



DEEPDIVE: CHATBOT TECHNOLOGY

TIPS & TRICKS

22 October 2019

Session Number: 5539

Track: Technical

CY2 SESSION OVERVIEW



Deep dive: Chatbot Technology

Session: 5539

Time: Oct 22 (12:05 PM - 12:55 PM)



Coordinating and monitoring your program logistics

Session: 5538

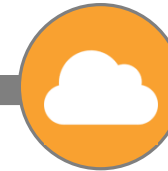
Time: Oct 23 (11:05 AM - 11:55 AM)



Fluid PE Self-service

Session: 5537

Time: Oct 23 (10:00 AM - 10:50 AM)



What's new in Student Cloud

Session: 5536

Time: Oct 23 (01:45 PM - 02:35 PM)



2019 Fluid projects at:



CY2 is a global reference partner of educational institutions for training and guidance. We aim for state of the art solutions related to student information services.

EMEA ALLIANCE 22-23 OCTOBER 2019

Academic Homepage **Advisor self service**

▼ **Search for a student**

Firstname
Surname
CRSId
USN

Search **Clear**

▼ **Quick filters & links**

To-do **11**

My current students

My new students

Director of Studies

Graduate Supervisor

My current students

Group students by:

Name My role College Department Tripos part Advisers **Start term** Course Residency

Michaelmas Term 2018 14 rows


Send email


Academic Homepage **Advisor Self Service**

Student Summary

Mr Marcelo Macdonald

Trinity College - ASNC Tripos - Admitted Michaelmas Term 2016 - Current Student

 USN 303366795 Born in United Kingdom.
Gender Male Citizen of United Kingdom.
CRS Id hf321 No Visa
Date of Birth 21/04/1997 Home residency.
Side D

Related Transactions 

Contact details Student record Exam results **Previous education** Questionnaire

▼ **Trinity College - ASNC Tripos - Admitted Michaelmas Term 2016 - Current Student**

Exam results

▼ **Easter Term 2019**

Results

Subject	Description	Total / Out of	Grade Boundaries
AST2	Anglo-Saxon, Norse and Celtic Tripos, Part II (Result)	-	.

Papers

Subject	Description	Number	Mark / Out of
AST2	The Anglo-Saxon Chancery (Exam)	1	-
AST2	The Sea Kings and the Celtic-speaking world, c.1014-1164 (Exam)	3	-
AST2	Writing women (Exam)	9	-
AST2	Dissertation (Dissertn)	DIS	-
AST2	The 'Angevin Empire', 1150s- 1230s (Exam)	14	-

Fluid Academic Self Service (a bespoke approach)

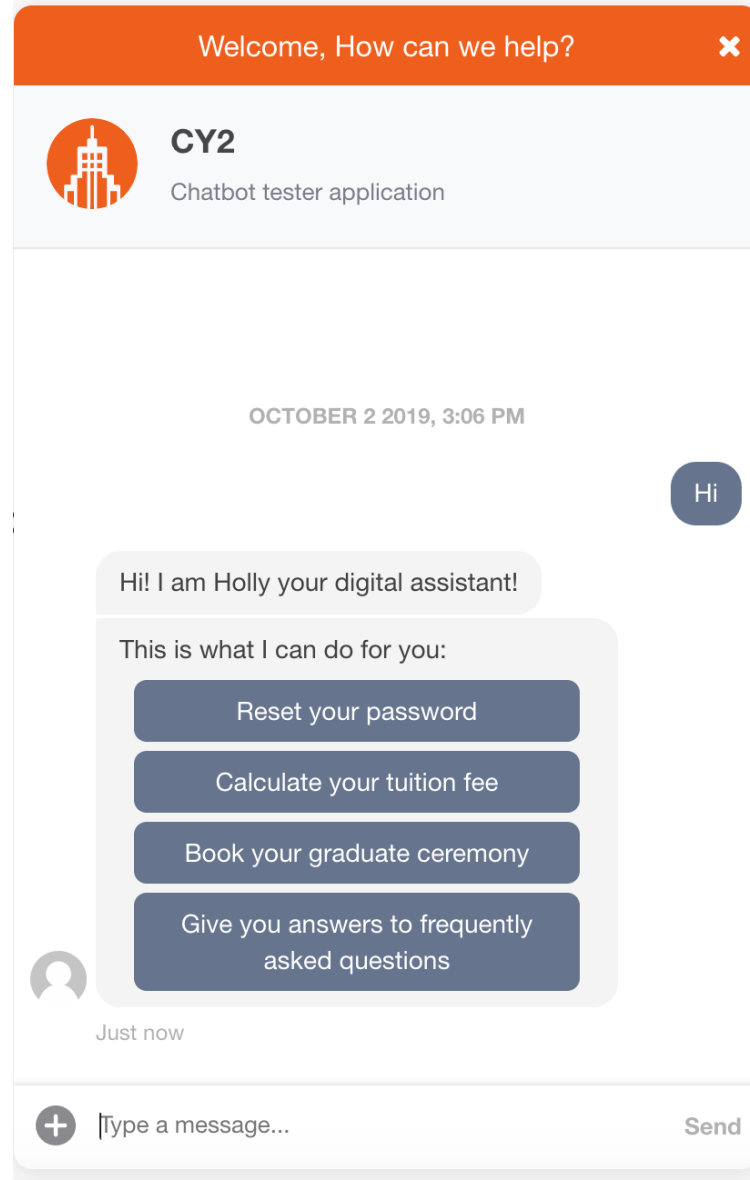
Session Number: 5503

Time: Oct 22, 2019 (02:55 PM - 03:45 PM)

EMEA ALLIANCE 22-23 OCTOBER 2019

CHATBOT TECHNOLOGY

DEEPDIVE





DEMO

SKEMA BUSINESS SCHOOL

A Business case

We don't want prospects to get stuck when applying ... **(students)**

Lower the burden on stressed-out faculty, as they no longer have to explain the same things over and over to different students **(staff)**

Implementation

Uses CY2 Fluid application form to apply for programs.

On every application page the chatbot is active.

Hereunder the details of the programme of your choice. Please fill in the missing details and click Create Application to start your application.

Hereunder an overview of your existing applications. Click Edit to edit a saved application, click on View to see a summary of your submitted application. The status of your application is shown in the Status column.

