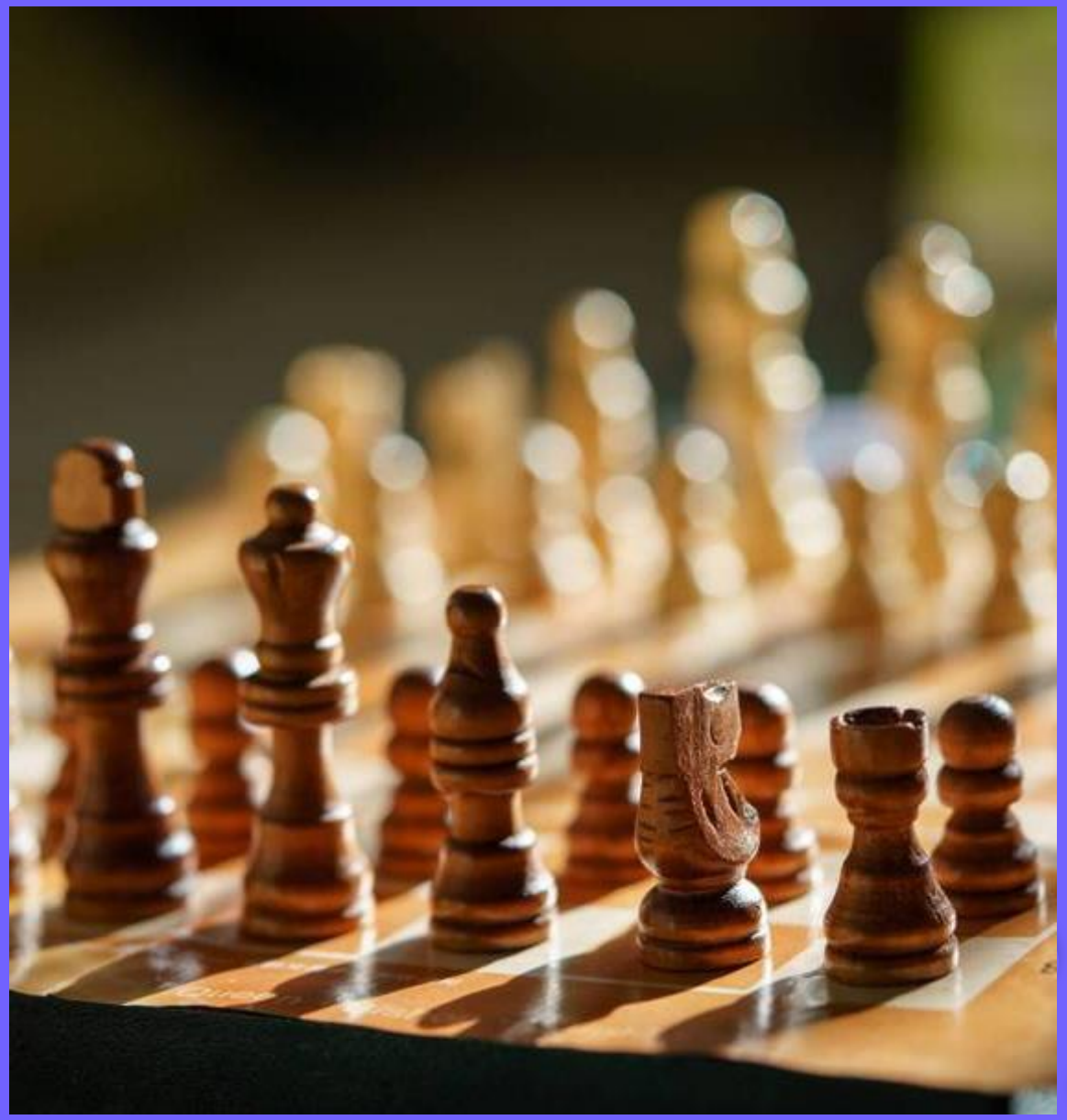
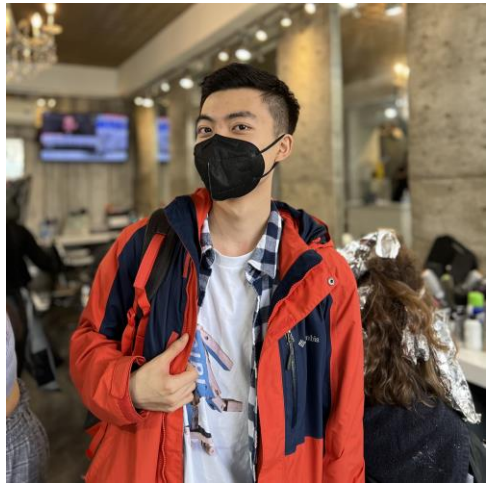
SPEAKCHESS

