AWS re:Invent

ALX302-R2

Build an engaging Alexa skill with Cake Walk

Karthik Ragubathy
Solutions Architect, Alexa Skills
Amazon Web Services

Sohan Maheshwar
Alexa Evangelist, Alexa Skills
Amazon Web Services

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Agenda

Introduction

How it works

Designing a skill

Build a skill in five minutes

Collect information turn by turn

Add memory to Cake Walk

Use settings API

Introduction

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Companion Course

alexa.design/cakewalk

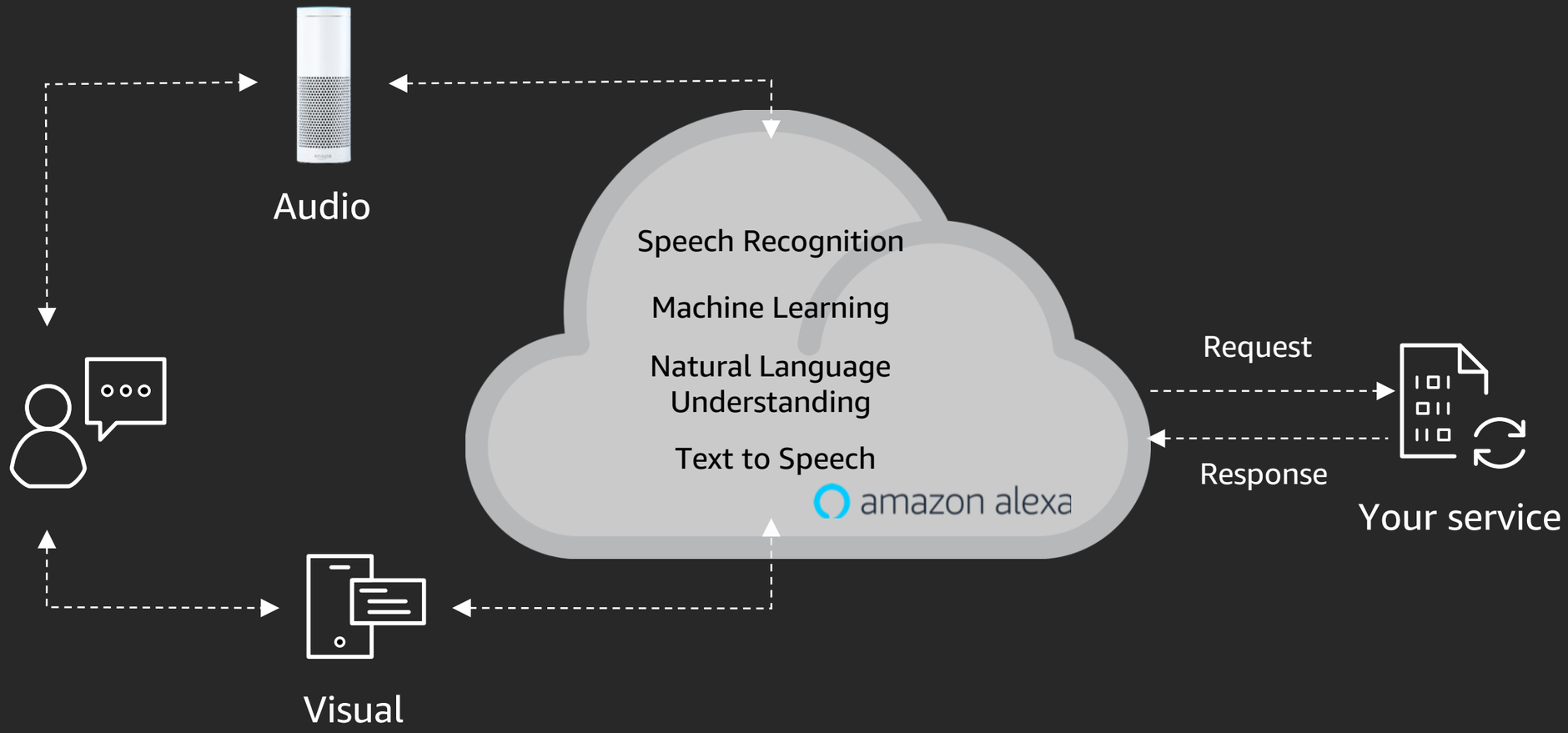
Cake Walk is a skill that celebrates your birthday! Tell it your birthday to have it count down the days. Interact with the skill on your special day to hear a happy birthday message.



How it works

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview


Audio
Speech Recognition
Machine Learning
Natural Language Understanding
Text to Speech
amazon alexa
Request
Response
Your service
Visual

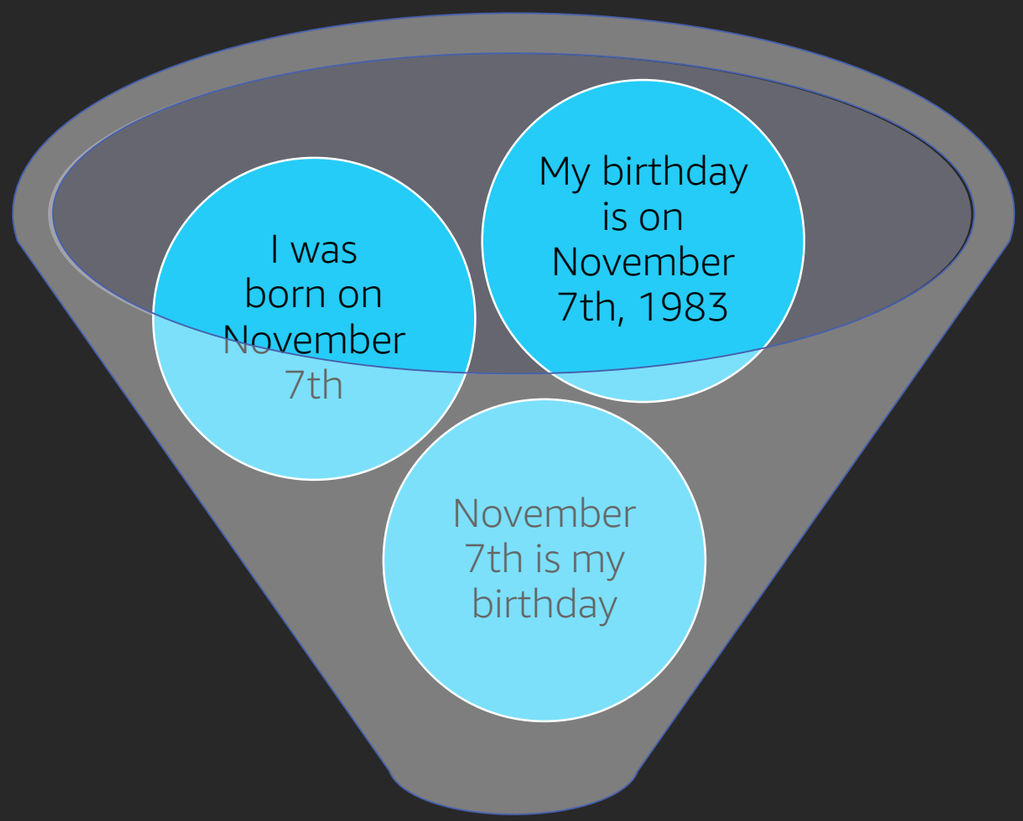
Intents, slots, utterances

First, a lesson in developer vocabulary

Parts of an utterance



Utterances and intents



Utterances

CaptureBirthdayIntent

Intent

Slots

Tell Cake Walk I was born on November 7th

Tell Cake Walk my birthday is the 7th

Tell Calk Walk I was born November 7th, 1983

Slots

My birthday is the 7th

My birthday is November

I was born November 7th

{day} {month} {year}

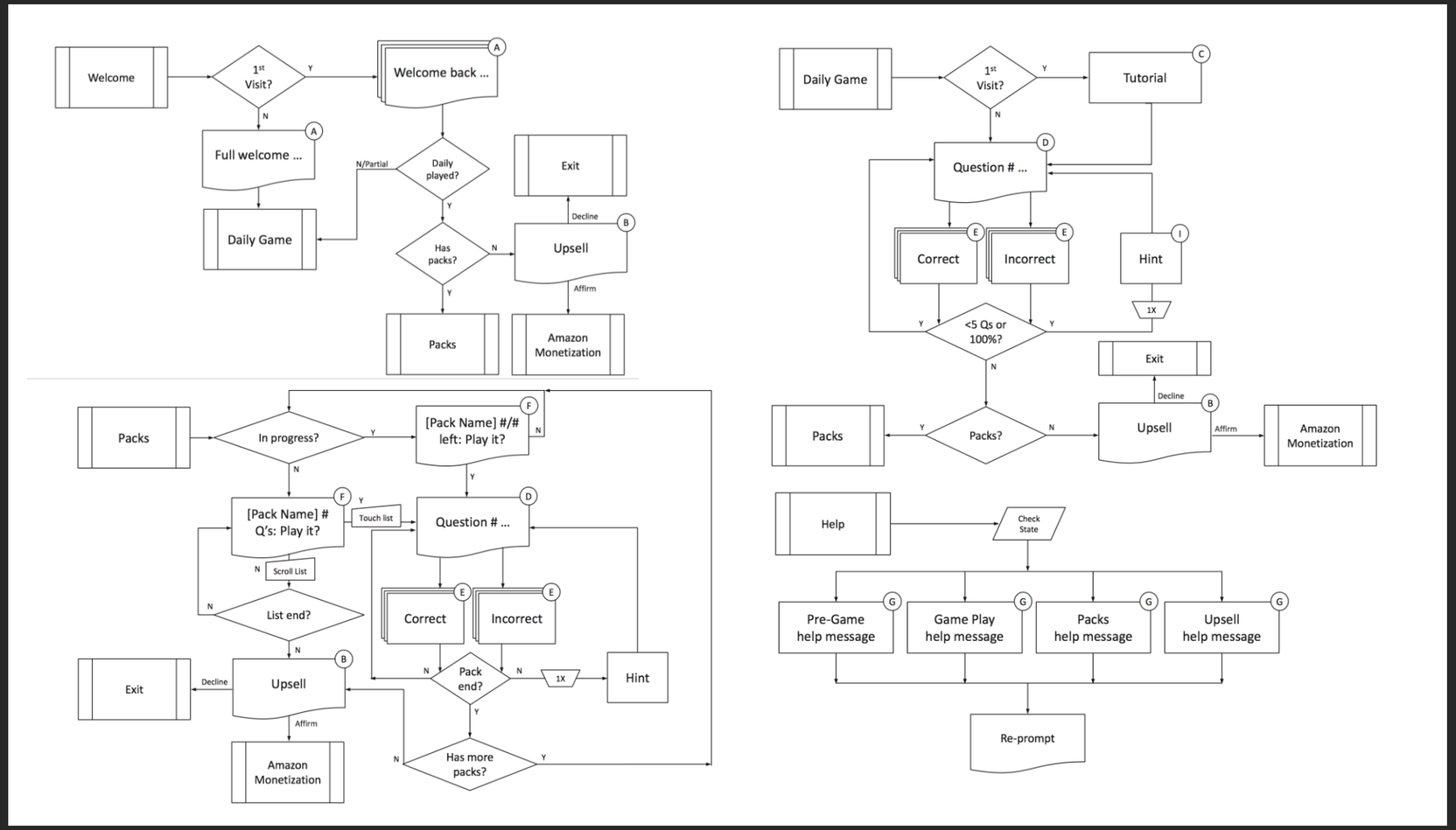
Slots

{slot}	Example values
day	7th, Seventh, Fifth
month	November, May
year	1983, 2001, 2015

Designing a skill

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Flowcharts aren’t conversational



Designing for conversation: Start with the script

BIRTHDAY UNKNOWN

USER
Alexa, open Cake Walk.

ALEXA
Hello! Welcome to Cake Walk.
What is your birthday?

USER
I was born November seventh
nineteen eighty-three.

ALEXA
Thanks, I'll remember that you
were born November seventh
nineteen eighty-three. Bye!

BIRTHDAY KNOWN

USER
Alexa, open Cake Walk.

ALEXA
Welcome back! There are 219 days
until your 36th birthday!

Who led the conversation? Alexa or the user?

The user leads the conversation

BIRTHDAY UNKNOWN

USER
Alexa, open Cake Walk.

ALEXA
Hello! Welcome to Cake Walk.
What is your birthday?

USER
I was born November seventh
nineteen eighty-three.