Application Number	Academic Career	Specialisation	Status
00044211	Postgraduate	Int Marketing & Bus Dev (2YR)	Posted
VIEW	MAINTAIN	PRINT RECEIPT	
00044212	Postgraduate	Digital Marketing (2YR)	Saved
EDIT	DELETE		
00044214	Undergraduate	E-Business (1YM)	Saved
EDIT	DELETE		
00044226	Postgraduate	Stratégie d'Entreprise et Man.	Saved
EDIT	DELETE		

Application Details

Program

Specialization

Campus

Intake

[CREATE APPLICATION](#)

LET'S SEE IT IN ACTION

LIVE DEMO

skema
BUSINESS SCHOOL

APPLY

Hereunder the details of the programme of your choice. Please fill in the missing details and click **Create Application** to start your application.

Hereunder an overview of your applications. To edit a saved application, click the **EDIT** button. To delete a submitted application, click the **DELETE** button. The status is shown in the Status column.

Application Number	Academic Career		
00044211	Postgraduate	VIEW	MAINTAIN
00044212	Postgraduate	EDIT	DELETE
00044214	Undergraduate	EDIT	DELETE
00044226	Postgraduate	EDIT	DELETE

Application Details

Program

Specialization

Campus

Intake

[CREATE APPLICATION](#)

skema
BUSINESS SCHOOL

APPLY

APPLICATION FORM

Introduction

Campus Choice

Personal Details

Contact Details

Education

Test Scores

Work Experience

Reason for applying

Survey

Attachments

Review the Application

Agreement

Application Fee

Submit

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

14

Details

Application Nbr 00044212

2YR Digital Marketing (2YR)

Intake September 2020

Question: internationalprograms@skema.edu

Mister

Ernst2

LA HAYE

01/06/2001

Ecuador

Unemployed

PREVIOUS STEP

SAVE

NEXT STEP

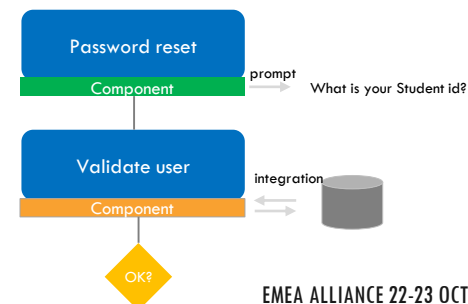
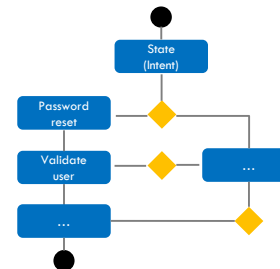
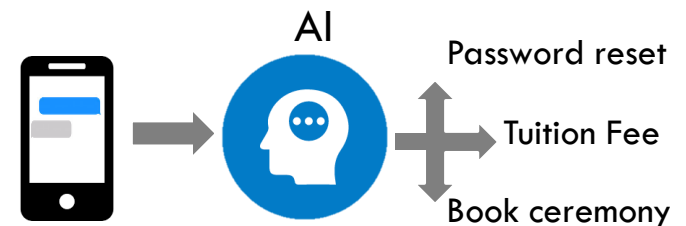
1

EMEA ALLIANCE 22-23 OCTOBER 2019

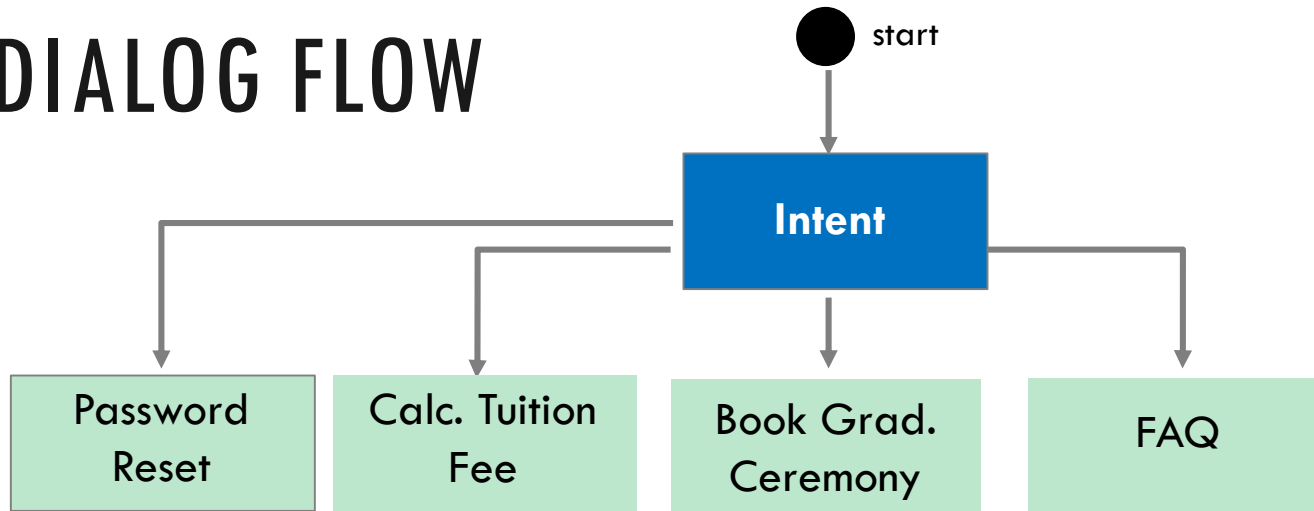
DEFINITIONS

The following definitions are used throughout this presentation.

