Engineering of AI intensive Systems
Vcon4 Documentation

Jan Holzweber
j.holzweber@gmx.at
K01556234

Niyazi Isgandarov
niyaziisgandarov07@gmail.com
K12349124

Description.....4
Goals.....4
Requirements.....4
Functional Requirements.....4
Non-Functional Requirements.....5
AI-Related Requirements.....5
Use Case Descriptions.....6
Implemented.....6
Not Implemented.....12
Use Case Diagram.....14
Traceability Matrix.....15
Domain Model.....16
Architecture Diagram.....17
Component Description.....18
Design Questions.....20
Questions.....20
Answers.....22

Description

We want to implement a simple app called Vcon4 (virtually connect four). It is an implementation of the Connect Four game, where two players drop colors in turns into six rows by seven columns field. The objective of the game is to be the first to connect four of his/hers colors either vertically, horizontally, or diagonally. Our game will not be played physically, but virtually and will be controlled through hand gestures via a camera.

Goals

The goal is to provide the user with a working connect 4 game, that can be controlled via hand gestures. It should not only allow you to play the game against another human player but also provide a single-player mode against an AI opponent.

Requirements

Functional Requirements

Req1. Vcon4 **shall** allow users to play by taking turns.

Req2. Vcon4 **shall** allow users to drop colors into columns.

Req3. Vcon4 **shall** display a winner when somebody connects four colors.

Req4. Vcon4 **shall** display which player has his/her turn.

Req5. Vcon4 **shall** allow users to control the game via hand gestures.

Req6. Vcon4 **shall** provide a 2-player mode to play against each other.

Req7. Vcon4 **shall** provide a single-player mode against an AI opponent.

Req8. When one of the players connects 4 colors, Vcon4 **shall** display a victory screen.

Req9. Vcon4 **shall** provide different difficulty levels for the AI opponent.

Req10. Vcon4 **shall** provide a main menu.

Req11. When the game is active, the gesture detection **shall** show what gesture it currently is detecting.

Req12. Vcon4 **shall** provide a high score menu.

Non-Functional Requirements

NfReq1. The AI player **shall** perform its move in less than 1 second.

NfReq2. The gesture detection **shall** work in real time.

AI-Related Requirements

Req7 & Req 9 & NfReq1 The AI Opponent

The AI player to play against is supposed to be quick, that's why it should make its move in within less than one second. Implementation-wise, will use a search tree with decision rules to provide the AI with the next move to make. These search trees give a look-ahead for all future moves. You can increase or decrease difficulty based on how many moves it looks ahead.

Req5 & NfReq2 & Req11 The Gesture Detecion

The gesture detection should work in real-time, as it is used to make moves, decide game mode, etc. Waiting for the controls is a no-go in casual games as they should stay casual. We plan to use OpenCV with contour and convex hull detection but this is still subject to change.
Regarding safety and data privacy, no images of people are stored or distributed. Furthermore, for explainability, gesture detection should display what it is focusing on and what it is currently detecting, so that the human knows, what is going on behind the scene.

Use Case Descriptions

Implemented

UC: Drop Color

Use case: Drop Color		
ID	UC1	
Description	Drops the color associated with player into the playing field	
Actors	Player, Hand Gesture Recognition Model, Game Engine	
Stakeholders:	Player, Game Company Manager	
Pre-Conditions	The game has started	
Success end condition:	The color has been dropped	
Failure end condition:	The game cannot identify a hand gesture	
Main Success Scenario		**Linked UCs**
1	The player makes the hand gesture to drop a color	
2	The system accesses the video	
3	The hand gesture recognition model identfies the hand gesture	SUC1
4	The game engine checks if it is a valid move	SUC2
5	The game engine drops the color at the correct position in the playing field	SUC5
Alternative Scenarios		
4.A1	The game engine determines that it is not a valid move (e.g. column is full)	
4.A2	The game displays a message, that the move is invalid	
4.A3	Go back to step 1	
Exception Scenario		
3.A1	The hand gesture recognition model can not identify the gesture	
3.A2	The game displays an error message with the problem	
3.A3	The game engine promts the user to redo the gesture	