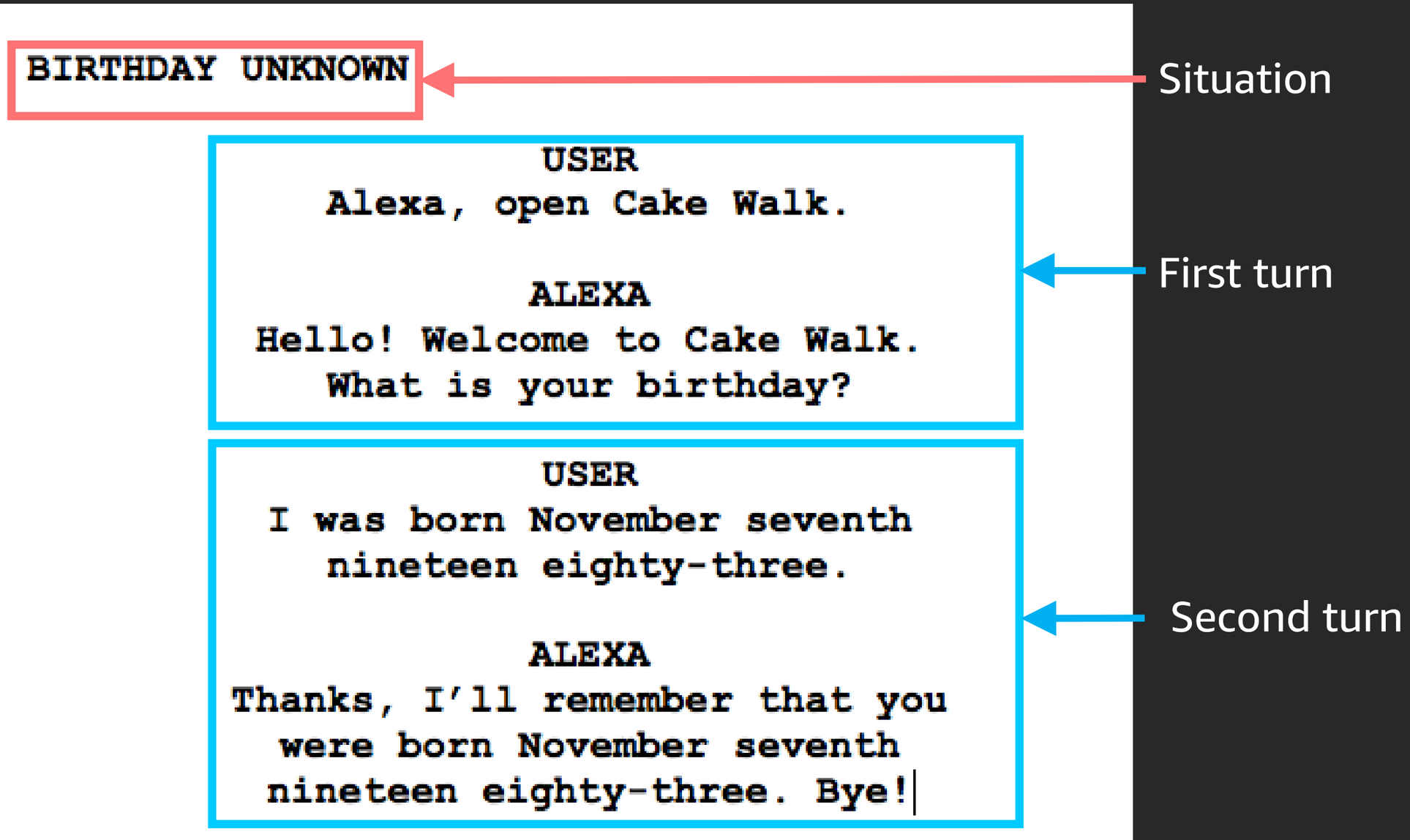
ALEXA
Thanks, I'll remember that you
were born November seventh
nineteen eighty-three. Bye!

BIRTHDAY KNOWN

USER
Alexa, open Cake Walk.

ALEXA
Welcome back! There are 219 days
until your 36th birthday!

Split your script into conversational turns



Represent each turn with a situational card

Utterance: What the user said

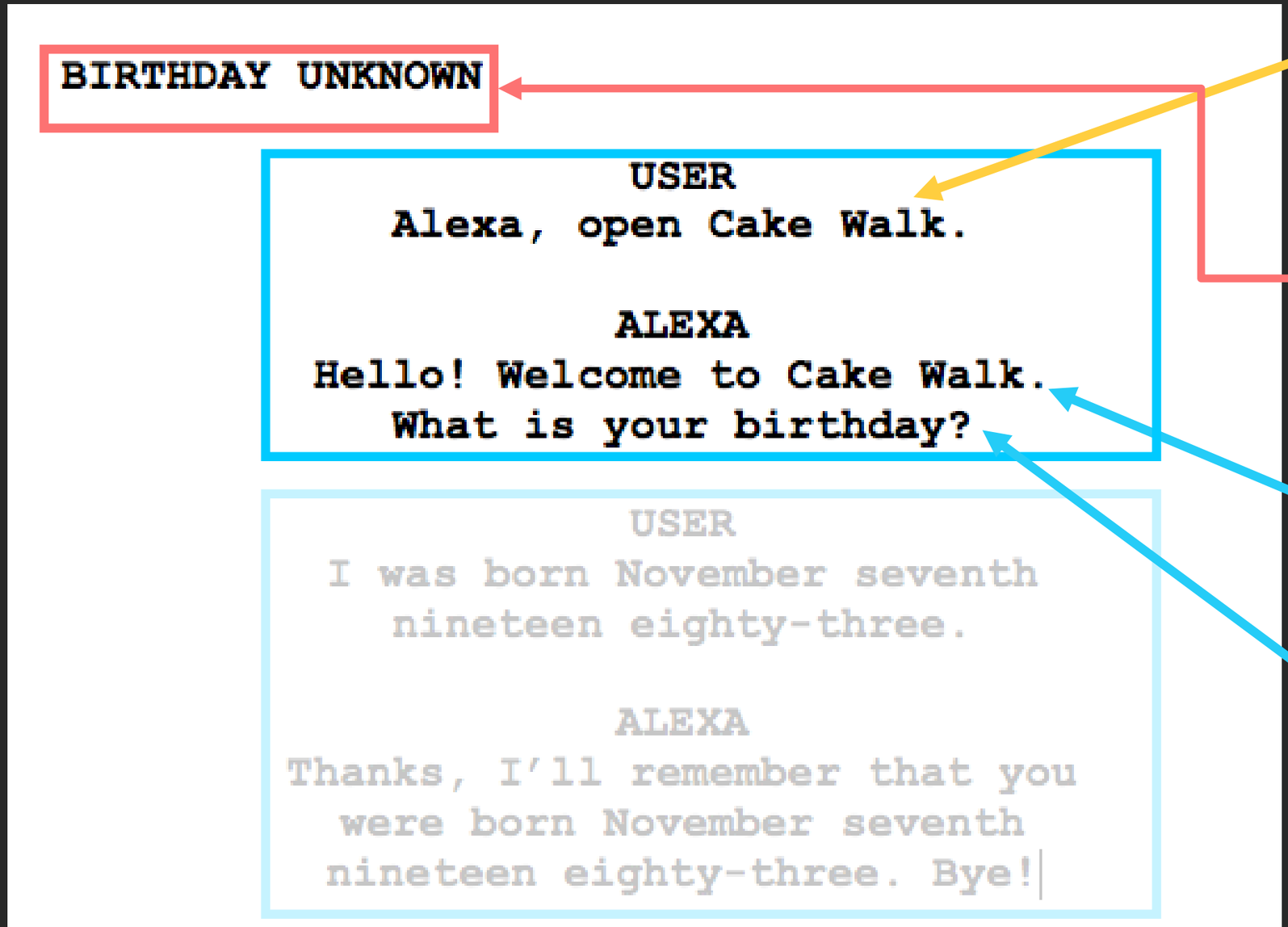
Situation: Context

Response: What Alexa says

Prompt: Question Alexa asks



Mapping scripts to cards – first turn



USER
Alexa, open Cake Walk

SITUATION

WHEN
1st time, birthday unknown

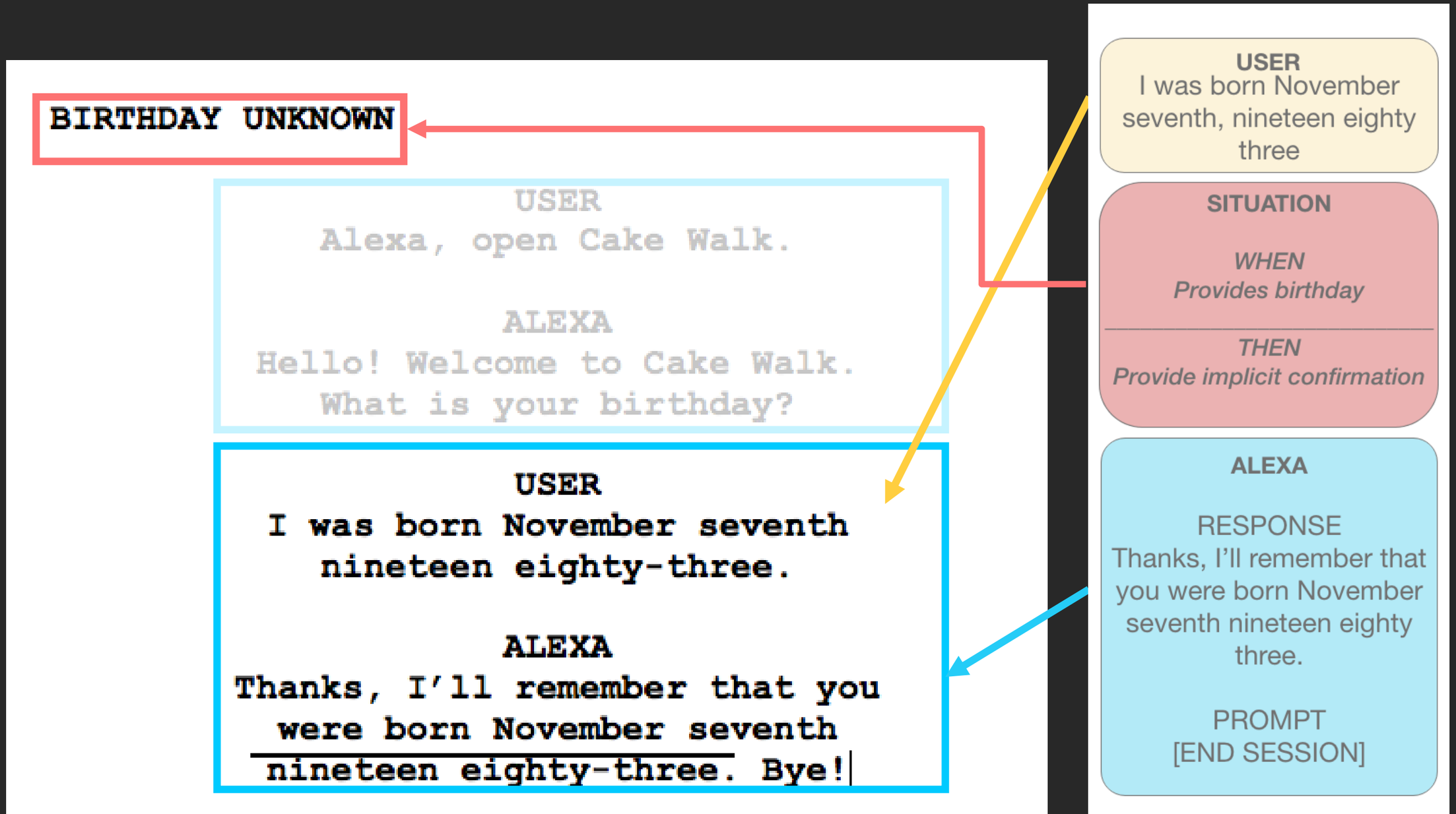
THEN
1st welcome, prompt for birthday

ALEXA

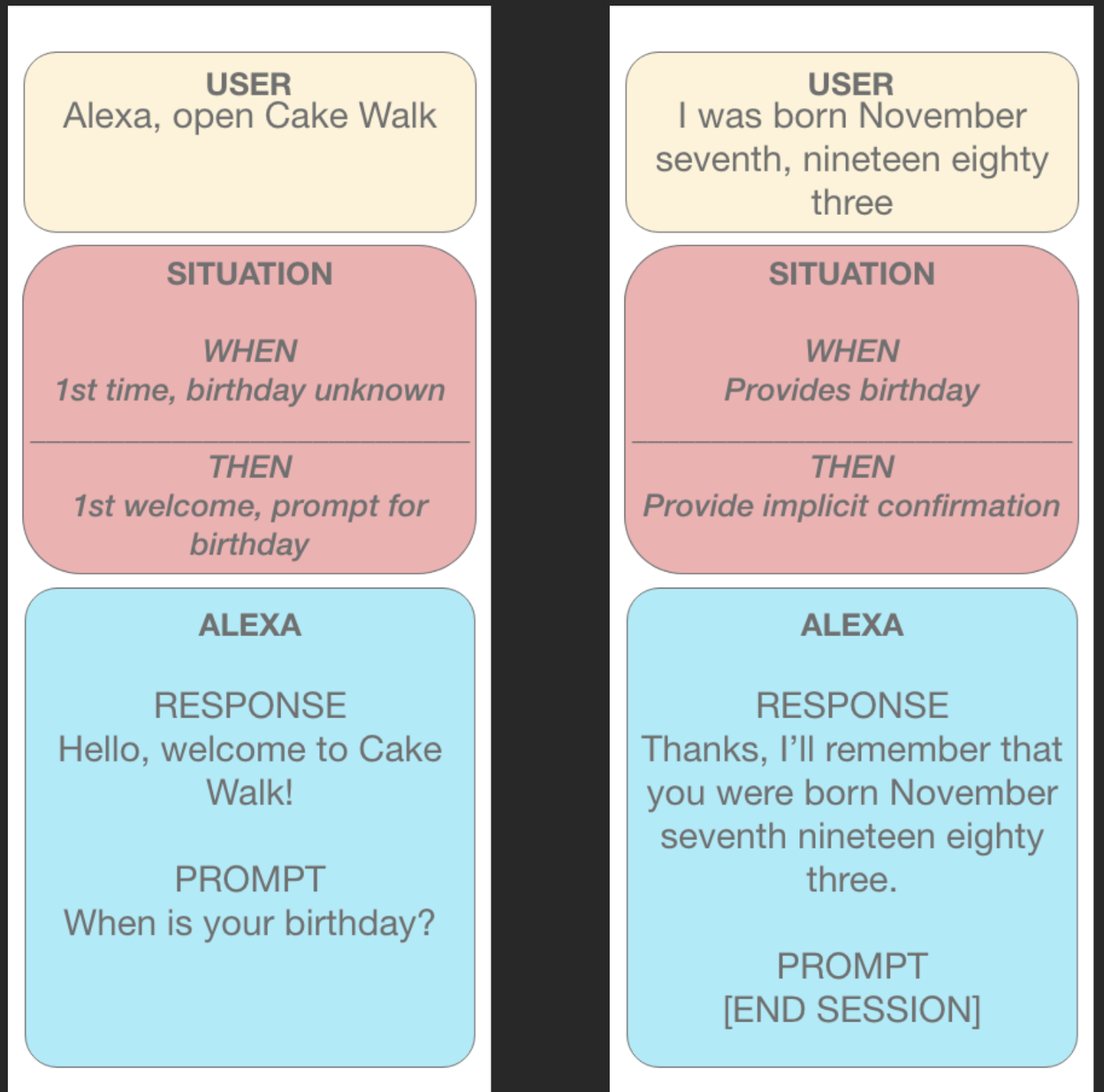
RESPONSE
Hello, welcome to Cake Walk!

