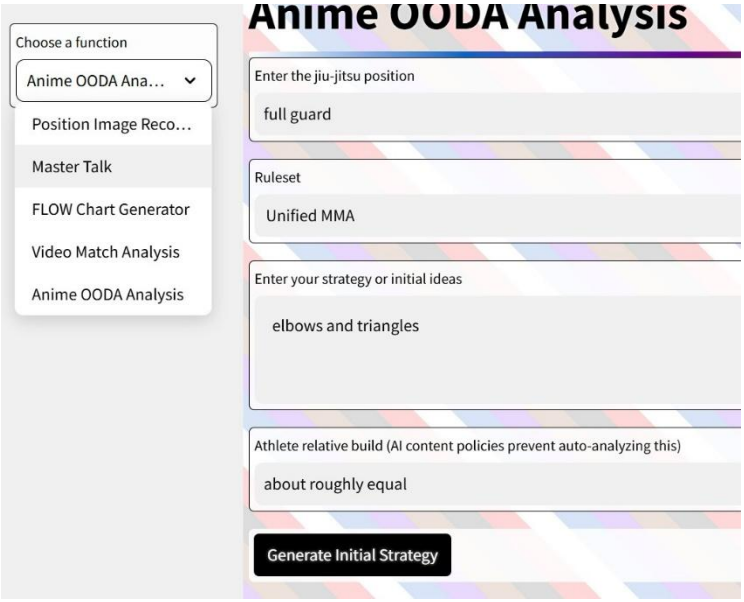

Figure 3: Sidebar and Initial UI formatting

The backend

5.A. Key Functional Modules

Each of the following bespoke functions serves a unique user-facing purpose, built on top of AI foundation models and custom formatting logic:

- **generate_grappling_plan(image, player_variable, isMMA, keywords)**
  Analyzes a still frame of a grappling position using image captioning and AI inference. Calls the GenAI.generate_image_description() method with a structured tactical prompt. Returns next-move recommendations in a bullet list.
- **gracie_talk(gracie_name, chat_history, user_message)**
  Simulates a conversation with a specific historical or modern jiu-jitsu master. The module generates replies through GenAI.generate_chat_response() and maintains continuity through persistent chat memory.
- **generate_flow_chart_with_start(measurables, starting_position, isMMA, favorite_ideas)**
  Constructs a **Mermaid.js-based strategy diagram** tailored to athlete attributes. Ensures grappling chains branch with both offensive and defensive logic using

adversarial reasoning prompts. This has subsidiary functions to also sanitize and render the mermaid diagram into an HTML streamlit frame.

- **next_move(flow_chart, move_text, measurables, isMMA, favorite_ideas)**
  Enables **flowchart traversal**, re-centering the strategic map on a selected node and regenerating transitions from that position. This supports recursive tactical planning and mimics real-life positional resets during matches.
- **analyse_grappling_match(video, player_variable, isMMA, keywords, start_time, end_time)**
  Uses frame extraction logic and the GenAI video description engine to assess a video segment. If time parameters are passed, **FFmpeg** is used to trim the video clip before analysis, supporting precise, scenario-based coaching.

5.B. Persistent State Management

One of the challenges in designing an interactive AI coaching tool is maintaining **continuity of context** as users move between modes or update inputs. To solve this, *Jiu-Jitsu Genie* uses Streamlit's st.session_state object to persist:

- current_image: file path for the uploaded grappling still
- current_flowchart: current Mermaid object being visualized
- current_chat: chat history for gracie_talk
- current_video: uploaded video file reference
- current_attributes: string describing athlete physical attributes

This design enables **multi-step analysis across UI sessions**, allowing users to refine insights, re-analyze positions, or shift focus without data loss.

5.C. Visual Features and Output Styling

The entire app uses a custom **black-and-white GUI with belt-rank-themed accent panels**, where strategy cards and responses are grouped using CSS classes like .white-belt, .black-belt, or .red-belt. These visual cues help users distinguish between entry-level tactics and more advanced, situational flows. Chat windows and analysis boxes are also styled to feel like corner-coach interfaces, supporting immersion.

