https://github.com/shenihachris/Chattbot/invitationsExplanation of complete steps that will be taken to put design of phase1 into transformation.

1. Setting Up Development Environment

Python installed on our system from the official Python website (https://www.python.org/downloads/) ,also a code editor like Visual Studio Code

2. Project Structure

directory structure:

```
chatbot/
│    main.py
│    user_input.py
│    response_generation.py
│    conversation_manager.py
│    config.py
│    data/
│        responses.json
```

main.py: The entry point of chatbot.

user_input.py: Handle user input processing.

response_generation.py: Generate responses.

conversation_manager.py: Manage the flow of the conversation.

config.py: Store configuration settings.

data/responses.json: A JSON file to store predefined responses.

3. Implement the Modules

a. user_input.py

Implement a function to process user input. Using NLP libraries like spaCy or NLTK for more advanced processing.

Extract user intents or keywords from the input.

b. response_generation.py

Create a function to generate responses based on user input. You can use predefined responses stored in responses.json.

Use context from the conversation manager to generate context-aware responses.

c. conversation_manager.py

Implement a class to manage the conversation flow. This class should keep track of the conversation history, user context, and any other relevant information.

Determine when to switch topics or context based on user input and the current conversation state.

d. config.py

Define configuration settings, such as API keys, default responses, or other parameters that your chatbot might use.

4. User Interaction Loop

In main.py, create the main loop for user interaction:

```
from user_input import
process_user_input
from response_generation import generate_response
from conversation_manager import ConversationManager

def main():
    # Initialize conversation manager
    conversation = ConversationManager()

    while True:
        user_message = input("User: ")
        if user_message.lower() == 'exit':
            break

        # Process user input
        processed_input = process_user_input(user_message)

        # Generate a response
        bot_response = generate_response(processed_input, conversation)

        # Display the bot's response
        print(f"Bot: {bot_response}")

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

This loop takes user input, processes it, generates a response, and displays it. The conversation manager helps maintain context.

5. Predefined Responses

Load predefined responses from responses.json in the response_generation.py module. You can use the json module in Python to read and parse the JSON file.

6. Testing and Debugging

Thoroughly testing chatbot by simulating various user inputs to ensure it handles different scenarios correctly. Debug any issues that arise during testing.

7. Deployment

chatbot is ready, you can deploy it on various platforms, such as a website, a messaging app, or a voice assistant, depending on your project's requirements.

Dataset:
https://storage.googleapis.com/kagglesdsdata/datasets/715041/1245709/dialogs.txt?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20231006%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20231006T062635Z&X-Goog-Expires=259200&X-Goog-SignedHeaders=host&X-Goog-Signature=820bdd32c5613d322e4709061cbe6fdb53ec09852ef8320f9cdca891258b0810fd50dcc68ae26413045b6712d1d0a6a0d7f8e597a4a714c1c6d29f0ba27932c2ae3b55b6e258cde6596d444c6b185bc3d2d4e954bb6f361ad2ccf1a57f7dcd4e31462f001f7dab455acaadd8b1fdd69c1138fa744452a035fef9c62c82e62d0a99ffe19c051672d8f41eca4d8e0962294a60c37e1f6e2a83a80ffe4b536c0cbedc85972dbfd7195b69d58135e06a1cc2eaabb45b3763ba5b2e2bcff1238feedeaaecb65255ae188f047f9bc8e3d1422ac98fb5f38ef29d8440064afbc518bada59d2478047a3a1033dfebbb788f817efd4574b2f8389003e465d12af4a58e428