Team Name: We don’t know



Team



Dumebi Osadebe

Josh Fernando

Eric Zhou

Daryus Ramkalawan

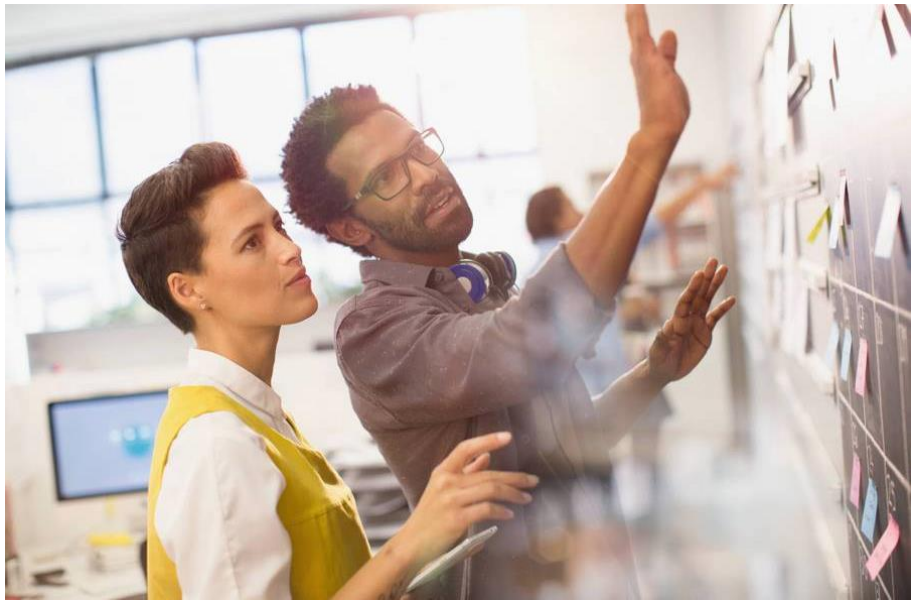
Agenda

+ Introduction
+ Our idea
+ Demo
+ Future Applications

Introduction

Chess is a game that has been enjoyed by millions of people around the world for centuries. However, for people with visual impairments or other disabilities, playing chess can be a challenging or impossible task.

Good afternoon/evening everyone. Today, We'd like to present to you a chess app that we named speakchess



