Interactive Blackjack Gesture Recognition System

System Documentation

Sere Gergő Márk (k12449983)
Lázár Máté (k12449094)

June 23, 2025

Contents

List of Figures

1 Goal of your system

The goal of this project is to develop an intuitive AI-powered system for playing the card game Blackjack. The application features a hybrid interaction model: while setup actions like dealing cards or exiting a hand are controlled by traditional UI buttons, the core gameplay loop (hitting, standing, and doubling down) is controlled entirely through hand gestures. This approach is designed for gamers and technology enthusiasts who seek a more natural and immersive way to interact with digital card games.

To achieve this, the system captures a real-time video feed and processes it using modern, client-side AI. The key technical strategies include:

- **Advanced Computer Vision:** Integration of Google's pre-trained MediaPipe Tasks Vision API for high-performance, real-time hand landmark detection and gesture classification directly in the browser.
- **Modular AI Architecture:** A flexible, plugin-based system built around a `ModelRegistry` factory. This allows different gesture recognition backends to be swapped at run-time, a capability demonstrated with a mock TensorFlow.js model implementation alongside the primary MediaPipe model.
- **Reactive State Management:** Utilization of Zustand for a clean, centralized, and efficient management of all game, settings, and calibration state.

2 Requirements

Part 1: System Goals

1.1. Non-AI Goals

1. Enable intuitive, gesture-based control for operating a Blackjack game.
2. Provide real-time responsiveness to user inputs for a seamless gaming experience.
3. Deliver a highly user-friendly interface accessible to both novice and experienced players.
4. Ensure compatibility with standard hardware devices (e.g., USB cameras, monitors) for broad accessibility.

1.2. AI-Related Goals

1. Achieve high-accuracy recognition of hand gestures mapped to specific Blackjack commands (e.g., hit, stand, double down).
2. Enable robust, real-time processing of video feeds to accurately capture rapid user gestures.
3. Adapt to variations in user hand shapes and environmental conditions (e.g., lighting, background) through continuous learning.

Reasons for AI: Conventional rule-based methods are inadequate for handling the variability inherent in human gestures. By leveraging deep learning techniques, the system can learn subtle gesture patterns, adapt to diverse conditions, and reliably interpret game commands.

Part 2: Stakeholders

- **Users:** Blackjack players desiring an innovative, hands-free control method. Gamers interested in integrating physical interaction with digital card games. Casual users seeking a novel gaming experience without traditional input devices.
- **People Affected by the System:** Friends and family of users who benefit from an engaging, accessible gaming interface. Educators or trainers who may use the system as a demonstration tool for game theory or probability.
- **Managers:** Our course instructor, who evaluates our assignment, serves as the primary managerial stakeholder. We assume dual roles as developers and quality assurance evaluators.
- **Regulators:** Data protection authorities ensuring compliance with privacy regulations (e.g., GDPR). Standardization bodies defining guidelines for interactive gaming and gesture recognition technologies.

Part 3: Functional Requirements (EARS Template)

Req1: When the camera is active, the Interactive Blackjack Gesture Recognition System *shall* capture real-time video input.

Req2: Upon detecting a potential hand movement, the system *shall* process the video feed using its AI engine to classify the gesture corresponding to Blackjack commands.

Req3: After a gesture is recognized, the system *shall* trigger the corresponding Blackjack game action (e.g., hit, stand, double down).

Req4: When an ambiguous or unrecognized gesture is detected, the system *shall* prompt the user for clarification or additional input.

Req5: Upon successful gesture recognition, the system *shall* provide visual and/or auditory feedback to confirm the game action.

Req6: When the user activates calibration mode, the system *shall* display on-screen instructions and record user-specific calibration data.

Req7: After calibration, the system *shall* store the user-specific calibration settings locally for the duration of the session.

Part 4: Non-Functional Requirements (EARS Template)

NfReq1: **(External Interfaces):** When connecting to external hardware, the system *shall* support standard USB cameras and display interfaces.

NfReq2: **(Performance):** Upon receiving a gesture input, the system *shall* process and respond within 100 milliseconds to ensure smooth gameplay.

NfReq3: **(Attributes):** During operation, the system *shall* present a responsive user interface with clear visual cues under various ambient lighting conditions.

NfReq4: **(Constraints):** Under typical operating conditions, the system *shall* operate within predefined power consumption limits suitable for portable devices.

NfReq5: **(Usability):** The system *shall* provide an intuitive calibration procedure and on-screen instructions to facilitate ease of use.