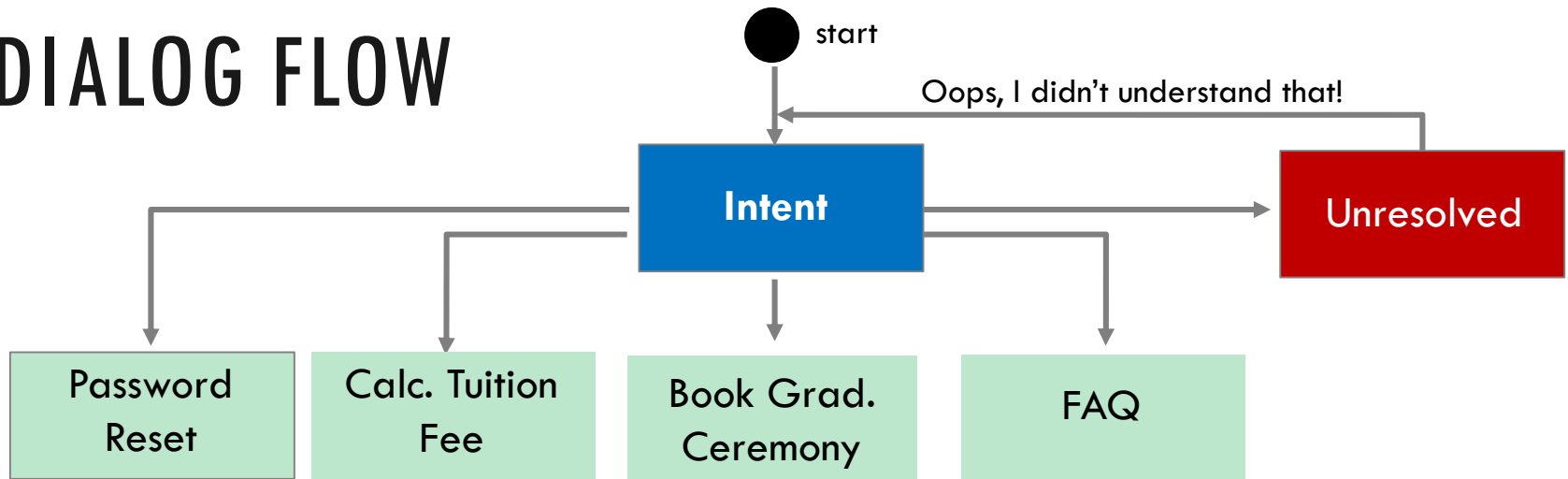
- 1. Intent**
Defines what the users wants (intention) and then starts a dialog flow.
- 2. Dialog flow**
Consist of a series of steps (states) that are executed in a given order.
- 3. Component**
Each state is tied to a component, responsible for things like outputting text or validating against a 3rd party system.



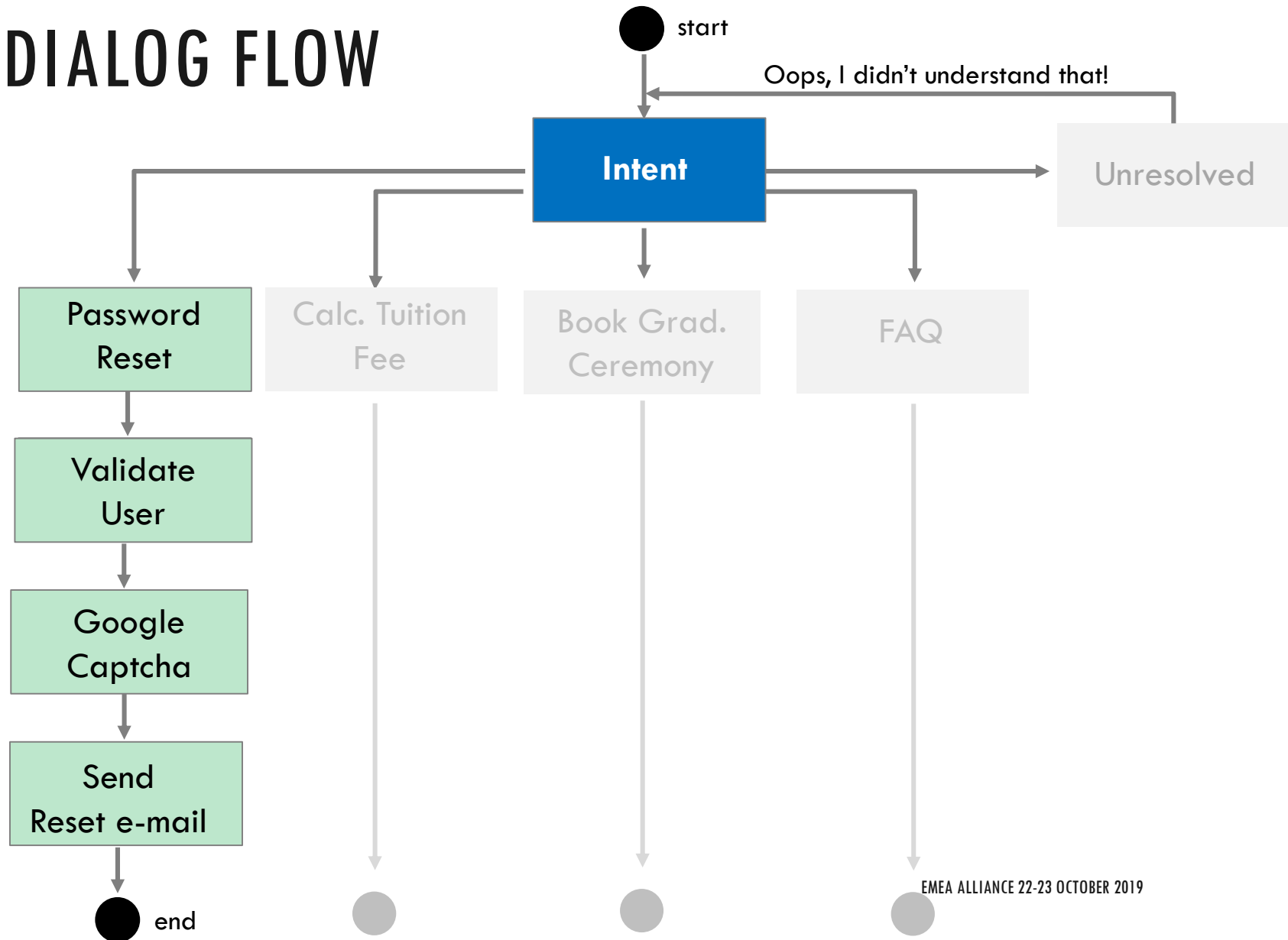
DIALOG FLOW



DIALOG FLOW



DIALOG FLOW



INTENT



Utterances

To make sure your intent has a high probability to resolve, enter a lot of utterances.

+ Intent

More ▾

Filter

🔍

Sort By

Created Ascending ▾

CancelPizza

×

OrderPizza

×

Page 1 of 1

⏪ ⏩ ⏴ ⏵

Description

Try It Out!

Conversation Name *

OrderPizza

Name *

OrderPizza

Description

Examples ⓘ

Filter

Enter your example utterances here.

Can I order a Pizza?

Do you have deep dish pizzas available?

Do you server gluten-free pizza?

How much does a slice of pizza cost?

I feel like eating some pizza

I want to order pizza for lunch

I'd like to order a Pie please

Let's order a cheese pizza

Order Pizza!

What types of pizzas can I order?

What's on the menu today?

Would love a large Pepperoni please!

Would you happen to have thin crust options on your Pizzas?

Examples ⓘ

Filter

Enter your example utterances here.

Can I order a Pizza?

Do you have deep dish pizzas available?

Do you server gluten-free pizza?

