chatbot.py

```
import json
import random
import string
import nltk
from nltk.tokenize import word_tokenize
from collections import defaultdict
import time

class SimpleChatbot:
    def __init__(self, knowledge_file="chatbot_knowledge.json"):
        self.knowledge_file = knowledge_file
        self.knowledge = self.load_knowledge()
        self.conversation_history = []

        try:
            nltk.data.find('tokenizers/punkt')
        except LookupError:
            print("Downloading required NLTK data...")
            nltk.download('punkt')

        self.auto_train()

    def load_knowledge(self):
        try:
            with open(self.knowledge_file, 'r') as f:
                return json.load(f)
        except FileNotFoundError:
            return {"responses": {}, "patterns": {}}

    def save_knowledge(self):
        with open(self.knowledge_file, 'w') as f:
            json.dump(self.knowledge, f, indent=2)

    def auto_train(self):
        training_data = {
                "hello": [
            "Hi there!",
            "Hello!",
            "Hey, how can I help you?",
            "Greetings!",
            "Welcome!"
        ],
```

```
    "hi": [
        "Hello!",
        "Hi there!",
        "Hey! How are you?",
        "Greetings!"
    ],
    "how are you": [
        "I'm doing well, thanks for asking!",
        "I'm great, how are you?",
        "Excellent! How about you?"
    ],
    "sup": [
        "Not much! Just here to help you. What's up with you?",
        "Hey! Just hanging out. How can I assist you?",
        "Sup! What's going on?",
        "Not much! How about you?",
        "Just here, ready to chat! What's up?",
        "Hey! What's happening?",
        "Sup! How can I help you today?",
        "Just chilling! What's on your mind?",
        "Hey there! What's new with you?",
        "Not much! How can I brighten your day?"
    ],
    "hey": [
        "Hey! How can I assist you today?",
        "Hello! What's on your mind?",
        "Hey there! What would you like to talk about?",
        "Hi! How are you doing?",
        "Hey! I'm here to help. What do you need?",
        "Hello! How can I make your day better?",
        "Hey! What's up?",
        "Hi there! What can I do for you?",
        "Hey! How can I brighten your day?",
        "Hello! What's going on?"
    ],
        # ... keep existing code (basic conversation patterns)

        # Add more complex emotional responses
        "i feel sad": [
            "I'm sorry you're feeling down. Would you like to talk about it?",
            "Sometimes sharing what's bothering you can help. I'm here to listen.",
            "Let's try to focus on something positive. What's something good that happened
recently?",
            "Would you like to hear something uplifting?"
```

```
],
"i feel happy": [
    "That's wonderful! What made your day special?",
    "Happiness is contagious! Tell me more about what's making you happy!",
    "I'm glad you're feeling good! Let's celebrate that feeling!"
],

# Add problem-solving responses
"i need help": [
    "I'll do my best to help you. What's the specific issue?",
    "Let's break down the problem and solve it together.",
    "I'm here to assist. Could you provide more details?"
],

# Add creative writing prompts
"give me a writing prompt": [
    "Write about a world where everyone has the same face.",
    "Describe your perfect day in vivid detail.",
    "Write about a character who wakes up with the ability to understand animals.",
    "Create a story about finding an old mysterious letter in your attic."
],

"i am bored": [
    "Let's play a word game!",
    "How about watching a movie? I can recommend some.",
    "Why not try learning a new skill?",
    "You could read a book, I have many suggestions!",
    "How about doing some exercise?",
    "Want to hear a joke?",
    "Let's have an interesting discussion about any topic you like!"
],
"tell me a joke": [
    "Why don't scientists trust atoms? Because they make up everything! 😄",
    "What do you call a bear with no teeth? A gummy bear! 🐻",
    "Why did the scarecrow win an award? Because he was outstanding in his field! 🌾",
    "What do you call a fish wearing a bowtie? So-fish-ticated! 🐠",
    "Why did the math book look sad? Because it had too many problems! 📚",
    "What do you call a fake noodle? An impasta! 🍝",
    "Why did the cookie go to the doctor? Because it was feeling crumbly! 🍪",
    "What do you call a sleeping bull? A bulldozer! 🐂",
    "Why don't eggs tell jokes? They'd crack up! 🥚",
    "What do you call a pig that does karate? A pork chop! 🥋"
],
"i feel anxious": [
```

"I'm really sorry to hear that you're feeling anxious. It's completely normal to feel this way sometimes.",
"Take a deep breath. I'm here for you, and we can talk about it if you'd like.",
"Anxiety can be tough. Just know that you're not alone in this.",
"It's okay to feel anxious. Sometimes it helps to talk about what's on your mind.",
"I understand that anxiety can be overwhelming. Let's take it one step at a time.",
"If you want, I can suggest some relaxation techniques to help you feel better.",
"It's important to acknowledge your feelings. You're doing great by reaching out.",
"Have you tried any breathing exercises? They can really help calm your nerves.",
"Sometimes, just talking about it can lighten the load. What's bothering you?",
"You're not alone in feeling this way. Many people experience anxiety.",
"If you want, we can distract ourselves with something fun. What do you think?",
"Remember, it's okay to ask for help. You don't have to go through this alone.",
"Let's find a way to tackle this together. You've got this!",
"If you need a moment, that's perfectly fine. Take your time.",
"Anxiety can feel like a heavy backpack. Let's see how we can lighten it.",
"Sometimes, a little humor can help. Want to hear a joke?",
"You're doing the right thing by talking about it. Keep going!",
"If you want, I can share some tips on managing anxiety.",
"It's okay to feel anxious. Let's find a way to make it better.",
"You're stronger than you think. Let's work through this together."