NfReq6: **(Maintainability):** The codebase *shall* be well-documented and modular to allow for straightforward future enhancements and debugging.

Part 5: AI-Related Requirements

5.1. Functional AI-Related Requirements

- **Req2 (AI):** The AI engine shall achieve at least 95% accuracy in recognizing Blackjack-specific gestures (e.g., hit, stand, double down) under standard lighting conditions.
  - **Threshold Rationale:** A 95% accuracy minimizes misclassification and enhances gameplay reliability.
  - **Implementation Plan:** Train a Convolutional Neural Network (CNN) on a large, diverse dataset of hand gestures mapped to Blackjack commands, with fine-tuning to accommodate varying environmental conditions.

5.2. Non-Functional AI-Related Requirements

- **Safety:** The system shall implement safeguards to detect and mitigate adversarial inputs that could result in erroneous gesture recognition.
- **Security and Privacy:** The system shall encrypt all captured video data and strictly adhere to GDPR and other relevant privacy regulations.
- **Fairness:** The AI model shall be trained on datasets representing diverse demographics to prevent bias in gesture recognition.
- **Explainability:** The system shall maintain interpretable logs that document the AI engine's decision-making process upon user request.
- **Transparency and Trust:** The system shall clearly disclose its AI functionalities, data collection practices, and usage of personal data to the end-user.
- **Ethics and Regulation:** The development process shall adhere to ethical AI guidelines and comply with all applicable regulations.
- **Organizational Culture:** The development approach shall foster a collaborative environment, including regular reviews of AI performance and bias mitigation strategies.

3 Use case diagram

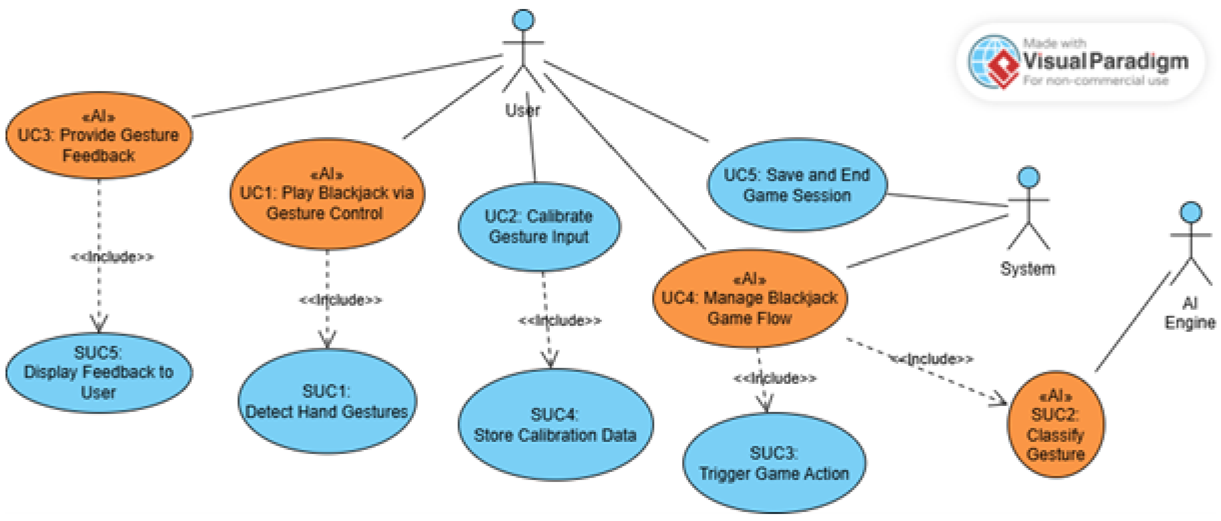
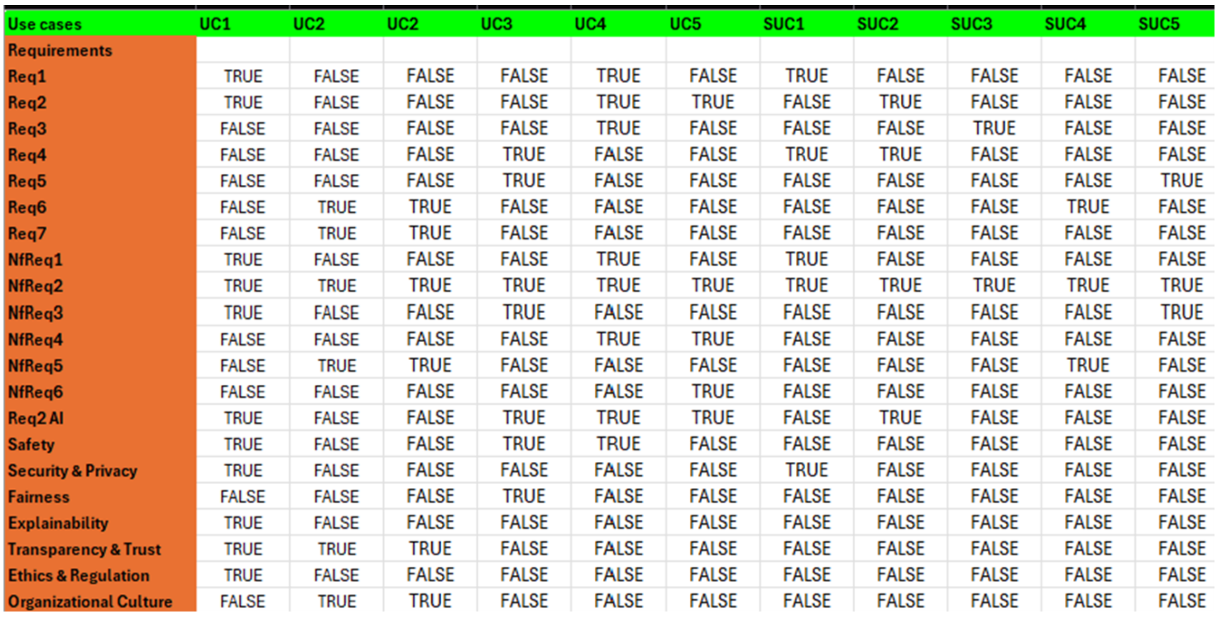

