# Hackathon: Team Yosemite - Design Document

Finalized Idea: **Presentation Creator & Notes Extractor:**

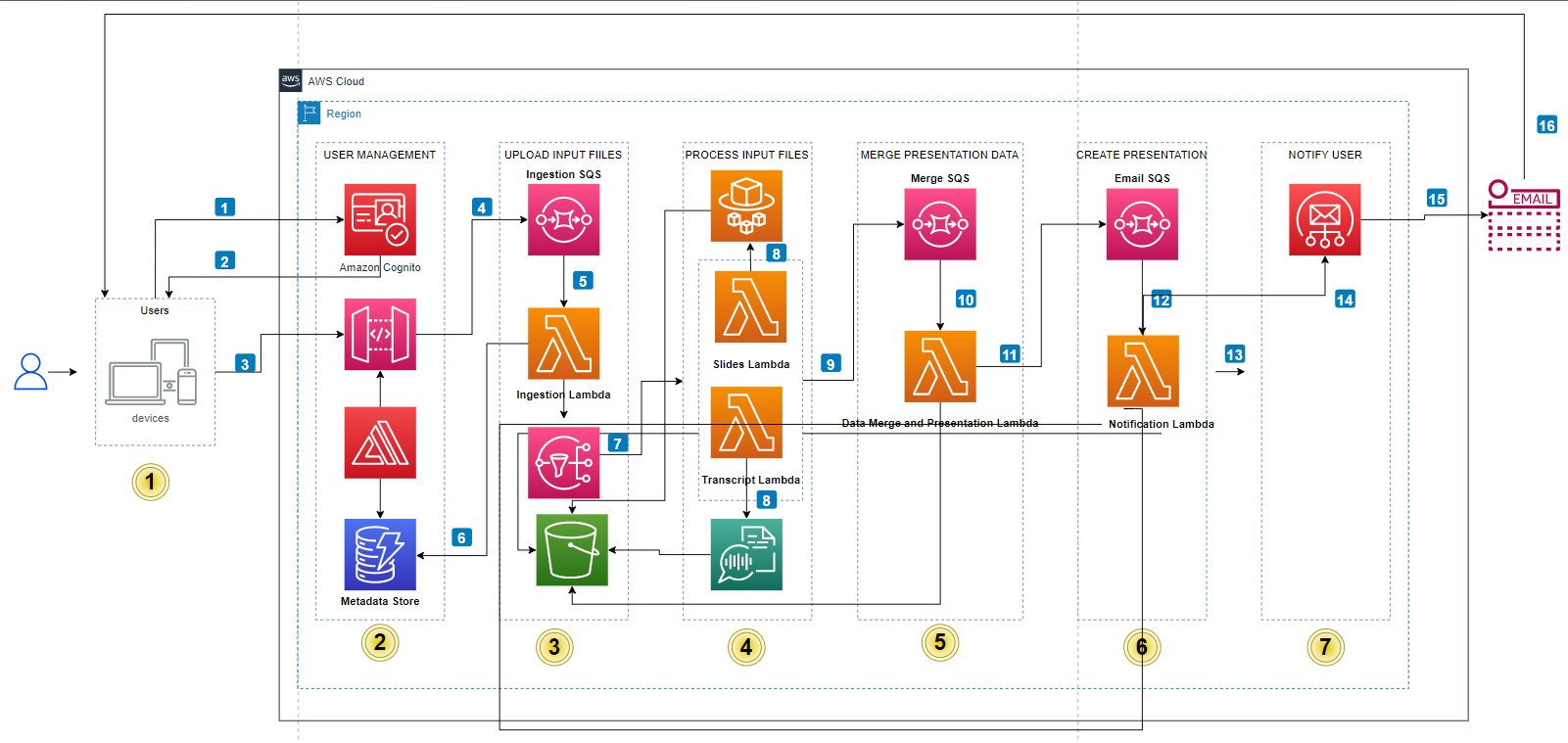
**Use case 1:**

As a member of an AWS field sales team, I  need to use a first call deck or other  presentation to deliver the same messaging as effectively as the original speaker. It is time-consuming to practice and leverage notes from original decks, videos and other channels to replicate the original messaging and content. The finalized idea from team Yosemite is to use a first call deck video file from re:invent/summit/broadcast or any video file and extract the audio, convert the audio to text and append it to the appropriate power point slide as notes for the field sales personnel to customize and use for their needs.