A defining UI feature of the app is the **Mermaid.js flowchart rendering** capability. The system programmatically generates valid Mermaid syntax from AI responses, then sanitizes and injects it into the frontend using Streamlit's html() method. These flowcharts display positional logic as radial graphs with arrows labeled by key transitions (e.g., "frame and hip escape", "post and underhook").

5.D. Media Handling and FFmpeg Integration

Video support is enabled through **FFmpeg**, which allows the user to define time-bound segments for tactical breakdown. Upon selecting a start and end time, the app runs FFmpeg via subprocess to clip the relevant portion of the video, saving it temporarily and

analyzing only the relevant exchange. This enables users to isolate scrambles, passes, or submission attempts for detailed feedback—mirroring how professional coaches perform film study.

6. Analysis Methods and Implementation Details

The core analytical capability of *Jiu-Jitsu Genie* lies in its use of **structured generative AI prompts** tailored to the grappling domain. The system's intelligence does not emerge solely from hardcoded logic or datasets, but from carefully engineered **natural language prompts** passed to large language models (LLMs), enabling dynamic, expert-like responses in the areas of strategy generation, match analysis, and conversational coaching.

This section documents the design of these prompts, explains logic differentiations between rule sets (MMA vs BJJ), and details implementation choices for resilience, adaptability, and future extensibility. More specific details of prompts are presented in Appendix A.

6.A. AI Prompt Design and Functionality

6.A.1. Flowchart Generation

The flowchart generation process uses the generate_flow_chart_with_start() function, which constructs a Mermaid.js diagram based on user inputs. More details on prompt structure are available in the appendix. The prompt attempts to induce the LLM to simulate **coach-level thinking**, including: Conditional transitions (e.g., "if opponent posts, sweep"); Body mechanics ("shift hips", "control wrist"); and Legal boundaries for each ruleset. This was not completely successful however, as the flow-charts generated by chatGPT are highly linear in nature at present. Attempts to generate more inventive charts often resulted in a significant decrease in quality.

6.A.2. Image Analysis (Still Frame)

For generate_grappling_plan(), the prompt structure is fundamentally an image analysis, with guiding rails for where we want the analysis to go. More details on prompt structure are available in the appendix. This allows the LLM to not just describe what's happening, but **recommend technically sound next steps**, similar to what a live coach might shout from the sidelines.

6.A.3. Chat Simulation (Gracie Coaching)

The gracie_talk() function, anchors the coaching in the fundamental style of the coach in question – particularly their notable historic victories. More details on prompt structure are available in the appendix. This simulates **personality-driven coaching**, embedding each Gracie's voice and principles into a dialogue engine that enhances user engagement and retention.

6.B. Design Decisions

6.B.1. Prompts Structured for Coaching Simulation

Each prompt is carefully worded to elicit **strategic, teachable responses** rather than encyclopedic explanations. The outputs are constrained to specific formats (bullets, diagrams, concise paragraphs) to maintain usability. Instructions also include formatting gates—like "return Mermaid only"—to **cleanly separate logic from display**.

6.B.2. MMA vs BJJ Logic Differences

The app prompts dynamically shift based on the selected ruleset. More details on prompt structure are available in the appendix.

6.C. Error Handling and Fallback Logic

Generative models occasionally return malformed or incomplete responses. To mitigate this, flowchart responses are passed through a sanitize_mermaid() function which: Removes debug text; Extracts valid Mermaid blocks from mixed output; and Substitutes a predefined fallback flowchart if no Mermaid syntax is detected. Image and video functions return debug logs alongside the primary response, helping users and developers isolate issues, and the interface is designed to enforce required fields and rejects incomplete requests (e.g., missing image or keywords).

6.C. Modular and Extensible Design

All major analysis functions are built as **top-level modules** with minimal interdependencies. This modular approach allows for easy extension of the platform into new future domains

6.D. Analysis and Implementation Summary

Through disciplined prompt engineering, ruleset-sensitive logic, and robust error-handling infrastructure, *Jiu-Jitsu Genie* achieves a level of tactical analysis and coaching simulation that reflects how expert grapplers think, react, and plan. The modular AI architecture not only powers the current features but provides a clear path for future growth into new domains of embodied intelligence and competitive strategy.

