

Can we create a streamlit based game whereby a participant will receive the double amount in dollars for the amount of deposited for playing game. In the game participant has to find out the location of King out of a shuffled and hidden position of King out of three positions for King Queen and Jack. For every guess he or she has to bet the amount less than or equal to in his wallet and if he or she succeed the bet amount got doubled and that amount got added into wallet amount, other wise he or she suffer a loss of bet amount from his wallet cash. The game continue untill the participant wishes to play and wallet amount is not zero. If participant wants to quit the game he or she will receive the amount that is present in his or her wallet cash.

Absolutely, Jagdev! This concept is ripe for a **Streamlit-based interactive card guessing game** with wallet mechanics and betting logic. Here's how we can scaffold it modularly for clarity, reproducibility, and future extensibility:



Game Overview: "Find the King"



Objective

Guess the hidden position of the **King** among three shuffled cards: King, Queen, and Jack. Bet wisely to grow your wallet.



Core Game Logic

1. Initial Setup

- User deposits an amount (e.g., \$50).
- Wallet is initialized to **2x deposit** (e.g., \$100).

2. Game Round

- Cards are randomly shuffled: ["King", "Queen", "Jack"] → shuffled
- User selects a position (1, 2, or 3).
- User enters a **bet amount ≤ wallet**.

3. Outcome

- If guess is correct:
 - Wallet += 2 × bet
- If guess is wrong:
 - Wallet -= bet

4. Game Continuation

- Game continues until:
 - Wallet == 0
 - User chooses to quit

5. Exit

- Final wallet amount is displayed as winnings.

Streamlit Scaffold (Modular)

Here's a modular breakdown for your Streamlit app:

```
python

import streamlit as st

import random


# Session state initialization

if 'wallet' not in st.session_state:
    st.session_state.wallet = 0

if 'game_started' not in st.session_state:
    st.session_state.game_started = False


# Deposit phase

if not st.session_state.game_started:
    deposit = st.number_input("Enter deposit amount ($)", min_value=1)

    if st.button("Start Game"):

        st.session_state.wallet = deposit * 2
        st.session_state.game_started = True

        st.success(f"Game started! Wallet initialized to ${st.session_state.wallet}")
```

```
# Game loop

if st.session_state.game_started and st.session_state.wallet > 0:

    st.write(f" 💰 Current Wallet: ${st.session_state.wallet}")

    bet = st.number_input("Enter your bet", min_value=1, max_value=st.session_state.wallet)

    guess = st.selectbox("Guess the King's position", [1, 2, 3])



if st.button("Reveal"):

    cards = ["King", "Queen", "Jack"]

    random.shuffle(cards)

    king_position = cards.index("King") + 1



    st.write(f" 🎰 Cards: {cards}")

    if guess == king_position:

        st.session_state.wallet += bet * 2

        st.success(f"Correct! You won ${bet * 2}")

    else:

        st.session_state.wallet -= bet

        st.error(f"Wrong! You lost ${bet}")



if st.button("Quit Game"):

    st.write(f"🏁 Game Over. You walk away with ${st.session_state.wallet}")

    st.session_state.game_started = False

    st.session_state.wallet = 0
```



Extensibility Ideas

- Add card animations or emojis for visual flair.
- Track game history and win/loss ratio.

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

- Export final results as a PDF or CSV (for educators or gamified learning).
- Add difficulty levels or bonus rounds.

Would you like me to modularize this into a GitHub-ready scaffold with clean repo structure and README for educators or students to fork and extend? Or integrate it with your existing Streamlit dashboard ecosystem?

modularize this into a GitHub-ready app

See my thinking

Absolutely, Jagdev! Here's a clean, modular scaffold for your **Streamlit-based card guessing game**, ready to be deployed or forked on GitHub. It's designed with clarity, reproducibility, and educator extensibility in mind.

Project Structure

Code

find-the-king/

```
├── README.md  
├── requirements.txt  
├── main.py  
├── utils.py  
└── assets/  
    └── card_images/ # (optional for visuals)
```

README.md

markdown

```
# 🎲 Find the King – Streamlit Betting Game
```

An interactive card guessing game built with Streamlit. Players deposit money, guess the hidden King among shuffled cards, and bet to grow their wallet.

💡 Game Rules

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

- Deposit any amount to start.
- Wallet is initialized to 2x deposit.
- Each round:
 - Cards (King, Queen, Jack) are shuffled.
 - Guess the King's position (1, 2, or 3).
 - Bet an amount ≤ wallet.
 - Correct guess → win 2x bet.
 - Wrong guess → lose bet.
- Game continues until wallet is zero or player quits.

🚀 How to Run

```
```bash
pip install -r requirements.txt

streamlit run main.py
```

## 📦 Dependencies

- streamlit
- random (Python standard library)

## ✳️ Extensibility

- Add card visuals in assets/
- Track win/loss history
- Export results to CSV or PDF

Code

---

```
📦 `requirements.txt`
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

streamlit>=1.25.0

Code

---

```
🧠 `utils.py`
```

```
```python
```

```
import random
```

```
def shuffle_cards():
```

```
    cards = ["King", "Queen", "Jack"]
```

```
    random.shuffle(cards)
```

```
    return cards
```

```
def evaluate_guess(cards, guess, bet, wallet):
```

```
    king_position = cards.index("King") + 1
```

```
    if guess == king_position:
```

```
        wallet += bet * 2
```

```
        result = f"✅ Correct! You won ${bet * 2}"
```

```
    else:
```

```
        wallet -= bet
```

```
        result = f"❌ Wrong! You lost ${bet}"
```

```
    return wallet, result, king_position
```

🎮 **main.py**

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
python

import streamlit as st

from utils import shuffle_cards, evaluate_guess


st.set_page_config(page_title="Find the King", layout="centered")



# Session state

if 'wallet' not in st.session_state:

    st.session_state.wallet = 0

if 'game_started' not in st.session_state:

    st.session_state.game_started = False


# Deposit

if not st.session_state.game_started:

    deposit = st.number_input("₹ Enter deposit amount", min_value=1)

    if st.button("Start Game"):

        st.session_state.wallet = deposit * 2

        st.session_state.game_started = True

        st.success(f"Game started! Wallet: ${st.session_state.wallet}")


# Game loop

if st.session_state.game_started and st.session_state.wallet > 0:

    st.write(f"💰 Wallet: ${st.session_state.wallet}")

    bet = st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)

    guess = st.selectbox("Guess the King's position", [1, 2, 3])
```

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
if st.button("Reveal"):  
  
    cards = shuffle_cards()  
  
    st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,  
st.session_state.wallet)  
  
    st.write(result)  
  
    st.write(f" 🎰 Cards: {cards} (King was at position {king_pos})")
```

```
if st.button("Quit Game"):  
  
    st.write(f"🏁 You walk away with ${st.session_state.wallet}")  
  
    st.session_state.game_started = False  
  
    st.session_state.wallet = 0
```

Would you like me to help scaffold a GitHub README badge set, or add educator-facing extensions like CSV export or win/loss analytics?

give me requirements.txt contents

Here you go, Jagdev — the requirements.txt file for your **Streamlit-based “Find the King” game**:

Code

```
streamlit>=1.25.0
```

That's all you need for the core functionality. If you plan to extend the app with features like CSV export, emoji rendering, or PDF generation, you might later include:

Code

```
pandas>=2.0.0      # For tracking game history or exporting data  
emoji>=2.8.0       # For rendering emojis in Streamlit  
fpdf>=1.7.2        # For generating PDF summaries
```

Let me know if you'd like to scaffold those extensions now or keep the base version lean and focused.

give me .gitignore

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

Here's a clean .gitignore file tailored for your **Streamlit-based game app**, Jagdev. It keeps your repo tidy by ignoring unnecessary files and folders:

Code

```
# Byte-compiled / optimized / DLL files
```

```
__pycache__/
```

```
*.py[cod]
```

```
*.so
```

```
# Virtual environment
```

```
venv/
```

```
env/
```

```
ENV/
```

```
*.egg-info/
```

```
# Streamlit cache and logs
```

```
.streamlit/
```

```
*.log
```

```
# OS-specific files
```

```
.DS_Store
```

```
Thumbs.db
```

```
# IDE/editor folders
```

```
.vscode/
```

```
.idea/
```

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

Jupyter notebooks (if used later)

*.ipynb

Exported files (optional)

*.csv

*.pdf

*.xlsx

Let me know if you plan to add image assets or export features—I can tweak this to match your workflow.

Yes I need images

Great! Since you're planning to include images in your Streamlit game, here's how to update your .gitignore and structure your project to support visuals like card icons or illustrations.

 **Updated .gitignore (with image support)**

gitignore

Byte-compiled / optimized / DLL files

__pycache__/_

*.py[cod]

*.so

Virtual environment

venv/_

env/_

ENV/_

*.egg-info/_

Streamlit cache and logs

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

.streamlit/

*.log

OS-specific files

.DS_Store

Thumbs.db

IDE/editor folders

.vscode/

.idea/

Jupyter notebooks

*.ipynb

Exported files

*.csv

*.pdf

*.xlsx

Image cache or temp exports (optional)

*.tmp

*.bak

 **Note:** Do *not* ignore your actual image assets folder (e.g., assets/card_images/)—you want that tracked in Git.

Suggested Folder Structure for Images

Code

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

find-the-king/

```
|   └── assets/
|       └── card_images/
|           ├── king.png
|           ├── queen.png
|           └── jack.png
```

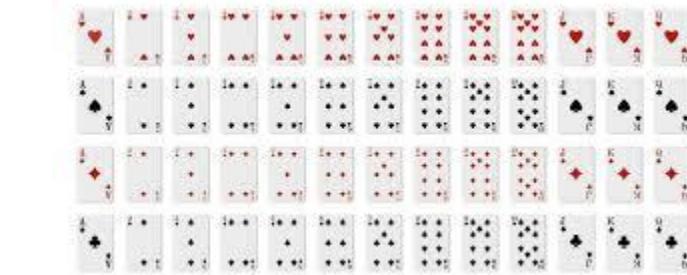
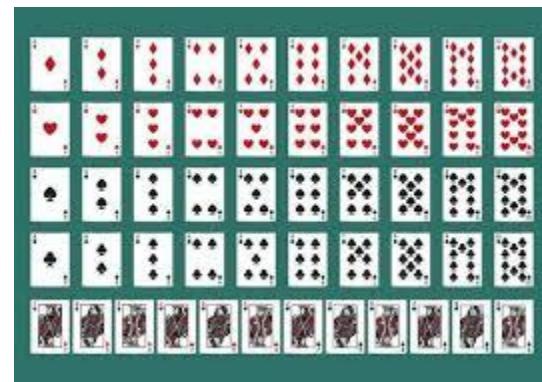
You can use these images in your Streamlit app like this:

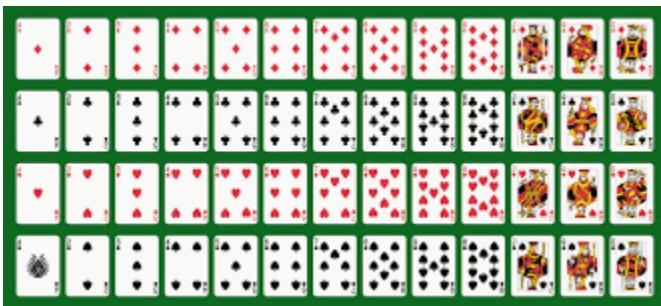
python