],
"i feel overwhelmed": [
"It sounds like you have a lot on your plate. It's okay to feel overwhelmed.",
"Take a moment to breathe. You can handle this.",
"Feeling overwhelmed is a sign that you need to take a break. It's important to rest.",
"You're doing your best, and that's all anyone can ask for.",
"Let's tackle this together. What can we focus on first?",
"If you need help prioritizing, I can assist you with that.",
"Remember, it's okay to ask for help when you need it.",
"Sometimes, just talking about what's overwhelming you can help.",
"You're not alone in feeling this way. Many people feel overwhelmed sometimes.",
"Let's break it down into smaller tasks. It'll feel more manageable.",
"If you want, we can create a plan to tackle everything step by step.",
"You're doing great! Just take it one day at a time.",
"If you need a moment to yourself, that's perfectly fine. Take your time.",
"It's okay to take a break and recharge. You deserve it!",
"Let's find a way to lighten your load. What's the most pressing issue?",
"Sometimes, a little humor can help. Want to hear a joke?",
"You're stronger than you think. Let's work through this together.",
"If you want, I can share some tips on managing overwhelm.",
"It's okay to feel overwhelmed. Let's find a way to make it better.",
"You're doing your best, and that's what matters most."
],

```
        "recommend a movie": [
            "If you're looking for a great movie, I recommend 'The Shawshank Redemption.' It's a classic!",
            "How about 'Inception'? It's a mind-bending thriller that keeps you guessing.",
            "'The Dark Knight' is always a solid choice if you enjoy superhero films.",
            "You might enjoy 'Pulp Fiction' if you're in the mood for something unique.",
            "'Parasite' is an amazing film that won several awards. It's definitely worth a watch.",
            "If you're looking for something light-hearted, 'The Grand Budapest Hotel' is delightful.",
            "For a good laugh, 'Superbad' is a hilarious coming-of-age film.",
            "If you want a tear-jerker, 'The Pursuit of Happyness' is incredibly inspiring.",
            "How about a classic romance? 'Casablanca' is timeless.",
            "If you're into sci-fi, 'Blade Runner 2049' is visually stunning.",
            "For a thrilling ride, 'Get Out' is a must-see!",
            "If you love animation, 'Spirited Away' is a beautiful masterpiece.",
            "For a fun family movie, 'Finding Nemo' is always a hit!",
            "If you're in the mood for a documentary, '13th' is eye-opening.",
            "How about 'Mad Max: Fury Road' for some high-octane action?",
            "If you want something quirky, 'Eternal Sunshine of the Spotless Mind' is a great choice.",
            "For a gripping drama, 'The Social Network' is fascinating.",
            "If you enjoy musicals, 'La La Land' is a delightful watch.",
            "For a feel-good movie, 'The Intouchables' is heartwarming.",
            "If you want a good mystery, 'Knives Out' is clever and entertaining."
        ],
        "recommend a tv show": [
            "You should check out 'Breaking Bad.' It's considered one of the best shows ever made.",
            "If you enjoy sci-fi, 'Stranger Things' is a fantastic choice.",
            "For something light-hearted, 'Ted Lasso' is a heartwarming and funny series.",
            "You might like 'The Crown' if you're into historical dramas.",
            "'The Office' is a classic workplace comedy that never gets old.",
            "If you're looking for a gripping drama, 'The Queen's Gambit' is a must-watch.",
            "'Parks and Recreation' is a hilarious take on local government.",
            "If you want something unique, 'Fleabag' is both funny and poignant.",
            "For a thrilling ride, 'Mindhunter' dives into the psychology of serial killers.",
            "If you love fantasy, 'Game of Thrones' is epic (despite the ending).",
            "For a great mystery, 'Sherlock' is clever and engaging.",
            "If you're into true crime, 'Making a Murderer' is a gripping watch.",
            "For a fun animated series, 'Bob's Burgers' is a delight.",
            "If you want a heartwarming story, 'This Is Us' is beautifully crafted.",
            "For a unique perspective, 'The Mandalorian' is a must for Star Wars fans.",
            "If you enjoy historical fiction, 'The Last Kingdom' is captivating.",
            "For a classic, 'Friends' is always a good choice for laughs.",
```

```
            "If you want something thought-provoking, 'Black Mirror' is a great anthology series.",
            "For a feel-good show, 'Schitt's Creek' is hilarious and heartwarming.",
            "If you enjoy competition, 'The Great British Bake Off' is delightful."
        ],
        "career advice": [
            "It's important to find a career that aligns with your passions. What are you interested in?",
            "Networking can open many doors. Don't hesitate to reach out to people in your field.",
            "Consider what skills you want to develop. Continuous learning is key.",
            "Sometimes a change can lead to great opportunities. Keep an open mind.",
            "Work-life balance is crucial. Make sure to prioritize your well-being.",
            "If you're unsure about your path, consider seeking a mentor for guidance.",
            "Don't be afraid to explore different fields until you find the right fit.",
            "Setting clear goals can help you stay focused on your career path.",
            "If you're feeling stuck, consider taking a course to learn something new.",
            "Remember, every experience is a stepping stone to your next opportunity."