PROMPT
When is your birthday?

Mapping scripts to cards – second turn

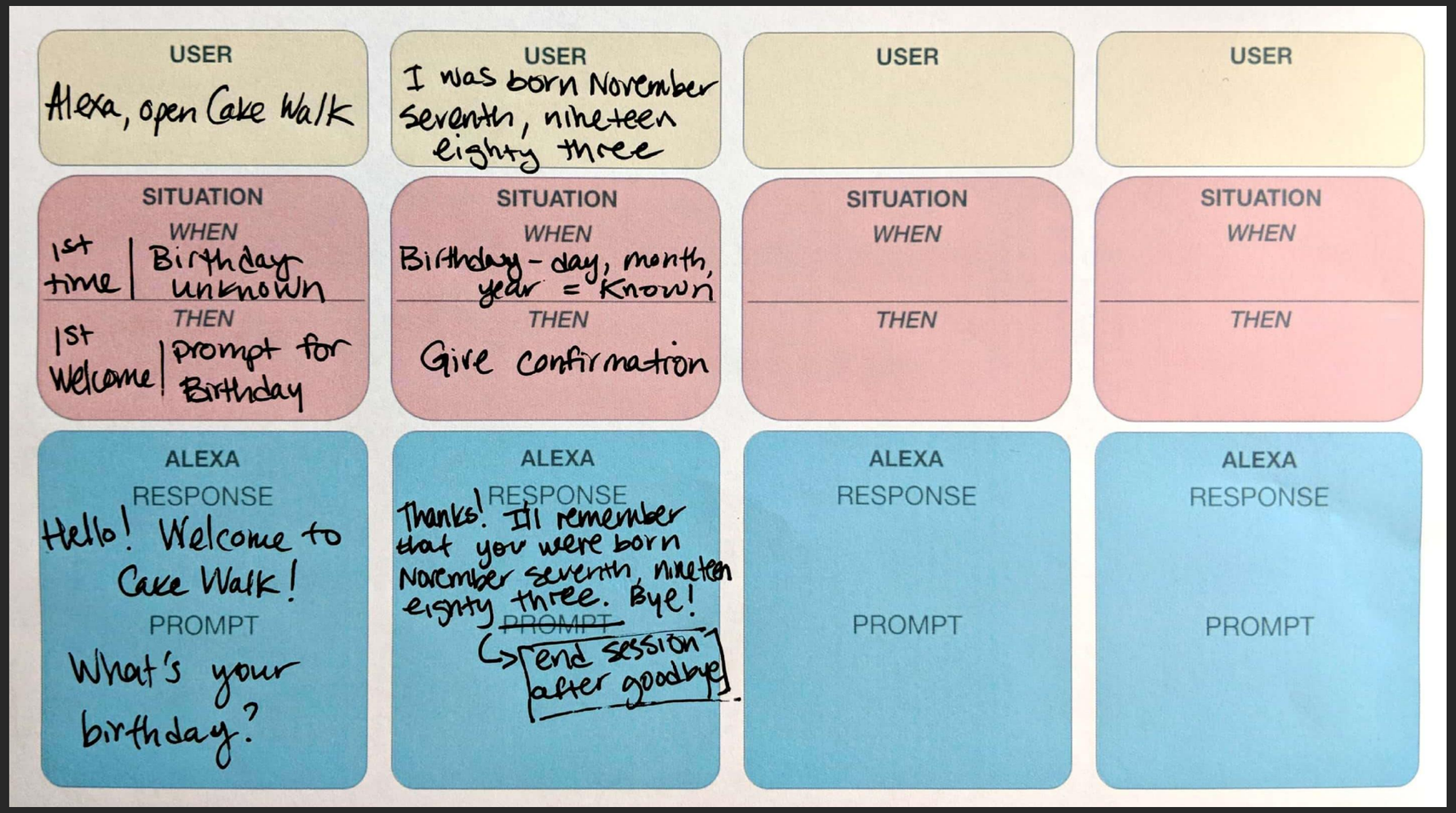


From script to storyboard



Activity: Create storyboards

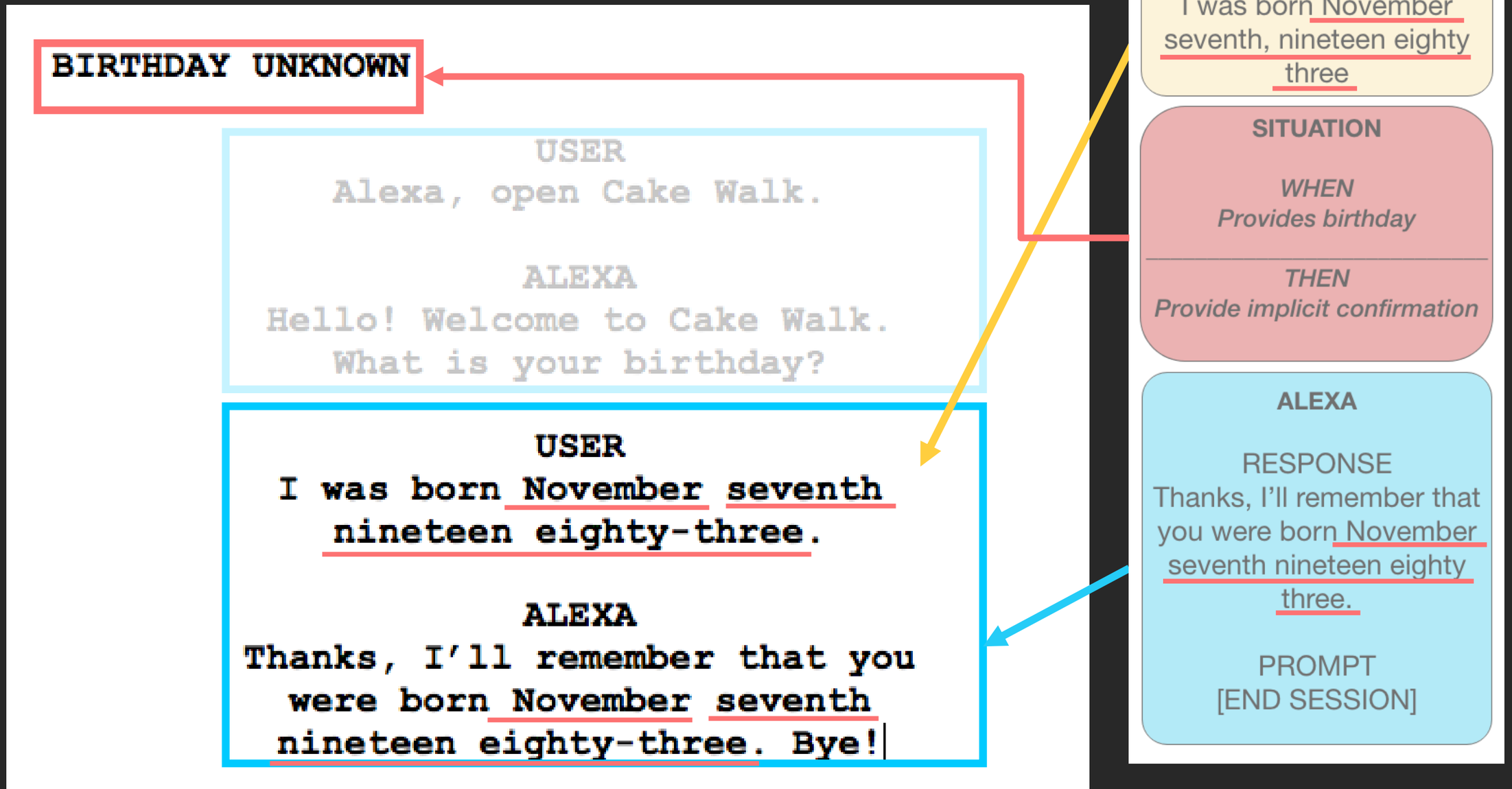
5 Min



Solution: Partial information

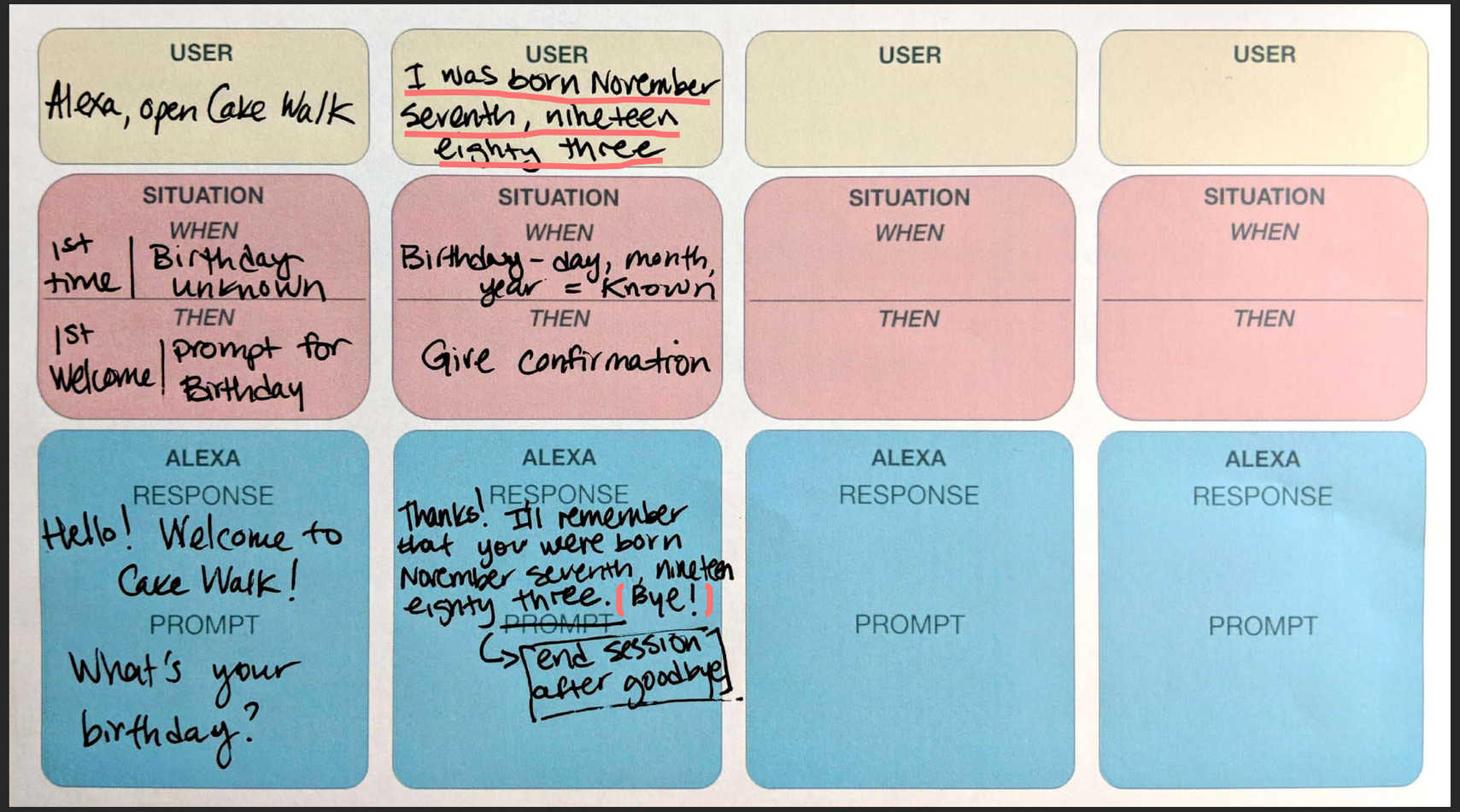