**Use case 2:**

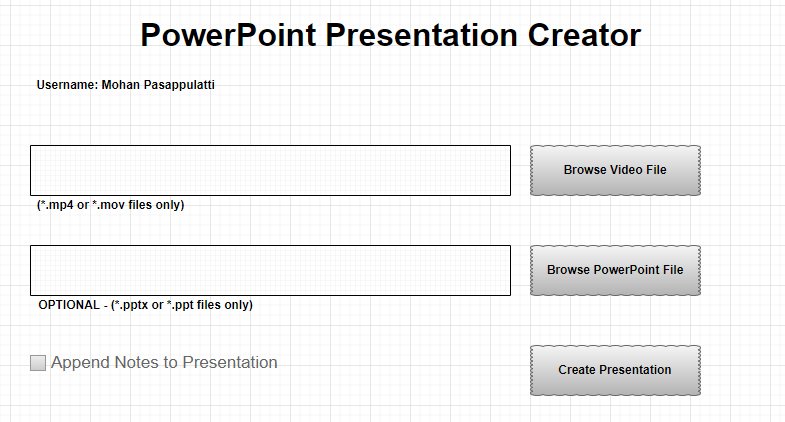
As a member of an AWS field sales team that is preparing a presentation to deliver to a customer peer or other audience, I would like to iterate and improve my messaging and content by automatically converting my speech to notes and insert it in to appropriate slides.

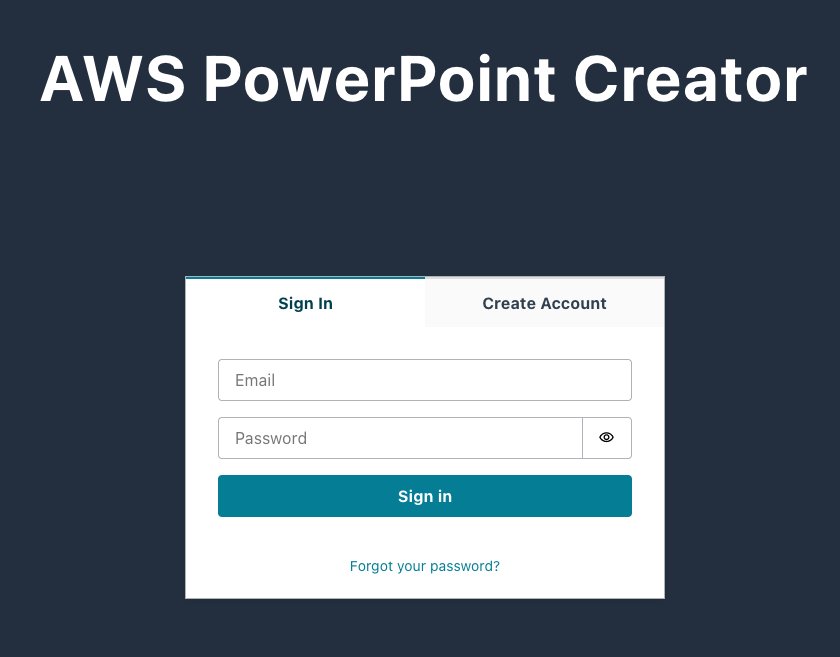
The architecture and data flow for the solution consists of 7 steps