        ],
        "financial advice": [
            "Building an emergency fund is a great first step. Aim for at least three to six months' worth of expenses.",
            "Budgeting can help you manage your finances better. Have you tried using a budgeting app?",
            "Investing early can really pay off in the long run. Even small amounts can grow significantly.",
            "It's important to set long-term financial goals. What do you want to achieve?",
            "Consider speaking with a financial advisor for personalized advice.",
            "Tracking your expenses can help you identify areas to save.",
            "Don't forget to review your financial plan regularly to stay on track.",
            "If you're looking to invest, consider starting with index funds.",
            "Paying off high-interest debt should be a priority. It can save you money in the long run.",
            "Remember, financial literacy is key to making informed decisions."
        ],
        "how to learn": [
            "Find a subject that excites you. Passion makes learning easier.",
            "The Pomodoro technique can help you stay focused while studying.",
            "Teaching others is a great way to reinforce your own learning.",
            "Breaking topics into smaller chunks can make them more manageable.",
            "Effective note-taking can enhance your understanding of the material.",
            "Online courses can provide structured learning opportunities.",
            "Don't hesitate to ask for help when you're stuck on something.",
            "Experiment with different learning styles to find what works best for you.",
            "Set specific goals for your learning journey to stay motivated.",
```

```
            "Remember, practice makes perfect! Keep at it!"
        ],
        "self improvement": [
            "Focus on small, consistent changes. They often lead to the best results.",
            "Setting SMART goals can help you track your progress effectively.",
            "Consider what habits you want to develop and start with one at a time.",
            "Reflecting on your progress can be motivating. Celebrate your achievements!",
            "Find inspiration in books, podcasts, or people you admire.",
            "Journaling can help you clarify your thoughts and goals.",
            "Remember, self-improvement is a journey, not a destination.",
            "Surround yourself with positive influences. They can uplift and inspire you.",
            "Don't be afraid to step out of your comfort zone. Growth happens there!",
            "Seek feedback from others. It can provide valuable insights for improvement."
        ],
        "healthy habits": [
            "Regular exercise is essential for both physical and mental health. Find an activity you
enjoy!",
            "Eating a balanced diet can boost your energy levels. Incorporate more fruits and
veggies!",
            "Staying hydrated is key! Aim for at least eight glasses of water a day.",
            "Prioritize sleep! A good night's rest can improve your mood and productivity.",
            "Consider mindfulness practices like meditation to reduce stress.",
            "Tracking your habits can help you stay accountable. Have you tried a habit tracker?",
            "Make time for self-care. It's important to recharge and relax.",
            "Find a workout buddy! Exercising with a friend can make it more fun.",
            "Set realistic goals for your health journey. Small steps lead to big changes.",
            "Remember, it's okay to indulge occasionally. Balance is key!"
        ],
        "mental health": [
            "Taking care of your mental health is just as important as physical health. Don't
neglect it!",
            "Talking to someone can be incredibly helpful. Have you considered therapy?",
            "Engaging in hobbies you love can boost your mood. What do you enjoy doing?",
            "Setting boundaries is crucial for your well-being. Protect your energy!",
            "Practicing gratitude can shift your perspective. Try writing down what you're thankful
for.",
            "Don't hesitate to reach out for support when you need it. You're not alone.",
            "Mindfulness and meditation can help you stay grounded. Have you tried it?",
            "Physical activity can also improve your mental health. Get moving!",
            "Remember, it's okay to take a break. Rest is essential for mental clarity.",
            "Find joy in the little things. They can make a big difference in your day!"
        ],
        "making friends": [
            "Joining clubs or groups can help you meet new people. What are your interests?",
```

```
            "Volunteering is a great way to connect with others while giving back.",
            "Don't be afraid to start conversations! A simple 'hello' can lead to new friendships.",
            "Consider attending local events or meetups. They can be fun and social!",
            "Online communities can also be a great way to make friends. Have you explored
any?",
            "Be open and approachable. A friendly smile can go a long way!",
            "Find common interests to bond over. Shared hobbies can spark connections.",
            "Don't rush the process. Building friendships takes time and effort.",
            "Be yourself! Authenticity attracts genuine connections.",
            "Remember, it's okay to step out of your comfort zone. New experiences can lead to
new friends!"
        ],
        "relationship advice": [
            "Communication is key in any relationship. Be open and honest with each other.",
            "Make time for each other. Quality time strengthens your bond.",
            "Expressing appreciation can go a long way. Let your partner know you value them.",
            "Don't shy away from discussing difficult topics. It's important to address issues
together.",
            "Consider setting boundaries to ensure both partners feel respected.",
            "Surprise each other with little gestures. They can keep the spark alive!",
            "Be willing to compromise. Relationships require give and take.",
            "Remember to have fun together! Shared experiences create lasting memories.",
            "Seek to understand each other's perspectives. Empathy is crucial.",
            "If challenges arise, consider seeking couples therapy for guidance."