The Idea

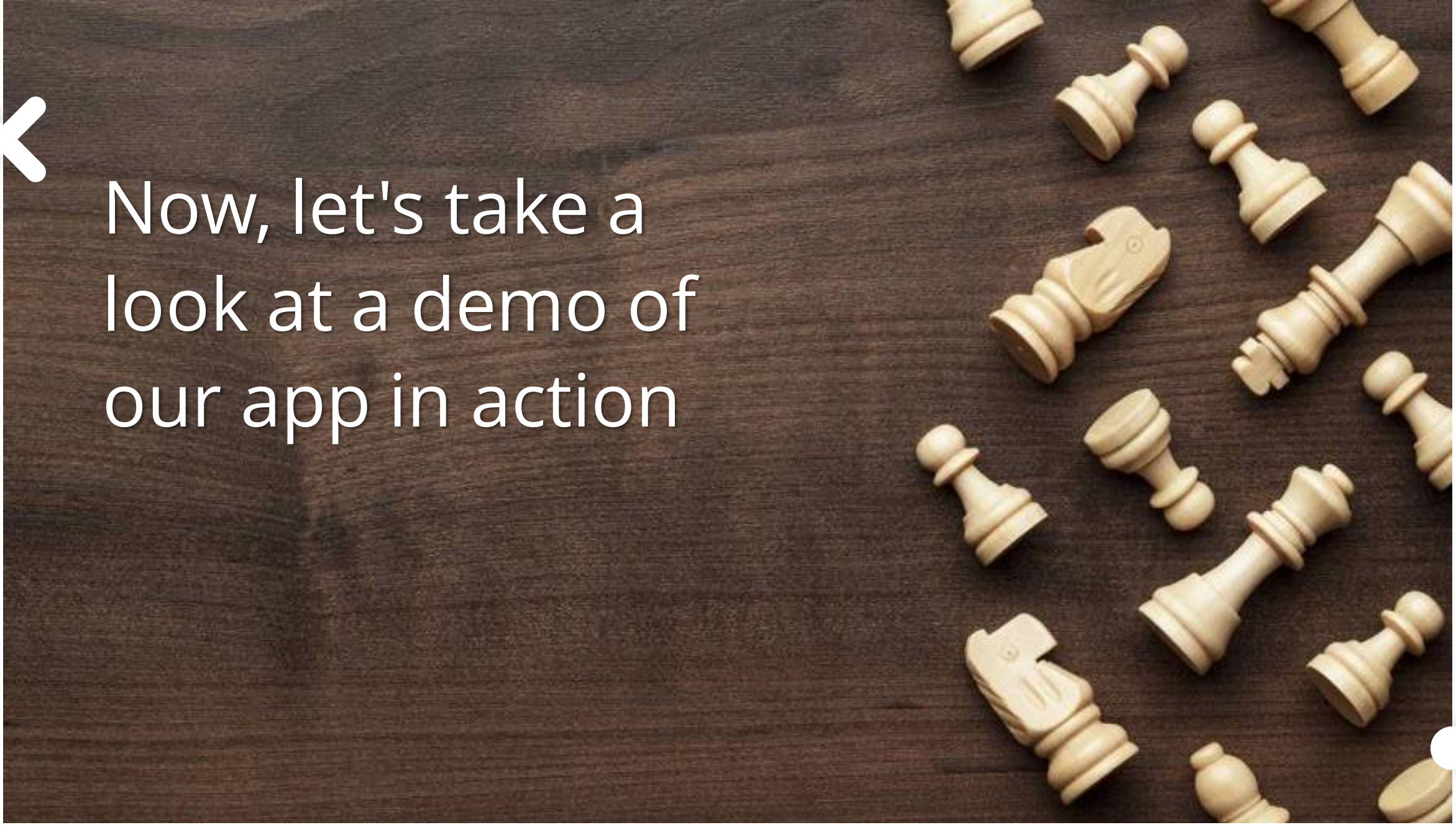
a chess app that utilizes voice recognition input and Cohere AI in python

Our app has several features that make it easy and fun to use:

Voice recognition input: Users can speak their moves instead of having to use a keyboard or mouse.

Board feedback: The app provides feedback to users about the current state of the board, including which pieces are in play and where they are located along with the possible positions it can move to

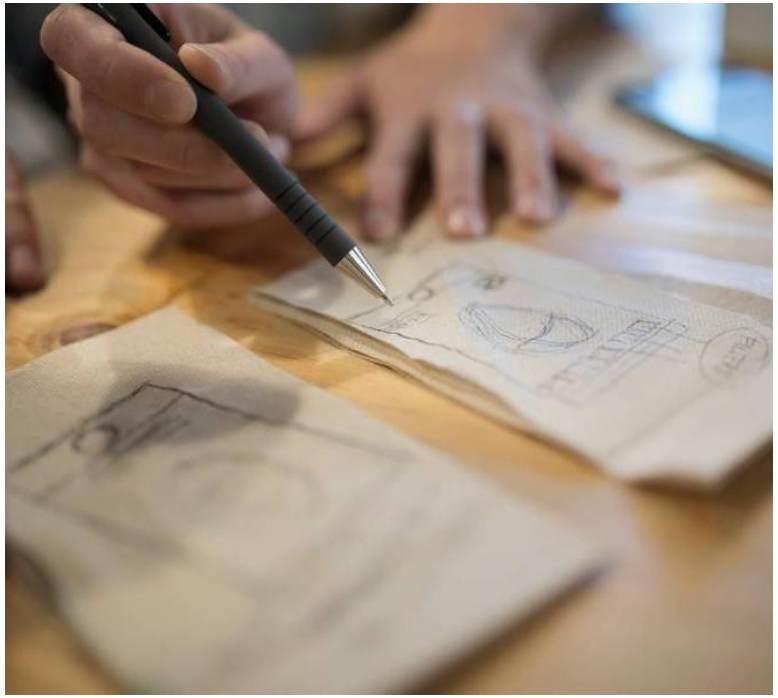
Now, let's take a look at a demo of our app in action



Demo

Subtitle

Subtitle



The future

We believe that our app has the potential to make chess more accessible to people with visual impairments or other disabilities. In the future, we plan to add additional features to the app, such as support for multiple languages and the ability to play against a computer opponent using the cohere ai library. Additionally, we can utilize cohere to train a language model on different chess moves and supply feedback by using a dataset from lichess