Identifying necessary information

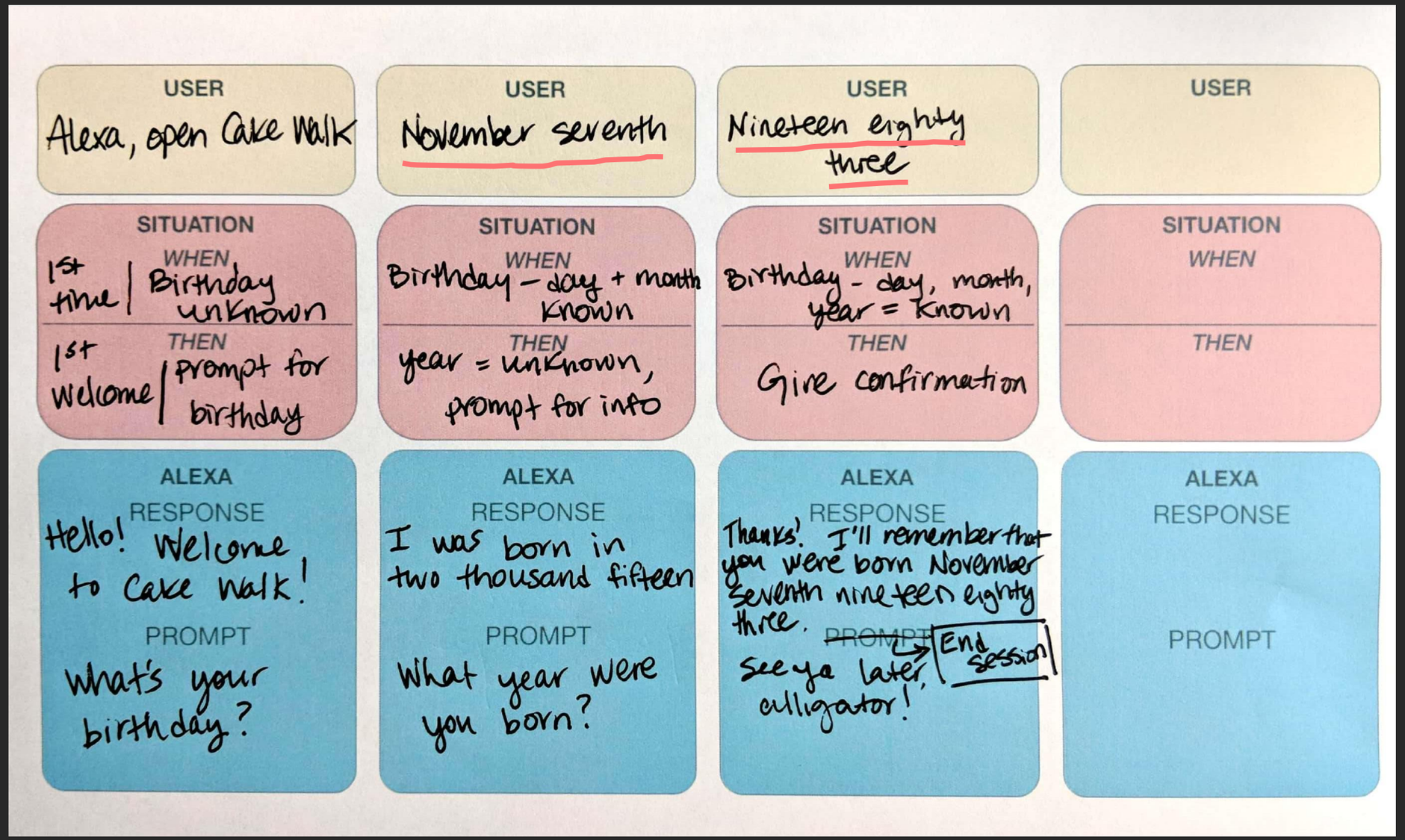


Activity: Underline your slots

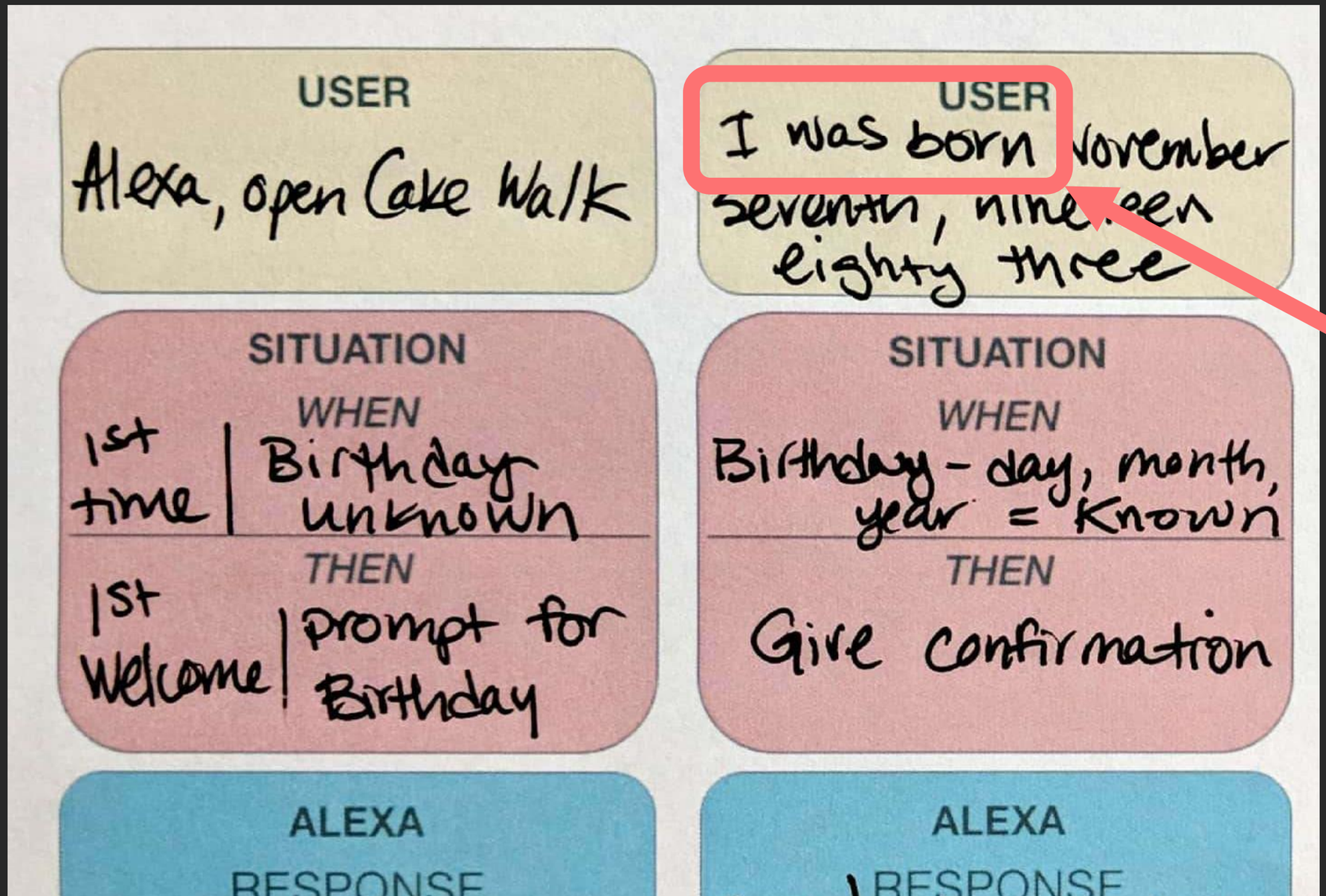
2 Min



Solution: Underlined slots for partial



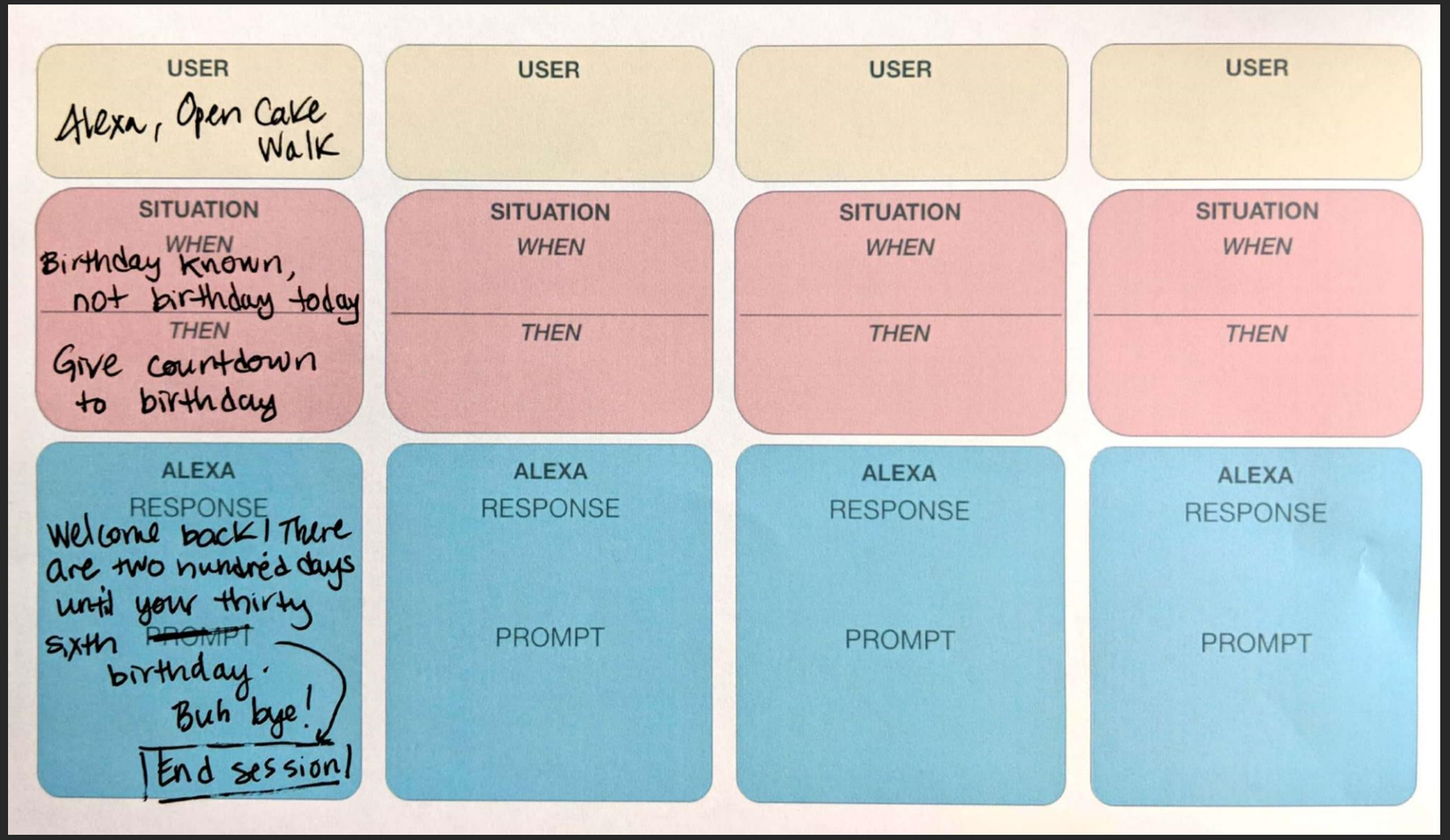
Did anyone underline this?