        ],
        "cooking tips": [
            "Start with simple recipes to build your confidence in the kitchen.",
            "Good knife skills can make cooking easier and safer. Practice makes perfect!",
            "Meal prepping can save you time during the week. Consider batch cooking!",
            "Experimenting with spices can elevate your dishes. Don't be afraid to get creative!",
            "Always taste as you go! Adjusting flavors can make a big difference.",
            "Invest in quality kitchen tools. They can make cooking more enjoyable.",
            "Don't rush the process. Cooking is an art that takes time to master.",
            "Try to use seasonal ingredients for the best flavor and freshness.",
            "Consider watching cooking shows for inspiration and new techniques.",
            "Remember, it's okay to make mistakes! Every cook has had their share of kitchen
disasters."
        ],
        "time management": [
            "Prioritize your tasks to focus on what matters most. What's your top priority today?",
            "Time-blocking can help you allocate specific times for tasks. Have you tried it?",
            "Set deadlines for yourself to stay on track. It can create a sense of urgency.",
            "Consider using productivity apps to help you stay organized.",
            "Break tasks into smaller, manageable steps. It makes them less overwhelming.",
```

```
            "Eliminate distractions while working. Find a quiet space to focus.",
            "Schedule breaks to recharge. A little downtime can boost your productivity
significantly.",
            "Review your progress regularly to adjust your plans as needed.",
            "Don't forget to celebrate your accomplishments, no matter how small!",
            "Remember, it's okay to say no to tasks that don't align with your goals."
        ],
        "tech help": [
            "What tech issue are you facing? I'm here to help you troubleshoot!",
            "Have you tried turning it off and on again? It's the classic tech fix!",
            "Let's tackle this problem step by step together!",
            "Updating your software can fix many issues! It's like giving your device a makeover!",
            "Check online guides or forums for specific problems. The internet is full of solutions!",
            "If you're having trouble with an app, try reinstalling it. Sometimes that does the trick!",
            "Make sure your device is compatible with the software you're trying to use.",
            "If you're dealing with a slow computer, consider clearing out unnecessary files.",
            "Backing up your data regularly can save you a lot of headaches later!",
            "If you need help with coding, there are plenty of resources available online!"
        ],
        "learn coding": [
            "What programming language are you curious about? Let's dive in!",
            "Online tutorials are a great way to start learning coding!",
            "Practice projects are essential for improving your skills! It's like leveling up!",
            "Consider joining a coding community for support and fun!",
            "Start with the basics and gradually build your knowledge!",
            "Don't hesitate to ask questions when you're stuck. The coding community is very
helpful!",
            "Try to work on real-world projects to apply what you've learned.",
            "Participating in coding challenges can sharpen your skills and boost your
confidence.",
            "Remember, everyone learns at their own pace. Be patient with yourself!",
            "If you enjoy learning visually, consider watching coding videos on platforms like
YouTube."
        ],
        "travel tips": [
            "What destination are you dreaming of visiting next?",
            "Creating a travel budget can help you manage expenses like a pro!",
            "Researching local customs can enhance your experience! It's like being a travel
ninja!",
            "Packing efficiently can save you time and hassle! Roll your clothes for more space!",
            "Traveling off-peak can save you money; consider it for your next adventure!",
            "Always have a backup plan in case things don't go as expected.",
            "Consider using travel apps to help you navigate and find the best spots.",
            "Don't forget to try local cuisine! It's one of the best parts of traveling.",
```

```
            "Keep your important documents safe and make copies just in case.",
            "Engage with locals to get insider tips on hidden gems!"
        ],
        "fitness advice": [
            "Set realistic fitness goals to keep yourself motivated!",
            "Find a workout routine that you enjoy! It's like a fun game for your body!",
            "Incorporating strength training can be beneficial for overall health!",
            "Staying active throughout the day is important; try to move regularly!",
            "Consider working with a personal trainer for guidance and support!",
            "Mix up your workouts to keep things interesting and prevent boredom.",
            "Don't forget to warm up before exercising and cool down afterward.",
            "Listen to your body! Rest days are just as important as workout days.",
            "Stay hydrated during your workouts to keep your energy levels up.",
            "Remember, consistency is key! Small efforts add up over time."
        ],
        "parenting tips": [
            "Establishing a routine can provide stability for your child!",
            "Positive reinforcement is effective in encouraging good behavior!",
            "Open communication is key to understanding your child's needs!",
            "Consider joining a parenting group for support and advice!",
            "Engaging in activities together can strengthen your bond!",
            "Be patient and flexible. Parenting is a learning experience for everyone.",
            "Encourage your child to express their feelings. It's important for their emotional
growth.",
            "Set boundaries to help your child understand expectations.",
            "Make time for family fun! It creates lasting memories.",
            "Don't hesitate to seek help when you need it. Parenting can be challenging!"
        ],
        "pet care": [
            "What type of pet do you have? Each one has unique needs!",
            "Establishing a feeding and exercise routine is important!",
            "Regular vet check-ups are essential for your pet's health!",
            "Training your pet can improve behavior and strengthen your bond!",
            "Make sure your pet has a safe and comfortable environment!",
            "Consider adopting from a shelter if you're looking for a new pet.",
            "Grooming is important for your pet's hygiene and comfort.",
            "Provide mental stimulation through toys and activities to keep them happy.",
            "Always have fresh water available for your pet.",
            "Remember, pets thrive on love and attention. Spend quality time with them!"