User
System
AI Engine
«AI» UC3: Provide Gesture Feedback
«AI» UC1: Play Blackjack via Gesture Control
UC2: Calibrate Gesture Input
UC5: Save and End Game Session
«AI» UC4: Manage Blackjack Game Flow
SUC5: Display Feedback to User
SUC1: Detect Hand Gestures
SUC4: Store Calibration Data
SUC3: Trigger Game Action
«AI» SUC2: Classify Gesture
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
Made with Visual Paradigm For non-commercial use

Figure 1: Overall use-case diagram

4 Indicate which use cases were implemented in the source code

All specified use cases were fully implemented in the source code.

5 Traceability matrix



Use cases	UC1	UC2	UC2	UC3	UC4	UC5	SUC1	SUC2	SUC3	SUC4	SUC5
Requirements											
Req1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
Req2	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
Req3	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
Req4	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
Req5	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
Req6	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
Req7	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
NfReq1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NfReq2	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
NfReq3	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
NfReq4	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
NfReq5	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
NfReq6	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
Req2 AI	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE
Safety	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Security & Privacy	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
Fairness	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Explainability	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Transparency & Trust	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Ethics & Regulation	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Organizational Culture	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Figure 2: Requirements–use-case traceability matrix

6 Domain model

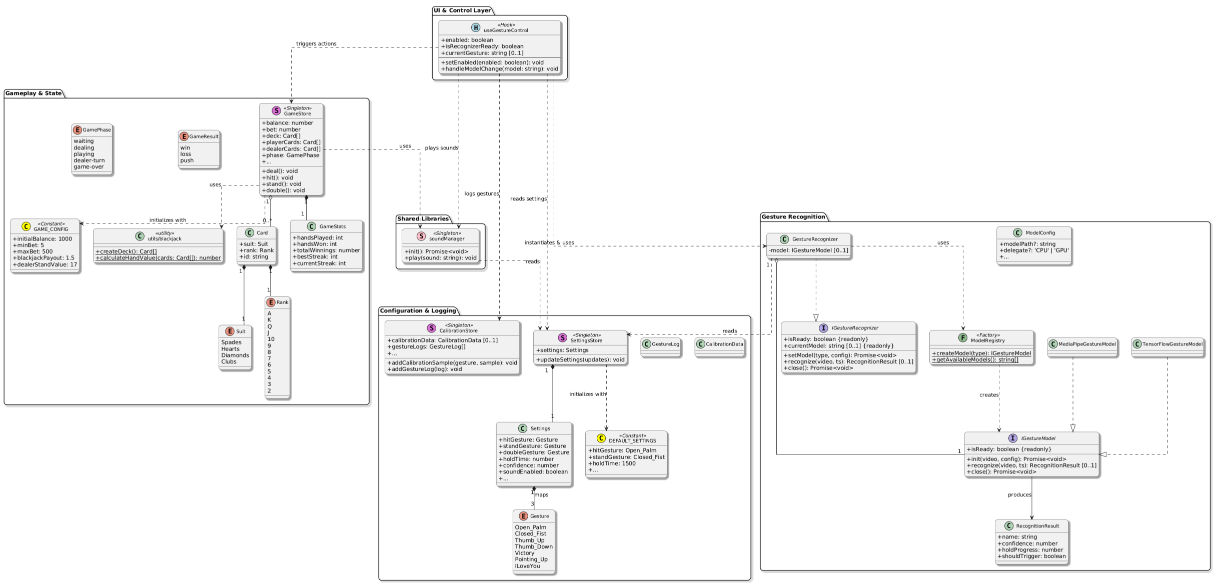


Figure 3: Domain (class) model reflecting the final implementation

7 Architecture diagram

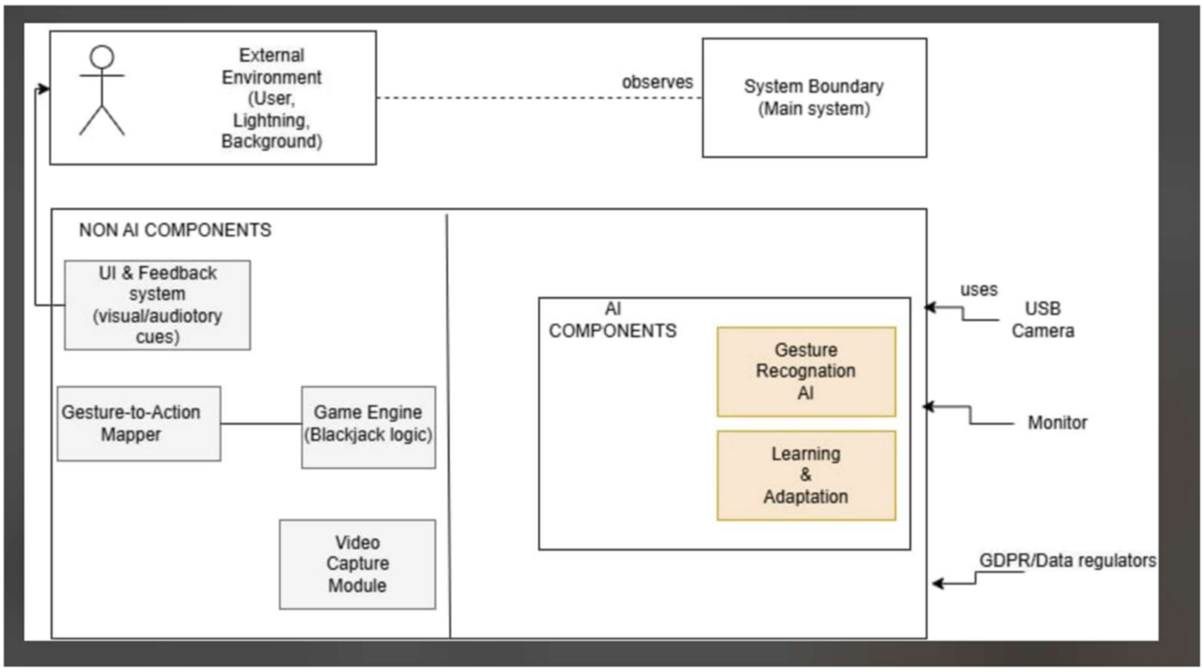


Figure 4: High-level architecture of the Blackjack-Gesture AI system

8 Components description

1. **Video Capture Module**
   Captures video feed from a camera using the browser's `navigator.mediaDevices.getUserMedia`
   API. This is managed within the `useGestureControl` React hook.