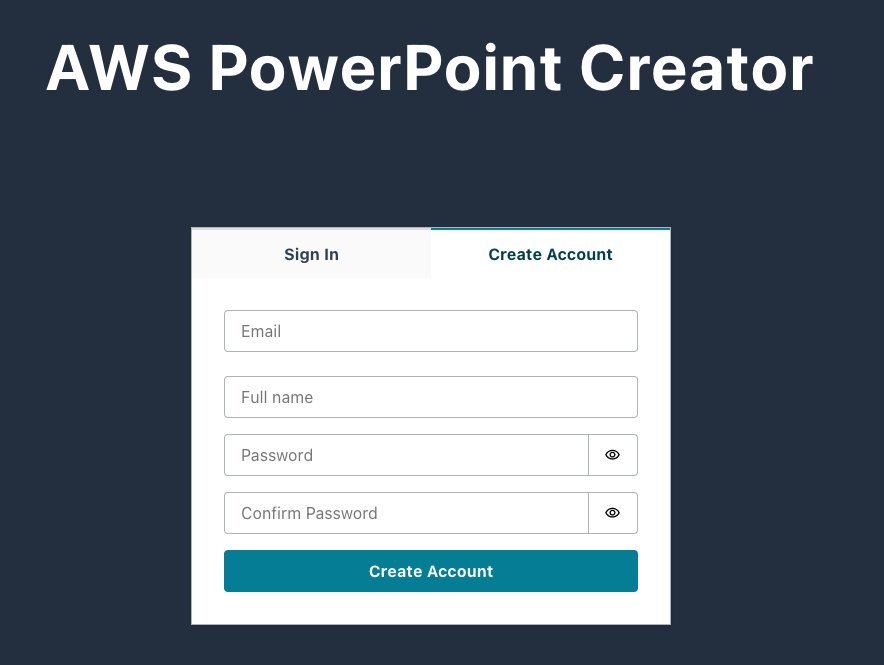


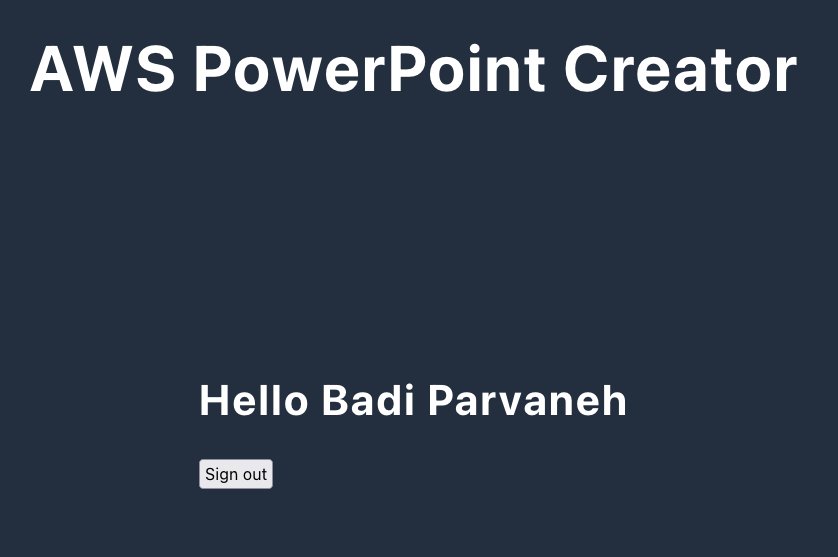
**Step 1:** Build a simple UX with option to  upload  1/Video and 2/PowerPoint (Optional) 3/Check the box if you want notes to be append and not over-written.

***Complexity: Low***









**Step 2 and 3:**

Build a simple user management and authentication mechanism using Amazon Cognito.