7. Challenges and Iterations

The development of *Jiu-Jitsu Genie* surfaced several technical and conceptual challenges, primarily centered around the unpredictability of large language models (LLMs) when operating in tightly constrained, domain-specific workflows. While the underlying AI models were impressively capable in general grappling logic, content restrictions and the computer vision of these models struggled at times. Also, ensuring **structured, usable,**

and repeatable outputs required a continuous cycle of refinement, validation, and troubleshooting.

7.A. Inconsistent Mermaid Syntax Output

One of the most frequent obstacles encountered involved **inconsistent or malformed Mermaid syntax** from generative models during flowchart generation. Despite instructing the model to return *only* valid Mermaid objects, outputs often included:

- Additional explanation or commentary before the graph block
- Misplaced or missing quotation marks around node text
- Use of graph TD instead of graph LR despite explicit instructions
- Improper indentation or stray Markdown formatting (e.g., triple backticks)

Solution: Prompt Refinement + Sanitization Pipeline
To address this, a two-pronged approach was developed:

- **Prompt Engineering:** Repeatedly refined prompts to be more directive and verbose (e.g., "ONLY return Mermaid syntax. DO NOT include any explanation or markdown formatting.").
- **Sanitization Script:** Created a sanitize_mermaid() function to:
    - Strip surrounding text
    - Isolate the valid Mermaid block
    - Remove Markdown artifacts
    - Inject a default backup flowchart if no valid syntax was found

This solution ensured flowchart rendering remained functional even when the LLM response failed to comply precisely with format constraints.

7.B. Overly Verbose or Redundant LLM Responses

Another consistent challenge was the tendency of LLMs to **generate verbose or repetitive content**, especially in:

- Grappling plan recommendations
- Chat simulation with Gracie figures
- Video analysis descriptions

In many cases, responses exceeded the needed scope, included disclaimers, or restated known context. This undermined clarity, slowed UI responsiveness, and overwhelmed the visual layout.

Solution: Format Constraints + Text Pruning
The following tactics were used:

- Prompts were explicitly scoped with phrases like “Respond in 3 concise bullets,” or “Summarize in <100 words.”
- When applicable, response text was programmatically truncated or filtered before rendering (e.g., response.split("RESPONSE:")[1]).
- For conversational AI, limits were placed on token usage and reinforcement of tone: “Keep replies brief, tactical, and motivational—like a real coach speaking between rounds.”

7.C. Claude vs ChatGPT Behavior Variability

Early code development used **Claude from Anthropic** due to its ability to handle nuanced technical writing and code generation with less verbosity. However, switching between Claude and OpenAI models occasionally resulted in:

- Different interpretations of Mermaid layout rules
- Shifts in tone or verbosity in Gracie chats
- Variability in prompt compliance (e.g., Claude following instructions more conservatively)

Solution: Model-Aware Prompt Templates
To standardize performance across AI providers:

- Separate prompt templates were created for each model type
- Claude prompts emphasized clarity and minimalism, while ChatGPT prompts used more hierarchical structure and formatting emphasis
- Response testing was done iteratively with both models on sample cases (e.g., “generate a flowchart from closed guard for a 5’6” MMA fighter”).

7.D. Managing Edge Cases in Media Input
The integration of video analysis and frame trimming (via **FFmpeg**) presented issues when users:

- Uploaded large video files
- Failed to define time ranges for clipping
- Provided unsupported file formats

Solution: Input Validation + Robust Fallback Logic

- The app includes checks for video format, size, and duration.
- When time inputs were missing, the full video was analyzed by default.
- For critical media errors, the app returns descriptive error messages instead of crashing.

7.E. Iterative UI Adjustments

Streamlit's rapid prototyping made early interface builds easy, but long-term usability required several iterations:

- Collapsible panels were added to prevent visual overload from long chat histories or flowcharts.
- Session state was extended to persist core inputs across multiple features.
- Custom CSS was introduced to create a **visual hierarchy** (white to red belt) to make analysis results more intuitive and emotionally resonant.

