

Guided Chinese Chess

Charlie Ai, Claire Lee, Yoyo Zhong

[overview \(claire\)](#)

[use case + design requirements \(claire\)](#)

[solution approach \(claire\)](#)

[complete solution \(yoyo\)](#)

[testing risks and mitigation \(yoyo\)](#)

[test strategies: LED \(claire\)](#)

[test strategies: CV \(yoyo\)](#)

[test strategies: LLM \(charlie\)](#)

[trade offs \(charlie\)](#)

[demo video \(charlie\)](#)

[gantt chart \(claire\)](#)



Guided Chinese Chess

A guided learning experience for Chinese Chess

- Lights up legal move suggestions and opponent's moves
- LLM to ask for strategies
- Computer vision that captures the board state



Use Case & Design Requirement

Use Case Requirement	Design requirement	Justification
Board state detection feels instantaneous	CV board detection latency $\leq 500\text{ms}$; piece recognition accuracy $\geq 95\%$	Users perceive delays $> 1\text{s}$ as lag; Cannot guarantee 100% accuracy with CV based board state detection
Move validation feels instant	Require a stable board state for $\geq 200\text{ms}$ Fixed overhead camera	Below human reaction time ($\sim 200\text{ms}$); appears instantaneous
Legal moves are clearly visible on board	90 individually addressable SK6812 LEDs; distinct colors	Physical board feedback minimizes screen reliance; color coding is intuitive
AI suggestions are always legal	Engine bounded LLM prompting; LLM restricted to explain engine evaluations and candidate moves only	Illegal suggestions destroy user trust; validator guarantees correctness

Use Case & Design Requirement

Use Case Requirement	Design requirement	Justification
System warns before blunders	Blunder detection triggers LED warning $\leq 500\text{ms}$ of piece lift	Proactive warning to encourage take back and help beginner players avoid easy to spot blunders
Guidance adapts to skill level	Three modes: Full Guidance (auto LEDs + explanations), Assistive (on-request), Challenge (minimal)	Supports learning progression; prevents information overload for advancing players
AI explanations are helpful and timely	LLM explanation latency $\leq 2.5\text{s}$; fine-tuned Qwen2-7B with RAG for Xiangqi knowledge	Acceptable wait for quality content; domain-specific training prevents hallucination
Players practice tactics	Puzzle mode activates when tactical pattern detected	Active recall more effective than passive hints

Solutions Approach

Components:

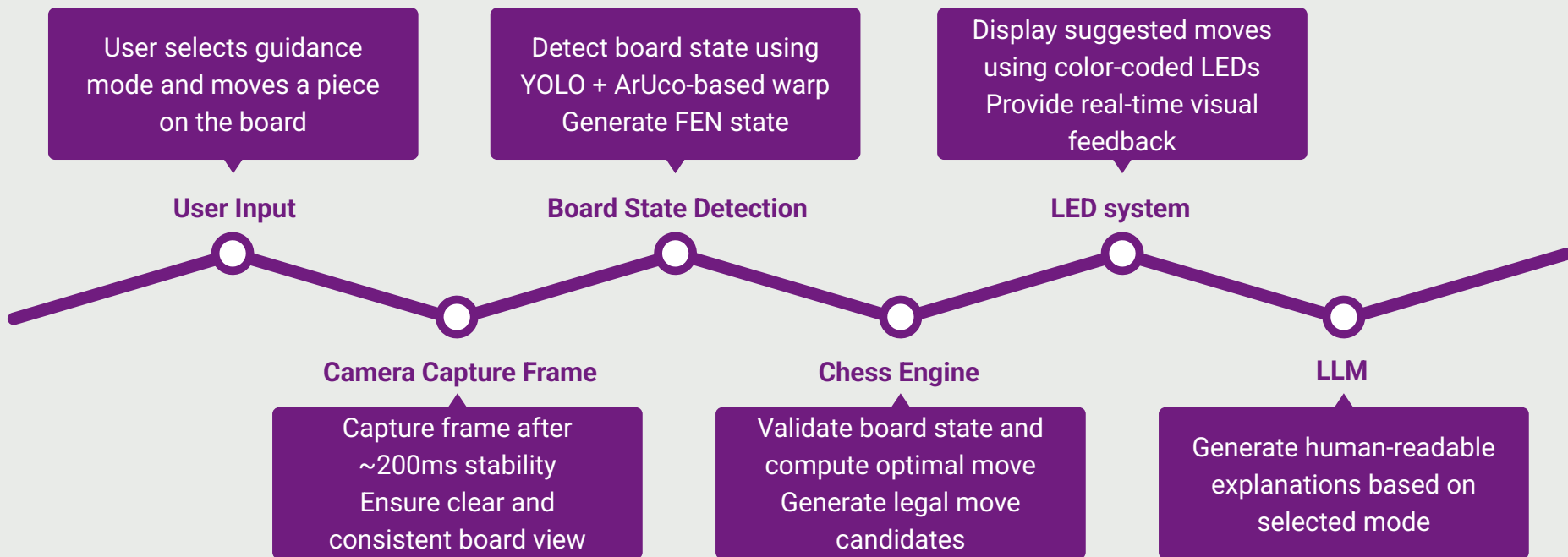
- Addressable LED strips: display move suggestions
- Camera: detect board state
- Custom Board: show grid lines and lights

