1)

A basic machine learning voice assistant consists of four distinct neural network algorithms.

1. **Wake Word Detection**

This confirms that only wake words such as "hey", "hello", "yo", "hi", etc. will wake up our voice assistant. Noises such as dog barks, wind hushing, keyboard clacking, other background chit-chat won't wake up our voice assistant or interrupt us in the middle.

2. **Speech Recognition**

This neural network algorithm will accept the incoming input in audio format and help the backend store and understand the voice query and transcribes to text.

3. **Natual Language Processing**

This AI algorithm maps transcribed text to entities and then those entities is mapped to response.

4. **Speech Synthesis**

This AI algorithm will generate sound waves and enable the voice assistant to reply back to us.

Apart from the above four we will be needing a co-ordinator module to integrate all the algorithms into one place. This will be called the core module.

We have to create the entitites which can be linked to the responses.

Our data is audio which is naturally occuring time-sequence, so using RNN will be best as it was invented for sequential data processing. For using huge amount of data I use Mozilla Common voice datasets.

So, there are five main components which are necessary to build a voice assistant:

- Voice interface - a frontend which users use to communicate with the assistant (web or mobile app, smart speaker, etc)
- Speech-to-text (STT) - a voice processing component which takes user input in an audio format and produces a text representation of it
- NLU - a component which takes user input in text format and extract structured data (intents and entities) which helps and assistant to understand what the user wants
- Dialogue management - a component which determines how an assistant should respond at specific state of the conversation and generates that response in a text format

- Text-to-speech (TTS) - a component which takes the response of the assistant in a text format and produces a voice representation of it which is then sent back to the user

If I could build this on top of Raspberry pi, it could give more privacy to the users unlike the MNCs. All this can be integrated on Raspberry pi.

While open source Rasa is a rather obvious choice for NLU and dialogue management, deciding on STT and TTS is a more difficult task simply because there aren't that many open source frameworks to choose from. After exploring the currently available options: CMUSphinx, Mozilla DeepSpeech, Mozilla TTS, Kaldi, I decided to go with Mozilla tools - Mozilla DeepSpeech and Mozilla TTS or else we can simply use SpeechRecongnition, pyttsx3, pyaudio.

Here is why:

Mozilla tools come with a set or pre-trained models, but you can also train your own using custom data. This allows you to implement things quickly, but also gives you all the freedom to build custom components.
In comparison to alternatives, Mozilla tools seem to be the most OS agnostic.
Both tools are written in Python which makes it slightly easier to integrate with Rasa.
It has a big and active open source community ready to help out with technical questions.

What is Mozilla DeepSpeech and Mozilla TTS?

Mozilla DeepSpeech is a speech-to-text framework which takes user input in an audio format and uses machine learning to convert it into a text format which later can be processed by NLU and dialogue system. Mozilla TTS takes care of the opposite - it takes the input (in our case - the response of the assistant produced by a dialogue system) in a text format and uses machine learning to create an audio representation of it.

NLU, dialogue management and voice processing components cover the backend of the voice assistant so what about the frontend?

Well, this is where the biggest problem lies - if you search for the open source voice interface widgets, you will end up finding none and that's why I developed my own Rasa voice interface which I used for this project.

So, to summarise, here are the four components of the open source voice assistant:

- Rasa
- Mozilla DeepSpeech
- Mozilla TTS
- Rasa Voice Interface

2)

```
- intent: greet
  examples: |
    - hey
    - hello
    - hi
    - hello there
    - good morning
    - good evening
    - moin
    - hey there
    - let's go
    - hey dude
    - goodmorning
    - goodevening
    - good afternoon

- intent: goodbye
  examples: |
    - good afternoon
    - cu
    - good by
    - cee you later
    - good night
    - bye
    - goodbye
    - have a nice day
    - see you around
    - bye bye
    - see you later

- intent: affirm
  examples: |
    - yes
    - y
    - indeed
    - of course
    - that sounds good
    - correct

- intent: deny
  examples: |
```

```
    - no
    - n
    - never
    - I don't think so
    - don't like that
    - no way
    - not really

- intent: mood_great
  examples: |
    - perfect
    - great
    - amazing
    - feeling like a king
    - wonderful
    - I am feeling very good
    - I am great
    - I am amazing
    - I am going to save the world
    - super stoked
    - extremely good
    - so so perfect
    - so good
    - so perfect

- intent: mood_unhappy
  examples: |
    - my day was horrible
    - I am sad
    - I don't feel very well
    - I am disappointed
    - super sad
    - I'm so sad
    - sad
    - very sad
    - unhappy
    - not good
    - not very good
    - extremly sad
    - so saad
    - so sad

- intent: bot_challenge
  examples: |
```

```
    - are you a bot?
    - are you a human?
    - am I talking to a bot?
    - am I talking to a human?

- intent: enquire
  examples: |
    - What's [X](company_name)?
    - What is so different about [X](company_name)?
    - What does [X](company_name) do?
    - What's [X]'s(company_name) ratings in globally?
    - What can [X](company_name) do?
    - Tell me about [X](company_name)
    - Tell me about your [company](company_name)
    - What's [X]'s(company_name) [products](product)?
    - What is so different about [X]'s(company_name) [products](product)?
    - What does [X](company_name) support in?
    - What [X]'s(company_name) [products](product) are offered globally?
    - What does [X](company_name) offer?
    - Tell me about [X]'s(company_name) [products](product)
    - Tell me about your [company]'s(company_name) [products](product)?
    - What's the [price range](price) of [medicines](product)?
    - What's the [price range](price) of [garage parts](product)?
    - What's the [price range](price) of [paint](product)?
    - Show me the [prices](price) of [medicines](product)
    - Show me the [prices](price) of [garage parts](product)
    - Show me the [prices](price) of [paint](product)
    - What's the [cost](price) of the [paracetamol](product)?
    - What is so different about [garage parts](product) sold by [X](company_name)?
    - How much does this [garage part](product) [costs](price)?
    - What's the [rate](price) of [paint](product) bought from [X](company_name)?
    - How much does [paint](product) [costs](price)?
    - How much does [medicines](product) [cost](price)?
    - How much does [garage parts](product) [cost](price)?
    - Tell me about [medicines'](product) [price](price)
    - Tell me about your [medicines'](product) [price](price)?
    - Tell me about [garage parts'](product) [price](price)
    - Tell me about your [garage parts'](product) [price](price)?
    - Tell me about [paint's](product) [price](price)
    - Tell me about your [paint's](product) [price](price)?

- intent: navigate
  examples: |
    - What's on your [website](company_name)?
```

```
    - Can you [recommend](recommender) some [products](product) for me?
    - Should I buy a [paracetamol](product) 250 mg or paracetamol 500 mg?
    - What's on your website?
    - Show me around
    - What do you got here?

- intent: inform
  examples: |
    - I am [Nabanita Dash](name)
    - My name is [Nabanita Dash](name)
    - My email address is [dashnabanita@gmail.com](email)
    - Email me to [dashnabanita@gmail.com](email)
    - Send an email to [b117030@iiit-bh.ac.in](email)
    - [dashnabanita@gmail.com](email)
    - Create a [sales ticket](sales_ticket) for me
    - Will you please create a [sales ticket](sales_ticket) on my behalf?
```

3)

Entities for voice assitant:

X - company_name

Company - company_name

Products - product

Price - price

Medicines - product

Garage Parts - product

Paint - product

Website - company_name

Recommend(s) - recommender

Cost - price

Navigate - navigate

Sales Ticket - sales_ticket

4)

- A good voice interface not only follows a specific command, but also considers what comes next.

- Designing a conversation is not just about creating nice prompts, it is also important for the assistant to know when it should shut up. Designers should be wary about over-informing the user.

- Users will remember commands and features of your assistant if they can discover features on their own rather than the features being pushed to them. When a user says “smile”, “1…2…3” or “say cheese!” on their android phones, the native camera clicks a picture.

- After a user has given a command, the assistant should confirm the input.

- Forcing the user to press the button every time they are going to talk can be cumbersome and unnatural. When talking to a real person, you don’t have to indicate every time you want to speak. You can use natural turn taking techniques like asking a question, pausing, waiting for a response and in some cases explicit direction. The easiest technique is to ask a question, users would naturally respond.

- Another turn-taking violation that systems need to keep in mind is putting an instruction after the question. “Would you like to save this? Say yes, no or repeat.”

- Conversational markers include:

Timelines (“First”, “At last”)
Acknowledgements (Thanks”, “Got it”)
Positive feedback (“Good job”, “Awesome”)
Assurance and generic replies (“Sorry to hear that”, “Oh”)

- No speech is detected
If a response is expected but is not detected, the system can handle it in two ways. Either call it out explicitly (“I’m sorry. I didn’t hear anything. Can you repeat that?”) or do nothing.

- Speech is detected but nothing is recognized
In the case where a response is detected but the automated speech recognition (ASR) tool could not recognize what was said, you can employ strategies similar to when no speech was detected. The system can call out explicitly (“Sorry, I didn’t get that”) or do nothing.

- Disambiguation
  For the system, there can be multiple correct responses for a given query. When a user says “What’s the weather in Rampur?” There are multiple places called Rampur. The system can ask “Did you mean the one in Uttar Pradesh or Rajasthan?” This method however is brute force and should be used only when no contextual information is available.
- Latency
  It is important to determine whether your system will have any latency or delays in answering queries. Latencies are often caused by poor connectivity, system processing or deeper database lookup. It can say things like “one moment please while I look this up”

- A user should be able to know where he is in the flow by asking for help anytime in the conversation. For example, when a user says ‘help’ when ordering pizza from the assistant, the assistant can prompt by telling the user a list go things he can do (Try saying ‘what have I ordered?’, ‘order another item’, ‘start the order again’ or ‘cancel order’).