How much does a slice of pizza cost?

I feel like eating some pizza

I want to order pizza for lunch

I'd like to order a Pie please

Let's order a cheese pizza

Order Pizza!

What types of pizzas can I order?

What's on the menu today?

Would love a large Pepperoni please!

Would you happen to have thin crust options on your Pizzas?

DIALOG FLOW



Hi there! What would you like me to echo back?

Hello!

Ola!

Vannakam!

Namaste!



Ola!

What is your name?



Stefan

Ola! Stefan

Present a list of options

And store it for later reference

Ask an open question

And store it for later reference too

Output the result

Using the previous two inputs

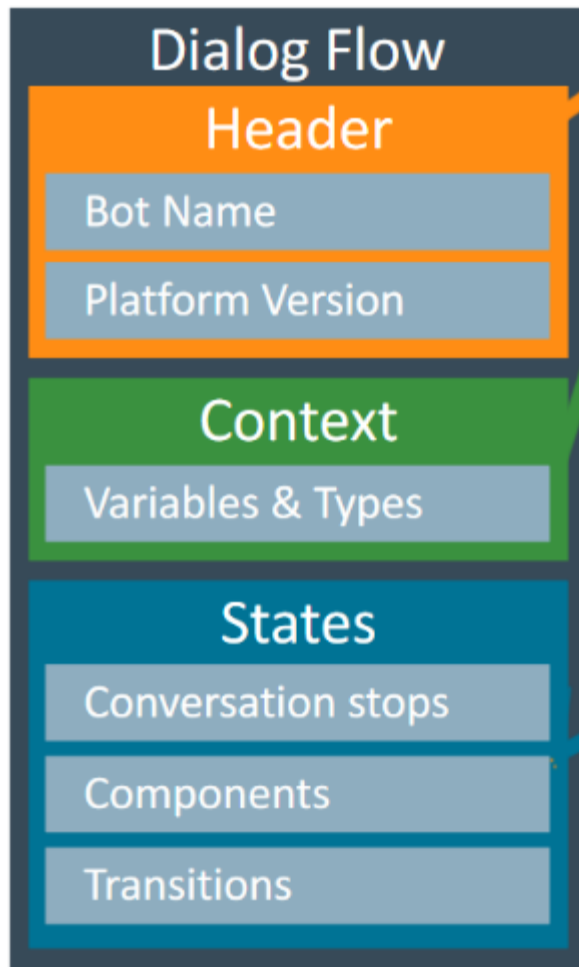
DIALOG FLOW



states

In this example there are three states.

We store the users input in a variable and then output it on the screen.



```
#-----  
# HEADER SECTION -----  
#-----  
metadata:  
  platformVersion: "1.0"  
  main: true  
  name: EchoSample  
  
#-----  
# CONTEXT SECTION -----  
#-----  
context:  
  variables:  
    greeting: "string"  
    name: "string"  
  
#-----  
# STATES SECTION -----  
#-----  
states:  
  
  askGreeting:  
    component: "System.List"  
    properties:  
      options: "Hello!, Ola!, Vannakam!, Namaste!"  
      prompt: "Hi there! What would you like me to echo back?"  
      variable: "greeting"  
  
  askName:  
    component: "System.Text"  
    properties:  
      prompt: "What is your name?"  
      variable: "name"  
  
  start:  
    component: "System.Output"  
    properties:  
      text: "${greeting.value} ${name.value}"  
    transitions:  
      return: "done"
```

COMPONENTS



```
#-----  
# STATES SECTION -----  
#-----
```

states:

```
askGreeting:  
  component: "System.List"  
  properties:  
    options: "Hello!, Ola!, Vannakam!, Namaste!"  
    prompt: "Hi there! What would you like me to echo back?"  
    variable: "greeting"
```

1

```
askName:  
  component: "System.Text"  
  properties:  
    prompt: "What is your name?"  
    variable: "name"
```

2

```
start:  
  component: "System.Output"  
  properties:  
    text: "${greeting.value} ${name.value}"  
  transitions:  
    return: "done"
```

3

Hi there! What would you like me to echo back?

Hello!

Ola!

Vannakam!

Namaste!



Ola!

What is your name?

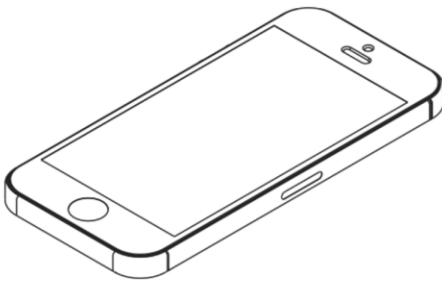


Stefan

Ola! Stefan

THREE INTERACTIVE QUESTIONS

Go to **www.menti.com**



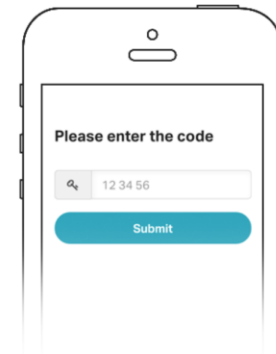
1

Grab your phone

www.menti.com|

2

Go to **www.menti.com**



3

Enter the code **85 24 20** and vote!