        ],
        "home improvement": [
            "What home improvement project are you considering?",
            "Have you set a budget for your renovations? It's important to plan!",
            "DIY projects can be fun and rewarding! What do you want to tackle?",
```

```
            "Choosing the right materials can make a big difference!",
            "Don't forget to enjoy the process of improving your space!",
            "Consider starting with small projects to build your confidence.",
            "Researching design ideas can inspire your renovations!",
            "Make a list of priorities to focus on what matters most.",
            "Involve family members in the projects for a fun bonding experience!",
            "Remember to take safety precautions while working on home improvements."
        ],
        "gardening tips": [
            "What type of garden are you interested in starting?",
            "Have you researched plants that thrive in your climate?",
            "Companion planting can be beneficial! What plants do you want to grow?",
            "What challenges do you face in gardening? I'm here to help!",
            "Have you considered growing herbs for cooking? They're easy to maintain!",
            "Start with easy-to-grow plants to build your confidence.",
            "Regularly check for pests to keep your garden healthy.",
            "Consider using organic fertilizers for a more sustainable approach.",
            "Make sure your plants get enough sunlight and water.",
            "Gardening can be therapeutic! Enjoy the process of nurturing your plants."
        ],
        "study tips": [
            "What subject are you studying? Let's find effective strategies!",
            "Have you tried creating a study schedule? It can help you stay organized!",
            "Active recall and spaced repetition can enhance retention!",
            "Would you like tips on creating a distraction-free study environment?",
            "How do you stay motivated while studying? Let's brainstorm some fun techniques!",
            "Consider forming a study group for collaborative learning.",
            "Use visual aids like charts and diagrams to reinforce concepts.",
            "Take regular breaks to recharge your brain.",
            "Practice past exams to familiarize yourself with the format.",
            "Remember, it's okay to ask for help when you're struggling!"
        ],
        "best games to play": [
            "Looking for some epic gaming recommendations? How about 'The Legend of Zelda: Breath of the Wild'? It's a magical adventure!",
            "If you're into strategy, 'Civilization VI' will have you conquering the world in no time!",
            "For a fun party game, 'Among Us' is a blast! Who doesn't love a little friendly deception?",
            "If you want to get lost in a story, 'The Witcher 3: Wild Hunt' is like diving into a fantasy novel!",
            "For some nostalgic fun, 'Mario Kart' is always a good time! Ready to race?",
            "If you enjoy puzzles, 'Portal 2' is a mind-bending experience!",
            "For a thrilling experience, 'Dark Souls' will test your skills!",
            "If you love simulation games, 'Stardew Valley' is a relaxing farming adventure!",
```

```
            "For a cooperative experience, 'Overcooked!' is chaotic fun with friends!",
            "If you're into RPGs, 'Final Fantasy VII Remake' is a must-play!"
        ],
        "fun facts": [
            "Did you know honey never spoils? Archaeologists have found pots of honey in ancient Egyptian tombs that are over 3000 years old!",
            "Octopuses have three hearts! Two pump blood to the gills, while one pumps it to the rest of the body!",
            "Bananas are berries, but strawberries aren't! Mind-blowing, right?",
            "A group of flamingos is called a 'flamboyance.' How fabulous is that?",
            "Wombat poop is cube-shaped! Nature sure has its quirks!",
            "Did you know that a day on Venus is longer than a year on Venus? Talk about a slow day!",
            "A jiffy is an actual unit of time! It’s about 1/100th of a second!",
            "The inventor of the frisbee was turned into a frisbee after he died. Talk about a flying legacy!",
            "Cows have best friends and get stressed when they are separated. How sweet is that?",
            "A snail can sleep for three years! Talk about a long nap!",
            "The world's largest desert is Antarctica! It's not just about sand and sun!",
            "A single strand of spaghetti is called a 'spaghetto.' How cute is that?",
            "The heart of a blue whale is so big that a human can swim through its arteries!",
            "Did you know that butterflies taste with their feet? That's one way to find a good meal!",
            "A group of crows is called a 'murder.' Sounds a bit dramatic, doesn't it?",
            "The Eiffel Tower can be 15 cm taller during the summer due to thermal expansion of the metal!",
            "Did you know that honeybees can recognize human faces? They're smarter than we think!",
            "Octopuses have three hearts! Two pump blood to the gills, while one pumps it to the rest of the body!",
            "Bananas are berries, but strawberries aren't! Mind-blowing, right?"
        ],
        "would you rather": [
            "Would you rather have the ability to fly or be invisible? Tough choice, right?",
            "Would you rather live in a world without music or a world without movies? What would you miss more?",
            "Would you rather have a pet dragon or a pet unicorn? Talk about magical companions!",
            "Would you rather time travel to the past or the future? Where would you go?",
            "Would you rather always have to sing instead of speaking or dance everywhere you go? Let's get groovy!",
            "Would you rather have unlimited pizza for life or unlimited tacos for life? Tough decision!",
```

```
            "Would you rather be able to talk to animals or speak every human language? What a superpower!",
            "Would you rather live in a treehouse or a houseboat? Both sound like fun!",
            "Would you rather have the power of super strength or super speed? Which would you choose?",
            "Would you rather explore space or the deep sea? Both are full of mysteries!"
        ]
    }

    for pattern, responses in training_data.items():
        if pattern not in self.knowledge["patterns"]:
            self.knowledge["patterns"][pattern] = responses

    self.save_knowledge()
    print("Auto-training completed with expanded knowledge base!")