UC: Start a Single Player Game

Use case: Start a Single Player Game		
ID	UC2	
Description	A player starts the game by selecting the "Start 1P Game" Option	
Actors	Player, Game Engine	
Stakeholders:	Player, Game Company Manager	
Pre-Conditions	The programm has been started	
Success end condition:	A 1P game starts	
Failure end condition:	The Hand Gesture Recognition Model fails to identify a hand gesture	
Main Success Scenario		**Linked UCs**
1	The player makes the hand gesture to choose "Start 1P Game"	
2	The system accesses the video feed	
3	The hand gesture recognition model identfies the hand gesture	SUC1
4	The system provides gesture calibration	
5	The game engine starts the 1P game	
Alternative Scenarios		
4.A1	The hand gesture recognition model starts a 2 player game instead	
4.A2	The game starts	
4.A3	The player manually ends the game	
4.A4	Go back to step 1	
Exception Scenario		
3.A1	The hand gesture recognition model can not identify the gesture	
3.A2	The game displays an error message with the problem	
3.A3	The game engine promts the user to redo the gesture	

UC: Win the game

Use Case: Win the game		
ID	UC3	
Description	A player or the AI Agent wins the game by connecting 4	
Actors	Game Engine	
Stakeholders:	Player, AI Agent, Game Engine, Developers	
Pre-Conditions	A player or the AI connected four of the same colors	
Success end condition:	A Victory Screen is displayed and adds the results to the highscore board	
Failure end condition:	The game engine fails to identify that the game is over	
Main Success Scenario		**Linked UCs**
1	The game engine recognizes that 4 the same color are connected	
2	The game engine processes the win	SUC2
3	The game engine displays a victory screen	
3	The game engine shifts to the high score screen	
4	The game engine displays where you rank with your score	
Alternative Scenarios		
4.A1	The player presses restart on the victory screen	
4.A2	The game engine omits showing highscores	
4.A3	The game engine restarts the game	
Exception Scenario		
1.A1	The game engine fails to identify the victory	
1.A2	The game engine fails to finish the game	
1.A3	The game has to be manually reset to the start	
1.A4	Restart the game	

UC: Choose Game difficulty

Use Case: Choosing the game difficulty		
ID	UC5	
Description	A player has option to choose the game level	
Actors	Game Engine, Players	
Stakeholders:	Players, Game Developers, Game Engine	
Pre-Conditions	The player has accessed the game settings or difficulty selection screen	
Success end condition:	The player successfully selects the desired game difficulty	
Failure end condition:	The player is unable to select the desired game difficulty, or the selection process encounters errors	
Main Success Scenario		**Linked UCs**
1	The player navigates to the game settings or difficulty selection screen	
2	The available difficulty options are displayed	
3	The player selects the desired game difficulty	SUC1
4	The selected game difficulty is confirmed and applied to the gameplay settings	
5	The player enters the game environment with the selected difficulty settings applied, ready to face the challenges ahead.	
Alternative Scenarios		
4.A1	Customization options within each difficulty level	
4.A2	Player-customized difficulty settings available	
4.A3	Dynamic difficulty adjustment based on player performance	
4.A4	Brief description provided for chosen difficulty level	
Exception Scenario		
1.A1	Error accessing game settings; displays retry message	
1.A2	Technical glitch prevents difficulty selection; prompt to restart	
1.A3	Player input failure due to software/hardware issue	
1.A4	Unexpected error during selection process; provides error message	

SUC: Analyze Video Feed

Supporting Use case: Analyze video feed		
ID	SUC1	
Description	Analyzes the video feed to identify the hand gesture	
Actors	Hand Gesture Recognition Model	
Stakeholders:	Player, Game Engine	
Pre-Conditions	The video feed is accessible	
Success end condition:	The identified hand gesture is returned to the game engine	
Failure end condition:	The hand gesture cannot be identified	
Main Success Scenario		
1	The Hand Gesture Recognition Model analyzes the video feed	
2	The Hand Gesture Recognition Model identifies the gesture	SUC3
3	The Hand Gesture Recognition Model returns the identified gesture	
Alternative Scenarios		
Exception Scenario		
2.A1	The Hand Gesture Recognition Model fails to identify the gesture	
2.A2	The Game Engine tasks the user to redo the gesture	