- Apart from help, the system should support universal commands such as ‘next’, ‘previous’, ‘go back’ or ‘good bye.’ Universals can be decided based on the assistant’s use case and the most common queries.

Users would engage with the system more if they realize that it is aware of their presence. We need certain strategies for the assistant to maintain an illusion of awareness. Harry Gottlieb wrote “The Jack Principles of the Interactive Conversation Interface” in 2002. In this paper, Gottlieb outlines tips for creating the illusion of awareness in a conversational system; specifically, he suggests responding with human intelligence and emotion to the following:

- The user’s actions

- The user’s inactions

- The user’s past actions

- A series of the user’s actions

- The actual time and space that the user is in (time is obvious, place can refer to geographical space, which app
  the user is in or a place in the house like kitchen, living room, etc.)

- The comparison of different users’ situations and actions

Gottlieb also outlined tips for maintaining the illusion of awareness:

- Use dialog that conveys a sense of intimacy

- Ensure that characters act appropriately while the user is interacting

- Ensure that dialog never seems to repeat

- Be aware of the number of simultaneous users

- Be aware of the gender of the users

- Ensure that the performance of the dialog is seamless

- Avoid the presence of characters when user input cannot be evaluated

Designing conversations on Sketch, Excel and Lucidchart can be tedious. Botsociety is the no-code tool that designers are looking for. The tools that can be used are https://botsociety.io/ for creating conversations and path flows. Design simple or complex interactions, preview the final user experience with one click, test with real users without having to build anything, and gather feedback. Then, export in video format, or directly to Dialogflow or Rasa.

5)

```
stories:
- story: happy path 1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_great
  - action: utter_happy

- story: happy path 2
  steps:
  - intent: greet
  - action: utter_greet
  - action: utter_inform
  - intent: inform
  - intent: navigate
  - action: utter_navigate
  - intent: mood_unhappy
  - action: utter_navigate
  - intent: mood_great
  - action: utter_happy

- story: happy path 3
  steps:
  - intent: greet
```

```
  - action: utter_greet
  - action: utter_inform
  - intent: inform
  - intent: enquire
  - action: utter_enquire
  - intent: mood_unhappy
  - action: utter_enquire
  - intent: navigate
  - action: utter_navigate
  - action: utter_did_that_help
  - intent: deny
  - action: utter_enquire
  - intent: navigate
  - action: utter_navigate
  - action: utter_did_that_help
  - intent: affirm
  - intent: mood_great
  - action: utter_happy

- story: sad path 1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_unhappy
  - action: utter_cheer_up
  - action: utter_did_that_help
  - intent: affirm
  - action: utter_happy

- story: sad path 2
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_unhappy
  - action: utter_cheer_up
  - action: utter_did_that_help
  - intent: deny
  - action: utter_goodbye

- story: sad path 3
  steps:
  - intent: greet
  - action: utter_greet
  - action: utter_inform
```

```
  - intent: inform
  - intent: navigate
  - action: utter_navigate
  - intent: mood_unhappy
  - action: utter_navigate
  - intent: mood_unhappy
  - action: utter_navigate
  - intent: mood_unhappy
  - action: utter_cheer_up
  - action: utter_did_that_help
  - intent: deny
  - action: utter_goodbye

- story: sad path 4
  steps:
  - intent: greet
  - action: utter_greet
  - action: utter_inform
  - intent: inform
  - intent: enquire
  - action: utter_enquire
  - intent: mood_unhappy
  - action: utter_enquire
  - intent: navigate
  - action: utter_navigate
  - action: utter_did_that_help
  - intent: deny
  - action: utter_enquire
  - intent: navigate
  - action: utter_navigate
  - action: utter_did_that_help
  - intent: deny
  - intent: navigate
  - action: utter_did_that_help
  - intent: mood_unhappy
  - action: utter_enquire
  - action: utter_did_that_help
  - intent: deny
  - action: utter_cheer_up
  - intent: mood_unhappy
  - action: utter_goodbye
```

6) submitted zip file

7)

When the assistant is ready and gets all the happy path covered, its the best time to start training in with data from real users. Rasa-X develops pre-existing Rasa open source models by interacting with real users, getting their feedback, recording their experience. This can be deployed using Docker Compose. If we are getting started then using Docker Compose is the most viable option.

This can be deployed using Cluster Environment such as Kubernetes or OpenShift. Deploying rasa-x on scalable architecture such as Kubernetes or OpenShift. Overtime container orchestration systems like Kubernetes have become the standard for deploying software in a way that is scalable, repeatable and reliable. Rasa has a helm chart required for training in cluster environment. Connect to local server by using kubectl and google cloud sdk.