def preprocess_text(self, text):
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))
    return text

def tokenize(self, text):
    return word_tokenize(text)

def find_best_match(self, user_input):
    preprocessed_input = self.preprocess_text(user_input)
    tokens = self.tokenize(preprocessed_input)

    best_match = None
    highest_similarity = 0

    for pattern in self.knowledge["patterns"]:
        pattern_tokens = self.tokenize(pattern)
        common_tokens = set(tokens) & set(pattern_tokens)
        similarity = len(common_tokens) / max(len(tokens), len(pattern_tokens))

        if similarity > highest_similarity:
            highest_similarity = similarity
            best_match = pattern

    return best_match if highest_similarity > 0.5 else None

def generate_response(self, user_input):
    best_match = self.find_best_match(user_input)
```

```
        if best_match:
            responses = self.knowledge["patterns"][best_match]
            # Choose a random response but avoid repeating the last one if possible
            if len(responses) > 1 and len(self.conversation_history) > 0:
                last_response = self.conversation_history[-1] if self.conversation_history else None
                available_responses = [r for r in responses if r != last_response]
                response = random.choice(available_responses if available_responses else
responses)
            else:
                response = random.choice(responses)

            self.conversation_history.append(response)
            return response
        else:
            # Provide a default response and ask for teaching
            default_response = "I'm still learning about that. Could you teach me how to respond to
that?"
            self.conversation_history.append(default_response)
            return default_response

    def learn_from_response(self, user_input, correct_response):
        preprocessed_input = self.preprocess_text(user_input)
        if preprocessed_input not in self.knowledge["patterns"]:
            self.knowledge["patterns"][preprocessed_input] = []
        self.knowledge["patterns"][preprocessed_input].append(correct_response)
        self.save_knowledge()
        print(f"Thank you! I've learned a new response for '{user_input}'")

def main():
    chatbot = SimpleChatbot()
    print("Chatbot initialized! Type 'quit' to exit.")

    while True:
        user_input = input("You: ").strip()

        if user_input.lower() == 'quit':
            print("Goodbye!")
            break

        response = chatbot.generate_response(user_input)
        print("Bot:", response)

        if "I'm still learning" in response:
```

```
            print("Would you like to teach me a response? (yes/no)")
            if input().lower() == 'yes':
                print("What should I respond with?")
                correct_response = input()
                chatbot.learn_from_response(user_input, correct_response)

if __name__ == "__main__":
    main()
```

Robot_interaction.py

```
import os
import cv2
import pickle
import subprocess
import time
from gtts import gTTS
import face_recognition
import torch
from PIL import Image
from torchvision import models, transforms
import numpy as np
import speech_recognition as sr  # Import the SpeechRecognition library

# === Load Emotion Detection Model ===
emotion_model = models.resnet50(weights="ResNet50_Weights.DEFAULT")
emotion_model.eval()

# === Paths ===
emotion_images_dir = 'emotion_images'
LLAMA_PATH = "/home/surya/llama.cpp/build/bin/llama-cli"
LLAMA_MODEL = "/home/surya/llama-2-7b-chat.ggmlv3.q4_0.bin"

# === Functions ===
def speak(text):
    """Function to speak text using Google TTS"""
    print(f"Robot: {text}")
    try:
        tts = gTTS(text=text, lang='en')
        tts.save("/home/surya/speech.mp3")
        os.system("mpg321 /home/surya/speech.mp3")
    except Exception as e:
        print(f"Error in TTS: {e}")
```

```
def capture_image():
    """Capture an image using libcamera"""
    filename = '/home/surya/captured_image.jpg'
    try:
        subprocess.run(['libcamera-still', '--output', filename, '--width', '640', '--height', '480'],
check=True)
        return filename
    except Exception as e:
        print(f"Error capturing image: {e}")
        return None

def recognize_face(known_face_encoding):
    """Recognize the face using face_recognition"""
    image_path = capture_image()
    if not image_path:
        speak("I don’t see anyone in my sight.")
        return False, None  # No image captured

    frame = cv2.imread(image_path)
    face_locations = face_recognition.face_locations(frame)
    if not face_locations:
        speak("I don’t see anyone in my sight.")
        return False, None

    face_encodings = face_recognition.face_encodings(frame, face_locations)
    for face_encoding in face_encodings:
        matches = face_recognition.compare_faces([known_face_encoding], face_encoding)
        if np.any(matches):
            return True, "happy"  # Placeholder for detected expression

    speak("Sorry, you are not my admin. Type 'quit' to exit.")
    return False, None

def detect_expression():
    """Detect emotion from the captured image."""
    image_path = capture_image()
    if not image_path:
        return "tired"  # Default to 'tired' if no image is captured

    # Dynamically load emotion labels from subdirectories
    emotion_labels = sorted(os.listdir(emotion_images_dir))

    img = Image.open(image_path).convert("RGB")
    preprocess = transforms.Compose([
```

```
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
    img_tensor = preprocess(img).unsqueeze(0)

    with torch.no_grad():
        output = emotion_model(img_tensor)
        _, predicted_class = torch.max(output, 1)

    # Ensure index is within the bounds of `emotion_labels`
    if predicted_class.item() < len(emotion_labels):
        return emotion_labels[predicted_class.item()]
    else:
        return "tired"  # Default to 'tired' if index is out of range

def chat_with_llama(user_input):
    """Chat function using LLaMA"""
    try:
        result = subprocess.run(
            [LLAMA_PATH, "-m", LLAMA_MODEL, "-p", user_input, "-n", "200", "-e"],
            capture_output=True,
            text=True,
        )
        response = result.stdout.strip().split("\n")[-1]
        return response
    except Exception as e:
        print(f"Error communicating with LLaMA: {e}")
        return "I'm sorry, I couldn't process your request."