Additional Considerations:

- make the board 3D to encapsulate all electrical components
- Inexpensive materials
- Robust wiring of LEDs

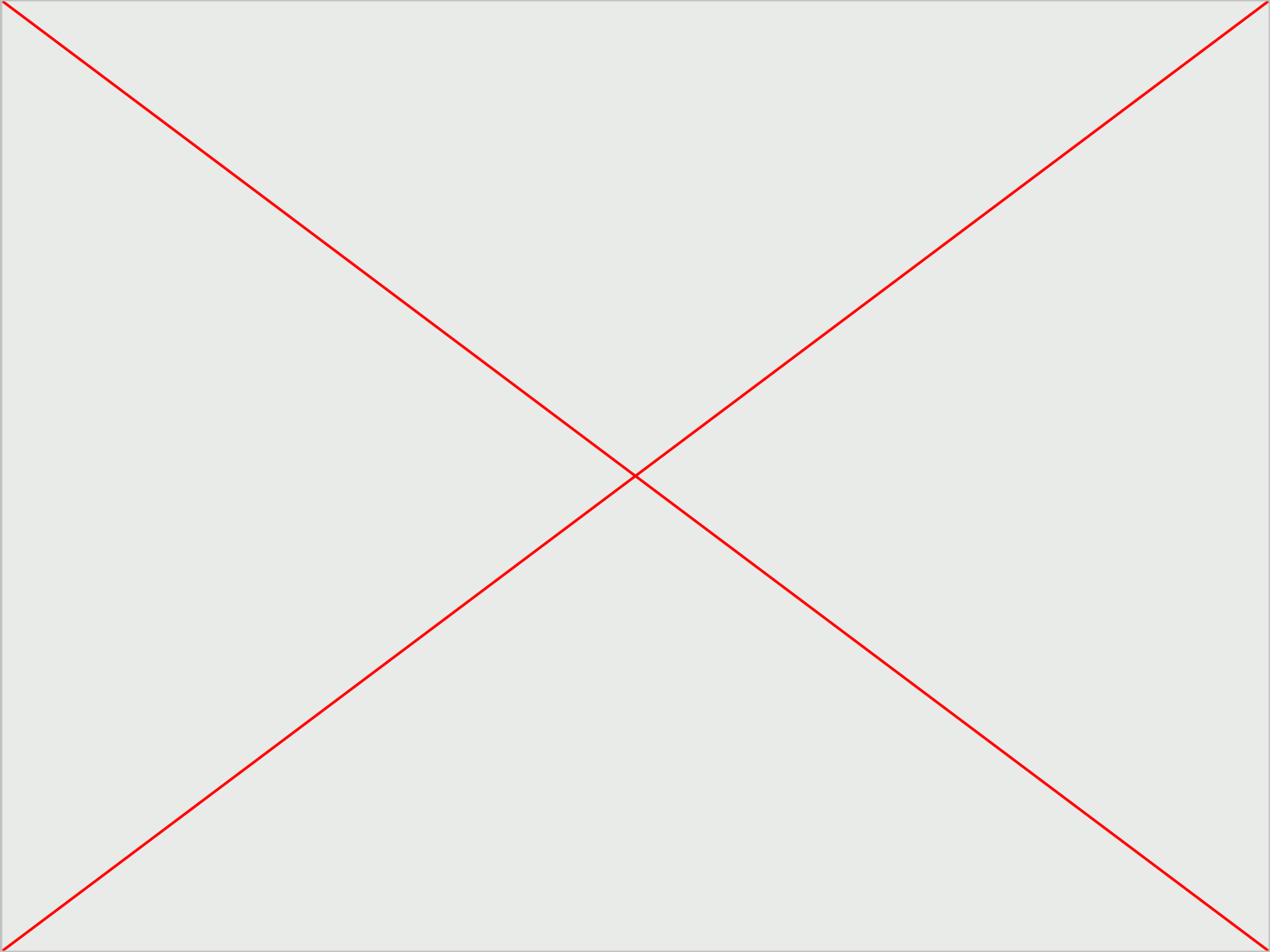


Complete Solution



Risks and mitigation

- **Lighting Variation**
 - Changes in lighting may affect detection accuracy
Mitigation: fixed camera setup and controlled indoor lighting
 - **Voice Command Noise**
 - Background noise may affect speech recognition
Mitigation: use predefined commands and noise filtering
 - **Invalid Board State**
 - Incorrect FEN may break downstream modules
Mitigation: sanity checks and fallback to last valid state
 - **Latency / System Delay**
 - Slow response affects user experience
Mitigation: event-based triggering and lightweight models
 -
-



TESTING: LED System