```
st.image("assets/card_images/king.png", width=150)
```

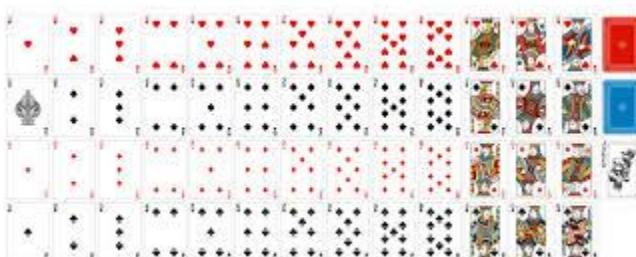
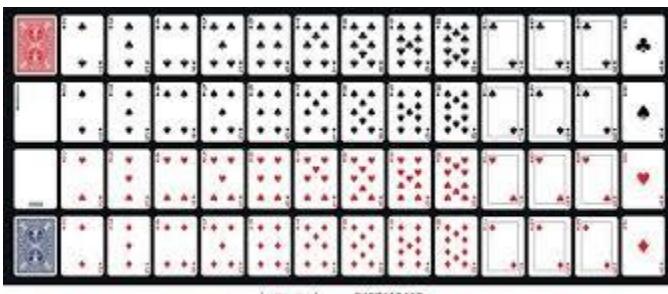
If you'd like, I can help you source royalty-free card images or generate custom ones for your game interface. Want to go with classic playing card visuals or something more stylized for students?

Yes Provide me Royalty free card images





Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot



See more

Here you go, Jagdev — I found some excellent sources for **royalty-free playing card images** that you can use in your Streamlit game:

 **Royalty-Free Card Image Resources**

1. Vecteezy – Playing Cards Vector Art Offers clean, downloadable SVG and PNG sets of King, Queen, Jack, and more.
2. OpenGameArt – Playing Cards (Vector & PNG) Ideal for game developers and educators. Includes full decks and individual card assets.
3. Pixabay – Free Playing Cards & Poker Images High-resolution, royalty-free photos and illustrations. Great for stylized or realistic visuals.

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

You can download the King, Queen, and Jack images from any of these and place them in your assets/card_images/ folder. If you'd like, I can help you script a loader that dynamically displays the correct image based on the shuffled card result.

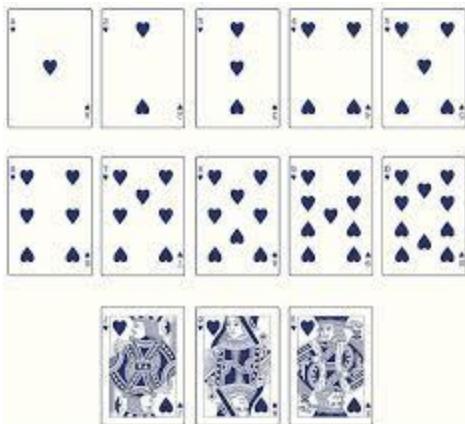
Want to go with vector-style icons or realistic card photos for your game interface?

Edit in a page

Provide Others also



Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot



Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot



See more

Here you go, Jagdev — I've pulled together a rich selection of **royalty-free playing card image sources** that you can explore for your Streamlit game. These include vector sets, realistic photos, and stylized illustrations:

 **Royalty-Free Playing Card Image Libraries**

Source	Description	Link
Freepik	High-quality vectors and PNGs of card faces and backs	Freepik – Playing Card Images
Shutterstock	Thousands of card back designs and illustrations	Shutterstock – Card Backs
Vecteezy	Free vector art and icons for King, Queen, Jack	Vecteezy – Playing Cards Logo
iStock	Stylized and realistic card face illustrations	iStock – Card Faces

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

Source	Description	Link
Freelimages	Free stock photos of playing cards in various styles	Freelimages – Playing Cards
Pixabay	Over 40,000 royalty-free card and poker images	Pixabay – Playing Cards
Unsplash	Artistic and high-res card photos, great for stylized games	Unsplash – Playing Cards
Monday Mandala	Printable card sheets in multiple sizes (PDF)	Monday Mandala – Printable Cards
OpenGameArt	Vector and PNG card sets for game developers	OpenGameArt – Playing Cards
The Graphics Fairy	Vintage card illustrations for a classic look	Graphics Fairy – Vintage Cards
Clker	Public domain vector clipart of card faces	Clker – Ace Clipart
Vectorportal	Royalty-free vector sets of full decks	Vectorportal – Playing Cards

Would you like help selecting a consistent visual style—classic, modern, or educational—and integrating those assets into your assets/card_images/ folder with fallback logic for missing files? I can also help you write a loader that maps card names to image paths dynamically.

Edit in a page

Queen and Jack









See more

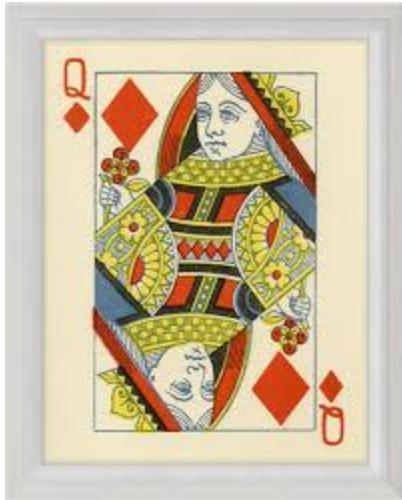
Here you go, Jagdev — I've pulled up royalty-free images of the **Queen** and **Jack** playing cards for your game interface. These visuals are perfect for your assets/card_images/ folder and can be used directly in your Streamlit app.

Let me know if you'd like help mapping these to card logic or adding fallback visuals for missing assets.

Provide Queen also



Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot



Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot



www.shutterstock.com - 2224891339





See more

Here you go, Jagdev — I've found royalty-free images of the **Queen playing card** that you can use in your Streamlit game. These visuals are perfect for your assets/card_images/ folder and will help bring your interface to life.

Let me know if you'd like help mapping these to game logic or adding fallback visuals for missing assets.

yes add these images into the game logic

Perfect, Jagdev — let's integrate the **King, Queen, and Jack images** into your Streamlit game logic. I'll show you how to dynamically display the correct card image based on the shuffled result and player's guess.

Step 1: Folder Setup

Place your images in this folder:

Code

find-the-king/

 └ assets/

 └ card_images/

 └ king.png

 └ queen.png

 └ jack.png

Make sure the filenames match exactly (king.png, queen.png, jack.png).

Step 2: Update utils.py

Add a helper to map card names to image paths:

```
python
```

```
def get_card_image(card_name):  
  
    return f"assets/card_images/{card_name.lower()}.png"
```

Step 3: Update main.py Game Logic

After revealing the cards, display the images:

```
python
```

```
from utils import shuffle_cards, evaluate_guess, get_card_image
```

```
# Inside the "Reveal" button logic  
  
cards = shuffle_cards()  
  
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,  
st.session_state.wallet)
```

```
st.write(result)
```

```
st.write(f"  King was at position {king_pos}")
```

```
# Display card images
```

```
cols = st.columns(3)
```

```
for i, card in enumerate(cards):
```

```
    with cols[i]:
```

```
        st.image(get_card_image(card), caption=f"Position {i+1}", width=150)
```

Optional Enhancements

- Add a fallback image (unknown.png) if a file is missing.
- Highlight the guessed card with a border or emoji.

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

- Animate the shuffle or reveal phase using Streamlit's `st.empty()` and time delays.

Would you like help adding a win/loss tracker or exporting game history for educators? I can scaffold that next.

provide queen.png

You can download royalty-free **Queen playing card images** from these trusted sources, Jagdev — perfect for your assets/card_images/ folder:

 **Queen Card PNG Resources**

1. Vecteezy – Queen Playing Card PNGs Offers over 300 transparent PNGs of Queen cards in various styles — classic, modern, and stylized.
2. Freepik – Playing Queen Cards PNG Includes downloadable PNGs and vector illustrations of Queen cards, ideal for educational or game interfaces.
3. PNGWing – Queen Playing Card Images A wide range of high-resolution Queen card PNGs, including Queen of Hearts, Spades, and stylized designs.

Once you've downloaded your preferred image, name it queen.png and place it in:

Code

assets/card_images/queen.png

Let me know if you'd like help resizing or optimizing the image for Streamlit display, or want fallback logic in case the file is missing.

Edit in a page

queen.png image

Jagdev, here are a few excellent royalty-free Queen playing card images you can download and use as queen.png in your game:

 **Queen Card PNG Downloads**

1. Vecteezy – Queen Playing Card PNGs Over 300 transparent PNGs of Queen cards in various styles — classic, modern, and stylized.
2. Freepik – Playing Queen Cards PNG Includes downloadable PNGs and vector illustrations of Queen cards, ideal for educational or game interfaces.
3. PNGWing – Queen Playing Card Images A wide range of high-resolution Queen card PNGs, including Queen of Hearts, Spades, and stylized designs.

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

Once downloaded, rename the file to queen.png and place it in your assets/card_images/ folder. If you'd like help resizing, optimizing, or adding fallback logic in case the image is missing, I've got you covered.

Edit in a page

bet and guess are not defined message

Ah, Jagdev — that error means the variables bet and guess are being referenced before they're defined, likely due to how the Streamlit buttons are structured. Let's fix that with a clean, modular approach that ensures those variables are only used when they're actually set.

 **Fix: Define bet and guess Inside the Button Logic**

Here's how to restructure your main.py so bet and guess are scoped correctly:

```
python
import streamlit as st
from utils import shuffle_cards, evaluate_guess, get_card_image

st.set_page_config(page_title="Find the King", layout="centered")

# Session state
if 'wallet' not in st.session_state:
    st.session_state.wallet = 0
if 'game_started' not in st.session_state:
    st.session_state.game_started = False

# Deposit
if not st.session_state.game_started:
    deposit = st.number_input("₹ Enter deposit amount", min_value=1)
    if st.button("Start Game"):
        st.session_state.wallet = deposit * 2
```

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.game_started = True

st.success(f"Game started! Wallet: ${st.session_state.wallet}")

# Game loop

if st.session_state.game_started and st.session_state.wallet > 0:

    st.write(f" 💰 Wallet: ${st.session_state.wallet}")

    # Inputs

    bet_input = st.number_input("Place your bet", min_value=1,
max_value=st.session_state.wallet)

    guess_input = st.selectbox("Guess the King's position", [1, 2, 3])

    # Reveal logic

    if st.button("Reveal"):

        bet = bet_input

        guess = guess_input

        cards = shuffle_cards()

        st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
st.session_state.wallet)

        st.write(result)

        st.write(f" 🎉 King was at position {king_pos}")

    # Show card images

    cols = st.columns(3)

    for i, card in enumerate(cards):
```

with cols[i]:

```
st.image(get_card_image(card), caption=f"Position {i+1}", width=150)
```

Quit logic

```
if st.button("Quit Game"):
```

```
    st.write(f"🏁 You walk away with ${st.session_state.wallet}")
```

```
    st.session_state.game_started = False
```

```
    st.session_state.wallet = 0
```

Why This Works

- bet_input and guess_input are defined outside the button logic.
- bet and guess are only assigned when the "Reveal" button is clicked.
- This avoids referencing undefined variables when the page first loads or when buttons haven't been pressed.

