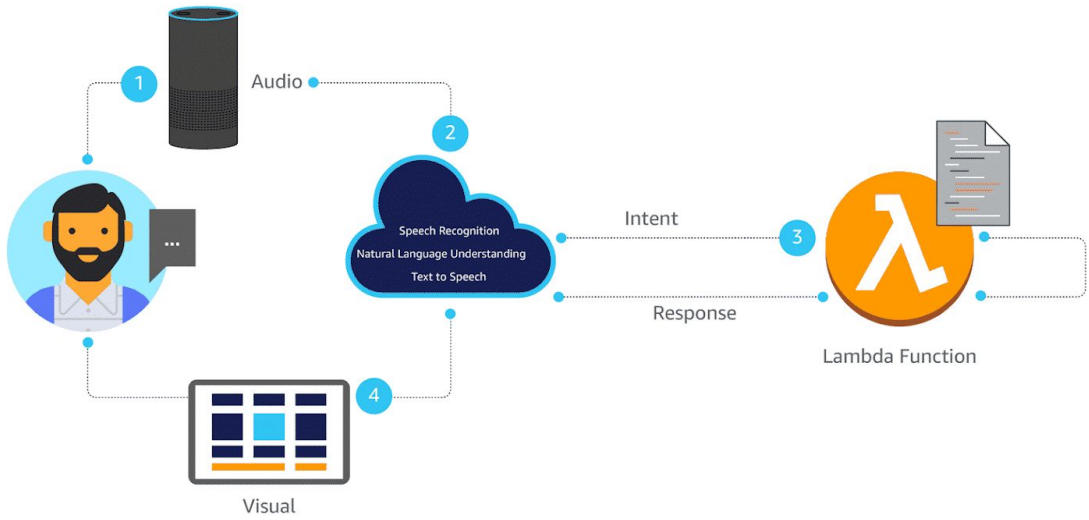
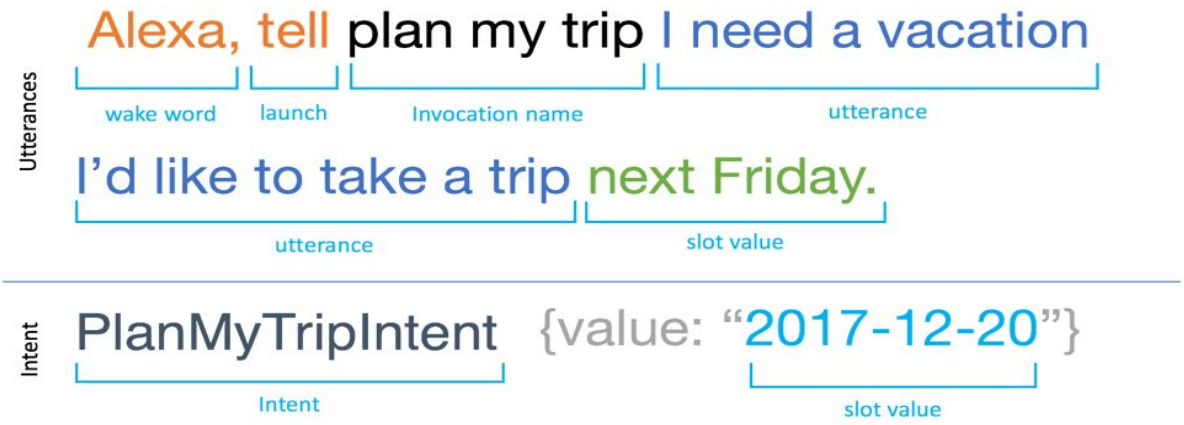
The Work Flow:



The developer should have the following roles:

- Define the request that the skill handles
- Define the name Alexa use to identify the skill
- Code to execute the request

Design VUI for Alexa:
Understand The Utterances and intents
Uses SSML (Speech Synthesis Markup Language) to process things like year 2014