SUC: Analyze Game State

Supporting Use case: Analyze Game State		
ID	SUC2	
Description	Analyzes the the game state for a win	
Actors	Game Engine	
Stakeholders:	Player, AI Agent, Game Engine	
Pre-Conditions	The game has started	
Success end condition:	A valid game state is returned	
Failure end condition:	The game engine fails to identify the game state	
Main Success Scenario		
1	The game engie analyzes the game state	
2	The game engine recognizes changes	
3	The game engine returns the updated game state to players	
Alternative Scenarios		
Exception Scenario		
2.A1	The game engine missidentifies the game state	
2.A2	The game engine stops and must be restarted	

SUC: Identify Hand Gesture

Supporting Use case: Identify hand gesture	
ID	SUC3
Description	The system accurately identifies a hand gesture
Actors	Hand Gesture Recognition Model
Stakeholders:	Players, Game Developers, Game Engine
Pre-Conditions	The system is ready to capture hand gestures
Success end condition:	The hand gesture is correctly identified by the system
Failure end condition:	The hand gesture is not recognized or misinterpreted by the system

Main Success Scenario	
1	The hand gesture recognition model activates in response to player input or system initialization
2	The system captures the live video feed or image containing the hand gesture
3	The hand gesture recognition model accurately identifies the hand gesture from the captured data
Alternative Scenarios	
Exception Scenario	
2.A1	The hand gesture recognition model fails to detect any hand gesture in the captured data
2.A2	The hand gesture recognition model detects a hand gesture, but it is misinterpreted as a different gesture or command

SUC: Modify Game State

Supporting Use case: Modify Game State	
ID	SUC5
Description	The system allows for modification of the game state, such as changing player position or altering game parameters
Actors	Game Engine, Game Designer
Stakeholders:	Players, Game Developers
Pre-Conditions	The game is running and in a state that allows modification
Success end condition:	The game state is successfully modified as intended
Failure end condition:	The game state is not modified or is modified incorrectly

Main Success Scenario	
1	The game engine receives instructions from the game designer or player to modify the game state
2	The game engine processes the modification request and updates the game state accordingly
3	The modified game state reflects the desired changes, such as updating player position, adjusting environmental elements, or altering game parameters
Alternative Scenarios	
Exception Scenario	
2.A1	Incorrect modification of game state leads to unintended consequences
2.A2	Technical issues prevent successful modification of game state