def listen_to_command():
    """Listen for a voice command and return the recognized text."""
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        audio = recognizer.listen(source)

    try:
        command = recognizer.recognize_google(audio)
        print(f"You said: {command}")
        return command.lower()
    except sr.UnknownValueError:
        speak("Sorry, I didn't catch that. Please try again.")
```

```
        return None
    except sr.RequestError as e:
        speak(" There was an error with the speech recognition service. Please check your internet
connection.")
        return None

# === Main Logic ===
if __name__ == "__main__":
    print("Initializing all systems...")
    speak("Initializing all systems. Please wait.")

    try:
        with open("face_encodings.pkl", "rb") as f:
            known_face_encoding = pickle.load(f)
            print("Face encoding loaded successfully.")  # Debugging line
    except FileNotFoundError:
        speak("Face encodings file not found. Exiting.")
        exit(1)

    speak("Ready to scan face. Please make sure you are visible to the camera.")
    running = True

    while True:
        if running:
            is_admin, expression = recognize_face(known_face_encoding)
            if not is_admin:
                continue

            # Greet admin
            speak("Hello Surya, my admin. I am your personal robot.")
            detected_emotion = detect_expression()  # Detect emotion
            speak(f"You seem {detected_emotion} today!")

            while running:
                user_input = listen_to_command()  # Use voice input instead of typing

                if user_input is None:
                    continue  # If no command was recognized, continue listening

                if user_input in ["exit", "bye", "quit talking"]:
                    speak("Shutting down all activities. Awaiting further instructions.")
                    running = False
                    break
```

```
        elif user_input == "start up":
            speak("Re-initializing all systems.")
            speak("Ready to scan face. Please make sure you are visible to the camera.")
            break

        elif "what does this look like" in user_input:
            speak("Let me check...")
            detected_object = detect_expression()  # Placeholder for object detection
            speak(f"This looks like {detected_object}")

        elif "time" in user_input:
            current_time = time.strftime("%H:%M:%S")
            speak(f"The current time is {current_time}")

        else:
            response = chat_with_llama(user_input)
            speak(response)
```

Updates robot interaction code

```
import os
import cv2
import pickle
import subprocess
import time
import json
import random
import string
import nltk
from nltk.tokenize import word_tokenize
from collections import defaultdict
import torch
from PIL import Image
from torchvision import models, transforms
import numpy as np
import speech_recognition as sr  # Import the SpeechRecognition library
from gtts import gTTS
```

```
# === Load Emotion Detection Model ===
emotion_model = models.resnet50(weights="ResNet50_Weights.DEFAULT")
emotion_model.eval()

# === Paths ===
emotion_images_dir = 'emotion_images'

# === Chatbot Class ===
class SimpleChatbot:
    def __init__(self, knowledge_file="chatbot_knowledge.json"):
        self.knowledge_file = knowledge_file
        self.knowledge = self.load_knowledge()
        self.conversation_history = []

        try:
            nltk.data.find('tokenizers/punkt')
        except LookupError:
            print("Downloading required NLTK data...")
            nltk.download('punkt')

        self.auto_train()

    def load_knowledge(self):
        try:
            with open(self.knowledge_file, 'r') as f:
                return json.load(f)
        except FileNotFoundError:
            return {"responses": {}, "patterns": {}}

    def save_knowledge(self):
        with open(self.knowledge_file, 'w') as f:
            json.dump(self.knowledge, f, indent=2)

    def auto_train(self):
        training_data = {
            # ... (keep the training data as in the original chatbot code)
        }

        for pattern, responses in training_data.items():
            if pattern not in self.knowledge["patterns"]:
                self.knowledge["patterns"][pattern] = responses

        self.save_knowledge()
        print("Auto-training completed with expanded knowledge base!")
```

```
    def preprocess_text(self, text):
        text = text.lower()
        text = text.translate(str.maketrans("", "", string.punctuation))
        return text

    def tokenize(self, text):
        return word_tokenize(text)

    def find_best_match(self, user_input):
        preprocessed_input = self.preprocess_text(user_input)
        tokens = self.tokenize(preprocessed_input)

        best_match = None
        highest_similarity = 0

        for pattern in self.knowledge["patterns"]:
            pattern_tokens = self.tokenize(pattern)
            common_tokens = set(tokens) & set(pattern_tokens)
            similarity = len(common_tokens) / max(len(tokens), len(pattern_tokens))

            if similarity > highest_similarity:
                highest_similarity = similarity
                best_match = pattern

        return best_match if highest_similarity > 0.5 else None

    def generate_response(self, user_input):
        best_match = self.find_best_match(user_input)

        if best_match:
            responses = self.knowledge["patterns"][best_match]
            if len(responses) > 1 and len(self.conversation_history) > 0:
                last_response = self.conversation_history[-1] if self.conversation_history else None
                available_responses = [r for r in responses if r != last_response]
                response = random.choice(available_responses if available_responses else
responses)
            else:
                response = random.choice(responses)

            self.conversation_history.append(response)
            return response
        else:
```

```
            default_response = "I'm still learning about that. Could you teach me how to respond to
that?"