8. Summary

In summary, *Jiu-Jitsu Genie* demonstrates how generative AI can be effectively applied to a high-context, adversarial domain like grappling strategy. The app was developed in response to a clear and persistent gap in training tools available to jiu-jitsu and MMA athletes—specifically, the absence of dynamic, individualized feedback that accounts for position, ruleset, and opponent behavior. By integrating structured natural language prompting with modular Python functions and a Streamlit-based user interface, the app delivers a flexible platform for tactical analysis, strategic planning, and simulated coaching conversations. Each module was built to reflect the real-time decision-making demands of combat sports, with persistent session state and customizable user inputs ensuring that output remains tailored and relevant across multiple interaction types.

Throughout development, the focus remained on generating technically sound, visually structured, and rule-compliant grappling recommendations. Flowcharts produced via Mermaid.js allow users to visually trace multiple strategy paths based on physical attributes and starting positions, with branching logic that incorporates likely counters and resets. Match and image analysis features offer fast, context-aware suggestions based on uploaded media or prompts, while the chat interface with named grappling masters enables character-driven, motivational feedback rooted in historical coaching styles. These outputs were made reliable through extensive prompt tuning, fallback handling, and testing with multiple language models (OpenAI and Claude), ensuring consistency across different use cases.

The architecture also prioritizes extensibility. Anime OODA Analysis, for example, reuses core prompt and visualization logic but shifts the domain to stylized, fictional combat scenarios—demonstrating the flexibility of the backend to accommodate both realistic and creative applications. Across all components, visual clarity, interactivity, and contextual accuracy were prioritized, resulting in an interface that supports both tactical exploration and reflective learning.

Ultimately, *Jiu-Jitsu Genie* offers more than technical novelty—it presents a new model for how combat athletes can engage with strategy development outside of the gym. By combining real-time interactivity, expert-level reasoning, and opponent-aware planning, the system functions as a digital coach and thinking partner, helping users refine their decision-making through structured, AI-guided feedback.

Appendix A: Sample Prompts and Outputs

Function	Full Prompt Used	Sample Output (Excerpt)	Format / Notes
generate_flow_chart_with_start()	Generate a technically detailed Mermaid flow chart showing branching jiu-jitsu sequences. STARTING POSITION: Half Guard The athlete has these attributes: 5'6", compact build, fast hip movement. Use MMA rules. Incorporate these specific techniques and ideas where technically sound: triangle, sweep, knee shield Important formatting requirements: 1. ONLY return the Mermaid diagram code with no explanations 2. Use "graph LR" 3. Use simple node IDs (A, B, C, etc.) 4. PUT DOUBLE QUOTES around all node text and arrow labels 5. Make node A the starting position 6. Create 4-6 branches with 2-3 follow-ups each	graph LR A["Half Guard"] -->|"Underhook & knee shield"| B["Deep Half"] A -->|"Frame and elevate"| C["Butterfly Hook Sweep"] B -->|"Drive under & lift"| D["Sweep to Top Half"] B -->|"Overhook switch"| E["Triangle Choke Setup"]	Parsed using sanitize_mermaid() and rendered via st.components.v1.html()
generate_grappling_plan()	The image is a still frame from a grappling match. The match has MMA rules. I want you to analyze: bottom player, and provide three options for the next immediate steps towards grappling moves that would work best for them given their current position. I want these steps in quick bullet format like: "1) <step>: toward <move>" Base the recommendations on historical MMA and Jiu-jitsu performance.	1) Pull opponent's head down: toward triangle choke 2) Control wrist and elevate hips: toward armbar setup 3) Lock right shin across back: toward clamp guard entry	Output cleaned using split("RESPONSE:"); styled as .highlight block