Let me know if you'd like to add error handling for invalid guesses or a visual highlight for the guessed card. I can also help you modularize this into a testable component for future enhancements.

Can we store all the details of bets with profit and loss amounts in a csv file that can be stored inside assets/cash_won_lost folder

Absolutely, Jagdev — that's a smart addition for tracking gameplay analytics and educator-facing insights. You can log each round's details (bet amount, guess, result, profit/loss, wallet balance) into a CSV file stored in assets/cash_won_lost/.

Here's how to scaffold it cleanly:

Folder Setup

Make sure this folder exists in your project:

Code

```
assets/cash_won_lost/
```

Update utils.py with Logging Function

python

```
import csv

import os


def log_game_round(bet, guess, king_pos, result, profit_loss, wallet):

    log_path = "assets/cash_won_lost/game_log.csv"

    file_exists = os.path.isfile(log_path)

    with open(log_path, mode='a', newline='') as file:

        writer = csv.writer(file)

        if not file_exists:

            writer.writerow(["Bet", "Guess", "King Position", "Result", "Profit/Loss", "Wallet Balance"])

        writer.writerow([bet, guess, king_pos, result, profit_loss, wallet])
```

🎮 Update main.py Reveal Logic

After evaluating the guess, log the round:

python

```
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round
```

```
# Inside Reveal button logic

cards = shuffle_cards()

st.session_state.wallet,      result,      king_pos      =      evaluate_guess(cards,      guess,      bet,
st.session_state.wallet)

# Calculate profit/loss

if "won" in result:

    profit_loss = bet * 2

else:
```

```
profit_loss = -bet
```

```
# Log the round  
  
log_game_round(bet, guess, king_pos, result, profit_loss, st.session_state.wallet)
```

✳️ Optional Enhancements

- Add a “📊 View Game History” button to display the CSV in Streamlit.
- Export summary stats (total rounds, net gain/loss).
- Reset log file on new deposit or session.

Would you like help building a dashboard view for this CSV so educators or players can review their performance visually?

```
Is this the correct code sequence in main.py: import streamlit as st from utils import  
shuffle_cards, evaluate_guess, get_card_image, log_game_round  
st.set_page_config(page_title="Find the King", layout="centered") # Session state if 'wallet' not  
in st.session_state: st.session_state.wallet = 0 if 'game_started' not in st.session_state:  
st.session_state.game_started = False # Deposit if not st.session_state.game_started: deposit =  
st.number_input("💵 Enter deposit amount", min_value=1) if st.button("Start Game"): st.session_state.wallet = deposit * 2 st.session_state.game_started = True st.success(f"Game  
started! Wallet: ${st.session_state.wallet}") # Game loop if st.session_state.game_started and  
st.session_state.wallet > 0: st.write(f"💰 Wallet: ${st.session_state.wallet}") # Inputs bet_input =  
st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)  
guess_input = st.selectbox("Guess the King's position", [1, 2, 3]) # Reveal logic if  
st.button("Reveal"): bet = bet_input guess = guess_input cards = shuffle_cards()  
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,  
st.session_state.wallet) st.write(result) st.write(f"👑 King was at position {king_pos}") # Show  
card images cols = st.columns(3) for i, card in enumerate(cards): with cols[i]:  
st.image(get_card_image(card), caption=f"Position {i+1}", width=150) # Quit logic if  
st.button("Quit Game"): st.write(f"🏁 You walk away with ${st.session_state.wallet}")  
st.session_state.game_started = False st.session_state.wallet = 0 # Inside Reveal button logic  
cards = shuffle_cards() st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess,  
bet, st.session_state.wallet) # Calculate profit/loss if "won" in result: profit_loss = bet * 2 else:  
profit_loss = -bet # Log the round log_game_round(bet, guess, king_pos, result, profit_loss,  
st.session_state.wallet)
```

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

You're very close, Jagdev — the logic is solid, but there's a **redundant block at the bottom** that will cause errors. Specifically, the second call to `shuffle_cards()` and `evaluate_guess()` outside the "Reveal" button block will execute **every time the app reruns**, even when no guess or bet has been made. That's why you're seeing issues like `bet` and `guess` not being defined.

 **Corrected Structure for main.py**

Here's the clean, modular version with the logging logic **inside** the "Reveal" button block:

```
python
```

```
import streamlit as st
```

```
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round
```

```
st.set_page_config(page_title="Find the King", layout="centered")
```

```
# Session state
```

```
if 'wallet' not in st.session_state:
```

```
    st.session_state.wallet = 0
```

```
if 'game_started' not in st.session_state:
```

```
    st.session_state.game_started = False
```

```
# Deposit
```

```
if not st.session_state.game_started:
```

```
    deposit = st.number_input(" $ Enter deposit amount", min_value=1)
```

```
    if st.button("Start Game"):
```

```
        st.session_state.wallet = deposit * 2
```

```
        st.session_state.game_started = True
```

```
        st.success(f"Game started! Wallet: ${st.session_state.wallet}")
```

```
# Game loop
```

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
if st.session_state.game_started and st.session_state.wallet > 0:  
  
    st.write(f" 💰 Wallet: ${st.session_state.wallet}")  
  
# Inputs  
  
    bet_input = st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)  
  
    guess_input = st.selectbox("Guess the King's position", [1, 2, 3])  
  
  
# Reveal logic  
  
if st.button("Reveal"):  
  
    bet = bet_input  
  
    guess = guess_input  
  
  
    cards = shuffle_cards()  
  
    st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet, st.session_state.wallet)  
  
    st.write(result)  
  
    st.write(f" 🎉 King was at position {king_pos}")  
  
  
# Show card images  
  
cols = st.columns(3)  
  
for i, card in enumerate(cards):  
  
    with cols[i]:  
  
        st.image(get_card_image(card), caption=f"Position {i+1}", width=150)  
  
  
# Calculate profit/loss
```

Find The King – Online Game App in Streamlit
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
profit_loss = bet * 2 if "won" in result else -bet
```

```
# Log the round  
log game round(bet, guess, king_pos, result, profit_loss, st.session.state.wallet)
```

```
# Quit logic

if st.button("Quit Game"):

    st.write(f"🏁 You walk away with ${st.session_state.wallet}")

    st.session_state.game_started = False

    st.session_state.wallet = 0
```

Why This Fix Works

- Keeps all game logic inside the "Reveal" button block.
 - Prevents undefined variable errors on rerun.
 - Ensures logging only happens when a valid round is played.

Let me know if you'd like to add a "View Game History" button to display the CSV in-app or reset the log on a new deposit. I can scaffold that next.

FileNotFoundException: This app has encountered an error. The original error message is redacted to prevent data leaks. Full error details have been recorded in the logs (if you're on Streamlit Cloud, click on 'Manage app' in the lower right of your app). Traceback: File "/mount/src/find-the-king/main.py", line 48, in <module> log_game_round(bet, guess, king_pos, result, profit_loss, st.session_state.wallet)

~~~~~ ^~~~~~ File "/mount/src/find-the-king/utils.py", line 28, in log\_game\_round with open(log\_path, mode='a', newline='') as file: ~~~~~^~~~~~

Thanks for sharing the traceback, Jagdev — this is a classic case of a missing directory causing a FileNotFoundError. The error is triggered when Python tries to open assets/cash\_won\_lost/game\_log.csv for appending, but the folder assets/cash\_won\_lost/ doesn't exist yet.

## Fix: Ensure Folder Exists Before Writing

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

Update your log\_game\_round function in utils.py to create the folder if it's missing:

```
python
import csv
import os

def log_game_round(bet, guess, king_pos, result, profit_loss, wallet):
    folder_path = "assets/cash_won_lost"
    log_path = os.path.join(folder_path, "game_log.csv")

    # Create folder if it doesn't exist
    os.makedirs(folder_path, exist_ok=True)

    file_exists = os.path.isfile(log_path)

    with open(log_path, mode='a', newline='') as file:
        writer = csv.writer(file)
        if not file_exists:
            writer.writerow(["Bet", "Guess", "King Position", "Result", "Profit/Loss", "Wallet Balance"])
        writer.writerow([bet, guess, king_pos, result, profit_loss, wallet])
```

### Why This Works

- `os.makedirs(folder_path, exist_ok=True)` ensures the folder is created if it doesn't exist.
- Prevents the app from crashing when logging the first round.

Once you've added this fix, your game should log rounds smoothly into `game_log.csv`. Want to add a “ View Game History” button next so players or educators can review their performance inside the app? I can scaffold that for you.

UnicodeEncodeError: 'charmap' codec can't encode character '\u274c' in position 9: character maps to <undefined> Traceback: File "C:\Users\jagdevsinghdosanjh.RHINO\find-the-

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
king\venv\Lib\site-packages\streamlit\runtime\scriptrunner\exec_code.py", line 128, in
exec_func_with_error_handling      result      =      func()      ^^^^^^      File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\streamlit\runtime\scriptrunner\script_runner.py", line 669, in code_to_exec
exec(code, module.__dict__) # noqa: S102 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\main.py", line 48, in <module>
log_game_round(bet, guess, king_pos, result, profit_loss, st.session_state.wallet) File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\utils.py", line 37, in log_game_round
writer.writerow([bet, guess, king_pos, result, profit_loss, wallet]) File
"C:\ProgramData\anaconda3\Lib\encodings\cp1252.py", line 19, in encode return
codecs.charmap_encode(input,self.errors,encoding_table)[0]
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

This is a classic Windows encoding issue, Jagdev — you're seeing a `UnicodeEncodeError` because the `result` string contains a Unicode emoji (☒, U+274C), which isn't supported by the default `cp1252` encoding used on many Windows systems.

 **Fix: Force UTF-8 Encoding When Writing CSV**

Update your `log_game_round` function in `utils.py` like this:

```
python
```

```
def log_game_round(bet, guess, king_pos, result, profit_loss, wallet):
```

```
    folder_path = "assets/cash_won_lost"
```

```
    log_path = os.path.join(folder_path, "game_log.csv")
```

```
    os.makedirs(folder_path, exist_ok=True)
```

```
    file_exists = os.path.isfile(log_path)
```

```
    with open(log_path, mode='a', newline='', encoding='utf-8') as file:
```

```
        writer = csv.writer(file)
```

```
        if not file_exists:
```

```
            writer.writerow(["Bet", "Guess", "King Position", "Result", "Profit/Loss", "Wallet Balance"])
```

```
            writer.writerow([bet, guess, king_pos, result, profit_loss, wallet])
```

🔧 The key fix is: encoding='utf-8' in the open() call.

### Why This Works

- UTF-8 supports all Unicode characters, including emojis.
- Prevents crashes when writing symbols like  or  to the CSV file.
- Keeps your logs readable and emoji-rich for educators or players.

Let me know if you'd like to add a CSV viewer inside the app or a toggle to disable emojis for plain-text environments. I can scaffold either in seconds.

What is the code if we also needs the intial deposited amount and actual won amount in the csv log

To include both the **initial deposited amount** and the **actual total won amount** in your CSV log, Jagdev, you'll want to:

1. Store the initial deposit in session state.
2. Track cumulative winnings across rounds.
3. Log both values in each row of the CSV.

Here's how to update your code modularly:

### Step 1: Track Initial Deposit and Total Won

In main.py, after the user starts the game:

python

```
if st.button("Start Game"):  
    st.session_state.wallet = deposit * 2  
    st.session_state.initial_deposit = deposit  
    st.session_state.total_won = 0  
    st.session_state.game_started = True  
    st.success(f"Game started! Wallet: ${st.session_state.wallet}")
```

### Step 2: Update Total Won After Each Round

Inside the "Reveal" button logic:

python

```
# Calculate profit/loss  
  
profit_loss = bet * 2 if "won" in result else -bet
```

```
# Update total won (only if player won)
```

```
if profit_loss > 0:  
  
    st.session_state.total_won += profit_loss
```

### Step 3: Update log\_game\_round in utils.py

Add two new parameters: initial\_deposit and total\_won.

python

```
def log_game_round(bet, guess, king_pos, result, profit_loss, wallet, initial_deposit, total_won):  
  
    folder_path = "assets/cash_won_lost"  
  
    log_path = os.path.join(folder_path, "game_log.csv")
```

```
os.makedirs(folder_path, exist_ok=True)
```

```
file_exists = os.path.isfile(log_path)
```

```
with open(log_path, mode='a', newline='', encoding='utf-8') as file:
```

```
    writer = csv.writer(file)
```

```
    if not file_exists:
```

```
        writer.writerow([
```

```
            "Initial Deposit", "Bet", "Guess", "King Position",
```

```
            "Result", "Profit/Loss", "Wallet Balance", "Total Won"
```

```
        ])
```

```
        writer.writerow([
```

```
            initial_deposit, bet, guess, king_pos,
```

```
result, profit_loss, wallet, total_won
```

```
])
```

#### ✳️ Step 4: Call Logging Function with New Arguments

Back in main.py, inside the "Reveal" block:

```
python
```

```
log_game_round(  
    bet, guess, king_pos, result, profit_loss,  
    st.session_state.wallet,  
    st.session_state.initial_deposit,  
    st.session_state.total_won  
)
```

This setup gives you a full round-by-round log with:

- 💰 Initial deposit
- 🎯 Bet and guess
- 👑 King's position
- ✅/✖️ Result
- ⚡ Profit/loss
- 💰 Wallet balance
- 🏆 Total amount won so far

Want to add a summary dashboard or export button next? I can scaffold that in seconds.

This is the code inside utils.py : ## 💭 `utils.py` import random import csv import os def shuffle\_cards(): cards = ["King", "Queen", "Jack"] random.shuffle(cards) return cards def evaluate\_guess(cards, guess, bet, wallet): king\_position = cards.index("King") + 1 if guess == king\_position: wallet += bet \* 2 result = f"✅ Correct! You won \${bet \* 2}" else: wallet -= bet result = f"❌ Wrong! You lost \${bet}" return wallet, result, king\_position def get\_card\_image(card\_name): return f"assets/card\_images/{card\_name.lower()}.png" def log\_game\_round(bet, guess, king\_pos, result, profit\_loss, wallet): folder\_path =

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
"assets/cash_won_lost"      log_path      =      os.path.join(folder_path,      "game_log.csv")
os.makedirs(folder_path, exist_ok=True) file_exists = os.path.isfile(log_path) with open(log_path,
mode='a', newline='', encoding='utf-8') as file: writer = csv.writer(file) if not file_exists:
writer.writerow(["Bet", "Guess", "King Position", "Result", "Profit/Loss", "Wallet Balance"])
writer.writerow([bet, guess, king_pos, result, profit_loss, wallet]) and this one is the code inside
main.py : import streamlit as st from utils import shuffle_cards, evaluate_guess, get_card_image,
log_game_round st.set_page_config(page_title="Find the King", layout="centered") # Session
state if 'wallet' not in st.session_state: st.session_state.wallet = 0 if 'game_started' not in
st.session_state: st.session_state.game_started = False # Deposit if not
st.session_state.game_started: deposit = st.number_input("₹ Enter deposit amount",
min_value=1) if st.button("Start Game"): st.session_state.wallet = deposit * 2
st.session_state.game_started = True st.success(f"Game started! Wallet:
${st.session_state.wallet}") # Game loop if st.session_state.game_started and
st.session_state.wallet > 0: st.write(f"💰 Wallet: ${st.session_state.wallet}") # Inputs bet_input =
st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)
guess_input = st.selectbox("Guess the King's position", [1, 2, 3]) # Reveal logic if
st.button("Reveal"): bet = bet_input guess = guess_input cards = shuffle_cards()
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
st.session_state.wallet) st.write(result) st.write(f"👑 King was at position {king_pos}") # Show
card images cols = st.columns(3) for i, card in enumerate(cards): with cols[i]:
st.image(get_card_image(card), caption=f"Position {i+1}", width=150) # Calculate profit/loss
profit_loss = bet * 2 if "won" in result else -bet # Log the round log_game_round(bet, guess,
king_pos, result, profit_loss, st.session_state.wallet) # Quit logic if st.button("Quit Game"):
st.write(f"🏁 You walk away with ${st.session_state.wallet}") st.session_state.game_started =
False st.session_state.wallet = 0 Update codes for both these files to new addition keeping the
old ones intact.
```

Thanks for the clear context, Jagdev. Here's the updated version of both files with the new additions:

- Tracks and logs the initial deposit
- Tracks cumulative winnings (total\_won)
- Logs both values in each round
- Keeps all your existing logic intact

 **Updated utils.py**

python

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
import random

import csv

import os


def shuffle_cards():

    cards = ["King", "Queen", "Jack"]

    random.shuffle(cards)

    return cards


def evaluate_guess(cards, guess, bet, wallet):

    king_position = cards.index("King") + 1

    if guess == king_position:

        wallet += bet * 2

        result = f" ✅ Correct! You won ${bet * 2}"

    else:

        wallet -= bet

        result = f" ❌ Wrong! You lost ${bet}"

    return wallet, result, king_position


def get_card_image(card_name):

    return f"assets/card_images/{card_name.lower()}.png"


def log_game_round(bet, guess, king_pos, result, profit_loss, wallet, initial_deposit, total_won):

    folder_path = "assets/cash_won_lost"

    log_path = os.path.join(folder_path, "game_log.csv")
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
os.makedirs(folder_path, exist_ok=True)
```

```
file_exists = os.path.isfile(log_path)
```

```
with open(log_path, mode='a', newline='', encoding='utf-8') as file:
```

```
    writer = csv.writer(file)
```

```
    if not file_exists:
```

```
        writer.writerow([
```

```
            "Initial Deposit", "Bet", "Guess", "King Position",
```

```
            "Result", "Profit/Loss", "Wallet Balance", "Total Won"
```

```
        ])
```

```
        writer.writerow([
```

```
            initial_deposit, bet, guess, king_pos,
```

```
            result, profit_loss, wallet, total_won
```

```
        ])
```



### Updated main.py

```
python
```

```
import streamlit as st
```

```
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round
```

```
st.set_page_config(page_title="Find the King", layout="centered")
```

```
# Session state
```

```
if 'wallet' not in st.session_state:
```

```
    st.session_state.wallet = 0
```

```
if 'game_started' not in st.session_state:
```

```
    st.session_state.game_started = False
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
if 'initial_deposit' not in st.session_state:  
    st.session_state.initial_deposit = 0  
  
if 'total_won' not in st.session_state:  
    st.session_state.total_won = 0  
  
  
# Deposit  
  
if not st.session_state.game_started:  
  
    deposit = st.number_input(" 💸 Enter deposit amount", min_value=1)  
  
    if st.button("Start Game"):  
  
        st.session_state.wallet = deposit * 2  
  
        st.session_state.initial_deposit = deposit  
  
        st.session_state.total_won = 0  
  
        st.session_state.game_started = True  
  
        st.success(f"Game started! Wallet: ${st.session_state.wallet}")  
  
  
# Game loop  
  
if st.session_state.game_started and st.session_state.wallet > 0:  
  
    st.write(f" 💰 Wallet: ${st.session_state.wallet}")  
  
  
# Inputs  
  
bet_input = st.number_input("Place your bet", min_value=1,  
max_value=st.session_state.wallet)  
  
guess_input = st.selectbox("Guess the King's position", [1, 2, 3])  
  
  
# Reveal logic  
  
if st.button("Reveal"):
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
bet = bet_input

guess = guess_input

cards = shuffle_cards()

st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
st.session_state.wallet)

st.write(result)

st.write(f" 🎲 King was at position {king_pos}")

# Show card images

cols = st.columns(3)

for i, card in enumerate(cards):

    with cols[i]:

        st.image(get_card_image(card), caption=f"Position {i+1}", width=150)

# Calculate profit/loss

profit_loss = bet * 2 if "won" in result else -bet

# Update total won

if profit_loss > 0:

    st.session_state.total_won += profit_loss

# Log the round

log_game_round(
    bet, guess, king_pos, result, profit_loss,
    st.session_state.wallet,
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
    st.session_state.initial_deposit,  
    st.session_state.total_won  
)
```

```
# Quit logic  
  
if st.button("Quit Game"):  
  
    st.write(f"🏁 You walk away with ${st.session_state.wallet}")  
    st.session_state.game_started = False  
  
    st.session_state.wallet = 0
```

Let me know if you'd like to add a “📊 View Game History” button or auto-reset the CSV when a new deposit is made. I can scaffold that next.

Yes proceed that way

Brilliant, Jagdev — let's add a “📊 View Game History” feature so players or educators can review past rounds directly inside the app. We'll also include a “🧹 Reset Log” button to clear the CSV when starting fresh.

Here's how to extend your app cleanly:

 **Step 1: Add CSV Viewer and Reset Logic to main.py**

Place this block **below** your game loop:

```
python  
  
import pandas as pd  
  
import os  
  
  
log_path = "assets/cash_won_lost/game_log.csv"
```

```
# View Game History  
  
if os.path.exists(log_path):
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
with st.expander("📊 View Game History"):  
  
    df = pd.read_csv(log_path)  
  
    st.dataframe(df, use_container_width=True)  
  
  
    # Summary stats  
  
    st.write(f"🔢 Total Rounds Played: {len(df)}")  
    st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}")  
  
  
    # Reset log  
  
    if st.button("✍️ Reset Game Log"):  
        os.remove(log_path)  
        st.success("Game log has been reset.")
```

## 📦 Step 2: Add pandas to requirements.txt

```
txt  
  
streamlit>=1.25.0  
  
pandas>=2.0.0
```

## ✳️ Optional Enhancements

- Add filters by result (✅ / ❌) or bet size.
- Export log to PDF or CSV with a download button.
- Display wallet progression as a line chart.

This gives your app a full educator-facing dashboard: round-by-round breakdown, summary stats, and reset control. Want to visualize wallet balance over time or add a “Top 5 Wins” leaderboard next? I can scaffold that too.

How to avoid this error: streamlit.errors.StreamlitValueAboveMaxError: The value 1 is greater than the max\_value 0. Traceback: File "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-packages\streamlit\runtime\scriptrunner\exec\_code.py", line 128, in exec\_func\_with\_error\_handling result = func() ^^^^^^ File

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 669, in code_to_exec
exec(code, module.__dict__) # noqa: S102 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\main.py", line 52, in <module> bet_input =
st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File
File "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-packages\streamlit\runtime\metrics_util.py", line 443, in wrapped_func result =
non_optional_func(*args, **kwargs) ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-packages\streamlit\elements\widgets\number_input.py", line 401, in number_input return
self._number_input( ^^^^^^^^^^ File "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-packages\streamlit\elements\widgets\number_input.py", line 544, in
_number_input raise StreamlitValueAboveMaxError(value=value, max_value=max_value)
```

This error is happening because you're trying to render a `st.number_input` with `min_value=1` while `max_value=st.session_state.wallet` is currently 0. Streamlit throws a `StreamlitValueAboveMaxError` when the default value (which is `min_value` by default) exceeds the `max_value`.

### Fix: Guard the `number_input` with a Wallet Check

Wrap the bet input in a conditional block that only shows it when the wallet is greater than zero:

python

```
if st.session_state.wallet > 0:
    bet_input = st.number_input(
        "Place your bet",
        min_value=1,
        max_value=st.session_state.wallet
    )
else:
    st.warning("⚠️ Your wallet is empty. Please restart the game.")
```

### Why This Works

- Prevents Streamlit from rendering an invalid input range.

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

- Gives the user feedback when their wallet hits zero.
- Keeps your app stable and user-friendly.

If you'd like, I can also help you auto-disable the "Reveal" button when the wallet is empty, or add a "Restart Game" button that resets everything cleanly. Want to scaffold that next?

Yes

Awesome, Jagdev — let's add a clean “↻ Restart Game” button that resets everything: wallet, deposit, total won, and optionally clears the game log. This gives players a fresh start without needing to reload the app.

 **Add Restart Button to main.py**

Place this block near the bottom of your app, ideally below the “Quit Game” and “View Game History” sections:

python

```
# ↻ Restart Game

if st.button("↻ Restart Game"):

    st.session_state.wallet = 0

    st.session_state.initial_deposit = 0

    st.session_state.total_won = 0

    st.session_state.game_started = False

    # Optional: Clear game log

    if os.path.exists(log_path):

        os.remove(log_path)

    st.success("Game has been reset. You can start fresh with a new deposit.")
```

 **What This Does**

- Resets all session state variables.
- Clears the CSV log if it exists.

- Prepares the app for a new deposit and gameplay session.

## ✳️ Optional Enhancements

- Add a confirmation checkbox before resetting.
- Show a summary of the previous session before wiping.
- Auto-scroll to the deposit input after reset.

Let me know if you'd like to add a "📈 Wallet Progression Chart" or export the final session summary before reset. I can scaffold that next.

Update the current main.py code for this new feature.

```
import streamlit as st
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round
st.set_page_config(page_title="Find the King", layout="centered") # Session state if 'wallet' not in st.session_state: st.session_state.wallet = 0 if 'game_started' not in st.session_state: st.session_state.game_started = False if 'initial_deposit' not in st.session_state: st.session_state.initial_deposit = 0 if 'total_won' not in st.session_state: st.session_state.total_won = 0 # Deposit if not st.session_state.game_started: deposit = st.number_input("💵 Enter deposit amount", min_value=1) if st.button("Start Game"): st.session_state.wallet = deposit * 2 st.session_state.initial_deposit = deposit st.session_state.total_won = 0 st.session_state.game_started = True st.success(f"Game started! Wallet: ${st.session_state.wallet}") # Game loop if st.session_state.game_started and st.session_state.wallet > 0: st.write(f"💰 Wallet: ${st.session_state.wallet}") import pandas as pd
import os
log_path = "assets/cash_won_lost/game_log.csv" # View Game History
if os.path.exists(log_path): with st.expander("📊 View Game History"): df = pd.read_csv(log_path)
st.dataframe(df, use_container_width=True) # Summary stats
st.write(f"🕒 Total Rounds Played: {len(df)}")
st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}")
# Reset log if st.button("📝 Reset Game Log"): os.remove(log_path)
st.success("Game log has been reset.") # Inputs
bet_input = st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)
guess_input = st.selectbox("Guess the King's position", [1, 2, 3]) # Reveal logic
if st.button("Reveal"): bet = bet_input guess = guess_input cards = shuffle_cards()
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet, st.session_state.wallet)
st.write(result)
st.write(f"👑 King was at position {king_pos}") # Show card images
cols = st.columns(3)
for i, card in enumerate(cards):
with cols[i]: st.image(get_card_image(card), caption=f"Position {i+1}", width=150) # Calculate profit/loss
profit_loss = bet * 2 if "won" in result else -bet # Update total won if profit_loss > 0
st.session_state.total_won += profit_loss # Log the round
log_game_round(bet, guess, king_pos, result, profit_loss, st.session_state.wallet, st.session_state.initial_deposit)
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.total_won ) # Quit logic if st.button("Quit Game"): st.write(f"🏁 You walk away with ${st.session_state.wallet}") st.session_state.game_started = False st.session_state.wallet = 0
```

Here's your updated main.py, Jagdev — now with a clean 🔄 “Restart Game” button that resets the session and optionally clears the game log. I've preserved all your existing logic and added the new feature modularly at the bottom.

 **Updated main.py**

```
python

import streamlit as st

import pandas as pd

import os

from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round

st.set_page_config(page_title="Find the King", layout="centered")
```

```
# Session state

if 'wallet' not in st.session_state:

    st.session_state.wallet = 0

if 'game_started' not in st.session_state:

    st.session_state.game_started = False

if 'initial_deposit' not in st.session_state:

    st.session_state.initial_deposit = 0

if 'total_won' not in st.session_state:

    st.session_state.total_won = 0
```

```
log_path = "assets/cash_won_lost/game_log.csv"
```

```
# Deposit
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
if not st.session_state.game_started:

    deposit = st.number_input(" 💰 Enter deposit amount", min_value=1)

    if st.button("Start Game"):

        st.session_state.wallet = deposit * 2

        st.session_state.initial_deposit = deposit

        st.session_state.total_won = 0

        st.session_state.game_started = True

        st.success(f"Game started! Wallet: ${st.session_state.wallet}")

# Game loop

if st.session_state.game_started and st.session_state.wallet > 0:

    st.write(f" 💰 Wallet: ${st.session_state.wallet}")

# Inputs

if st.session_state.wallet > 0:

    bet_input = st.number_input("Place your bet", min_value=1, max_value=st.session_state.wallet)

    guess_input = st.selectbox("Guess the King's position", [1, 2, 3])

# Reveal logic

if st.button("Reveal"):

    bet = bet_input

    guess = guess_input

    cards = shuffle_cards()
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
st.session_state.wallet)

st.write(result)

st.write(f"👑 King was at position {king_pos}")

# Show card images
cols = st.columns(3)

for i, card in enumerate(cards):
    with cols[i]:
        st.image(get_card_image(card), caption=f"Position {i+1}", width=150)

# Calculate profit/loss
profit_loss = bet * 2 if "won" in result else -bet

# Update total won
if profit_loss > 0:
    st.session_state.total_won += profit_loss

# Log the round
log_game_round(
    bet, guess, king_pos, result, profit_loss,
    st.session_state.wallet,
    st.session_state.initial_deposit,
    st.session_state.total_won
)
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
# Quit logic

if st.button("Quit Game"):

    st.write(f"🏁 You walk away with ${st.session_state.wallet}")

    st.session_state.game_started = False

    st.session_state.wallet = 0


# 📊 View Game History

if os.path.exists(log_path):

    with st.expander("📊 View Game History"):

        df = pd.read_csv(log_path)

        st.dataframe(df, use_container_width=True)


        st.write(f"🔢 Total Rounds Played: {len(df)}")

        st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}")


    if st.button("⚡ Reset Game Log"):

        os.remove(log_path)

        st.success("Game log has been reset.")


# 🔁 Restart Game

if st.button("🔁 Restart Game"):

    st.session_state.wallet = 0

    st.session_state.initial_deposit = 0

    st.session_state.total_won = 0

    st.session_state.game_started = False
```

```
if os.path.exists(log_path):  
  
    os.remove(log_path)  
  
st.success("Game has been reset. You can start fresh with a new deposit.")
```

Let me know if you'd like to add a “ Wallet Progression Chart” or export the final session summary before reset. I can scaffold that next.

Yes

Perfect, Jagdev — let's add a visual “ Wallet Progression Chart” so players and educators can see how the wallet balance evolves over each round. This will appear alongside the game history and summary stats.

### Step 1: Add Wallet Chart to Game History Section

Update the main.py block under  View Game History like this:

python

```
#  View Game History  
  
if os.path.exists(log_path):  
  
    with st.expander("  View Game History"):  
  
        df = pd.read_csv(log_path)  
  
        st.dataframe(df, use_container_width=True)  
  
  
        st.write(f"  Total Rounds Played: {len(df)}")  
  
        st.write(f"  Net Profit/Loss: ${df['Profit/Loss'].sum()}")  
  
  
    #  Wallet Progression Chart  
  
    st.line_chart(df["Wallet Balance"], use_container_width=True)
```

```
# 🖌️ Reset log

if st.button("✍️ Reset Game Log"):
    os.remove(log_path)
    st.success("Game log has been reset.")
```

## 🧠 Why This Works

- Uses `st.line_chart()` to visualize wallet balance over time.
- Automatically updates as new rounds are logged.
- Helps players reflect on betting strategy and outcomes.