            self.conversation_history.append(default_response)
            return default_response

    def learn_from_response(self, user_input, correct_response):
        preprocessed_input = self.preprocess_text(user_input)
        if preprocessed_input not in self.knowledge["patterns"]:
            self.knowledge["patterns"][preprocessed_input] = []
        self.knowledge["patterns"][preprocessed_input].append(correct_response)
        self.save_knowledge()
        print(f"Thank you! I've learned a new response for '{user_input}'")

# === Functions ===
def speak(text):
    """Function to speak text using Google TTS"""
    print(f"Robot: {text}")
    try:
        tts = gTTS(text=text, lang='en')
        tts.save("/home/surya/speech.mp3")
        os.system("mpg321 /home/surya/speech.mp3")
    except Exception as e:
        print(f"Error in TTS: {e}")

def capture_image():
    """Capture an image using libcamera"""
    filename = '/home/surya/captured_image.jpg'
    try:
        subprocess.run(['libcamera-still', '--output', filename, '--width', '640', '--height', '480'],
check=True)
        return filename
    except Exception as e:
        print(f"Error capturing image: {e}")
        return None

def recognize_face(known_face_encoding):
    """Recognize the face using face_recognition"""
    image_path = capture_image()
    if not image_path:
        speak("I don’t see anyone in my sight.")
        return False, None  # No image captured

    frame = cv2.imread(image_path)
    face_locations = face_recognition.face_locations(frame)
```

```
    if not face_locations:
        speak("I don’t see anyone in my sight.")
        return False, None

    face_encodings = face_recognition.face_encodings(frame, face_locations)
    for face_encoding in face_encodings:
        matches = face_recognition.compare_faces([known_face_encoding], face_encoding)
        if np.any(matches):
            return True, "happy"  # Placeholder for detected expression

    speak("Sorry, you are not my admin. Type 'quit' to exit.")
    return False, None

def detect_expression():
    """Detect emotion from the captured image."""
    image_path = capture_image()
    if not image_path:
        return "tired"  # Default to 'tired' if no image is captured

    # Dynamically load emotion labels from subdirectories
    emotion_labels = sorted(os.listdir(emotion_images_dir))

    img = Image.open(image_path).convert("RGB")
    preprocess = transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
    img_tensor = preprocess(img).unsqueeze(0)

    with torch.no_grad():
        output = emotion_model(img_tensor)
        _, predicted_class = torch.max(output, 1)

    if predicted_class.item() < len(emotion_labels):
        return emotion_labels[predicted_class.item()]
    else:
        return "tired"  # Default to 'tired' if index is out of range

def listen_to_command():
    """Listen for a voice command and return the recognized text."""
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
```

```
        print("Listening...")
        audio = recognizer.listen(source)

    try:
        command = recognizer.recognize_google(audio)
        print(f"You said: {command}")
        return command.lower()
    except sr.UnknownValueError:
        speak("Sorry, I didn't catch that. Please try again.")
        return None
    except sr.RequestError as e:
        speak("There was an error with the speech recognition service. Please check your internet
connection.")
        return None

# === Main Logic ===
if __name__ == "__main__":
    print("Initializing all systems...")
    speak("Initializing all systems. Please wait.")

    try:
        with open("face_encodings.pkl", "rb") as f:
            known_face_encoding = pickle.load(f)
            print("Face encoding loaded successfully.")  # Debugging line
    except FileNotFoundError:
        speak("Face encodings file not found. Exiting.")
        exit(1)

    chatbot = SimpleChatbot()  # Initialize the chatbot
    speak("Ready to scan face. Please make sure you are visible to the camera.")
    running = True

    while True:
        if running:
            is_admin, expression = recognize_face(known_face_encoding)
            if not is_admin:
                continue

            # Greet admin
            speak("Hello Surya, my admin. I am your personal robot.")
            detected_emotion = detect_expression()  # Detect emotion
            speak(f"You seem {detected_emotion} today!")

            while running:
```

```
user_input = listen_to_command()  # Use voice input instead of typing

if user_input is None:
    continue  # If no command was recognized, continue listening

if user_input in ["exit", "bye", "quit talking"]:
    speak("Shutting down all activities. Awaiting further instructions.")
    running = False
    break

elif user_input == "start up":
    speak("Re-initializing all systems.")
    speak("Ready to scan face. Please make sure you are visible to the camera.")
    break

elif "what does this look like" in user_input:
    speak("Let me check...")
    detected_object = detect_expression()  # Placeholder for object detection
    speak(f"This looks like {detected_object}")

elif "time" in user_input:
    current_time = time.strftime("%H:%M:%S")
    speak(f"The current time is {current_time}")

else:
    response = chatbot.generate_response(user_input)  # Use chatbot response
    speak(response)
    if "I'm still learning" in response:
        speak("Would you like to teach me a response? (yes/no)")
        if listen_to_command() == "yes":
            speak("What should I respond with?")
            correct_response = listen_to_command()
            if correct_response:
                chatbot.learn_from_response(user_input, correct_response)
```