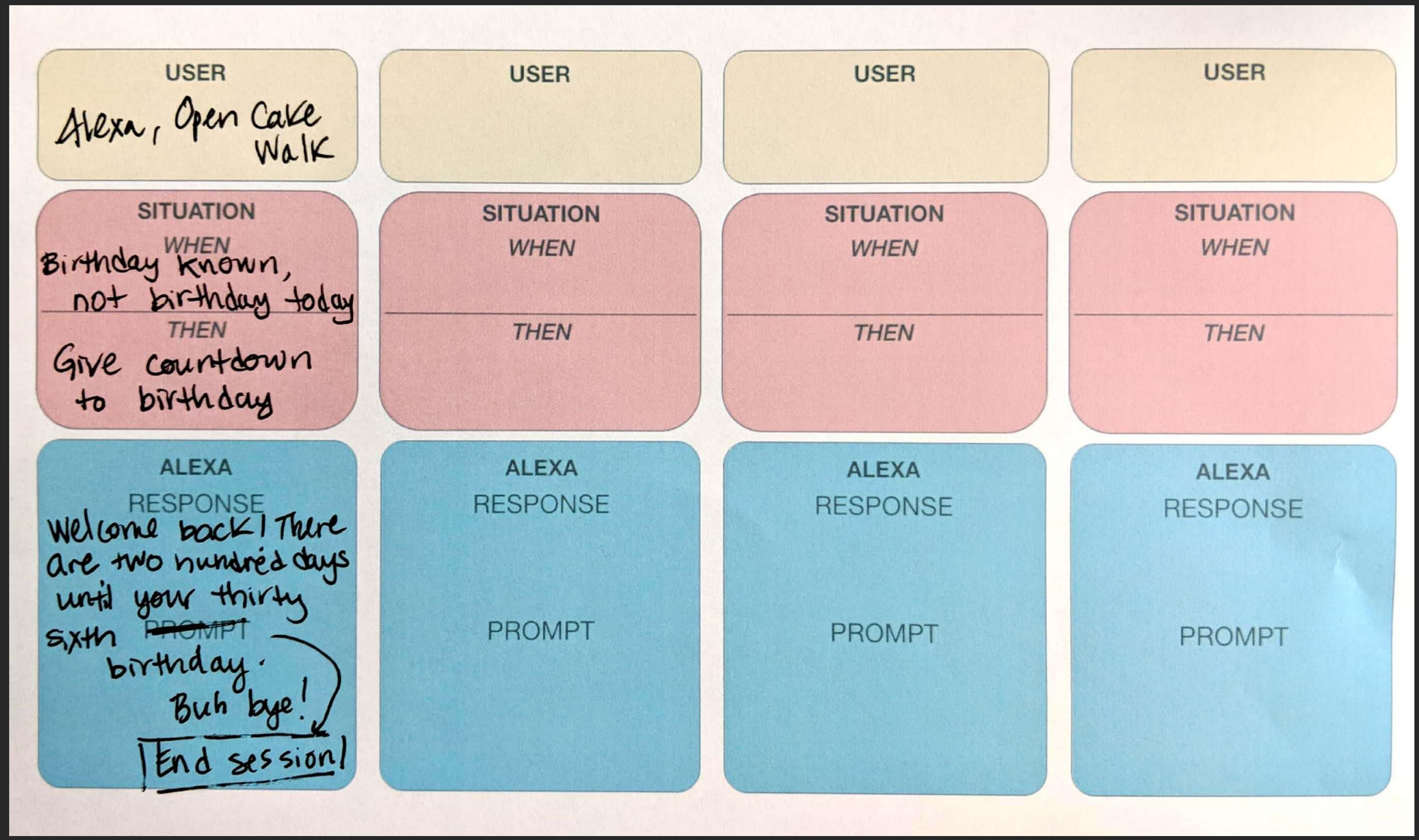
There are many ways to say this, so it should be a slot.

Activity: Create more storyboards

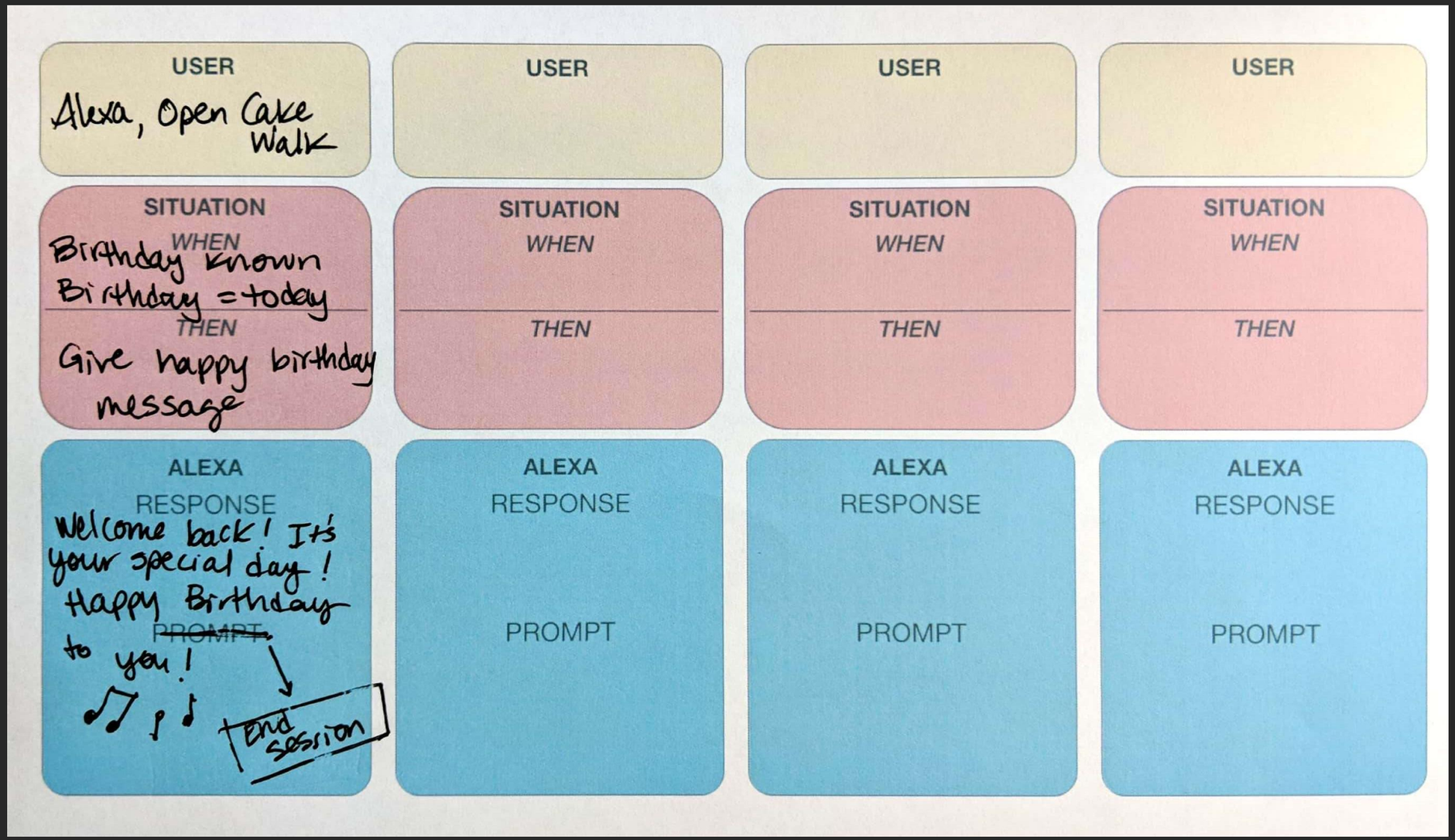
5 Min



Solution: Birthday countdown

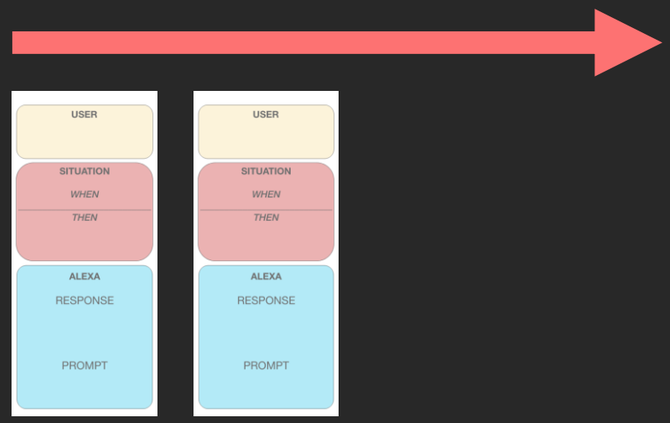


Solution: Birthday today!

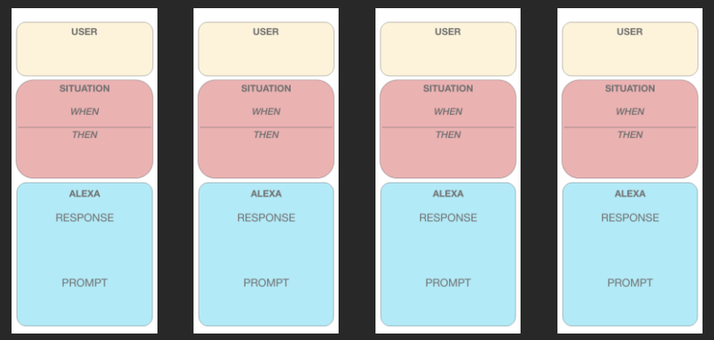


Storyboards and variations

Storyboard scenarios



Scenario 2



Deck of cards – Variations by theme



Build a skill in five minutes

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

What you'll create

1. Cake Walk skill
2. Interaction model
3. Backend resource

How to host your backend resource

- Node.js SDK
  - Best with Node.js v8.10 or higher
- Python SDK
- Java SDK

- AWS Lambda function
- Amazon CloudWatch Logs
- Amazon Simple Storage Service (Amazon S3) bucket
- AWS Identity and Access Management (IAM) role
- Amazon DynamoDB table

How to host your backend resource

- Node.js SDK
  - Best with Node.js v8.10 or higher
- Python SDK
- Java SDK

- AWS Lambda function
- Amazon CloudWatch Logs
- Amazon S3 bucket
- IAM role
- Amazon DynamoDB table

Provisioning your web service

Choose a method to host your skill's backend resources

You can provision your own backend resources or you can have Alexa host them for you. If you decide to have Alexa host your skill, you'll get access to our code editor, which will allow you to deploy code directly to AWS Lambda from the developer console.

Provision your own

Provision your own endpoint and backend resources for a skill. With this option, you will not gain access to the console's code editor.

Alexa-Hosted (Node.js)

SELECTED

Alexa will host skills in your account up to the AWS Free Tier limits and get you started with a Node.js template. You will gain access to an AWS Lambda endpoint, 5 GB of media storage with 15 GB of monthly data transfer, and a table for session persistence. Learn more

Alexa-Hosted (Python)

Alexa will host skills in your account up to the AWS Free Tier limits and get you started with a Python template. You will gain access to an AWS Lambda endpoint, 5 GB of media storage with 15 GB of monthly data transfer, and a table for session persistence. Learn more

Understanding the SDK

```
const handler = {
    canHandle(handlerInput) {
        // Returns true if the handler can service the request
    },
    handle(handlerInput) {
        // Code to handle the request
    }
}
```

Understanding the SDK

```
const handler = {
    canHandle(handlerInput) {
        // Returns true if the handler can service the request
    },
    handle(handlerInput) {
        // Code to handle the request
    }
}
```

Understanding the SDK

```
const handler = {
    canHandle(handlerInput) {
        // Returns true if the handler can service the request
    },
    handle(handlerInput) {
        // Code to handle the request
    }
}
```

Understanding the SDK

```
const handler = {
    canHandle(handlerInput) {
        // Returns true if the handler can service the request
    },
    handle(handlerInput) {
        // Code to handle the request
    }
}
```