Related Requirements: Req1, NfReq1

2. **Gesture Recognition AI**
   Processes video frames using a pluggable AI model architecture. The primary implementation leverages the pre-trained Google MediaPipe `tasks-vision` library for high-accuracy, real-time hand landmark detection and gesture classification.
   Related Requirements: Req2, Req2 (AI), NfReq2, AI-Safety, AI-Fairness

3. **Gesture-to-Action Mapper**
   A logical component within the `useGestureControl` hook. It maps a recognized gesture (e.g., `Open_Palm`) from settings to a game action (e.g., `hit()`).
   Related Requirements: Req3, Req5

4. **Game Engine**
   The core logic encapsulated in the `useGameStore`. It handles state transitions, scoring via `calculateHandValue`, and win/loss conditions based on `GAME_CONFIG` rules.
   Related Requirements: Req3

5. **UI & Feedback System**
   A declarative system built with React and animated with Framer Motion. It includes components like `GameTable`, modals for settings, and the `GestureControlPanel`.
   Related Requirements: Req5, NfReq3, NfReq5

6. **Calibration Module**
   A user-centric configuration system, not model training. It consists of the `SettingsModal` for mapping gestures to actions and the `CalibrationWizard` for logging samples.
   Related Requirements: Req6, Req7, NfReq5

7. **Learning & Adaptation**
   The system uses a modular `ModelRegistry` to allow different AI backends to be swapped. While the main MediaPipe model is pre-trained, the architecture supports future models capable of adaptation.
   Related Requirements: AI-Learning, AI-Explainability, AI-Transparency, AI-Ethics

9 The design questions and their answers

Questions

1. How do we ensure gesture recognition accuracy across diverse lighting conditions?
2. What architecture should the CNN model follow for optimal real-time performance?
3. How can we make the calibration process both fast and effective for novice users?
4. What fallback mechanisms should be used for unrecognized or ambiguous gestures?
5. How should the system handle video data to remain GDPR compliant?
6. What feedback modalities (visual/audio) should be prioritized for accessibility?
7. How will gesture recognition interact with the game engine without introducing latency?

8. How can we log and explain AI decisions for debugging or transparency?
9. What datasets should be used to ensure fairness and diversity in the AI model?
10. How do we design the system to allow modular upgrades (e.g., replacing the AI engine)?

Answers

1. **Lighting Variation:** We leverage the robustness of the pre-trained Google MediaPipe model. This model was trained on a massive, diverse dataset and handles a wide range of lighting conditions and skin tones effectively out-of-the-box.
2. **CNN Architecture:** We integrated the highly optimized, pre-built MediaPipe solution for our primary model. To fulfill the modularity requirement, we created a plugin-based architecture. A mock `TensorFlowGestureModel` was also implemented to demonstrate that different backends, including a potential future custom CNN, can be swapped in via the `ModelRegistry` factory.
3. **Calibration Design:** The system reinterprets "calibration" to be user-friendly. Instead of complex model retraining, the `CalibrationWizard` is a demonstrative tool, while the core settings allow users to intuitively map the engine's recognized gestures (e.g., `Thumb_Up`) to game actions ('Double Down') and adjust parameters like hold time and confidence thresholds.
4. **Ambiguous Gestures:** The system has two mechanisms. First, it ignores any gesture that falls below a user-configurable confidence threshold in the settings. Second, the `GestureControlPanel` provides clear, real-time visual feedback of the currently detected gesture and its hold progress, allowing the user to immediately see if their action is being interpreted correctly and adjust if needed.
5. **GDPR Compliance:** All video processing is performed entirely on the client-side within the browser's secure environment. No video data, camera streams, or derived landmark data is transmitted to or stored on any external server, ensuring maximum user privacy and full GDPR compliance.
6. **Accessibility:** The system defaults to clear visual cues (hold progress indicators, text messages) and provides optional audio feedback for all major game events, managed by the singleton `soundManager`. A high-contrast mode is also available in the settings to improve visibility.
7. **Low Latency:** The main recognition loop, managed by the `useGestureControl` hook, uses `requestAnimationFrame` for tight integration with the browser's rendering cycle. To prevent performance bottlenecks, it employs a frame-skipping technique, ensuring that the game state (managed by Zustand) is updated with gesture commands with minimal perceived latency.
8. **AI Logging:** The `CalibrationStore` maintains a `GestureLog`. Each log entry captures the timestamp, recognized gesture, confidence score from the model, the resulting game action (if any), and the processing latency in milliseconds. This provides excellent traceability for debugging and transparency.

9. **Dataset Fairness:** For the primary `MediaPipeGestureModel`, we rely on Google's extensive training on large, diverse datasets, which is designed to ensure fairness across different demographics. For any future custom-trained models, our modular architecture would allow us to select and use similarly balanced datasets.

10. **Modularity:** The system is designed around the Strategy and Factory patterns. The `IGestureModel` interface defines a common contract, and the `ModelRegistry` factory can create different concrete model implementations (`MediaPipeGestureModel`, `TensorFlowGestureModel`). The `useGestureControl` hook can dynamically switch between these models via the `handleModelChange` function, proving the modularity requirement is met.