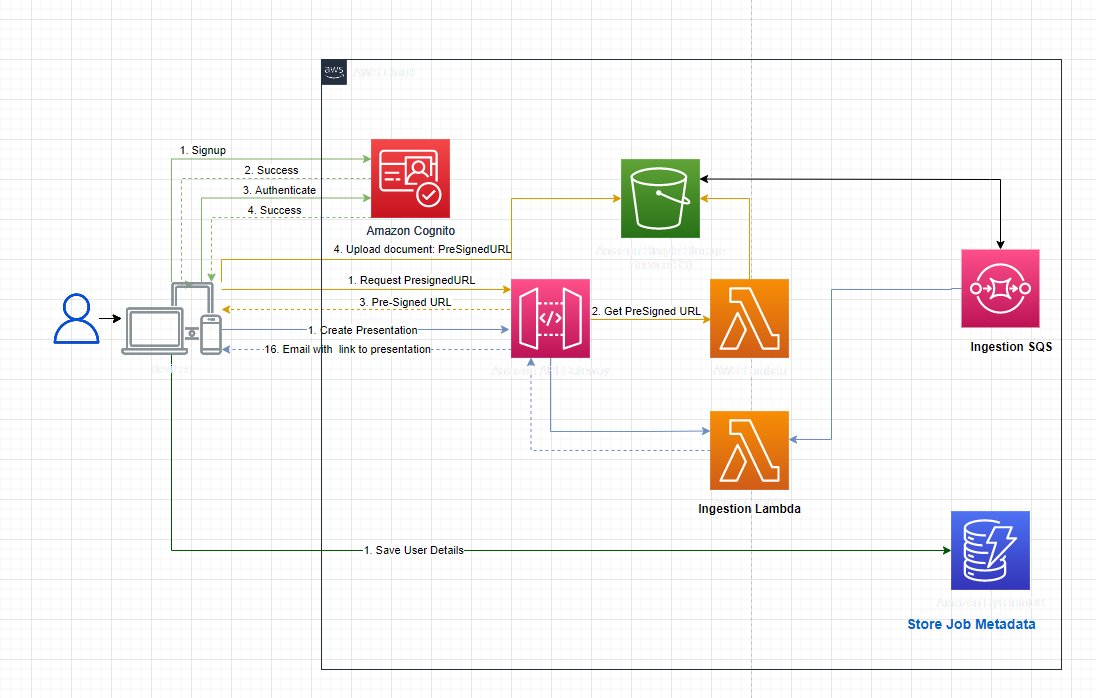
<https://cloudcompiled.com/tutorials/aws-cognito-hosted-ui/>

We will be using just email to signup as a new user. Get the first name & last name for the user.

Once user selects the file and submits the same for upload, get a pre-signed URL to upload and store the file in S3 and store the file location with some metadata in DynamoDB.