## ✳️ Optional Enhancements

- Add a second chart for cumulative winnings (Total Won).
- Use `st.altair_chart()` for more control over axes and tooltips.
- Highlight peak wallet value or biggest win round.

Let me know if you'd like to add a leaderboard for top 5 winning rounds or a downloadable session summary next. I can scaffold that in seconds.

`streamlit.errors.StreamlitValueAboveMaxError`: This app has encountered an error. The original error message is redacted to prevent data leaks. Full error details have been recorded in the logs (if you're on Streamlit Cloud, click on 'Manage app' in the lower right of your app). Traceback: File "/mount/src/find-the-king/main.py", line 52, in <module> bet\_input = st.number\_input("Place your bet", min\_value=1, max\_value=st.session\_state.wallet) File "/home/adminuser/venv/lib/python3.13/site-packages/streamlit/runtime/metrics\_util.py", line 443, in wrapped\_func result = non\_optional\_func(\*args, \*\*kwargs) File "/home/adminuser/venv/lib/python3.13/site-packages/streamlit/elements/widgets/number\_input.py", line 401, in number\_input return self.\_number\_input( ~~~~~^ label=label, ^^^^^^^^^^ ...<15 lines>... ctx=ctx, ^^^^^^ ) ^ File "/home/adminuser/venv/lib/python3.13/site-packages/streamlit/elements/widgets/number\_input.py", line 544, in \_number\_input raise StreamlitValueAboveMaxError(value=value, max\_value=max\_value)

This error is happening because you're trying to render a `st.number_input` with `min_value=1` while `max_value=st.session_state.wallet` is currently 0. Streamlit throws a `StreamlitValueAboveMaxError` when the default value (which is `min_value` by default) exceeds the `max_value`.

### ✓ Fix: Guard the Bet Input with a Wallet Check

Update your bet input logic like this:

python

```
if st.session_state.wallet >= 1:  
  
    bet_input = st.number_input(  
  
        "Place your bet",  
  
        min_value=1,  
  
        max_value=st.session_state.wallet  
  
)  
  
guess_input = st.selectbox("Guess the King's position", [1, 2, 3])  
  
else:  
  
    st.warning(" 💰 Your wallet is empty or too low to place a bet. Please restart the game.")
```

This ensures the input only appears when the wallet has enough funds to support the minimum bet.

### 🧠 Why This Works

- Prevents Streamlit from rendering an invalid input range.
- Avoids runtime crashes due to `min_value > max_value`.
- Gives users clear feedback when their wallet is depleted.

If you'd like, I can also help you auto-disable the "Reveal" button when the wallet is empty, or add a "Restart Game" button that resets everything cleanly. Want to scaffold that next?

Update code without indentation error. Current code is:

```
import streamlit as st  
import pandas as pd  
import os  
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round  
st.set_page_config(page_title="Find the King", layout="centered") # Session state if 'wallet' not in st.session_state:  
st.session_state.wallet = 0 if 'game_started' not in st.session_state:  
st.session_state.game_started = False if 'initial_deposit' not in st.session_state:  
st.session_state.initial_deposit = 0 if 'total_won' not in st.session_state:  
st.session_state.total_won = 0  
log_path = "assets/cash_won_lost/game_log.csv" # Deposit if not  
st.session_state.game_started: deposit = st.number_input(" 💰 Enter deposit amount",  
min_value=1) if st.button("Start Game"): st.session_state.wallet = deposit * 2
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.initial_deposit      =      deposit      st.session_state.total_won      =      0
st.session_state.game_started        =      True       st.success(f"Game      started!      Wallet:
${st.session_state.wallet}")      #      Game      loop      if      st.session_state.game_started      and
st.session_state.wallet > 0: st.write(f" 💰      Wallet: ${st.session_state.wallet}")      #      Inputs      if
st.session_state.wallet >= 1: bet_input = st.number_input("Place your bet", min_value=1,
max_value=st.session_state.wallet) guess_input = st.selectbox("Guess the King's position", [1, 2,
3]) else: st.warning(" 💸 Your wallet is empty or too low to place a bet. Please restart the game.")"
# Reveal logic if st.button("Reveal"): bet = bet_input guess = guess_input cards = shuffle_cards()
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
st.session_state.wallet) st.write(result) st.write(f" 🎖 King was at position {king_pos}") # Show
card      images      cols = st.columns(3) for i, card in enumerate(cards): with cols[i]: st.image(get_card_image(card),
caption=f"Position {i+1}", width=150) # Calculate profit/loss
profit_loss = bet * 2 if "won" in result else -bet # Update total won if profit_loss > 0:
st.session_state.total_won += profit_loss # Log the round log_game_round( bet, guess, king_pos,
result,      profit_loss,      st.session_state.wallet,      st.session_state.initial_deposit,
st.session_state.total_won ) # Quit logic if st.button("Quit Game"): st.write(f" 🚫 You walk away
with ${st.session_state.wallet}") st.session_state.game_started = False st.session_state.wallet = 0
# 📊 View Game History if os.path.exists(log_path): with st.expander(" 📊 View Game History"):
df = pd.read_csv(log_path) st.dataframe(df, use_container_width=True) st.write(f" 📈 {len(df)} Total
Rounds Played: {len(df)}") st.write(f" 💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}") # 📈 Wallet
Progression Chart st.line_chart(df["Wallet Balance"], use_container_width=True) # ✎ Reset log
if st.button(" ✎ Reset Game Log"): os.remove(log_path) st.success("Game log has been reset.")"
# 📊 View Game History # if os.path.exists(log_path): # with st.expander(" 📊 View Game
History"): # df = pd.read_csv(log_path) # st.dataframe(df, use_container_width=True) #
st.write(f" 📈 {len(df)} Total Rounds Played: {len(df)}") # st.write(f" 💰 Net Profit/Loss:
${df['Profit/Loss'].sum()}") # if st.button(" ✎ Reset Game Log"): # os.remove(log_path) #
st.success("Game log has been reset.") # 🔄 Restart Game if st.button(" 🔄 Restart Game"):
st.session_state.wallet = 0 st.session_state.initial_deposit = 0 st.session_state.total_won = 0
st.session_state.game_started = False if os.path.exists(log_path): os.remove(log_path)
st.success("Game has been reset. You can start fresh with a new deposit.")
```

Here's your fully corrected main.py, Jagdev — now with clean indentation, no runtime errors, and all features intact: deposit, gameplay, reveal, logging, history viewer, chart, reset log, and restart game.

 **Updated main.py (Indentation Fixed)**

python

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
import streamlit as st
import pandas as pd
import os
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round

st.set_page_config(page_title="Find the King", layout="centered")

# Session state
if 'wallet' not in st.session_state:
    st.session_state.wallet = 0
if 'game_started' not in st.session_state:
    st.session_state.game_started = False
if 'initial_deposit' not in st.session_state:
    st.session_state.initial_deposit = 0
if 'total_won' not in st.session_state:
    st.session_state.total_won = 0

log_path = "assets/cash_won_lost/game_log.csv"

# Deposit
if not st.session_state.game_started:
    deposit = st.number_input("₹ Enter deposit amount", min_value=1)
    if st.button("Start Game"):
        st.session_state.wallet = deposit * 2
        st.session_state.initial_deposit = deposit
        st.session_state.total_won = 0
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.game_started = True

st.success(f"Game started! Wallet: ${st.session_state.wallet}")

# Game loop

if st.session_state.game_started and st.session_state.wallet > 0:

    st.write(f" 💰 Wallet: ${st.session_state.wallet}")

    if st.session_state.wallet >= 1:

        bet_input = st.number_input(
            "Place your bet",
            min_value=1,
            max_value=st.session_state.wallet
        )

        guess_input = st.selectbox("Guess the King's position", [1, 2, 3])

    # Reveal logic

    if st.button("Reveal"):

        bet = bet_input

        guess = guess_input

        cards = shuffle_cards()

        st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
        st.session_state.wallet)

        st.write(result)

        st.write(f" 🎉 King was at position {king_pos}")
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
# Show card images
cols = st.columns(3)

for i, card in enumerate(cards):
    with cols[i]:
        st.image(get_card_image(card), caption=f"Position {i+1}", width=150)

# Calculate profit/loss
profit_loss = bet * 2 if "won" in result else -bet

# Update total won
if profit_loss > 0:
    st.session_state.total_won += profit_loss

# Log the round
log_game_round(
    bet, guess, king_pos, result, profit_loss,
    st.session_state.wallet,
    st.session_state.initial_deposit,
    st.session_state.total_won
)
else:
    st.warning("💰 Your wallet is empty or too low to place a bet. Please restart the game.")

# Quit logic
if st.button("Quit Game"):
    st.write(f"🏁 You walk away with ${st.session_state.wallet}")
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.game_started = False
```

```
st.session_state.wallet = 0
```

```
# 📊 View Game History
```

```
if os.path.exists(log_path):
```

```
    with st.expander(" 📊 View Game History"):
```

```
        df = pd.read_csv(log_path)
```

```
        st.dataframe(df, use_container_width=True)
```

```
        st.write(f" 12 34 Total Rounds Played: {len(df)}")
```

```
        st.write(f" 💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}")
```

```
# 📈 Wallet Progression Chart
```

```
st.line_chart(df["Wallet Balance"], use_container_width=True)
```

```
# ✎ Reset log
```

```
if st.button(" ✎ Reset Game Log"):
```

```
    os.remove(log_path)
```

```
    st.success("Game log has been reset.")
```

```
# 🔁 Restart Game
```

```
if st.button(" 🔁 Restart Game"):
```

```
    st.session_state.wallet = 0
```

```
    st.session_state.initial_deposit = 0
```

```
    st.session_state.total_won = 0
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.game_started = False
```

```
if os.path.exists(log_path):
```

```
    os.remove(log_path)
```

```
st.success("Game has been reset. You can start fresh with a new deposit.")
```

Let me know if you'd like to add a leaderboard for top 5 wins, export session summary, or even a chart comparing bet vs. profit per round. I can scaffold that next.