Build your first skill

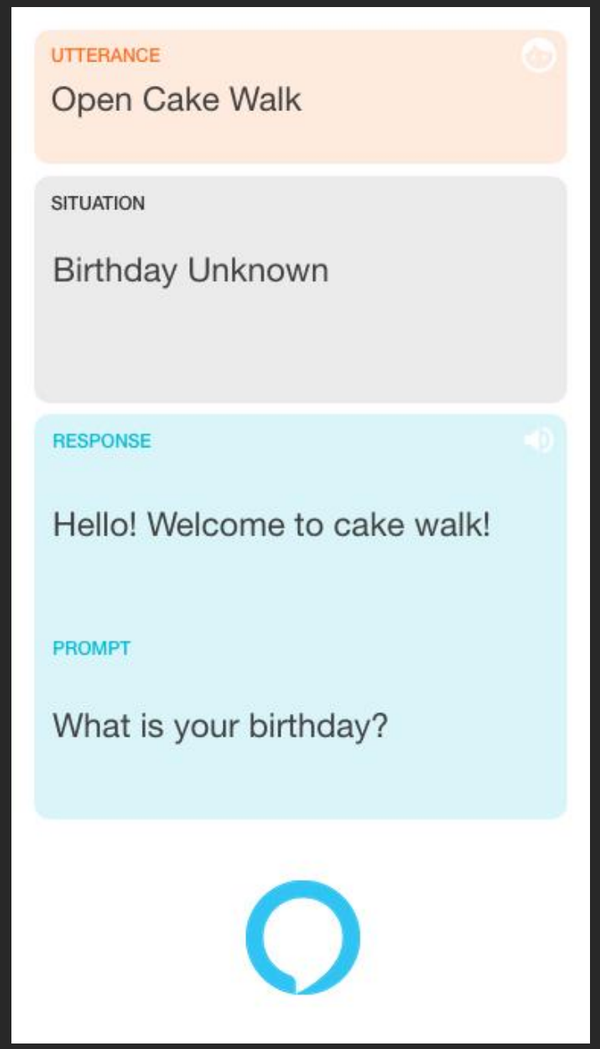
5 min

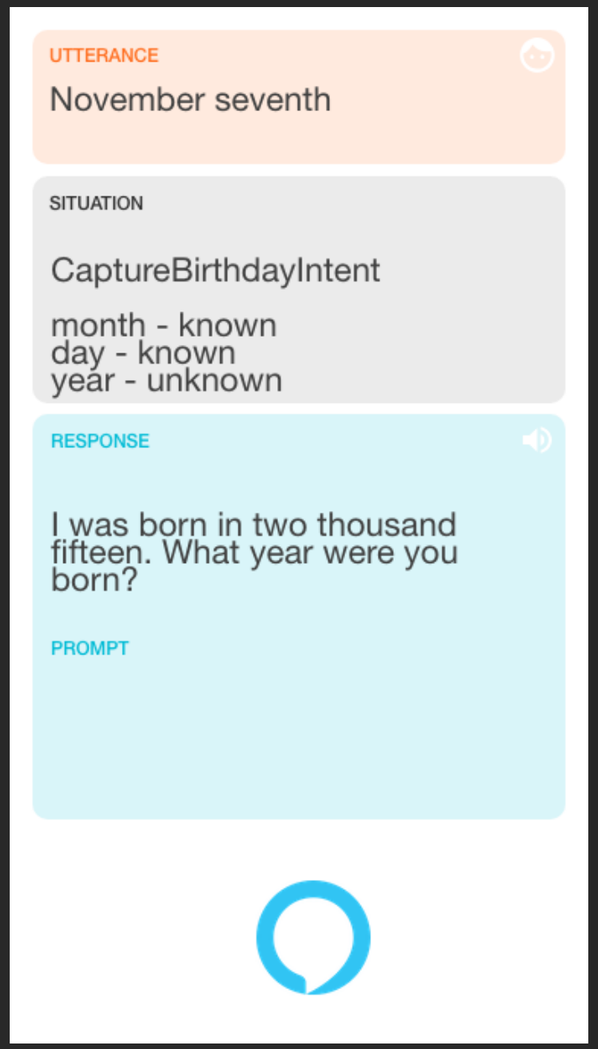
alexa.design/cake-walk-one

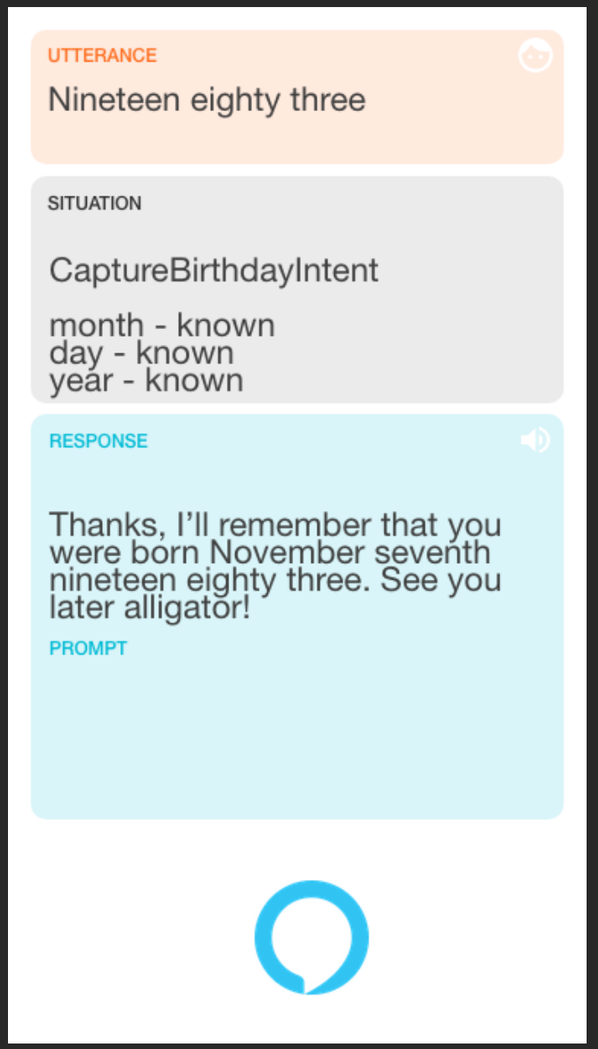
Collect slots turn by turn

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ask follow-up questions



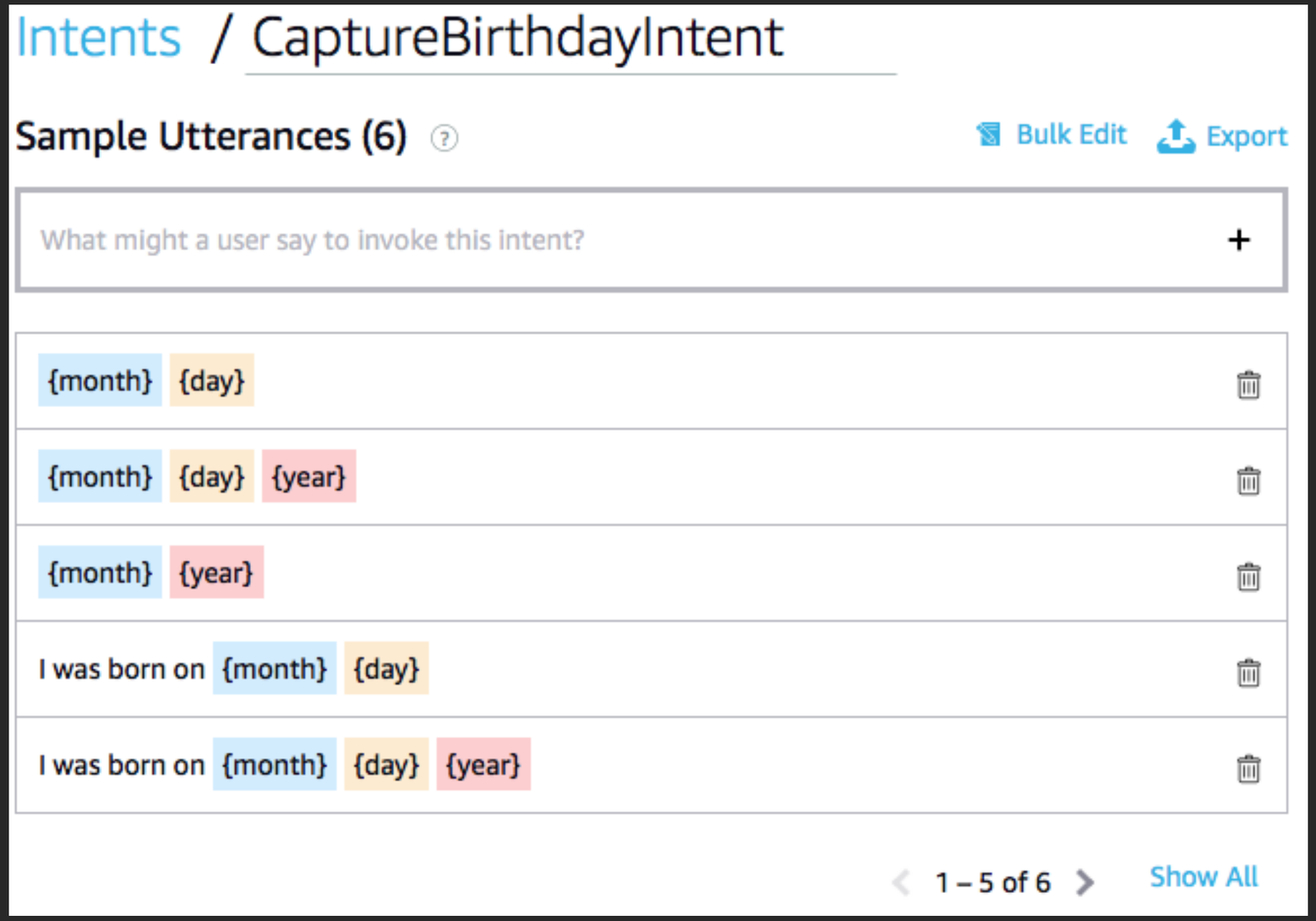




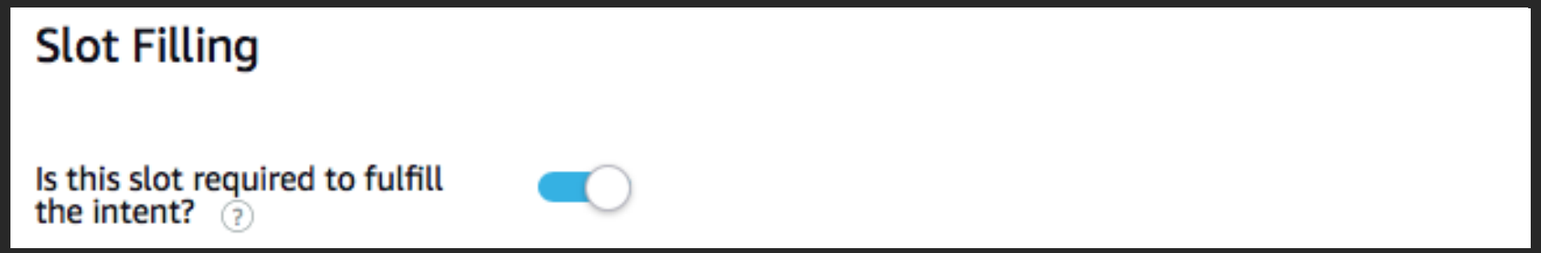
How to activate auto-delegation

1. Define slots
2. Mark them required
3. Define prompts
4. Choose a delegation strategy

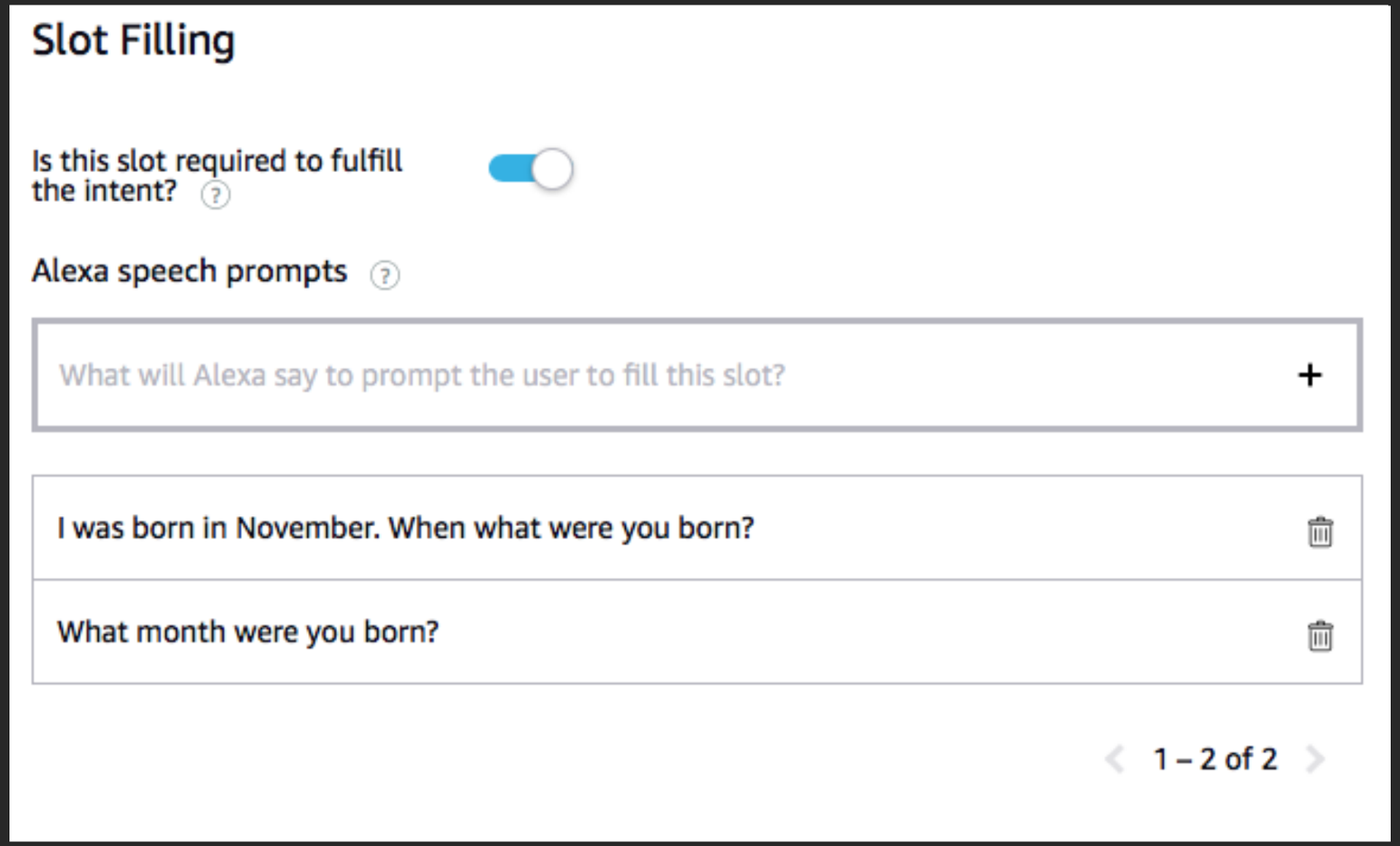
Define slots



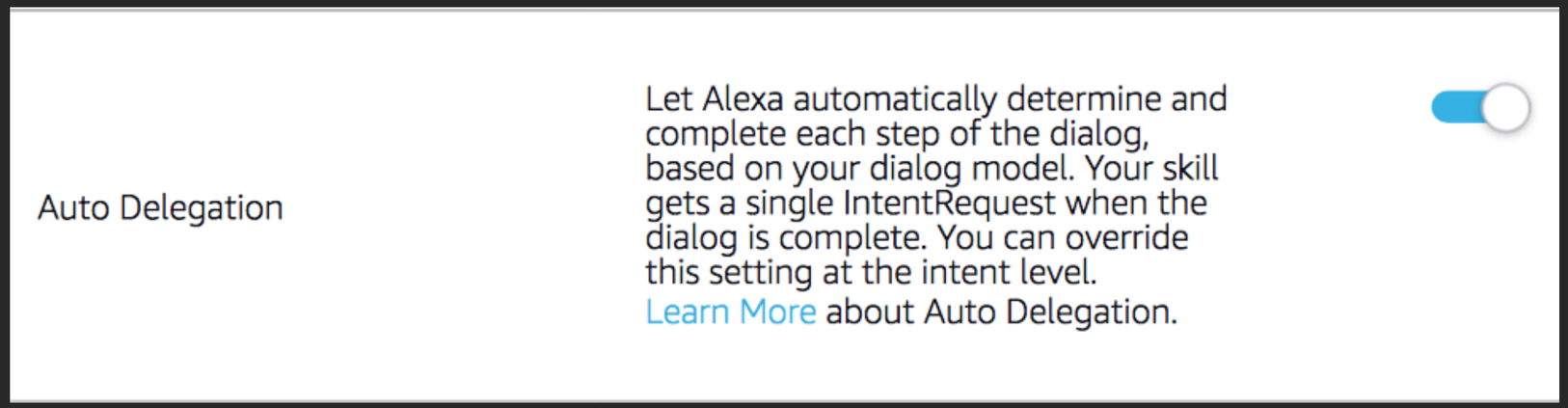
Mark required



Define prompts



Choose a delegation strategy

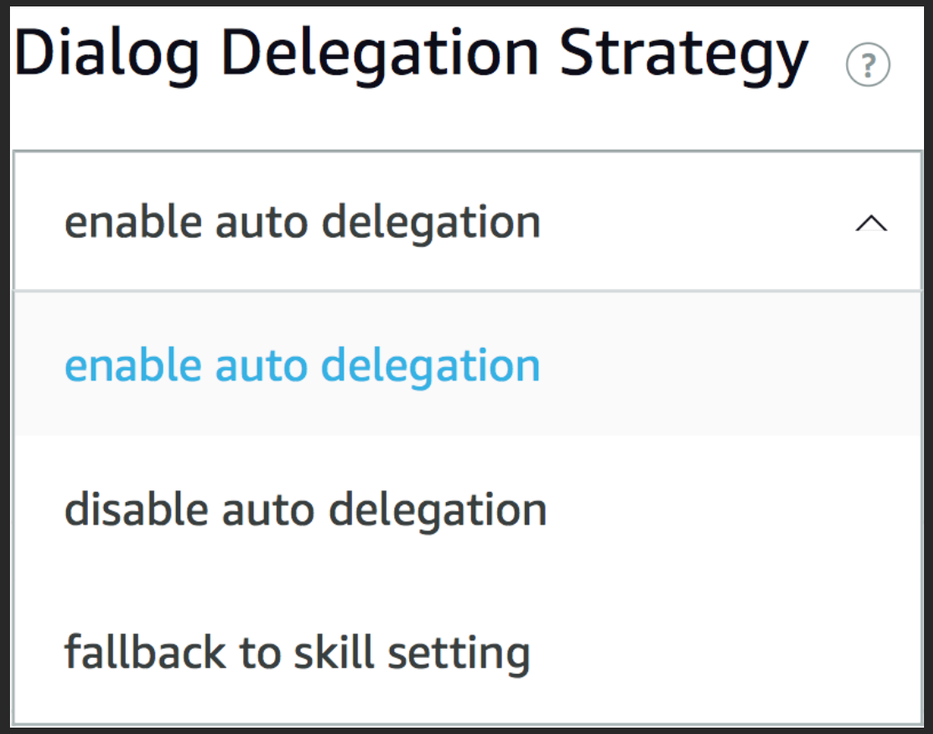


Auto-delegation vs. dialog management

- Hands-off
- Alexa service takes over until all slots are collected
- Skill receives a request after all required slots are collected

- Hands-on
- Skill receives a request from Alexa at each turn
- Skill returns a dialog directive to have Alexa prompt for the next slot

Enable, disable, or fallback



Collect information turn by turn

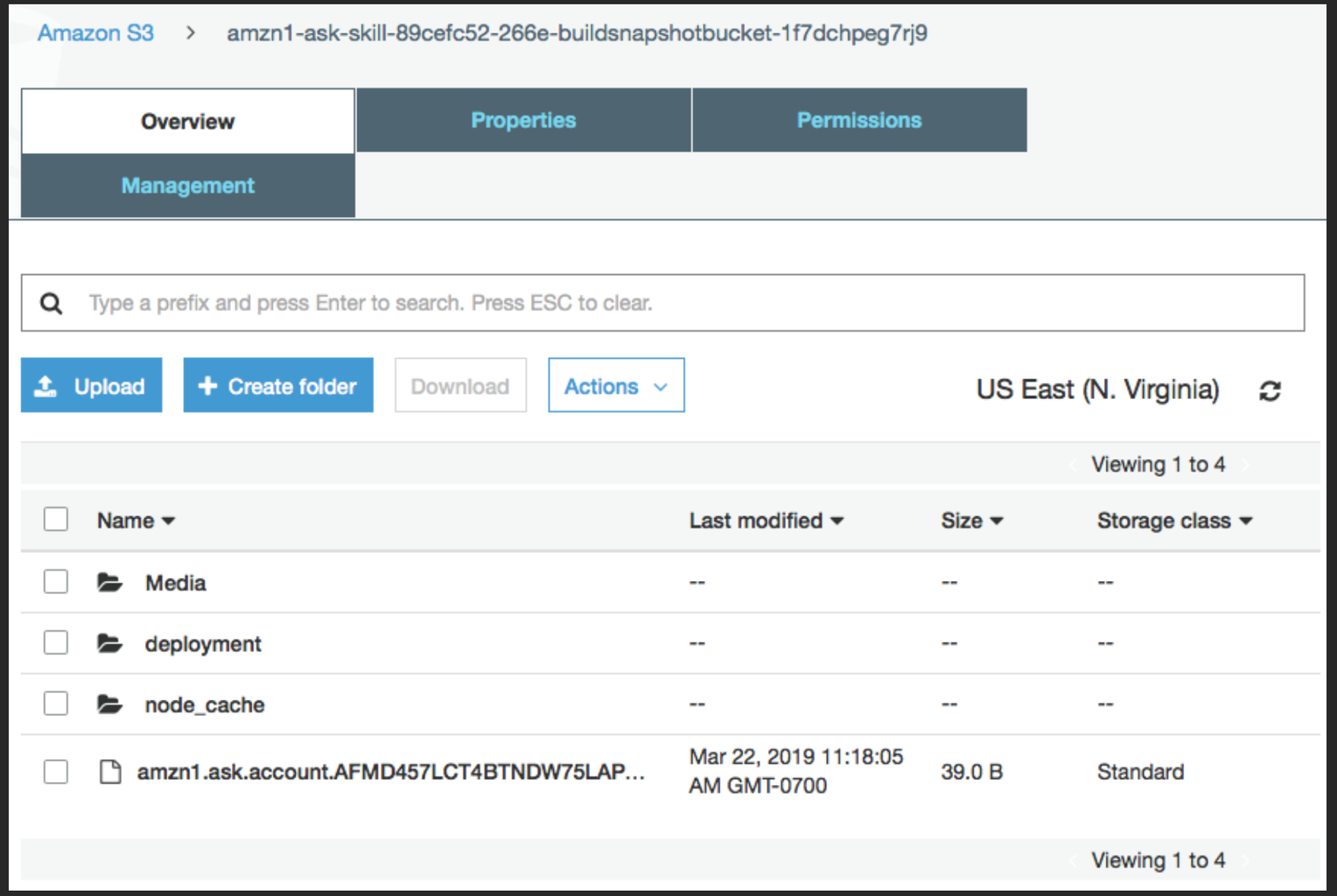
20 min

alexa.design/cake-walk-two

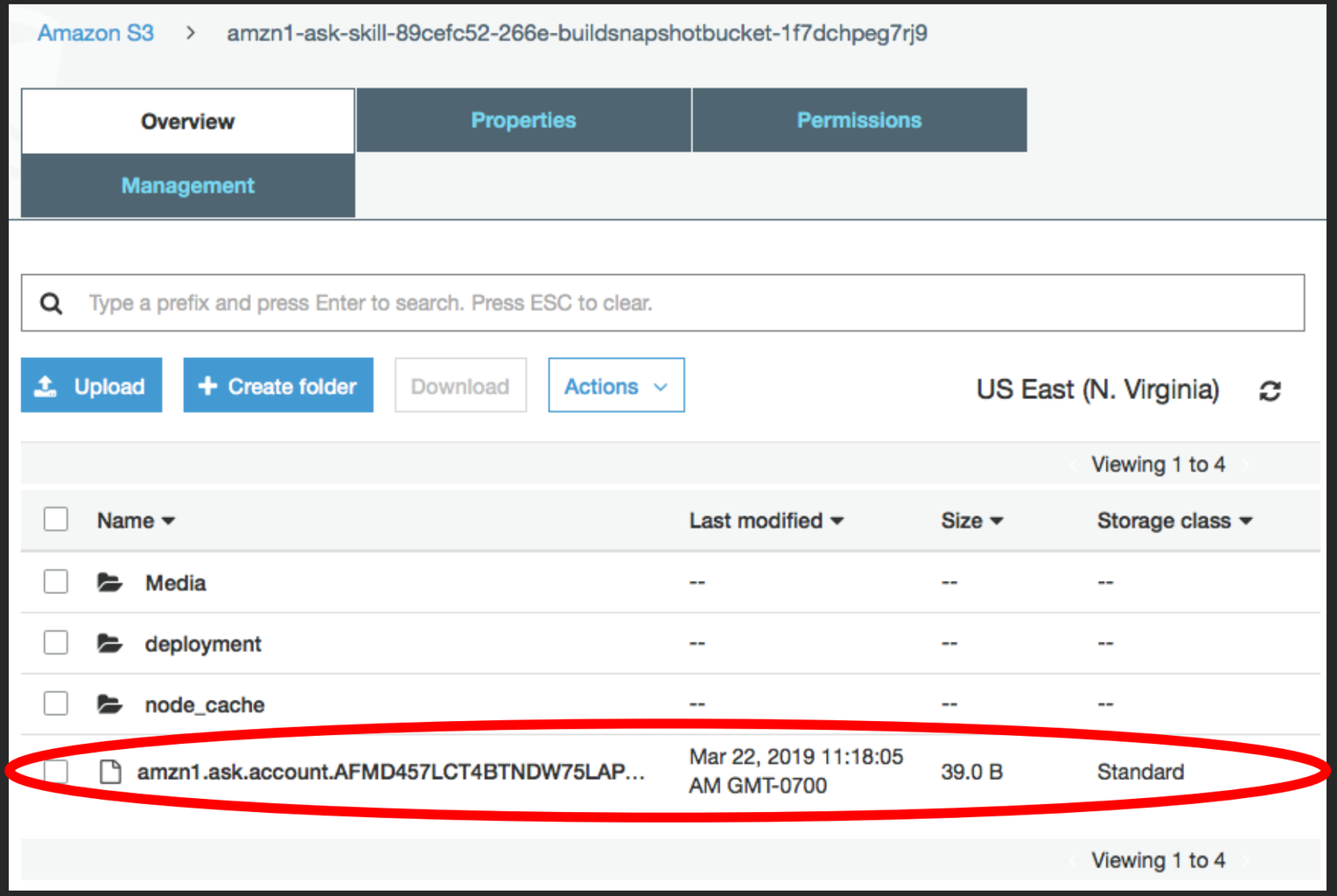