Not Implemented

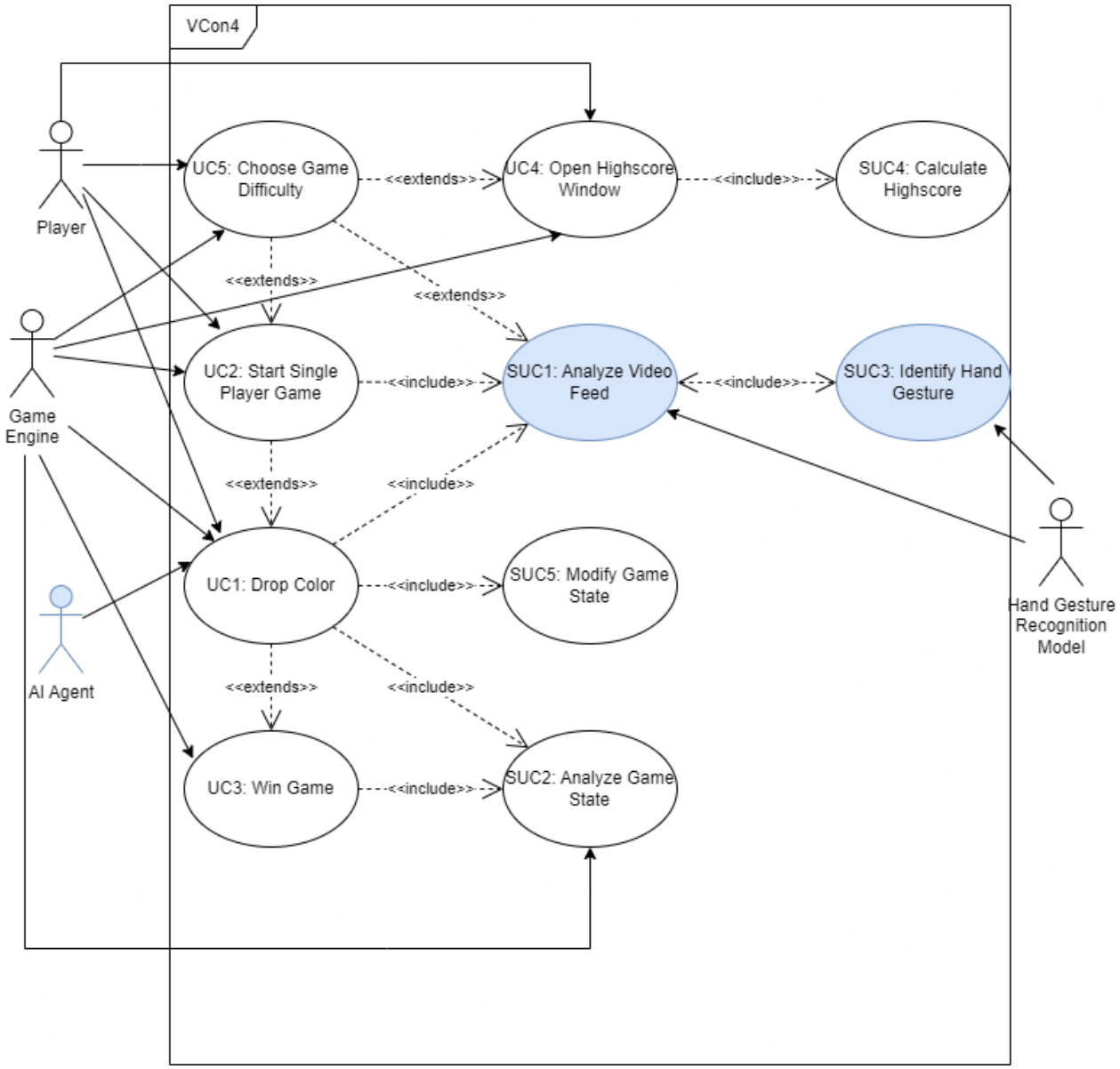
UC: Open Highscore Window

Use Case: Open Highscore Window		
ID	UC4	
Description	The player opens the highscore window to display all highscores	
Actors	Game Engine, Players	
Stakeholders:	Players, Game Developers	
Pre-Conditions	The game is running, The highscore data is available and accessible	
Success end condition:	The highscore windows is successfully displayed, showing top scores achieved by players	
Failure end condition:	The highscore window fails to open or display properly	
Main Success Scenario		**Linked UCs**
1	The player navigates to the game menu	
2	Within the game menu, the player selects the option to view the highscore window	
3	The system responds to the player's input by opnening the highscore window	
4	The highscore window is displayed, showing the list of top scores achieved players.	SUC4
5	The player can scroll thgrough the highscore list to view different entries and ther associated scores	
Alternative Scenarios		
4.A1	Highscore window displays additional info like date and mode	
4.A2	Highscore window offers filtering options	
4.A3	Highscore window includes interactive sorting buttons	
4.A4	Highscore window opening accompanied by visual effects	
Exception Scenario		
1.A1	Error accessing highscore data; displays retry message	
1.A2	Technical glitch prevents highscore window from opening	
1.A3	Insufficient memory leads to incomplete highscore window	
1.A4	Player input failure due to software or hardware issue	

SUC: Calculate Highscore

Supporting Use case: Calculate Highscore		
ID	SUC4	
Description	The system accurately calculates the player's highscore	
Actors	Game Engine	
Stakeholders:	Players, Game Developers, Game Engine	
Pre-Conditions	The player has completed a gameplay session	
Success end condition:	The highscore is correctly calculated and displayed	
Failure end condition:	The highscore calculation encounters errors or inaccuracies	
Main Success Scenario		
1	The game engine receives data regarding the player's performance during the gameplay session	
2	The game engine applies the appropriate scoring algorithm to the player's actions and achievements	
3	The game engine accurately calculates the player's highscore based on the applied scoring algorithm	
Alternative Scenarios		
Exception Scenario		
2.A1	Inconsistent scoring algorithms lead to varying highscores	
2.A2	Technical limitations cause issues with handling extremely high scores	

