**P0: Opening**

Next, let me share technical solution part for Digital Sales Coach, I will focus on high level part, EA team will introduce some detail solution related to Al.

In this part, we will cover 4 sub-topics, the 1st is DSC High Level Solution, the 2nd is DSC Application Architecture, the 3rd is DSC Integration Architecture, the last one is DSC Data Architecture.

Let’s move to the 1st one.

**P1: DSC High Level Solution**

For DSC application, we planned to develop it with popular web framework React, it can be built as SDK to integrate with existing mobile or web application, like iSmart.

* These red flags indicate data ingestion

1. About content source, we provide 2 approaches to ingest data.

The 1st one is automatically ingesting content from external system per ETL, after filtering, extracting, cleaning, chunking, and embedding, store into vector database finally.

1. The 2nd , we can manually manage above documents per admin portal.

* The yellow part is core RAG process.

1. App access exposed APls provided from backend Knowledge Service, Knowledge Service embedded

user question, and query knowledge chunk related to user's question to generate the most optimal prompt and send to Azure Open Al Service, Open Al will generate the answer.

* The green part is Avatar function.

1. User access exposed APls provided from backend Avatar Service, according to input, voice config and

avatar config, Avatar Service call APls from Azure TTS Avatar Service to generated real time video,

1. Admin can configure user/role permission, generating avatar rule, manage version, generating various report, etc.

**P2: DSC Application Architecture**

This is DSC application architecture diagram, it is a popular three-tier implementation, consists of a separated presentation tier, backend tier, and data tier.

1. Like what we said before, DSC application is developed with React web framework. can be built as an

SDK, and be integrated into existing mobile, like iSmart. It contains these function modules, like product learning, sales strategy learning, etc.

1. Admin portal web application to manage use, manage content, review learning report, etc.
2. We use Azure APIM to manage and protect our exposed APls, of course we can use HK existing software

AG webMethods to do this.

1. Avatar Service based on python tech stack, is responsible to generating video according to input, voice

config and avatar config.

1. Knowledge Service also based on python tech stack, is responsible to process content, communicate

with embedding module, vector database and azure open Al, etc.

1. Admin portal backend service. provide system administrate capabilities combine frontend application.
2. For storage, introduce Azure SQL Server to store structural data, vector database to store knowledge

base, azure blob storage to store orinigal content, like video, documents, etc., and azure cache Redis to store some hot data for improving performance.

1. Using Okta to authenticate DSC users.

**P3: DSC Integration Architecture**

This diagram is integration architecture, it consists of the applications, backend services, networks, security, data, azure cloud services, and other dependent components that support the whole DSC application environment, we use azure cloud to deploy DSC application in this design.

1. This is existing application that integrate with DSC SDK

Because all users are from internet, so we need to access backend APls per external API Gateway.

1. This is Admin portal application, because it is an internal admin tool, only access from intranet, it can

directly access admin portal backend service.

1. Sales Coach namespace in AKS, hosts Avatar service, Knowledge service and Integration service.

All services in this namespace need to support user access from internet.

1. Admin Portal namespace in AKS, hosts admin portal service and admin portal web application.

All services in this namespace need to support user access from intranet.

1. These are dependent database, blob storage, vector database and Redis.
2. These are other external Azure Al dependencies, so far, we plan to use
   * Azure Al Translator is responsible to translating text across Traditional Chinese, Simplified Chinese and English language.
   * Azure Al Speech is responsible to converting text to speech, speech to text, building Al avatar, etc.
   * Azure Al Document is responsible to extracting text, key-value pairs, tables, and structures from documents automatically and accurately.
   * Azure Open Al Service is responsible to providing REST APls access to OpenAl's powerful language models including GPT-4o, GPT-4 Turbo with Vision, GPT-4, GPT-3.5-Turbo, and Embeddings model series.
3. Here, you can see other existing service that need to integrate with, we have many trainings and learning

materials in iAcademy, maybe need to integrate it firstly.

1. For Monitoring and observability, we use these azure existing services to monitor and observe running

application and services on cloud.

**P4: DSC Data Architecture**

This diagram is Data architecture, it consists of the data source, data ingestion, data storing, data processing, and data consuming.

* The left part is source content related to Al, the right part is traditional user tracking and analytics.

1. We can automatically ingest content from iAcademy or other system per ETL.
2. Manually update content, manage DSC user feedback content.
3. Save this content into azure blob storage.
4. Process the content, like filter some useless resource, extract text from video/audio/documents, etc.
5. Save embedding data into vector database (can use Azure Al Search/Elastic Search/PostgreSQL plus

Pgvector plugin)

1. Azure Open Al combining the user question and trained knowledge-base to answer user.

* This is tracking user operation behavior and issue diagnosis

1. Design and implement tracking user operation behavior and issue diagnosis
2. Save these user data into SQL Server database.
3. Analyzing user data.
4. Build user operation/behavior report using Power Bl, continuous improvement and enhanced user

experience.