PASSWORD RESET CHATBOT

FLOW OF THE CLICK-THROUGH

1

Intent creation

Recognize greetings and
Password reset Intents

2

Use Google Translate API

Let google do the translation work for you
in any language

Integration

3

Add a skill summary

So people know what your bot can do
and can't do

4

Check if student exists

Using Query Webservice in Campus

Integration

5

Validating student

Ask his/her address or national id

6

Use Google Captcha

To prevent bot attacks

Integration

7

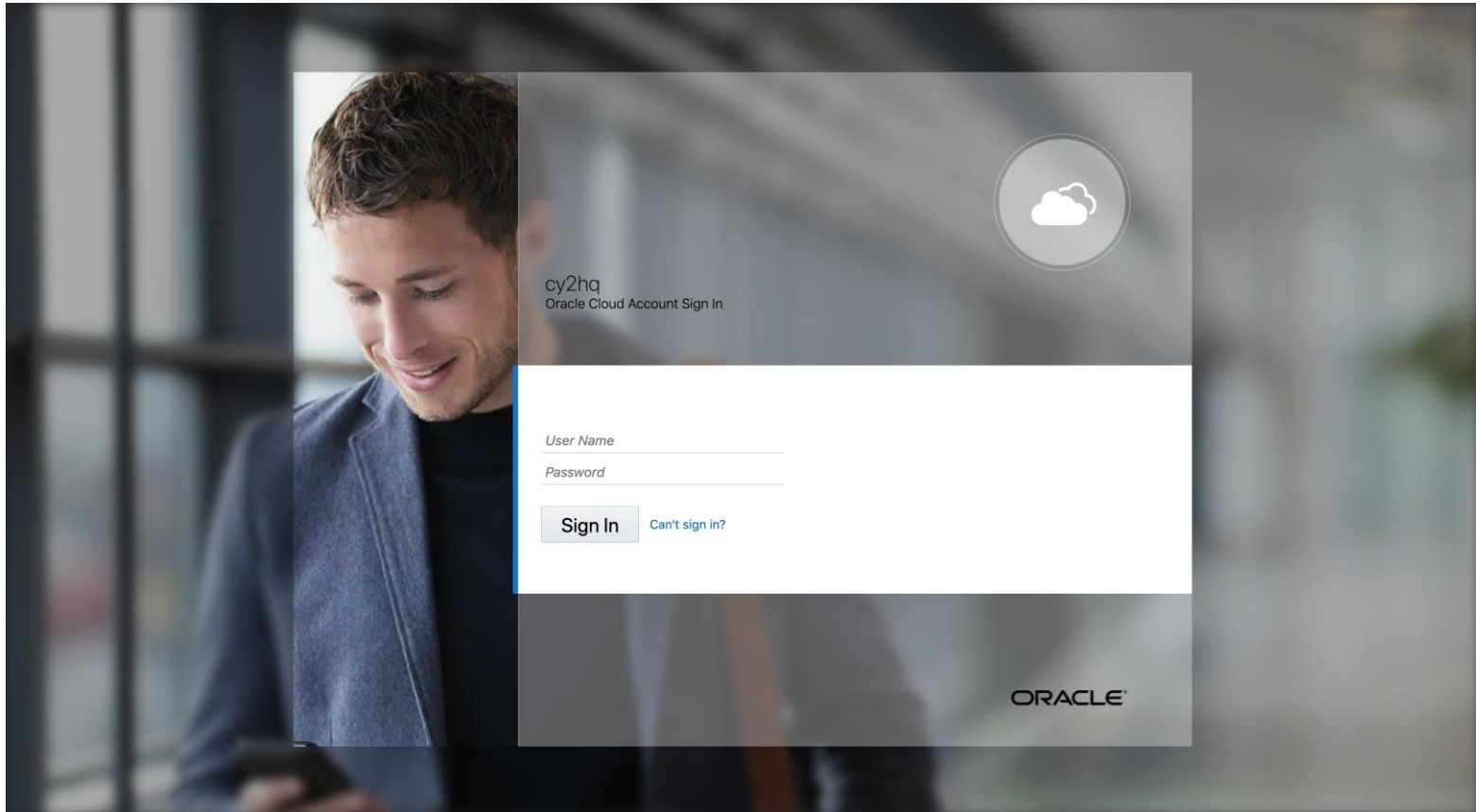
Close conversation

Using random responses



BUILD THE BOT FROM SCRATCH

LIVE DEMO



BOTS CLIENT SDK (NODE.JS)

TO CREATE YOUR CUSTOM COMPONENTS

1. Node.js is a run-time environment that executes **JavaScript** code outside of a browser.



2. Node.js is **Fast** and **Highly scalable**

Netflix, Linkedin, Trello, Uber, PayPal, Medium, eBay

3. **Conclusion:** We need to know a bit of JavaScript to make our customizations in Oracle digital assistant.

GENERATE CODE TEMPLATE

OUR FIRST CUSTOM COMPONENT

1. Prerequisite: installed **node.js** and **npm**

<https://www.taniarascia.com/how-to-install-and-use-node-js-and-npm-mac-and-windows/>

<https://nodejs.org/en/download/package-manager/>

<https://oracle.github.io/bots-node-sdk/>

Create the Custom Component

Use the Bots Node.js SDK CLI to generate a skeleton framework and template files, then add your custom code to it.

1. Install the Bots Node.js SDK:

```
npm install -g @oracle/bots-node-sdk
```

2. Create a directory on your development system and `cd` into it.

3. Generate the starter template files:

```
bots-node-sdk init cy2.campusServices --component-name cy2.helloWorld
```

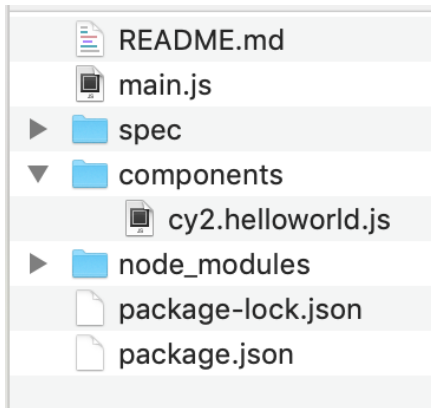
4. Modify the template files to add your custom code.

5. Run the following command to create a package:

```
npm pack
```

GENERATE CODE TEMPLATE

OUR FIRST CUSTOM COMPONENT



```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'cy2.helloworld',
    properties: {
      human: { required: true, type: 'string' },
    },
    supportedActions: ['weekday', 'weekend']
  }),
  invoke: (conversation, done) => {
    // perform conversation tasks.
    const { human } = conversation.properties();
    // determine date
    const now = new Date();
    const dayOfWeek = now.toLocaleDateString('en-US', { weekday: 'long' });
    const isWeekend = [0, 6].indexOf(now.getDay()) > -1;
    // reply
    conversation
      .reply(`Hello world ${human}`)
      .reply(`Today is ${now.toLocaleDateString()}, a ${dayOfWeek}`)
      .transition(isWeekend ? 'weekend' : 'weekday');
    done();
  }
};
```

GENERATE CODE TEMPLATE

OUR FIRST CUSTOM COMPONENT

```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'cy2.helloworld',
    properties: {
      human: { required: true, type: 'string' },
    },
    supportedActions: ['weekday', 'weekend']
  }),
  invoke: (conversation, done) => {
    // perform conversation tasks.
    const { human } = conversation.properties();
    // determine date
    const now = new Date();
    const dayOfWeek = now.toLocaleDateString('en-US', { weekday: 'long' });
    const isWeekend = [0, 6].indexOf(now.getDay()) > -1;
    // reply
    conversation
      .reply(`Hello world ${human}`)
      .reply(`Today is ${now.toLocaleDateString()}, a ${dayOfWeek}`)
      .transition(isWeekend ? 'weekend' : 'weekday');
    done();
  }
};
```

helloWorld:
component: "cy2.helloworld"
properties:
 human: "Stefan"
transitions:
 actions:
 weekday: "weekday"
 weekend: "weekend"