LED Strip

- Connectivity: tested every 5 LEDs on the strip while building board: can show light at all and all 4 colors RGBW

Raspberry Pi Move Suggestions:

- LATENCY: LED response <100ms after the player has selected a piece
 - timestamp of when the FEN string is outputted from the computer vision system - timestamp of when each of the LEDs were turned on.
- ACCURACY: Want > 95% accuracy. Used 20 different board states and selected 5 pieces from each board state to question their potential moves Log the accuracy of the 100 attempts.



TESTING: Computer Vision

Latency

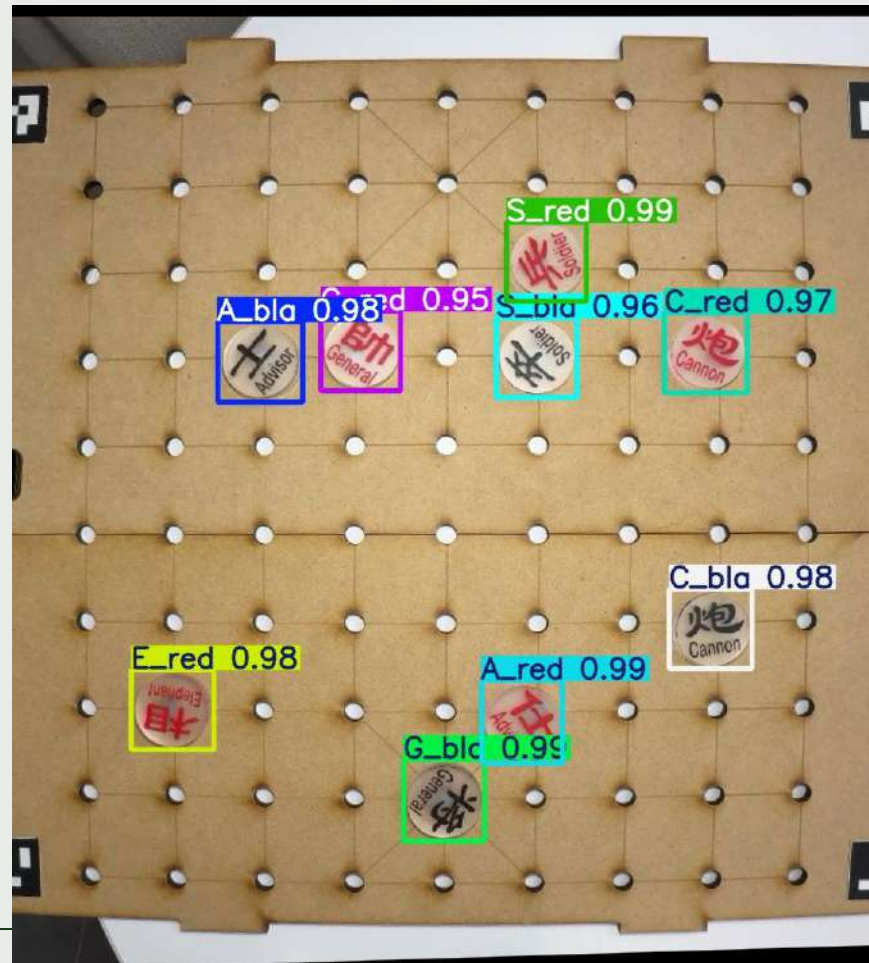
- End-to-end CV processing $\approx 80\text{--}120$ ms per frame
 - Includes capture, warp, detection, and mapping
 - Supports real-time interaction