W.I.Ps

Training the model

```python
import pandas as pd
import cohere

client = cohere.Client(api_key='jyKUvNcdwgPZb2aR8MFRAKvkn6HsWz1xXZXLqsfo')

# Define dataset path
dataset_path = "C:\\Users\\daryu\\Downloads\\archive\\games.csv"

# Load and preprocess chess dataset
def load_dataset():
    dataset = pd.read_csv(dataset_path)
    return dataset

def preprocess_dataset(dataset):
    # Extract moves and corresponding coordinates from chess dataset
    moves = dataset["move"]
    coordinates = dataset["coordinate"]

    # Tokenize and encode moves and coordinates
    encoded_moves = client.encode_text(moves.tolist(), model_name="text-gpt2")
    encoded_coordinates = client.encode_text(coordinates.tolist(), model_name="t

    # Create input and output sequences
    input_sequences = []
    output_sequences = []
    for i in range(len(encoded_moves)):
        input_sequence = encoded_moves[i] + encoded_coordinates[i]
        output_sequence = encoded_moves[i+1] if i < len(encoded_moves)-1 else [0
        input_sequences.append(input_sequence)
        output_sequences.append(output_sequence)

    # Create input and output datasets
    input_dataset = cohereai.TensorDataset(input_sequences)
    output_dataset = cohereai.TensorDataset(output_sequences)

    return input_dataset, output_dataset
```

Upgrading the speech recognition using cohere ai

```python
import cohereai
import chess
import speech_recognition as sr

# Initialize Cohere AI client
client = cohere.Client(api_key="jyKUvNcdwgPZb2aR8MFRAKvkn6HsWz1xXZXLqsfo")

# Initialize speech recognition engine
r = sr.Recognizer()

# Initialize chess board
board = chess.Board()

# Define function for getting user move input
def get_move_input():
    with sr.Microphone() as source:
        print("Please say your move:")
        audio = r.listen(source)
    try:
        move_input = r.recognize_google(audio)
        return move_input
    except:
        print("Sorry, could not recognize your move.")
        return None

# Define function for processing user move input
def process_move_input(move_input):
    # Use Cohere AI's text classification API to determine if move input is vali
    classification = client.classify_text(move_input, model_name="chess-move-cla
    if classification[0]["label"] == "valid":
        # Use Cohere AI's named entity recognition API to extract move coordinat
        entities = client.extract_entities(move_input, model_name="chess-ner")
        if "from_square" in entities and "to_square" in entities:
            # Convert move coordinates to chess.Move object and return
            from_square = chess.SQUARE_NAMES.index(entities["from_square"].upper
            to_square = chess.SQUARE_NAMES.index(entities["to_square"].upper())
            move = chess.Move(from_square, to_square)
            return move
    print("Invalid move input.")
    return None
```

```python
# Define function for processing user move input
def process_move_input(move_input):
    # Use Cohere AI's text classification API to determine if move input is va
    classification = client.classify_text(move_input, model_name="chess-move-c
    if classification[0]["label"] == "valid":
        # Use Cohere AI's named entity recognition API to extract move coordin
        entities = client.extract_entities(move_input, model_name="chess-ner")
        if "from_square" in entities and "to_square" in entities:
            # Convert move coordinates to chess.Move object and return
            from_square = chess.SQUARE_NAMES.index(entities["from_square"].upp
            to_square = chess.SQUARE_NAMES.index(entities["to_square"].upper()
            move = chess.Move(from_square, to_square)
            return move
    print("Invalid move input.")
    return None

# Main game loop
while not board.is_game_over():
    # Get user move input
    move_input = get_move_input()

    # Process move input
    move = process_move_input(move_input)

    # Make move if valid
    if move in board.legal_moves:
        board.push(move)
        print("Your move:", move)
        # Use text-to-speech API to tell user the coordinates of their move
        client.synthesize_text(str(move), model_name="text-tts")

        # Make computer move
        computer_move = chess.engine.SimpleEngine.ponder(board).move
        board.push(computer_move)
        print("Computer move:", computer_move)
        # Use text-to-speech API to tell user the coordinates of the computer'
        client.synthesize_text(str(computer_move), model_name="text-tts")
```

Also plans to implement cohere to uses it's natural language processing APIs to interpret user moves and ensure they are valid

Thank You

- Thank you for taking the time to learn about our app. We hope that you will give it a try and see how it can make playing chess more accessible and enjoyable for everyone.