Add memory to Cake Walk

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Persistent attributes are visible in Amazon S3



Persistent attributes are visible in Amazon S3



Persistent attributes file

```
{
    "year": "1983",
    "month": "november",
    "day": "7"
}
```

What you'll need to save

1. Node module: ask-sdk-s3-persistence-adapter

2. Adapter: persistenceAdapter.S3PersistenceAdapter

3. Manager: AttributesManager

Amazon S3 persistence adapter node module

```
{
  "name": "calk-walk",
  "version": "0.9.0",
  "description": "a skill that counts the days to the customer’s birthday and wishes them happy
birthday.",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Amazon Alexa",
  "license": "ISC",
  "dependencies": {
    "ask-sdk-s3-persistence-adapter": "^2.0.0",
    "ask-sdk-core": "^2.0.7",
    "ask-sdk-model": "^1.4.1",
    "aws-sdk": "^2.326.0"
  }
}
```

Define the persistence adapter

```
exports.handler = Alexa.SkillBuilders.custom()
    .withPersistenceAdapter(
        new persistenceAdapter.S3PersistenceAdapter({bucketName:process.env.S3_PERSISTENCE_BUCKET})
    )
    .addRequestHandlers(…)
    .addErrorHandlers(ErrorHandler)
.lambda();
```