Hello world Stefan

Today is 2019-10-17,
a Thursday

CUSTOM COMPONENT

TAKE THE HELLOWORLD EXAMPLE AND ADJUST IT

```
"use strict";

var request = require("request");

module.exports = {
  metadata: () => ({
    name: "cy2.validateEmplid",
    properties: {
      emplid: { required: true, type: "string" },
      result: { required: true, type: "string" }
    },
    supportedActions: ["success", "failure", "warning"]
  }),
  invoke: (conversation, done) => {
    // perform conversation tasks.

    const { emplid } = conversation.properties();
    const { result } = conversation.properties();

    conversation.transition("success");
    done();
  }
};
```

```
#-----
# Validate student -----
#-----
```

```
validateEmplid:
  component: "cy2.validateEmplid"
  properties:
    emplid: ${studentNumber}
    result: "result"
  transitions:
    actions:
      success: "studentFound"
      failure: "studentNotFound"
      warning: "connectionError"
```

```
studentFound:
  component: "System.Output"
  properties:
    text: "Student found!"
  transitions: {}
```


CUSTOM COMPONENT

RUN NPM PACK


It generates a compressed file (.tgz)

```
[steve:cy2.helloworld Steve$ npm pack
```

```
> my-custom-component@1.0.0 prepack /Users/Steve/Developments/oda/cy2.helloworld  
> npm run bots-node-sdk -- pack --dry-run
```

```
> my-custom-component@1.0.0 bots-node-sdk /Users/Steve/Developments/oda/cy2.helloworld  
> bots-node-sdk "pack" "--dry-run"
```

Component Package 'cy2.helloworld' is valid!

```
npm notice  my-custom-component@1.0.0  
npm notice === Tarball Contents ===  
npm notice 721B components/cy2.helloworld.js  
npm notice 60B main.js  
npm notice 511B package.json  
npm notice 2.3kB README.md  
npm notice === Tarball Details ===  
npm notice name: my-custom-component  
npm notice version: 1.0.0  
npm notice filename: my-custom-component-1.0.0.tgz  
npm notice package size: 1.8 kB  
npm notice unpacked size: 3.6 kB  
npm notice shasum: 1d2a0a1363a3654c3a2b6655f320677af5f986ed  
npm notice integrity: sha512-0hMMdWpXX/qIf[...]31FYnWnfKhZ4Q==  
npm notice total files: 4  
my-custom-component-1.0.0.tgz
```

CUSTOM COMPONENT

BASIC STRUCTURE WITH METADATA AND INVOKE

```
#-----  
# Validate student -----  
#-----
```

```
validateEmplid:  
  component: "cy2.validateEmplid"  
  properties:  
    emplid: ${studentNumber}  
    result: "result"  
  transitions:  
    actions:  
      success: "studentFound"  
      failure: "studentNotFound"  
      warning: "connectionError"
```

```
studentFound:  
  component: "System.Output"  
  properties:  
    text: "Student found!"  
  transitions: {}
```

I can reset your
password if you like



Please type in your
student number..

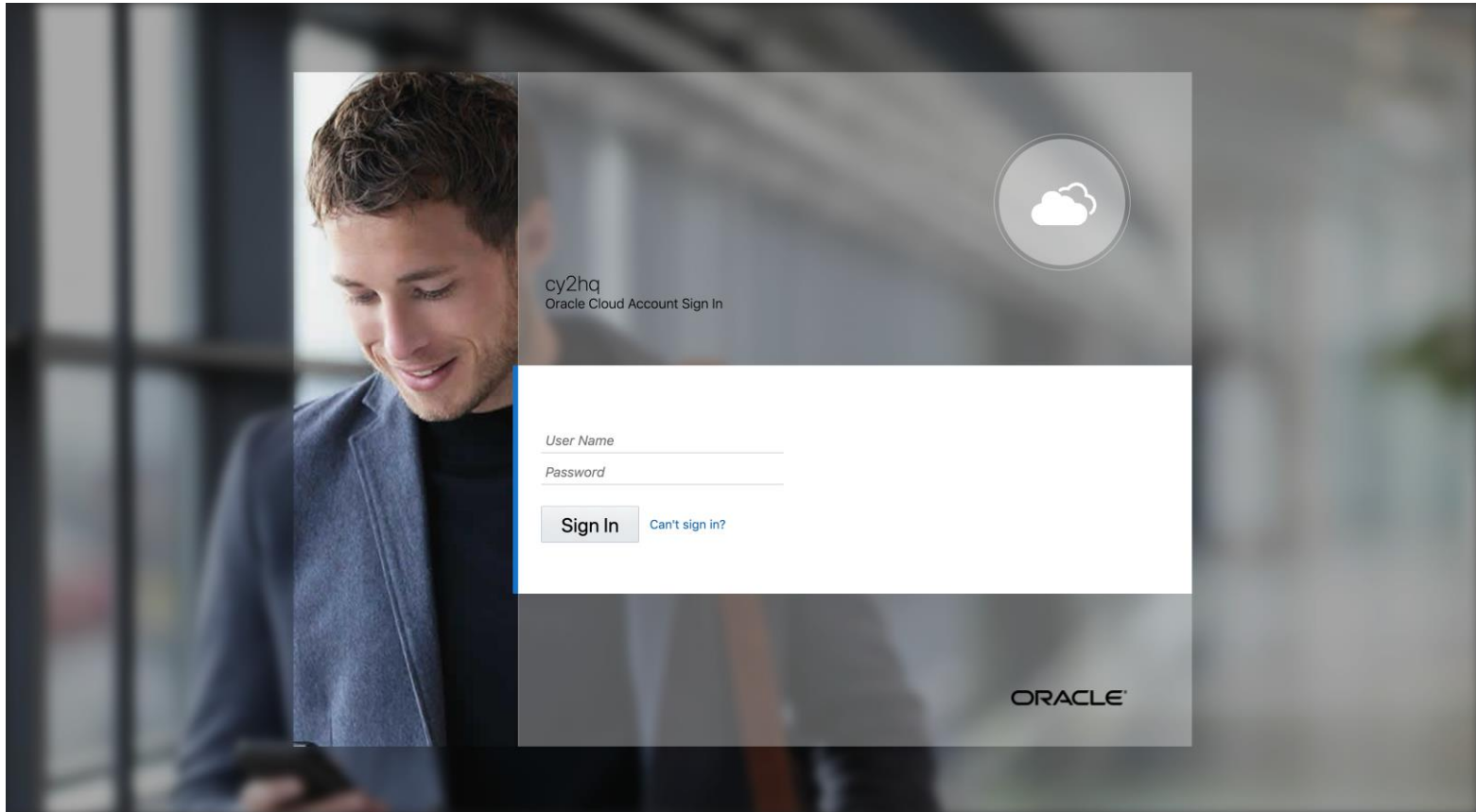


GA0001

Student Found!