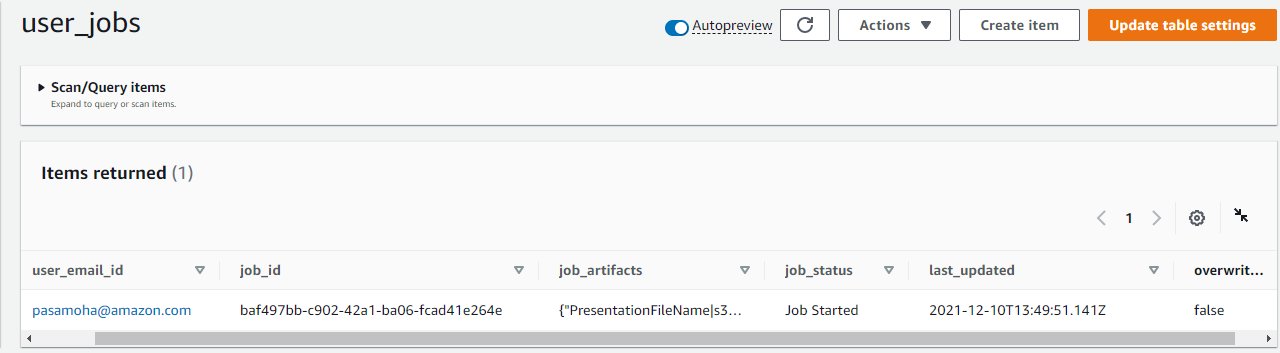
The file should go

***Complexity: Medium***



**Job Metadata stored in DynamoDB:**

{  
  "user\_email\_id": "pasamoha@amazon.com",  
  "job\_id": "baf497bb-c902-42a1-ba06-fcad41e264e",  
  "last\_updated": "2022-03-30 13:44:08.843439",  
  "overwrite\_notes": false,  
  "update\_counter": 0,  
  "VideoFilename": "data/users/pasamoha@amazon.com/baf497bb-c902-42a1-ba06-fcad41e264e/my-presentation.mp4",  
  "BucketName": "sagemaker-us-west-2-653878369704",  
  "job\_status": "Job Created",  
  "PresentationFileName": "data/users/pasamoha@amazon.com/baf497bb-c902-42a1-ba06-fcad41e264e/my-presentation.pptx",  
  "job\_artifacts": {  
    "PresentationTemplateFilename": "data/templates/presentationtemplate.pptx",  
    "SlideTranscriptionsFilename": "data/users/pasamoha@amazon.com/baf497bb-c902-42a1-ba06-fcad41e264e/slidetranscriptions.json",  
    "TranscriptionsFilename": "data/users/pasamoha@amazon.com/baf497bb-c902-42a1-ba06-fcad41e264e/transcriptions.json",  
    "SlidesPrefix": "data/users/pasamoha@amazon.com/baf497bb-c902-42a1-ba06-fcad41e264e/slides",  
    "SlideTransitionsFilename": "data/users/pasamoha@amazon.com/baf497bb-c902-42a1-ba06-fcad41e264e/slidetransitions.json"  
  }  
}



**App Metadata stored in DynamoDB:**

{  
 "application\_id": "1db38593-e6d5-4657-8e09-dc71b8df3d58",  
 "SendEmailQueueName": "ugh-pca\_Send\_Email",  
 "BucketPrefix": "data",  
 "TranscriptionsFilename": "transcriptions.json",  
 "PresentationTemplateFilename": "presentationtemplate.pptx",  
 "CreatePresentationQueueName": "ugh-pca\_Create\_Presentation",  
 "application\_name": "unicorn-gym-hackathon-presentation-creator-app",  
 "TemplatesPrefix": "|BucketPrefix/templates",  
 "application\_code": "uga\_pca",  
 "SlideTransitionsFilename": "slidetransitions.json",  
 "SlideTranscriptionsFilename": "slidetranscriptions.json",  
 "SlidesPrefix": "|JobPrefix/slides",  
 "BucketName": "unicorn-gym-hackathon-presentation-creator-app",  
 "JobPrefix": "|BucketPrefix/users/|user\_email\_id/|job\_id"  
}

**NOTE:**

Use the user **email/generated GUID** as the prefix in the S3 bucket. All the other metadata is saved to DynamoDB. The final JSON payload that is put in the SQS is as shown below:

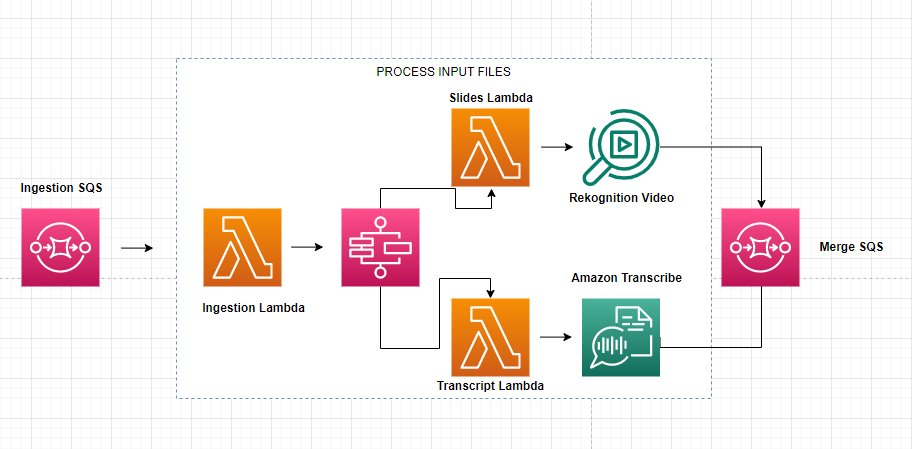
{  
 "user\_email\_id": "pasamoha@amazon.com",  
 "job\_id": "baf497bb-c902-42a1-ba06-fcad41e264e"  
}

**Step 4:**

Process: For each uploaded file, leverage step-functions and process the same for Elemental(Audio Extraction) and Transcribe (Audio to text), need to explore using Rekognition video segmentation API or other image processing python libraries.