Create and save persistent attributes

```
async handle(handlerInput) {
        const year = handlerInput.requestEnvelope.request.intent.slots.year.value;
        const month = handlerInput.requestEnvelope.request.intent.slots.month.value;
        const day = handlerInput.requestEnvelope.request.intent.slots.day.value;

        const attributesManager = handlerInput.attributesManager;

        const birthdayAttributes = {
            "year": year,
            "month": month,
            "day": day

        };
        attributesManager.setPersistentAttributes(birthdayAttributes);
        await attributesManager.savePersistentAttributes();
```

Create and save persistent attributes

```
async handle(handlerInput) {
        const year = handlerInput.requestEnvelope.request.intent.slots.year.value;
        const month = handlerInput.requestEnvelope.request.intent.slots.month.value;
        const day = handlerInput.requestEnvelope.request.intent.slots.day.value;

        const attributesManager = handlerInput.attributesManager;

        const birthdayAttributes = {
            "year": year,
            "month": month,
            "day": day

        };
        attributesManager.setPersistentAttributes(birthdayAttributes);
        await attributesManager.savePersistentAttributes();
```

Create and save persistent attributes

```
async handle(handlerInput) {
        const year = handlerInput.requestEnvelope.request.intent.slots.year.value;
        const month = handlerInput.requestEnvelope.request.intent.slots.month.value;
        const day = handlerInput.requestEnvelope.request.intent.slots.day.value;

        const attributesManager = handlerInput.attributesManager;

        const birthdayAttributes = {
            "year": year,
            "month": month,
            "day": day

        };
        attributesManager.setPersistentAttributes(birthdayAttributes);
        await attributesManager.savePersistentAttributes();
```

Create and save persistent attributes

```
async handle(handlerInput) {
        const year = handlerInput.requestEnvelope.request.intent.slots.year.value;
        const month = handlerInput.requestEnvelope.request.intent.slots.month.value;
        const day = handlerInput.requestEnvelope.request.intent.slots.day.value;

        const attributesManager = handlerInput.attributesManager;

        const birthdayAttributes = {
            "year": year,
            "month": month,
            "day": day

        };
        attributesManager.setPersistentAttributes(birthdayAttributes);
        await attributesManager.savePersistentAttributes();
```

Create and save persistent attributes

```
async handle(handlerInput) {
        const year = handlerInput.requestEnvelope.request.intent.slots.year.value;
        const month = handlerInput.requestEnvelope.request.intent.slots.month.value;
        const day = handlerInput.requestEnvelope.request.intent.slots.day.value;

        const attributesManager = handlerInput.attributesManager;

        const birthdayAttributes = {
            "year": year,
            "month": month,
            "day": day

        };
        attributesManager.setPersistentAttributes(birthdayAttributes);
        await attributesManager.savePersistentAttributes();
```

Create and save persistent attributes

```
async handle(handlerInput) {
        const year = handlerInput.requestEnvelope.request.intent.slots.year.value;
        const month = handlerInput.requestEnvelope.request.intent.slots.month.value;
        const day = handlerInput.requestEnvelope.request.intent.slots.day.value;

        const attributesManager = handlerInput.attributesManager;

        const birthdayAttributes = {
            "year": year,
            "month": month,
            "day": day

        };
        attributesManager.setPersistentAttributes(birthdayAttributes);
        await attributesManager.savePersistentAttributes();
```

We can now remember (save) their birthday, but how do we recall (get) it?

Steps to recall their birthday

1. Set up connection to Amazon S3

2. Read the data for the current user

3. Restore context

Set up the connection to Amazon S3 – done!

```
exports.handler = Alexa.SkillBuilders.custom()
    .withPersistenceAdapter(
        new
persistenceAdapter.S3PersistenceAdapter({bucketName:process.env.S3_PERSISTENCE_BUCKET})
    )
    .addRequestHandlers(…)
    .addErrorHandlers(ErrorHandler)
.lambda();
```

Read the data for the current user

```
const attributesManager = handlerInput.attributesManager;

Const attributes = await attributesManager.getPersistentAttributes() || {};
```

Read the data for the current user

```
const attributesManager = handlerInput.attributesManager;

Const attributes = await attributesManager.getPersistentAttributes() || {};
```

Read the data for the current user

```
const attributesManager = handlerInput.attributesManager;

const attributes = await attributesManager.getPersistentAttributes() || {};
```

Read the data for the current user

```
const attributesManager = handlerInput.attributesManager;

const attributes = await attributesManager.getPersistentAttributes() || {};
```

Restore context

```
const attributesManager = handlerInput.attributesManager;

const attributes = await attributesManager.getPersistentAttributes() || {};


const year = attributes.hasOwnProperty('year') ? attributes.year : 0;

const month = attributes.hasOwnProperty('month') ? attributes.month : 0;

const day = attributes.hasOwnProperty('day') ? attributes.day : 0;


if (year && month && day) {

        // do what you need with the birthday

    }

}
```

Restore context

```
const attributesManager = handlerInput.attributesManager;

const attributes = await attributesManager.getPersistentAttributes() || {};


const year = attributes.hasOwnProperty('year') ? attributes.year : 0;

const month = attributes.hasOwnProperty('month') ? attributes.month : 0;

const day = attributes.hasOwnProperty('day') ? attributes.day : 0;


if (year && month && day) {

        // do what you need with the birthday

    }

}
```

Avoid reading from Amazon S3 twice

HasBirthdayLaunchIntentHandler

canHandle

- Returns true if
  - type === 'LaunchRequest’
  - has birthday entry in Amazon S3

handle

- Use birthday entry from Amazon S3 to compute the number of days until birthday

Recall our steps to remember their birthday

1. Set up connection to Amazon S3

2. Read the data for the current user

3. Restore context

Recall our steps to remember their birthday

1. Set up connection to Amazon S3
2. Define an interceptor
3. Read the data for the current user
4. Restore context

Interceptors

- Built into the SDK
- Allow for processing requests **before** your handlers
- Allow for processing responses **after** your handlers

LoadBirthdayInterceptor

```
const LoadBirthdayInterceptor = {
    async process(handlerInput) {
        const attributesManager = handlerInput.attributesManager;
        const sessionAttributes = await attributesManager.getPersistentAttributes() || {};

        const year = sessionAttributes.hasOwnProperty('year') ? sessionAttributes.year : 0;
        const month = sessionAttributes.hasOwnProperty('month') ? sessionAttributes.month : 0;
        const day = sessionAttributes.hasOwnProperty('day') ? sessionAttributes.day : 0;

        if (year && month && day) {
            attributesManager.setSessionAttributes(sessionAttributes);
        }
}}
```

LoadBirthdayInterceptor

```
const LoadBirthdayInterceptor = {
    async process(handlerInput) {
        const attributesManager = handlerInput.attributesManager;
        const sessionAttributes = await attributesManager.getPersistentAttributes() || {};

        const year = sessionAttributes.hasOwnProperty('year') ? sessionAttributes.year : 0;
        const month = sessionAttributes.hasOwnProperty('month') ? sessionAttributes.month : 0;
        const day = sessionAttributes.hasOwnProperty('day') ? sessionAttributes.day : 0;

        if (year && month && day) {
            attributesManager.setSessionAttributes(sessionAttributes);
        }
}}
```

LoadBirthdayInterceptor

```
const LoadBirthdayInterceptor = {
    async process(handlerInput) {
        const attributesManager = handlerInput.attributesManager;
        const sessionAttributes = await attributesManager.getPersistentAttributes() || {};

        const year = sessionAttributes.hasOwnProperty('year') ? sessionAttributes.year : 0;
        const month = sessionAttributes.hasOwnProperty('month') ? sessionAttributes.month : 0;
        const day = sessionAttributes.hasOwnProperty('day') ? sessionAttributes.day : 0;

        if (year && month && day) {
            attributesManager.setSessionAttributes(sessionAttributes);
        }
}}
```

Persistent attributes vs. session attributes

- When you need to remember data across multiple sessions
- Stored in data stores

- When you need to retain data during the session
- Stored in the request – response JSON

Register the interceptor

```
exports.handler = Alexa.SkillBuilders.custom()
    .withPersistenceAdapter(
        new
persistenceAdapter.S3PersistenceAdapter({bucketName:process.env.S3_PERSISTENCE_BUCKET})
    )
    .addRequestHandlers(…)
    .addRequestInterceptors(
        LoadBirthdayInterceptor
    )
    .addErrorHandlers(ErrorHandler)
    .lambda();
```

Add memory to Cake Walk

20 min

alexa.design/cake-walk-three

Use Alexa settings API to look up the time zone

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Time zone settings are device dependent

```
…
"context": {
    "System": {
        "application": {
            "applicationId": "amzn1.ask.skill.XXXX"
        },
        "user": {
            "userId": "amzn1.ask.account.XXXX"
        },
        "device": {
            "deviceId": "amzn1.ask.device.XXXX",
            "supportedInterfaces": {}
        },
…
```

Time zone settings are device dependent

```
…
"context": {
    "System": {
        "application": {
            "applicationId": "amzn1.ask.skill.XXXX"
        },
        "user": {
            "userId": "amzn1.ask.account.XXXX"
        },
        "device": {
            "deviceId": "amzn1.ask.device.XXXX",
            "supportedInterfaces": {}
        },
…
```

Time zone settings are device dependent

```
…
"context": {
    "System": {
        "application": {
            "applicationId": "amzn1.ask.skill.XXXX"
        },
        "user": {
            "userId": "amzn1.ask.account.XXXX"
        },
        "device": {
            "deviceId": "amzn1.ask.device.XXXX",
            "supportedInterfaces": {}
        },
…
```

```
const deviceId = handlerInput.requestEnvelope
                             .context.System.device.deviceId;
```

Define the default API client

```
exports.handler = Alexa.SkillBuilders.custom()
    .withPersistenceAdapter(
        new
persistenceAdapter.S3PersistenceAdapter({bucketName:process.env.S3_PERSISTENCE_BUCKET})
    )
    .addRequestHandlers(…) handlers
    .addErrorHandlers(ErrorHandler)
    .addRequestInterceptors(
        LoadBirthdayInterceptor
    )
    .withApiClient(new Alexa.DefaultApiClient())
    .lambda();
```

Use the user profile settings service client

```
async handle(handlerInput) {


    const serviceClientFactory = handlerInput.serviceClientFactory;
    const deviceId = handlerInput.requestEnvelope.context.System.device.deviceId;


    const upsServiceClient = serviceClientFactory.getUpsServiceClient();
    userTimeZone = await upsServiceClient.getSystemTimeZone(deviceId);
    …
}
```

Use the user profile settings service client

```
async handle(handlerInput) {

    const serviceClientFactory = handlerInput.serviceClientFactory;
    const deviceId = handlerInput.requestEnvelope.context.System.device.deviceId;


    const upsServiceClient = serviceClientFactory.getUpsServiceClient();
    userTimeZone = await upsServiceClient.getSystemTimeZone(deviceId);
    …
}
```

Use the user profile settings service client

```
async handle(handlerInput) {


    const serviceClientFactory = handlerInput.serviceClientFactory;
    const deviceId = handlerInput.requestEnvelope.context.System.device.deviceId;


    const upsServiceClient = serviceClientFactory.getUpsServiceClient();
    userTimeZone = await upsServiceClient.getSystemTimeZone(deviceId);
    …
}
```

Use the user profile settings service client

```
async handle(handlerInput) {


    const serviceClientFactory = handlerInput.serviceClientFactory;
    const deviceId = handlerInput.requestEnvelope.context.System.device.deviceId;


    const upsServiceClient = serviceClientFactory.getUpsServiceClient();
    userTimeZone = await upsServiceClient.getSystemTimeZone(deviceId);
    …
}
```

Use the user profile settings service client

```
async handle(handlerInput) {


    const serviceClientFactory = handlerInput.serviceClientFactory;
    const deviceId = handlerInput.requestEnvelope.context.System.device.deviceId;


    const upsServiceClient = serviceClientFactory.getUpsServiceClient();
    userTimeZone = await upsServiceClient.getSystemTimeZone(deviceId);
    …
}
```

Use the user profile settings service client

```
async handle(handlerInput) {


    const serviceClientFactory = handlerInput.serviceClientFactory;
    const deviceId = handlerInput.requestEnvelope.context.System.device.deviceId;


    const upsServiceClient = serviceClientFactory.getUpsServiceClient();
    userTimeZone = await upsServiceClient.getSystemTimeZone(deviceId);
    …
}
```

Use the settings API

Until you're done

alexa.design/cake-walk-four

How many days until your next happy birthday?



Follow-up and more resources

- Read the course
- Download the Cake Walk source code
- Interact with the community

Resources:

- alexa.design/cdw
- alexa.design/guide

Follow us:

 @AlexaDevs

 twitch.tv/amazonalexa

Learn Alexa skills with AWS Training and Certification

Build and validate skills to help you build solutions for 100+ million Alexa-enabled devices



Free digital courses cover topics related to Alexa skill-building, including Alexa for Developers and Designing for Conversation



Validate expertise with the
AWS Certified Alexa Skill Builder - Specialty exam

Visit aws.amazon.com/certification/certified-alexa-skill-builder-specialty

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

Karthik Ragubathy
pkartikr@amazon.com

Sohan Maheshwar
sohanm@amazon.com

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.