Wake word: Tells Alexa to start listen
Launch Word: The skill's invocation name is to follow
Invocation Name: Skill Name
Utterance: Many ways in which user can form their requests (spoken requests)
Slots: Specific inputs given to user spoken requests : time, when, where, what (things that can directly be used by lambda function)
Intent: Actions that user wants to perform (intents can have slots as a compulsory or optional argument)

How to identify slots from utterance:

Utterance	Maps to
"I am going on a trip Friday."	TRAVEL_DATE
"I want to visit Portland."	TO_CITY
"I want to travel from Seattle to Portland next Friday."	FROM_CITY, TO_CITY, and TRAVEL_DATE
"I'm driving to Portland to go hiking."	MODE_OF_TRAVEL, TO_CITY, and ACTIVITIES

Tip: Optionally, if your skill is complex and has a lot of back-and forth-conversation (multi-turn conversation), create a dialog model for the skill. A dialog model is a structure that identifies the steps for a multi-turn conversation between your skill and the user to collect all the information needed to fulfill each intent. This simplifies the code you need to write to ask the user for information

Interaction model: Combination of intent, slots and utterances:
Mention all in json : Write down all the intents and utterances on paper (revise to cover most possible combinations of them)

Remember to use **sticky skills** whenever you have something that alexa should remember on calling the skill second time eg. remember your favourite color, name, etc

BUILDING THE CAKE WALK:

Backend:

➔ Create a skill:
- ◆ Creat custom skill
- ◆ Select lambda language
- ◆ Name the skill

WRITE A WELCOME NOTE

➔ Go to Code:
- ◆ Find handle() function in code
- ◆ It uses responseBuilder function to compose and return response to user
- ◆ Within the handle() function, find the line that begins const speakOutput. This variable contains the string of words the skill should say back to the user when they launch the skill

```
def handle(self, handler_input):
    # type: (HandlerInput) -> Response
    speak_output = "Hello! Welcome to cake walk. That was a piece of cake! Bye!"

    return (
      handler_input.response_builder
        .speak(speak_output)
        .ask(speak_output)
        .response
    )
```

Handler_input.response_builder handles the response to be sent where speak_output is the welcome note that alexa will say on invocation

➔ Save → Deploy → Test

MAINTAIN A DIALOGUE

- Ask the user a question
- Listen for the answer
- Respond to the user

➔ Add reprompt text to .ask(reprompt) attr. Remember giving example helps user understand the context

```
reprompt_text = "I was born on Nov. 6th, 2015. When are you born?"
```

Alexa can now listen and ask but can not respond, we need frontend for the same

---

Frontend:

GET INTENT AND SLOTS:

- ➔ Go to build: add intent, slots and utterances
- ➔ Go to dialog management (for slots being required or optional)
  - ◆ In slots, click on edit slot
  - ◆ Turn the toggle for required field on and type the prompt that alexa should show
  - ◆ Press + to add the prompt
- ➔ Save and build the model

Backend:

ENTER CAPTURE BIRTHDAY INTENT HANDLER:

- ➔ Write code in handler (refer cake walk skill)
- ➔ Export the class handler at the end of the code
- ➔ Save and deploy → test

At this point, alexa captures birthday, prompts for missing fields, responses
BUT doesn't remember the bday.

Now writing code for it to remember the birthday from S3 bucket

---

This SDK has something called as attribute manager: With the manager, your read/write code can remain the same, even if you change where you save your data later

```
session_attr = handler_input.attributes_manager.session_attributes
    session_attr['year'] = year
    session_attr['month'] = month
    session_attr['day'] = day
    # save session attributes as persistent attributes
    handler_input.attributes_manager.persistent_attributes = session_attr
    handler_input.attributes_manager.save_persistent_attributes()
```

This process is supposed toi be async to make it more efficient.

We need a new handler to read the data from s3 bucket

- Attribute Manager saves values to S3 (these attributes are persistant)

- **Default intent types:**

- **AMAZON.NoIntent, AMAZON.CancelIntent, AMAZON.StopIntent:**
    - Handle user request when user declines to interact with the skill

- AMAZON. FallbackIntent:
    - Handles when user speech doesn't match any of the utterances

- AMAZON.HelpIntent:
    - User speech requests for help