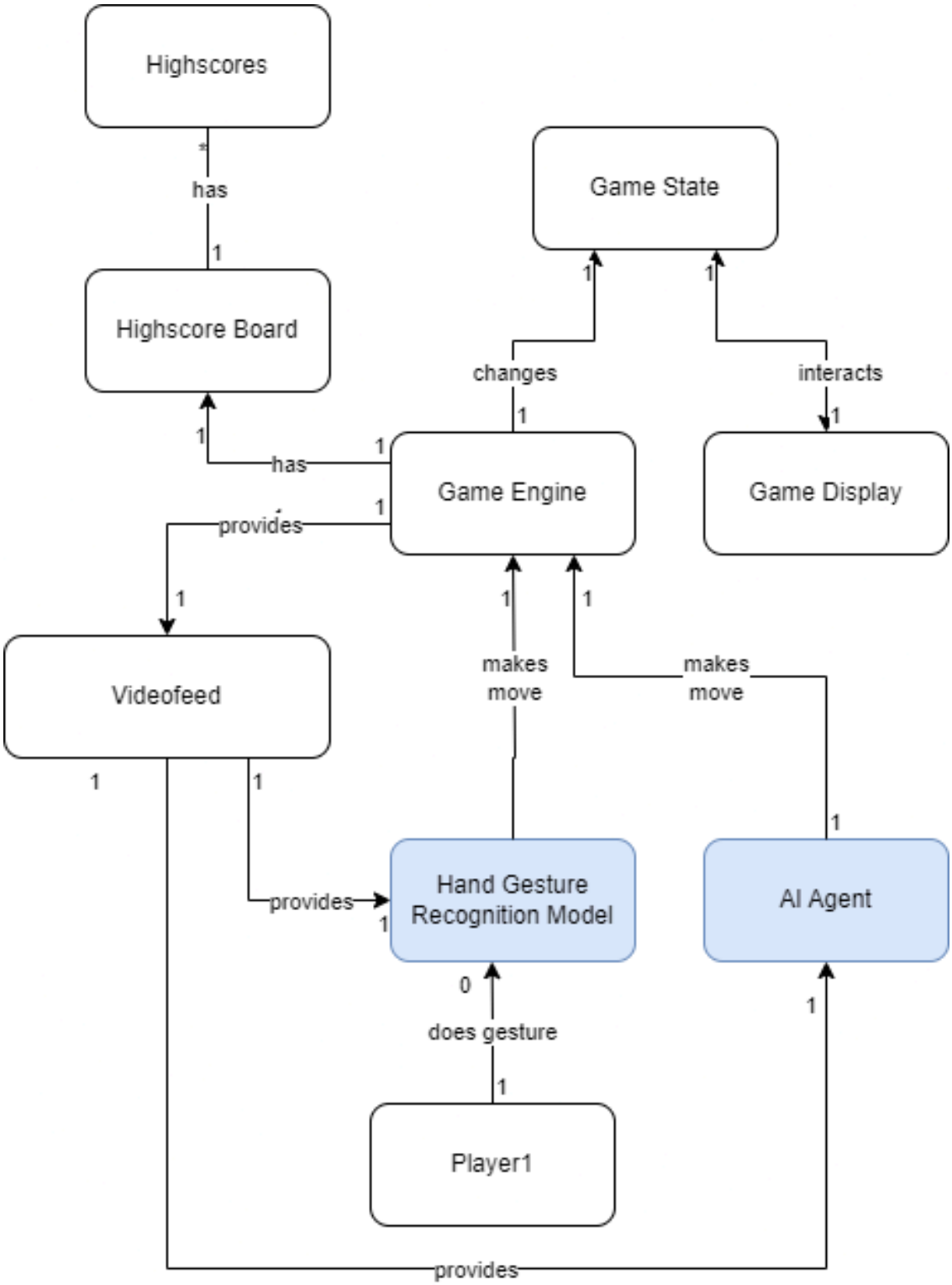
Use Case Diagram



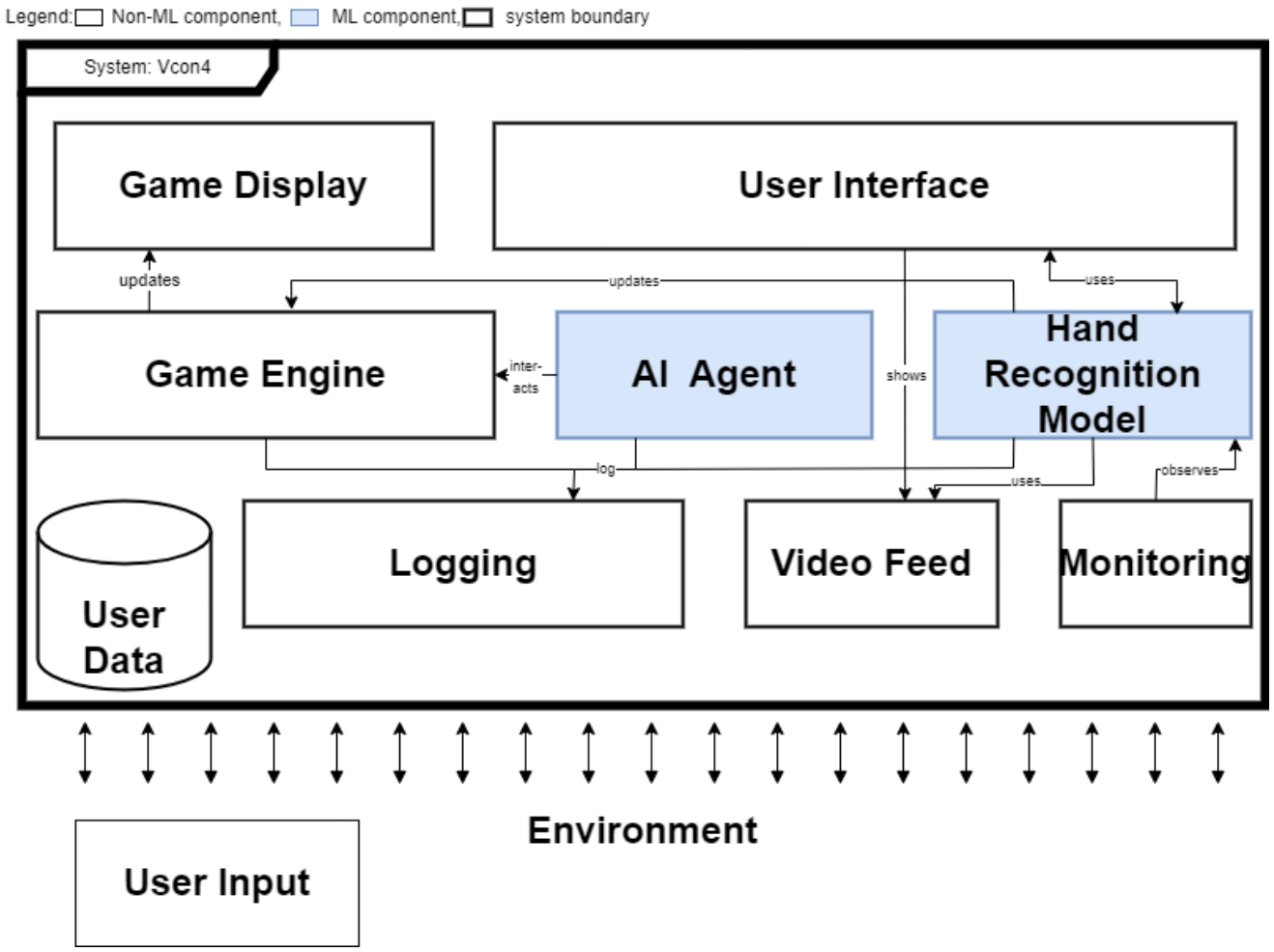
Traceability Matrix

Use cases	UC1 Drop Color	SUC1 Analyze video feed	UC2 Start a Single Player Game	SUC2 Analyze Game State	UC3 Win the game	SUC3 Identify hand gesture	UC4 Open Highscore Window	SUC4 Calculate Highscore	UC5 Choosing the game difficulty	SUC5 Modify Game State	Use cases
Req1	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	Req1
Req2	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	Req2
Req3	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	Req3
Req4	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	Req4
Req5	☐	☐	☐	☐	☑	☐	☐	☐	☑	☐	Req5
Req6	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	Req6
Req7	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	Req7
Req8	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	Req8
Req9	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐	Req9
Req10	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	Req10
Req11	☐	☐	☐	☑	☐	☑	☐	☐	☐	☐	Req11
Req12	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	Req12
NfReq1	☑	☐	☐	☑	☐	☑	☐	☐	☐	☐	NfReq1
NfReq2	☐	☑	☐	☐	☐	☐	☑	☑	☐	☐	NfReq2

Domain Model



Architecture Diagram



Component Description

Game Display

The display shows the overall game state (all the colors and it can only be changed and updated by the game engine, so we strictly adhere to the MVC design pattern.