Detection Performance

- YOLOv8
 - $\text{mAP@0.5} = 0.989$
 - $\text{mAP@0.5:0.95} = 0.961$
- YOLOX
 - $\text{mAP@0.5} = 0.871$
 - $\text{mAP@0.5:0.95} = 0.829$

System-Level Accuracy

- Dataset split: 70% train / 20% val / 10% test
- Tested across 60 board configurations
- Evaluated correctness of piece detection and grid assignment



Chess Engine, Move Generation

Rust chess engine

- optimized move generation with move ordering, LVM, quiescence
- output top candidate moves and evaluation metrics

-Explainability

to accommodate for latency and effectively manage token cost, we have added two separate paths for generating quality explanations

-Fast track: heuristic generated explanation based on current board position, capturing material gains, blunders, center control, king safety dependent on game stages displayed after every move

-Slow track: fine-tuned Qwen 2.5 7B for combining RAG explanations invoked by user questions and between multiple moves, aimed at explanation multi move strategies

Testing:

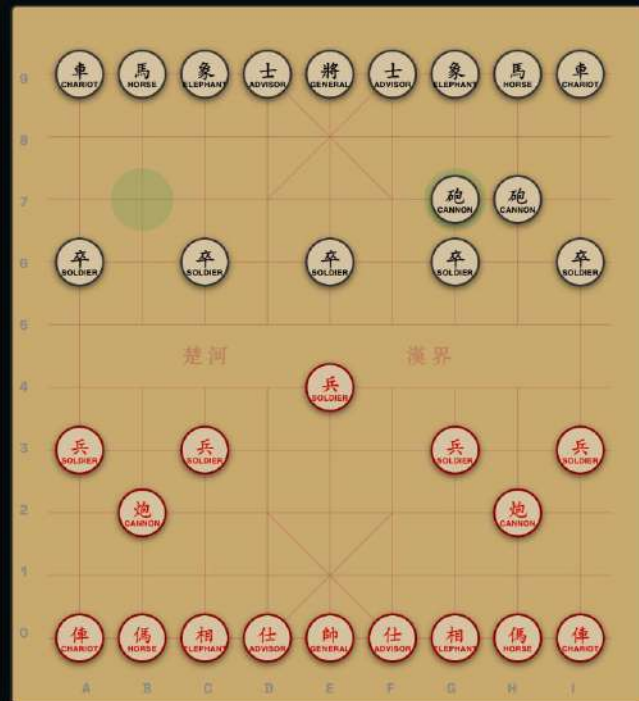
Inclusive suites of unit tests to ensure correctness

Performance benchmarked against baseline min max model displayed significant reduction in branching factor

=== Opening ===					
Depth	Time(ms)	Nodes	Nodes/sec	EBF	Score
1	0.81	90	111380	0.00	0
2	12.55	496	39514	5.51	0
3	34.15	4686	137208	9.45	0
4	176.92	17655	99789	3.77	50
5	802.72	117089	145866	6.63	50

=== Mid-game ===					
Depth	Time(ms)	Nodes	Nodes/sec	EBF	Score
1	0.21	81	385485	0.00	1850
2	2.44	510	209302	6.30	1850
3	15.21	3363	221065	6.59	1850
4	62.64	14071	224627	4.18	1850
5	250.02	90731	362897	6.45	1850

=== Opening vs Mid-game Comparison ===			
Depth	Opening NPS	Midgame NPS	Ratio
1	111380	385485	0.29
2	39514	209302	0.19
3	137208	221065	0.62