USE CUSTOM COMPONENT

LIVE DEMO



PASSWORD RESET CHATBOT

FLOW OF THE CLICK-THROUGH

1

Intent creation

Recognize greetings and
Password reset intents

2

Use Google Translate API

Let Google do the translation work for you
in any language

Integration

3

Add a skill summary

So people know what your bot can do
and can't do

4

Check if student exists

Using Query Webservice in Campus

Integration

5

Validating student

Ask his/her address or national id

6

Use Google Captcha

To prevent bot attacks

Integration

7

Close conversation

Using random responses

GOOGLE'S RECAPTCHA

<https://developers.google.com/recaptcha/intro>

Home > Products > reCAPTCHA > Guides



Developer's Guide

Welcome to the reCAPTCHA developer documentation.

reCAPTCHA protects you against spam and other types of automated abuse. Here, we explain how to add reCAPTCHA to your site or application.

Audience

This documentation is designed for people familiar with HTML forms, server-side processing or mobile application development. To install reCAPTCHA, you will probably need to edit some code.

We hope you find this documentation easy to follow. Make sure to join the [reCAPTCHA developer forum](#) to give feedback and discuss the API.

You can find a reCAPTCHA codelab [here](#).

Overview

To start using reCAPTCHA, you need to [sign up for an API key pair](#) for your site. The key pair consists of a site key and secret key. The site key is used to invoke reCAPTCHA service on your site or mobile application. The secret key authorizes communication between your application backend and the reCAPTCHA server to [verify the user's response](#). The secret key needs to be kept safe for security purposes.

GOOGLE'S RECAPTCHA

READY-TO-GO COMPONENT



You are a human. Nice to meet you.

Note: Because a locally deployed webview is used, the article and the samples require Oracle Digital Assistant 19.1.5 or later.

READ FULL ARTICLE (PDF)

Sample Download

You can download the sample skill, components and sources from here (18 MB):

recaptcha.zip

Related Content

TechExchange - Tutorial: How-to Debug Custom Component Services Deployed to Oracle Digital Assistant Skill Bot Local Component Container

Oracle Bots Node.js SDK: Building Custom Component Services in and for Oracle Autonomous Mobile Cloud Made Easy

Very useful step-by-step instruction of how to install the reCaptcha in digital assistant.

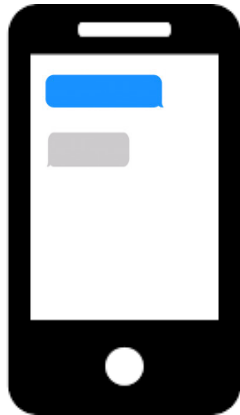
<https://blogs.oracle.com/mobile/techexchange%3a-integrating-google-recaptcha-in-oracle-digital-assistant-chatbot-conversations-using-systemwebview-and-a-custom-component>

GOOGLE'S RECAPTCHA

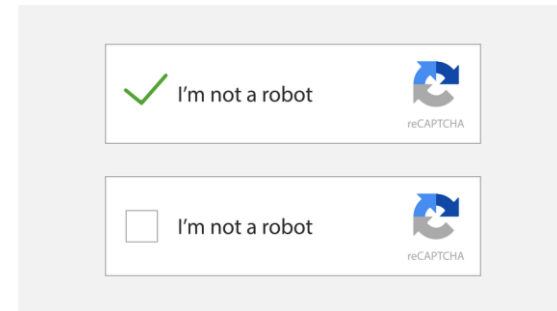
READY-TO-GO COMPONENT

Dialog flow

(custom components)



Webview (browser tab)