User Interface

The user interface is the display of the computer where the game is played on. It shows the video feed and enables the user to use hand gesture to make moves in the game. Together with the hand recognition model it acts as the controller in the MVC design pattern.

Game Engine

The game engine controls the game. It stores all related information, like the game state, which user has his turn, etc. It acts as the model in the MVC design pattern.

AI Agent

The AI agent is the computer player we can play against. He does his turns based on heuristics and a search algorithm. Depending on how sophisticated this heuristic is, we can adjust the difficulty of the player.

Hand Recognition Model

The hand recognition model acts as the brain of the hand gesture recognition. Based on these gestures, it sends the move updates to the game engine. Together with the user interface, it acts as the controller in the MVC design pattern.

Video Feed

The video feed is what the camera detects. It should stream the video so we can detect hand gestures in real-time.

Monitoring

The monitoring works as a check for hand gesture recognition so we know what is happening inside this model.

Logging

The logging module provides us with information on what is currently happening within the game. It provides us with updates on the game engine and the display, so we can check what is going on and if moves were done the right way

User Data

The user data module stores player results and accounts.

User Input

The user input is the gestures the player does when making moves.

Design Questions

Questions

1. Game Display:

    - How can we ensure the game display remains responsive and visually appealing across different devices and screen sizes?
    - What strategies can be implemented to optimize rendering performance while maintaining high-quality graphics?

2. User Interface:

    - How can we design the user interface to be intuitive and easily navigable for players of all ages and skill levels?
    - What accessibility features should be incorporated to accommodate users with disabilities or special needs?

3. Game Engine:

    - What design patterns and data structures should be utilized to efficiently manage game state and logic?
    - How can we ensure the game engine remains modular and extensible to accommodate future feature additions or updates?

4. AI Agent:

    - What algorithms and techniques should be employed to balance the AI opponent's difficulty levels effectively?
    - How can we design the AI to adapt its strategies based on the player's skill level and gameplay patterns?

5. Hand Recognition Model:

- What preprocessing techniques should be applied to enhance the accuracy and robustness of hand gesture detection?
- How can we optimize the hand recognition model to achieve real-time performance without sacrificing accuracy?

6. Video Feed:

- How can we ensure consistent and reliable video feed quality across different lighting conditions and camera setups?
- What compression methods or streaming protocols should be implemented to minimize latency and bandwidth usage?

7. Monitoring and Logging:

- What key metrics should be monitored to assess the performance and stability of the overall system?
- How can we implement effective logging mechanisms to capture relevant data for debugging and analysis purposes?

8. User Data:

- What measures should be taken to ensure the security and privacy of user data, especially considering the use of hand gestures?
- How can we provide users with control over their data and comply with relevant data protection regulations?

9. User Input:

- How can we optimize input processing to minimize latency and improve responsiveness during gameplay?
- What input validation techniques should be implemented to prevent unintended actions or malicious inputs?

10. Component Integration:

- How can we facilitate seamless integration and communication between different components of the system?
- What protocols or APIs should be established to enable interoperability and data exchange between modules?

Answers

1. Game Display:

Implement SVG graphics for responsiveness and visual appeal, ensuring compatibility across devices.

2. User Interface:

Design an intuitive interface with clear navigation, incorporating accessibility features for diverse users.

3. Game Engine:

Utilize MVC architecture to separate game logic from presentation, optimizing performance with efficient data structures.

4. AI Agent:

Implement minimax algorithm with alpha-beta pruning for quick decision-making within one second.

5. Hand Recognition Model:

Utilize OpenCV for real-time hand gesture detection, displaying focused areas for user feedback.

6. Video Feed:

Use OpenCV for consistent video feed quality, applying contour and convex hull detection for hand recognition.

7. Monitoring and Logging:

Monitor real-time performance metrics such as frame rate and gesture accuracy, log events for analysis.

8. User Data:

Encrypt and securely manage user data, ensuring compliance with privacy regulations and no storage of identifiable images.

9. User Input:

Optimize input processing for minimal latency, validate inputs to prevent unintended actions.

10. Component Integration:

Establish clear communication protocols between components, conduct integration testing to ensure seamless operation.