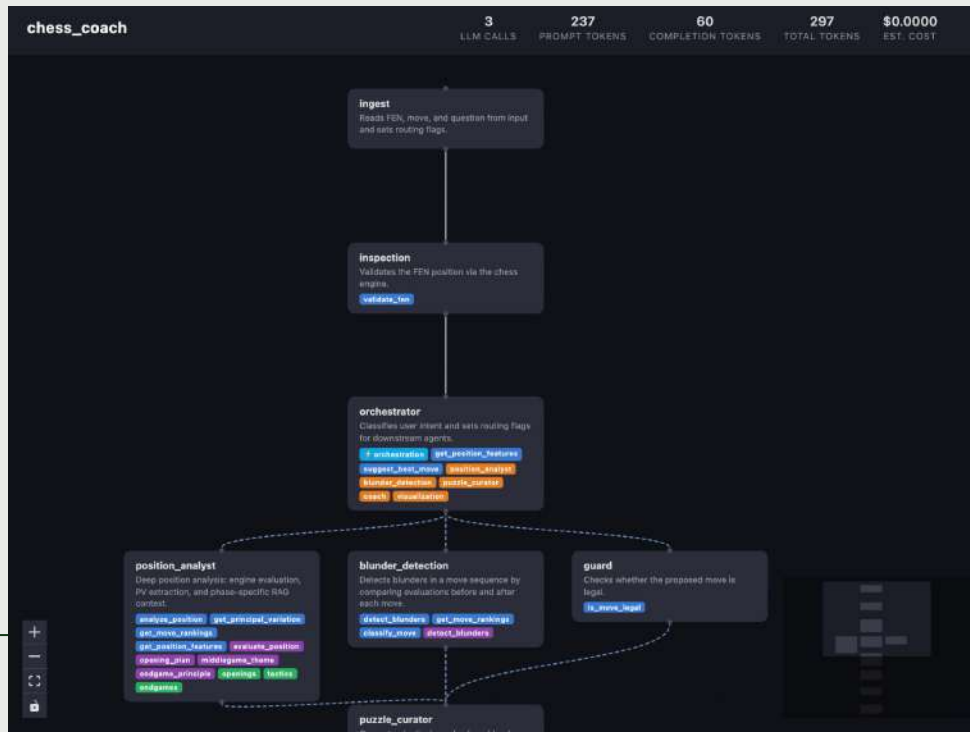


=== Feature Comparison (Depth 8, Opening) ===				
Config	Time(ms)	Nodes	Nodes/sec	Score
Baseline (no features)	9365.43	318755		34035
Move Ordering only	487.59	13498		27683
MO + TT	504.79	12414		24593
MO + Quiescence	611.18	24346		39835
All features	542.96	17655		32516

LLM Agent Orchestration - Charlie

- Built a custom LLM orchestration pipeline in Go, enabling precise control over agent execution and real-time debugging
- Integrated Grafana & Prometheus for full-stack observability metrics, latency tracking, and system health at a glance
- Developed a React dashboard surfacing agent state, skill usage patterns, and performance benchmarks to pinpoint bottlenecks

Kibo is a digital avatar that transforms chess gameplay into a personalized, interactive experience and reflects on your progress over time, surfacing insights about your growth as a player
Makes every game feel like a coaching session



Trade Offs

- YOLOv8 model instead of YOLOX
 - Higher average precision with small datasets
 - Better overall accuracy in our setting
 - Requires less manual tuning
- A "move done" button in UI rather than automatic LED switching
 - Avoids constant LED flashing
 - Improves user experience and visual comfort
 - Less automation, relies on user input to trigger capture
- 90 meaningful board positions out of 300+ LEDs:
 - pros: simpler wiring and less cost
 - cons: more indexing complexity, 60–70% LED utilization
- Engine "best" move computation being not the actual "most optimal" but based on more shallow characteristics
 - pros: reduced latency and allowing for real time feedback, providing up for intermediate play performance; Utilize move generation metrics in building detailed explanations
 - cons: reduced accuracy compared to fully CPU optimized NNUE engines
- Model Trade off:
 - Qwen 2.5 7B provides a balance of foundational knowledge of chess, being trained on chinese literature and token cost and inference latency

Testing and verification

Latency

- Time between player completes their move and opponents move must be $< 5s$
Metric: Average latency across all computer generated moves across 10 full games

- LEDs light up $< 100ms$ after player completes their move

Metric: Print the timestamp of FEN string input and output of LED display and subtract the difference

- Performance: compare move generation across depth with different optimization methods enabled,

Accuracy

- Board State Detection: $> 95\%$ correct detection rate across 100 trials, and notify user in misalignment

Metric: Set up board with 100 different states with different numbers of pieces

- Move generation: the opponent's generated moves should always be legal

Metric: Across 10 full games, log the percentage of correctly generated moves

UI

- Verify correct board rendering and validation of moves
 - Recruit 3-5 beginners with the task of finding the best move through the UI
 - keep note of any confusion of what certain information/button mean
 - if they are looking of specific information but we don't include it
 - time to successfully complete the task
-

PLAYER MOVES

red - selected piece for move selection

white - legal move

orange - capture move

green - best move

OPPONENT MOVES

blue - where the opponent moves from

purple - where the opponent moves to

BOARD ZONES

blue - black side

green - red side

cyan - river

red - palaces

Gantt Chart