Output: Save the audio file and the transcribed text in a location. Update the file metadata with the location of the audio and transcribed text. ***NOTE: Marking the slide transition is key.***

***Complexity***: **High**



**NOTE:**

The final JSON payload that is put in the SQS is as shown below:

{  
 "user\_email\_id": "pasamoha@amazon.com",  
 "job\_id": "baf497bb-c902-42a1-ba06-fcad41e264e"  
}

**TranscriptionTextFile Sample:**

{  
    "items": [  
        {  
            "start\_time": "0.04",  
            "end\_time": "1.08",  
            "alternatives": [  
                {  
                "confidence": "1.0",  
                "content": "slide"  
                }  
            ],  
            "type": "pronunciation"  
        },  
        {  
            "start\_time": "1.08",  
            "end\_time": "1.28",  
            "alternatives": [  
                {  
                "confidence": "1.0",  
                "content": "number"  
                }  
            ],  
            "type": "pronunciation"  
        },  
        {  
            "start\_time": "1.28",  
            "end\_time": "1.4",  
            "alternatives": [  
                {  
                "confidence": "1.0",  
                "content": "one"  
                }  
            ],  
            "type": "pronunciation"  
        }  
    ]  
}

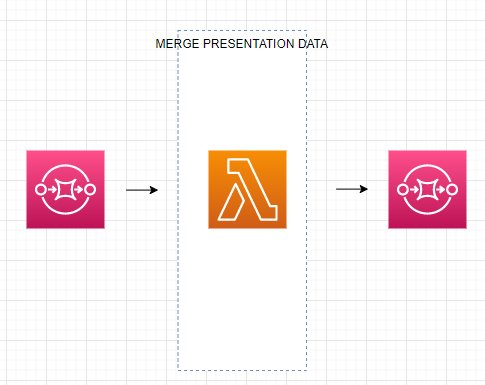
**SlideTransitionFile Sample:**

{  
    "slide\_prefix" : "slide-",  
    "slide\_image\_mime\_type" : "png",  
    "Slides": [  
        {  
            "slide\_number": 1,  
            "start\_time": "0",  
            "end\_time": "9.62"  
        },  
        {  
            "slide\_number": 2,  
            "start\_time": "10.19",  
            "end\_time": "20.86"  
        },  
        {  
            "slide\_number": 3,  
            "start\_time": "21.51",  
            "end\_time": "37.52"  
        }  
    ]  
}

**Step 5:**

Merge the PowerPoint **SlideTransitionFile** slides information with the notes from the transcribed results in **TranscriptionTextFile** based on the timestamp data.

***Complexity: Medium***



The merged output sample is shown below.

{  
  "slides": [  
    {  
      "slidenumber": 1,  
      "slideimage": "slide-1.png",  
      "transcript": " This is The slide number one."  
    },  
    {  
      "slidenumber": 2,  
      "slideimage": "slide-2.png",  
      "transcript": "This is slide number two."  
    },  
    {  
      "slidenumber": 3,  
      "slideimage": "slide-3.png",  
      "transcript": "This is a slide number three."  
    }  
  ]  
}

**Step 6 and 7:**

Create or update the PowerPoint with notes from Step5. If over-write is false then just append to the notes section, but include a timestamp to identify the data/time of updating the notes.

Created a pre-signed URL and send an email to the end-user with a link to the pre-signed URL to download the presentation.

***Complexity: Medium***