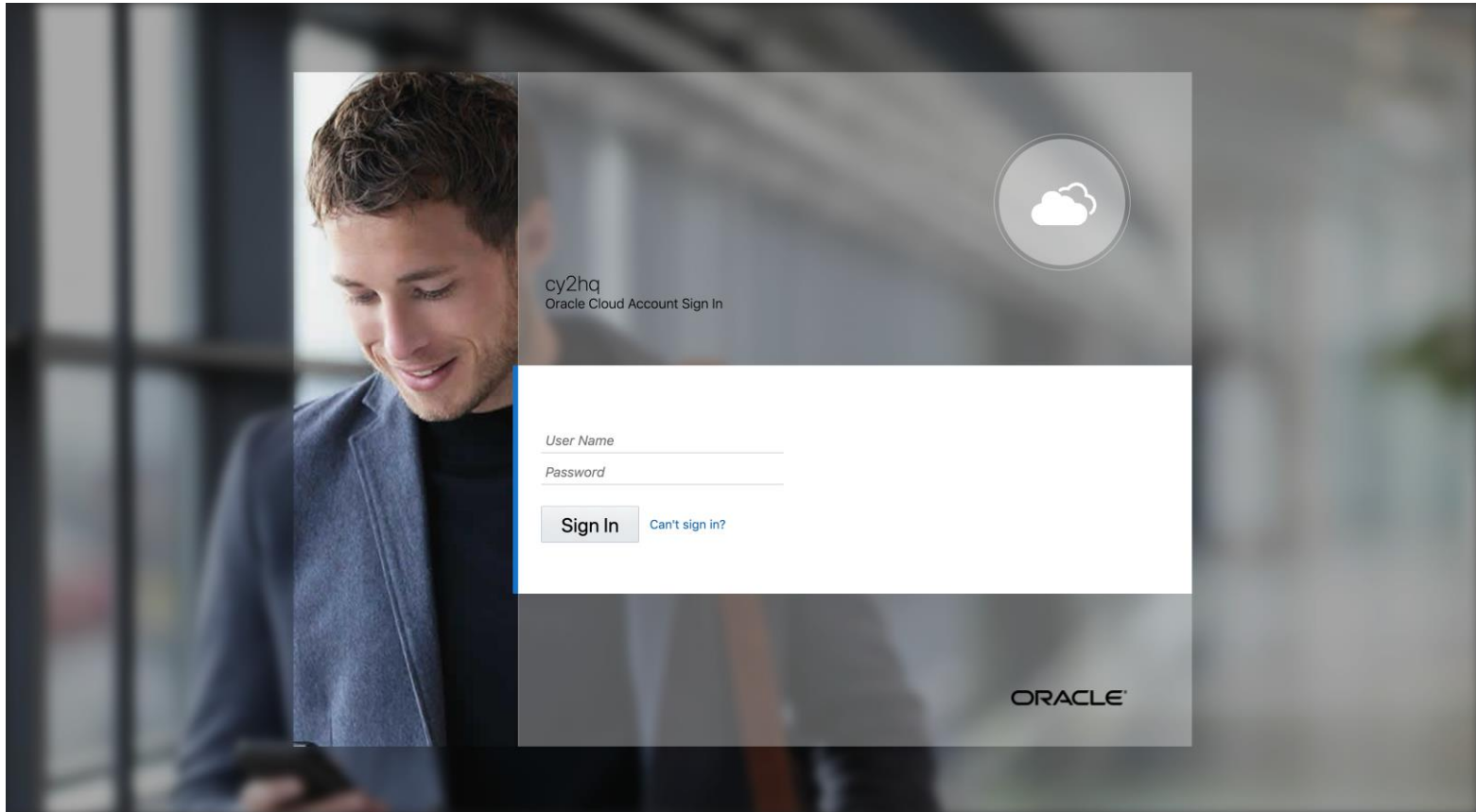


From the dialog flow we move to the webview

After we hit ***I'm not a robot*** we
move back to the dialog flow again

IMPLEMENT GOOGLE'S RECAPTCHA

LIVE DEMO



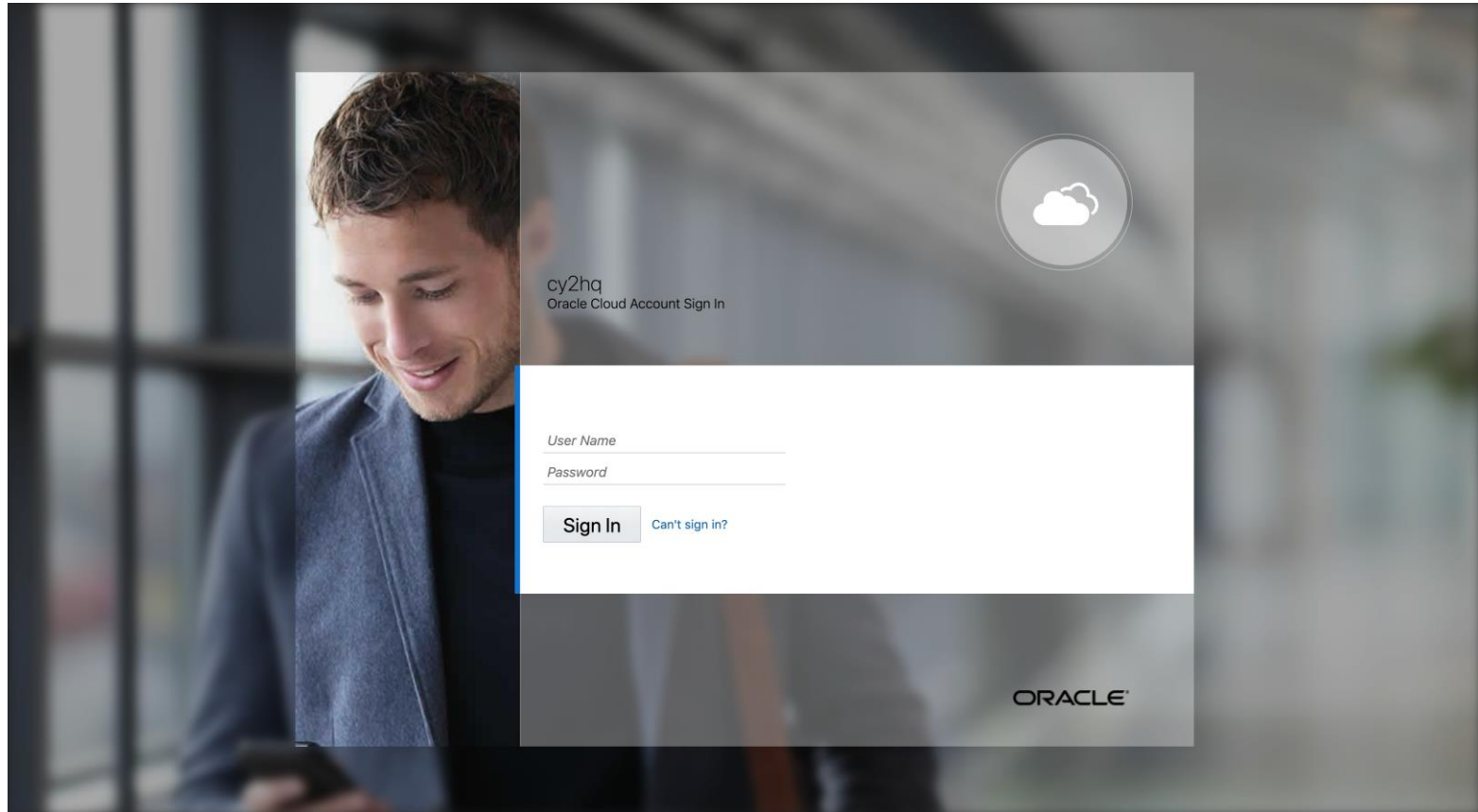
CONVERSATIONAL DESIGNER



1. Enables you to quickly build a functioning skill by writing a typical user-skill conversation.
2. You don't need to create intents, entities, or write a dialog flow definition. (**What?!**)
3. Typically used to sketch a quick dialog flow for demo purposes or basis for further development.

CONVERSATIONAL DESIGNER

LIVE DEMO



WHAT WE DIDN'T COVER

1. **Resource bundles**

Like Message catalogue, define your responses in one place

2. **Digital assistant**

A master bot, that routes between different skills

3. **Channels**

How we can deploy our bot to different channels like Web, facebook messenger, slack, ios, android, etc

4. **Analytics**

Optimize your bot based on insights

FURTHER READING

One-stop shop

<https://docs.oracle.com/en/cloud/paas/digital-assistant/index.html>

Contains:

- Tutorials
- Videos
- Documentation
- Tips and Tricks



THANK YOU!

PRESENTER



Stefan van Liempt

stefan@cy2.nl

**ALL ALLIANCE PRESENTATIONS WILL BE AVAILABLE FOR
DOWNLOAD FROM THE CONFERENCE SITE**