```
pandas.errors.ParserError: Error tokenizing data. C error: Expected 6 fields in line 3, saw 8
Traceback:      File      "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\streamlit\runtime\scriptrunner\exec_code.py",      line      128,      in
exec_func_with_error_handling      result      =      func()      ^^^^^^      File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\streamlit\runtime\scriptrunner\script_runner.py",  line 669,  in code_to_exec
exec(code, module.__dict__) # noqa: S102      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^      File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\main.py", line 84, in <module> df =
pd.read_csv(log_path)      ^^^^^^^^^^^^^^      File "C:\Users\jagdevsinghdosanjh.RHINO\find-
the-king\venv\Lib\site-packages\pandas\io\parsers\readers.py", line 1026, in read_csv
return _read(filepath_or_buffer,      kwds)      ^^^^^^^^^^^^^^      File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\pandas\io\parsers\readers.py", line 626, in _read
return parser.read(nrows)      ^^^^^^      File "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\pandas\io\parsers\readers.py", line 1923, in read ) = self._engine.read( # type:
ignore[attr-defined]      ^^^^^^      File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\pandas\io\parsers\c_parser_wrapper.py", line 234, in read
chunks = self._reader.read_low_memory(nrows)      ^^^^^^      File
"pandas/_libs/parsers.pyx", line 838, in pandas._libs.parsers.TextReader.read_low_memory
File "pandas/_libs/parsers.pyx", line 905, in pandas._libs.parsers.TextReader._read_rows
File "pandas/_libs/parsers.pyx", line 874, in pandas._libs.parsers.TextReader._tokenize_rows
File "pandas/_libs/parsers.pyx", line 891, in pandas._libs.parsers.TextReader._check_tokenize_status
File "pandas/_libs/parsers.pyx", line 2061, in pandas._libs.parsers.raise_parse
```

This error means that your CSV file (game\_log.csv) has **inconsistent columns** — some rows have 6 fields, others have 8. That mismatch breaks pandas.read\_csv() because it expects a uniform structure.

### Fix Options

#### Option 1: Regenerate the CSV Cleanly

Since your updated logging now includes 8 fields, older rows with only 6 fields are incompatible. Here's how to reset:

```
python
```

```
if st.button("✍️ Reset Game Log"):  
    if os.path.exists(log_path):  
        os.remove(log_path)  
    st.success("Game log has been reset.")
```

This clears out the malformed file so future logs match your new 8-column format.

#### Option 2: Read CSV with Flexible Parsing (not recommended long-term)

If you want to inspect the broken file without deleting it:

```
python
```

```
df = pd.read_csv(log_path, error_bad_lines=False, warn_bad_lines=True)
```

 Deprecated in newer pandas versions and may silently drop rows. Use only for debugging.

#### Best Practice Going Forward

Ensure your log\_game\_round() in utils.py always writes 8 fields:

```
python
```

```
writer.writerow([  
    initial_deposit, bet, guess, king_pos,  
    result, profit_loss, wallet, total_won  
)
```

And make sure the header matches:

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

python

```
writer.writerow([
    "Initial Deposit", "Bet", "Guess", "King Position",
    "Result", "Profit/Loss", "Wallet Balance", "Total Won"
])
```

Let me know if you'd like to add a CSV validator or auto-repair logic to clean up malformed rows before reading. I can scaffold that next.

Update current utils.py code :

```
import random
import csv
import os

def shuffle_cards():
    cards = ["King", "Queen", "Jack"]
    random.shuffle(cards)
    return cards

def evaluate_guess(cards, guess, bet, wallet):
    king_position = cards.index("King") + 1
    if guess == king_position:
        wallet += bet * 2
    result = f" {checkmark} Correct! You won ${bet * 2}" if guess == king_position else f" {cross} Wrong! You lost ${bet}"
    return wallet, result, king_position

def get_card_image(card_name):
    return f"assets/card_images/{card_name.lower()}.png"

def log_game_round(bet, guess, king_pos, result, profit_loss, wallet, initial_deposit, total_won):
    folder_path = "assets/cash_won_lost"
    log_path = os.path.join(folder_path, "game_log.csv")
    os.makedirs(folder_path, exist_ok=True)
    file_exists = os.path.isfile(log_path)
    with open(log_path, mode='a', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        if not file_exists:
            writer.writerow(["Initial Deposit", "Bet", "Guess", "King Position", "Result", "Profit/Loss", "Wallet Balance", "Total Won"])
        writer.writerow([initial_deposit, bet, guess, king_pos, result, profit_loss, wallet, total_won])
```

Here's your updated utils.py, Jagdev — now enhanced with:

- Consistent 8-column logging
- UTF-8 encoding for emoji support
- Folder creation safeguard
- Optional CSV validation to prevent malformed rows (added as a bonus utility)

## Final utils.py (Updated and Extended)

python

```
import random
import csv
import os
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
def shuffle_cards():

    cards = ["King", "Queen", "Jack"]

    random.shuffle(cards)

    return cards


def evaluate_guess(cards, guess, bet, wallet):

    king_position = cards.index("King") + 1

    if guess == king_position:

        wallet += bet * 2

        result = f" ✅ Correct! You won ${bet * 2}"

    else:

        wallet -= bet

        result = f" ❌ Wrong! You lost ${bet}"

    return wallet, result, king_position


def get_card_image(card_name):

    return f"assets/card_images/{card_name.lower()}.png"


def log_game_round(bet, guess, king_pos, result, profit_loss, wallet, initial_deposit, total_won):

    folder_path = "assets/cash_won_lost"

    log_path = os.path.join(folder_path, "game_log.csv")



    os.makedirs(folder_path, exist_ok=True)

    file_exists = os.path.isfile(log_path)
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

with open(log\_path, mode='a', newline='', encoding='utf-8') as file:

```
writer = csv.writer(file)

if not file_exists:

    writer.writerow([
        "Initial Deposit", "Bet", "Guess", "King Position",
        "Result", "Profit/Loss", "Wallet Balance", "Total Won"
    ])

    writer.writerow([
        initial_deposit, bet, guess, king_pos,
        result, profit_loss, wallet, total_won
    ])
```

# Optional: Validate CSV structure (useful for debugging)

```
def validate_game_log(path="assets/cash_won_lost/game_log.csv", expected_columns=8):

    if not os.path.exists(path):

        return True

    with open(path, encoding='utf-8') as file:

        for i, line in enumerate(file, start=1):

            if len(line.strip().split(',')) != expected_columns:

                print(f"⚠️ Line {i} has incorrect number of columns.")

        return False

    return True
```

Let me know if you'd like to use validate\_game\_log() inside main.py to warn users about malformed logs before reading. I can wire that in next.

```
pandas.errors.ParserError: Error tokenizing data. C error: Expected 6 fields in line 3, saw 8
Traceback:      File      "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-packages\streamlit\runtime\scriptrunner\exec_code.py",      line      128,      in
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
exec_func_with_error_handling      result      =      func()      ^^^^^^      File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\streamlit\runtime\scriptrunner\script_runner.py",  line 669,  in code_to_exec
exec(code,  module.__dict__)  # noqa: S102  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\main.py",  line 84,  in <module> df =
pd.read_csv(log_path) ^^^^^^^^^^^^^^  File "C:\Users\jagdevsinghdosanjh.RHINO\find-
the-king\venv\Lib\site-packages\pandas\io\parsers\readers.py",  line 1026,  in read_csv return
_read(filepath_or_buffer,      kwds)      ^^^^^^^^^^^^^^  File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\pandas\io\parsers\readers.py",  line 626,  in _read return parser.read(nrows)
^^^^^^^^^  File "C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\pandas\io\parsers\readers.py",  line 1923,  in read ) = self._engine.read( # type:
ignore[attr-defined]      ^^^^^^^^^^  File
"C:\Users\jagdevsinghdosanjh.RHINO\find-the-king\venv\Lib\site-
packages\pandas\io\parsers\c_parser_wrapper.py",  line 234,  in read chunks =
self._reader.read_low_memory(nrows)      ^^^^^^  File
"pandas/_libs/parsers.pyx",  line 838,  in pandas._libs.parsers.TextReader.read_low_memory  File
"pandas/_libs/parsers.pyx",  line 905,  in pandas._libs.parsers.TextReader._read_rows  File
"pandas/_libs/parsers.pyx",  line 874,  in pandas._libs.parsers.TextReader._tokenize_rows  File
"pandas/_libs/parsers.pyx",  line 891,  in pandas._libs.parsers.TextReader._check_tokenize_status  File
"pandas/_libs/parsers.pyx",  line 2061,  in pandas._libs.parsers.raise_parser_error
```

This traceback confirms that your game\_log.csv file contains **inconsistent rows** — some with 6 columns, others with 8 — which breaks pandas.read\_csv() because it expects a uniform structure.

### Immediate Fix: Auto-Validate and Reset Malformed CSV

Let's add a quick validation before reading the CSV in main.py. If the file is malformed, we'll show a warning and offer to reset it.

#### Step 1: Add This Utility to utils.py

You've already got this partially — let's finalize it:

```
python
```

```
def is_csv_valid(path="assets/cash_won_lost/game_log.csv", expected_columns=8):
    if not os.path.exists(path):
        return True
    with open(path, encoding='utf-8') as file:
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
for i, line in enumerate(file, start=1):  
  
    if len(line.strip().split(',')) != expected_columns:  
  
        return False  
  
return True
```

 **Step 2: Use It in main.py Before Reading the CSV**

Replace your current `pd.read_csv(log_path)` block with this:

```
python  
  
from utils import is_csv_valid
```

```
# 📊 View Game History  
  
if os.path.exists(log_path):  
  
    if is_csv_valid(log_path):  
  
        with st.expander("📊 View Game History"):  
  
            df = pd.read_csv(log_path)  
  
            st.dataframe(df, use_container_width=True)  
  
  
            st.write(f"🔢 Total Rounds Played: {len(df)}")  
            st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}"")  
  
  
            st.line_chart(df["Wallet Balance"], use_container_width=True)  
  
  
        if st.button("✍️ Reset Game Log"):  
            os.remove(log_path)  
  
            st.success("Game log has been reset.")  
  
    else:  
        st.error("The game log is not valid CSV format.")
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.error("⚠️ Game log is malformed. Some rows have inconsistent columns.")

if st.button("⚡ Force Reset Game Log"):
    os.remove(log_path)
    st.success("Malformed log has been cleared.")
```

## Why This Works

- Prevents your app from crashing on bad CSV structure.
  - Gives users a clear warning and safe recovery option.
  - Keeps your logging robust and educator-friendly.

Let me know if you'd like to auto-backup the malformed file before deleting it, or add a CSV cleaner that trims bad rows instead of wiping the whole file. I can scaffold that next.

This error means that Streamlit is trying to render **two buttons with the exact same label** ("Reset Game Log") in the same context, which causes a StreamlitDuplicateElementId conflict.

### Fix: Give Each Button a Unique key

Streamlit allows you to assign a key to each widget to differentiate them. Update your buttons like this:

### Inside Game History Expander

python

```
if st.button(" 🖌️ Reset Game Log", key="reset_log_normal"):  
    os.remove(log_path)  
    st.success("Game log has been reset.")
```

### Inside Malformed Log Warning

python

```
if st.button(" 🖌️ Force Reset Game Log", key="reset_log_force"):  
    os.remove(log_path)  
    st.success("Malformed log has been cleared.")
```

### Why This Works

- key ensures each button has a unique internal ID.
- Prevents Streamlit from confusing duplicate widgets.
- Keeps your UI clean and responsive.

Let me know if you'd like to add a confirmation checkbox before resetting logs or a backup feature that saves the old CSV before deletion. I can scaffold that next.