	I want you to also take special note of the following keywords: triangle, posture control, wrist grip The athlete has these attributes: 5'6", compact torso, strong grip		
analyse_grappling_match()	These images are frames from a grappling match. The match has IBJJF jiu-jitsu rules. I want you to analyze: top player, and provide an analysis of what went right or wrong during the match, along with some recommendations for the top player in their next match. I want you to also take special note of the following keywords for your analysis: butterfly hook, base, underhook	The top player failed to break the bottom player's butterfly hook early, allowing elevation. They also didn't pressure the underhook side. In future exchanges, shift hips forward earlier and control the near-side sleeve.	Plain text returned by generate_video_description(); displayed in scroll box
gracie_talk()	You are the jiu-jitsu master Rickson Gracie. I want you to have a conversation with the user based on that information, and have the conversation focused around improving the user's grappling and self-defense. Keep the tone direct and motivational, as if coaching in person. User Message: I'm struggling to hold mount on heavier opponents	Rickson: "You're thinking about control wrong. It's not about pushing down. It's about sticking to them. Lower your center of gravity. Wrap their hips. Control before you strike."	Stored in st.session_state.current_chat[] and displayed with markdown
next_move()	Dynamically reuses flowchart prompt, replacing STARTING POSITION with selected node text from user. E.g., "Triangle Choke Setup".	graph LR A["Triangle Choke Setup"] -->|"Adjust angle and lock"| B["Triangle Finish"] A -->|"Post and elevate hips"| C["Armbar Entry"] A -->|"Underhook and sweep"| D["Mount Transition"]	Enables flowchart traversal and recursive tactical planning

Appendix B: Development Challenges and Solutions Overview (Detailed)

Challenge	Description (from Implementation)	Solution (Aligned to Prompts & Architecture)
Inconsistent Mermaid Syntax from LLMs	Prompts like: 'ONLY return the mermaid object, and not any shoulder text whatsoever' were inconsistently followed. LLMs (especially GPT) often included markdown wrappers (` ``` `), narrative setup, or non-graph content before and after the flowchart, breaking rendering in the `generate_mermaid()` and `display_graph()` pipeline.	Refined prompts to emphasize exclusion rules. Wrote `sanitize_mermaid()` to extract valid syntax. Added default Mermaid fallback logic tied to position/ruleset.
Overly Verbose or Misaligned AI Output	LLMs exceeded prompt constraints, e.g., generating six recommendations when only three were requested, or writing historical background in `gracie_talk()`.	Used stricter prompt templates (e.g., “3 bullets max”), text slicing (`split('RESPONSE:')`), and Claude's more literal model when needed.
Ruleset Context Not Fully Integrated in Output	Despite ruleset indicators in prompt (e.g., 'Use MMA rules'), LLMs occasionally returned illegal techniques (e.g., strikes in IBJJF scenarios).	Prompt conditions now include ruleset-specific constraints (e.g., 'DO NOT include strikes' for IBJJF). MMA prompts request striking sequences explicitly.
Prompt-Driven Functions Generating Side Effects	`next_move()` and `generate_flow_chart_with_start()` reused chart logic but could misinterpret node labels as instructions if unescaped or ambiguous.	Ensured all recursive prompts specify `STARTING POSITION` with node text verbatim. Validated Mermaid begins with `graph LR`.
LLM Disobedience in Structured Image Prompts	`generate_grappling_plan()` prompts were ignored partially, e.g., analyzing the wrong player or omitting keywords.	Reworded prompt: 'ONLY analyze the bottom player. Base ALL steps on these keywords.' Claude used in critical contexts.
Claude vs GPT Prompt Interpretation Drift	Claude followed instructions precisely but lacked detail. GPT was richer but less disciplined.	Model-aware prompt variants were created. Claude prompts emphasized context repetition; GPT prompts emphasized formatting rules.
User Error with Uploads and State Loss	Missing inputs (images, measurables, timestamps) caused silent function failures or reset app context on rerun.	UI now includes validation checks. Inputs stored in `st.session_state` to persist between app states and reruns.
No Output Fallbacks = Broken UI Elements	When LLMs failed to return valid Mermaid, the UI showed blank spaces.	Inserted backup Mermaid objects based on position and ruleset, ensuring UI continuity and guidance even on API failure.