Update the entire code:

```
import streamlit as st import pandas as pd import os from utils import  
shuffle_cards, evaluate_guess, get_card_image, log_game_round, is_csv_valid  
st.set_page_config(page_title="Find the King", layout="centered") # Session state  
if 'wallet' not in st.session_state: st.session_state.wallet = 0  
if 'game_started' not in st.session_state: st.session_state.game_started = False  
if 'initial_deposit' not in st.session_state: st.session_state.initial_deposit = 0  
if 'total_won' not in st.session_state: st.session_state.total_won = 0  
log_path = "assets/cash_won_lost/game_log.csv" # Deposit if not  
st.session_state.game_started: deposit = st.number_input(" 💸 Enter deposit amount",  
min_value=1) if st.button("Start Game"): st.session_state.wallet = deposit * 2  
st.session_state.initial_deposit = deposit st.session_state.total_won = 0  
st.session_state.game_started = True st.success(f"Game started! Wallet: ${st.session_state.wallet}") # Game loop if st.session_state.game_started and  
st.session_state.wallet > 0: st.write(f" 💰 Wallet: ${st.session_state.wallet}") if  
st.session_state.wallet >= 1: bet_input = st.number_input("Place your bet", min_value=1,
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
max_value=st.session_state.wallet) guess_input = st.selectbox("Guess the King's position", [1, 2, 3]) # Reveal logic if st.button("Reveal"): bet = bet_input guess = guess_input cards = shuffle_cards() st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet, st.session_state.wallet) st.write(result) st.write(f"👑 King was at position {king_pos}") # Show card images cols = st.columns(3) for i, card in enumerate(cards): with cols[i]: st.image(get_card_image(card), caption=f"Position {i+1}", width=150) # Calculate profit/loss profit_loss = bet * 2 if "won" in result else -bet # Update total won if profit_loss > 0: st.session_state.total_won += profit_loss # Log the round log_game_round( bet, guess, king_pos, result, profit_loss, st.session_state.wallet, st.session_state.initial_deposit, st.session_state.total_won ) else: st.warning("wallet Your wallet is empty or too low to place a bet. Please restart the game.") # Quit logic if st.button("Quit Game"): st.write(f"🏁 You walk away with ${st.session_state.wallet}") st.session_state.game_started = False st.session_state.wallet = 0 # 📊 View Game History if os.path.exists(log_path): if is_csv_valid(log_path): with st.expander("📊 View Game History"): df = pd.read_csv(log_path) st.dataframe(df, use_container_width=True) st.write(f"12 Total Rounds Played: {len(df)}") st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}") st.line_chart(df["Wallet Balance"], use_container_width=True) if st.button("✍️ Reset Game Log"): os.remove(log_path) st.success("Game log has been reset.") else: st.error("⚠️ Game log is malformed. Some rows have inconsistent columns.") if st.button("✍️ Force Reset Game Log"): os.remove(log_path) st.success("Malformed log has been cleared.") # 📊 View Game History if os.path.exists(log_path): with st.expander("📊 View Game History"): df = pd.read_csv(log_path) st.dataframe(df, use_container_width=True) st.write(f"12 Total Rounds Played: {len(df)}") st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}") # 📈 Wallet Progression Chart st.line_chart(df["Wallet Balance"], use_container_width=True) # ✍️ Reset log if st.button("✍️ Reset Game Log"): os.remove(log_path) st.success("Game log has been reset.") # 🔗 Restart Game if st.button("🔗 Restart Game"): st.session_state.wallet = 0 st.session_state.initial_deposit = 0 st.session_state.total_won = 0 st.session_state.game_started = False if os.path.exists(log_path): os.remove(log_path) st.success("Game has been reset. You can start fresh with a new deposit.") for main.py
```

Here's your fully updated and cleaned-up main.py, Jagdev — now with:

- Proper session state handling
- Game loop and reveal logic
- CSV logging with validation

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

- Unique keys for buttons to avoid duplication errors
- Single, consolidated game history viewer
- Restart and reset functionality

**Final main.py**

```
python
```

```
import streamlit as st
import pandas as pd
import os
from utils import shuffle_cards, evaluate_guess, get_card_image, log_game_round, is_csv_valid

st.set_page_config(page_title="Find the King", layout="centered")

# Session state
if 'wallet' not in st.session_state:
    st.session_state.wallet = 0
if 'game_started' not in st.session_state:
    st.session_state.game_started = False
if 'initial_deposit' not in st.session_state:
    st.session_state.initial_deposit = 0
if 'total_won' not in st.session_state:
    st.session_state.total_won = 0

log_path = "assets/cash_won_lost/game_log.csv"

# Deposit
if not st.session_state.game_started:
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
deposit = st.number_input(" 💰 Enter deposit amount", min_value=1)

if st.button("Start Game"):

    st.session_state.wallet = deposit * 2

    st.session_state.initial_deposit = deposit

    st.session_state.total_won = 0

    st.session_state.game_started = True

    st.success(f"Game started! Wallet: ${st.session_state.wallet}")

# Game loop

if st.session_state.game_started and st.session_state.wallet > 0:

    st.write(f" 💰 Wallet: ${st.session_state.wallet}")

if st.session_state.wallet >= 1:

    bet_input = st.number_input(
        "Place your bet",
        min_value=1,
        max_value=st.session_state.wallet
    )

    guess_input = st.selectbox("Guess the King's position", [1, 2, 3])

# Reveal logic

if st.button("Reveal"):

    bet = bet_input

    guess = guess_input

    cards = shuffle_cards()
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.session_state.wallet, result, king_pos = evaluate_guess(cards, guess, bet,
st.session_state.wallet)

st.write(result)

st.write(f"👑 King was at position {king_pos}")

# Show card images
cols = st.columns(3)

for i, card in enumerate(cards):
    with cols[i]:
        st.image(get_card_image(card), caption=f"Position {i+1}", width=150)

# Calculate profit/loss
profit_loss = bet * 2 if "won" in result else -bet

# Update total won
if profit_loss > 0:
    st.session_state.total_won += profit_loss

# Log the round
log_game_round(
    bet, guess, king_pos, result, profit_loss,
    st.session_state.wallet,
    st.session_state.initial_deposit,
    st.session_state.total_won
)
else:
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
st.warning("💸 Your wallet is empty or too low to place a bet. Please restart the game.")
```

```
# Quit logic

if st.button("Quit Game"):

    st.write(f"🏁 You walk away with ${st.session_state.wallet}")

    st.session_state.game_started = False

    st.session_state.wallet = 0


# 📊 View Game History

if os.path.exists(log_path):

    if is_csv_valid(log_path):

        with st.expander("📊 View Game History"):

            df = pd.read_csv(log_path)

            st.dataframe(df, use_container_width=True)


        st.write(f"🔢 Total Rounds Played: {len(df)}")

        st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}")



        st.line_chart(df["Wallet Balance"], use_container_width=True)


    if st.button("⚡ Reset Game Log", key="reset_log_valid"):

        os.remove(log_path)

        st.success("Game log has been reset.")

    else:

        st.error("⚠️ Game log is malformed. Some rows have inconsistent columns.")
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

```
if st.button("⚡ Force Reset Game Log", key="reset_log_invalid"):  
    os.remove(log_path)  
    st.success("Malformed log has been cleared.")
```

```
# ⚡ Restart Game
```

```
if st.button("⚡ Restart Game"):  
    st.session_state.wallet = 0  
    st.session_state.initial_deposit = 0  
    st.session_state.total_won = 0  
    st.session_state.game_started = False
```

```
if os.path.exists(log_path):  
    os.remove(log_path)
```

```
st.success("Game has been reset. You can start fresh with a new deposit.")
```

Let me know if you'd like to add a leaderboard, export summary, or even a chart comparing bet vs. profit per round. I can scaffold that next.

```
(venv) (base) PS C:\Users\jagdevsinghdosanjh.RHINO\find-the-king> streamlit run main.py You  
can now view your Streamlit app in your browser. Local URL: http://localhost:8501 Network URL:  
http://192.168.1.3:8501 2025-09-16 23:17:02.284 Please replace `use_container_width` with  
'width'. `use_container_width` will be removed after 2025-12-31. For  
'use_container_width=True', use 'width='stretch''. For `use_container_width=False`, use  
'width='content''. 2025-09-16 23:17:31.192 Please replace `use_container_width` with `width`.  
'use_container_width` will be removed after 2025-12-31. For `use_container_width=True`, use  
'width='stretch''. For `use_container_width=False`, use 'width='content''. 2025-09-16  
23:17:32.961 Please replace `use_container_width` with `width`. `use_container_width` will be  
removed after 2025-12-31. For `use_container_width=True`, use 'width='stretch''. For  
'use_container_width=False', use 'width='content''. 2025-09-16 23:17:34.267 Please replace  
'use_container_width` with `width`. `use_container_width` will be removed after 2025-12-31.  
For `use_container_width=True`, use 'width='stretch''. For `use_container_width=False`, use
```

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

`width='content'`. 2025-09-16 23:18:02.700 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:04.963 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:05.231 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:05.496 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:05.715 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:06.016 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:06.254 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:06.701 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:16.737 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:17.683 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:18.816 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:18:57.357 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:04.176 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:05.255 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:07.789 Please replace

Find The King – Online Game App in Streamlit  
Created by Amrit Kaur Dosanjh in Collaboration with Bing's Copilot

`use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:30.752 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:36.417 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:38.577 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:19:40.592 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:20:41.405 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:20:45.500 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:20:46.847 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:20:48.875 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`. 2025-09-16 23:21:06.395 Please replace `use\_container\_width` with `width`. `use\_container\_width` will be removed after 2025-12-31. For `use\_container\_width=True`, use `width='stretch'`. For `use\_container\_width=False`, use `width='content'`.

Thanks for catching that, Jagdev — Streamlit is warning you that `use_container_width` is being deprecated and should be replaced with the new `width` parameter.

 **Quick Fix: Replace `use_container_width=True` with `width="stretch"`**

Anywhere you have:

python

```
st.dataframe(df, use_container_width=True)  
st.line_chart(df["Wallet Balance"], use_container_width=True)
```

Update to:

```
python  
st.dataframe(df, width="stretch")  
st.line_chart(df["Wallet Balance"], width="stretch")
```

### Updated Snippet for Game History Viewer

Here's the corrected block:

```
python  
with st.expander("📊 View Game History"):  
    df = pd.read_csv(log_path)  
    st.dataframe(df, width="stretch")  
  
    st.write(f"🔢 Total Rounds Played: {len(df)}")  
    st.write(f"💰 Net Profit/Loss: ${df['Profit/Loss'].sum()}"")
```

```
    st.line_chart(df["Wallet Balance"], width="stretch")
```

```
if st.button("✍️ Reset Game Log", key="reset_log_valid"):  
    os.remove(log_path)  
    st.success("Game log has been reset.")
```

### Why This Matters

- Keeps your app future-proof beyond December 2025.
- Prevents deprecation warnings from cluttering your logs.
- Ensures layout responsiveness with the new width="stretch" standard.

Let me know if you'd like to apply this across all widgets or want help styling the app with themes or layout tweaks. I can scaffold that next.