God Bless You

Group members: 陳昱翔、薛耀智、張高睿

Motivation and Objectives

When faced with difficulties, many people seek spiritual support or guidance. In Taiwan, practices like temple visits or fortune-telling are common but often face three issues:

- Lack of instant responses
- Limited consultation time
- Unsatisfactory consultation quality

To address these, we developed "God Bless You," integrating artificial intelligence (AI) and cloud technology with traditional fortune-telling. Our goal is to provide real-time facial analysis and fortune-telling advice for a convenient, instant, and exceptional user experience.

2. Related Work/Market Survey

Fortune-telling services have increasingly shifted to online platforms, offering users greater convenience and accessibility. However, existing solutions, like live-streaming on YouTube or template-based websites such as "明明觀止" and "算命王官網," often lack personalization and emotional depth. Photo-based platforms like Testharo, which allow facial interpretation, suffer from inconsistencies, further undermining user trust.

The market demand highlights the need for innovation. According to a 2021 survey by the Institute of Sociology at Academia Sinica, over two-thirds of Taiwanese express interest in spiritual guidance services, including 27.9% identifying with folk religions, 19.8% with Buddhism, and 18.7% with Taoism. Globally, the market for spiritual and mental wellness services generates tens of billions of dollars annually, with steady growth projected. For instance, the global mental health market reached $36.73 billion in 2023 and is expected to grow to $37.67 billion by 2028, reflecting rising demand.

"God Bless You" addresses these gaps by combining AI with traditional metaphysical knowledge to deliver real-time, personalized, and empathetic interpretations. Our service prioritizes accessibility and emotional support, setting it apart from

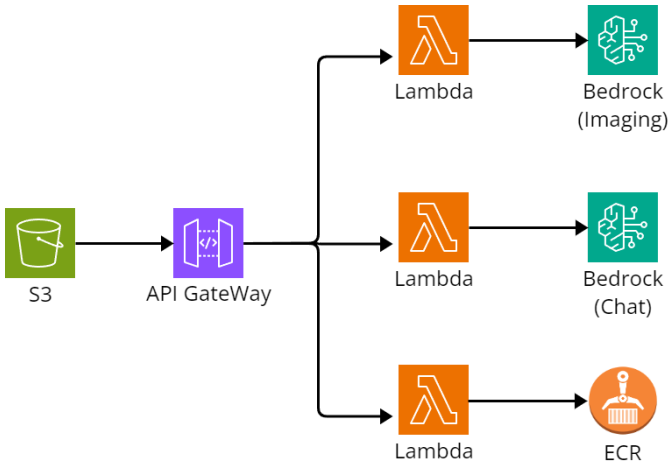
competitors. Target users include individuals seeking emotional support, religious believers, and curious explorers, representing a broad and diverse audience for innovative solutions.

3. Solution Overview

Introduction to the implemented application

- **Facial Interpretation**: Allows users to upload photos, which are analyzed to reveal personality traits and potential. This integrates AI with metaphysical principles.
- **Fate Consultation**: Provides personalized advice through an AI-powered chatbot using BaZi calculations and astrological insights.
- **Visualized Fortune**: Converts interpretation results into visually symbolic fortune charts.

System architecture diagram



Cloud Technologies

Our architecture uses AWS services for serverless, efficiency and scalability :

1. Website Front-End

   The web frontend consists of index.html, scripts.js, styles.css, and an images folder containing images. The deployment process involves first creating an S3 bucket, uploading all the aforementioned files into the bucket, and enabling the static website hosting service. This way, the S3 bucket automatically generates a website endpoint that redirects users to the index.html file within the bucket, allowing them to access the website.

The website we designed is primarily composed of the three features mentioned above: Facial Interpretation, Fate Consultation, and Visualized Fortune. All three features are implemented using Amazon Lambda functions. The frontend communicates with these features by calling the function URLs generated by Amazon Lambda for data transmission and reception. Additionally, we configured Cross-Origin Resource Sharing (CORS) policies in the S3 bucket to facilitate smooth HTTP POST calls to the target Lambda functions.

2. Facial Interpretation

We provide a facial interpretation service that extracts users' facial features and compares them with a database to find the most similar match. This service was built using AWS Elastic Container Registry (ECR), where we used AWS-provided image URIs as the base for our Docker container. On top of this base, we added our custom code to enable the container to perform the required functionalities once executed.

To extract facial features, we implemented the Scale-Invariant Feature Transform (SIFT) algorithm from the cv2 library and Google's Mediapipe framework . These algorithm identifies critical facial landmarks such as the eyes, nose, and mouth and converts them into feature arrays. We compare these feature arrays with those in our database, calculate the differences, and select the face with the smallest cumulative distance as the closest match. Due to the limited size of our photo database, we embedded the facial data directly into the Docker container for simplicity and efficiency.

To optimize cost and scalability, we integrated AWS Lambda in image execution mode. This approach enables a serverless "pay-as-you-go" model that mimics SageMaker’s capabilities while significantly reducing the cost of cloud computing resources.

3. Fate Consultation

We provide users with a chat interface on the website, allowing them to engage in conversations with a chatbot and ask questions related to their future fortune. The chatbot utilizes the Amazon Titan text model provided by Amazon Bedrock to respond to users and facilitate subsequent interactions.

The chat history between the user and the chatbot is transmitted from the frontend to a Lambda function. Within the Lambda function, the chat history

is passed to the Amazon Titan text model by calling its API. Once the model generates a response, the Lambda function transmits the response back to the web frontend, where it is displayed in the chat window.

4. Visualized Fortune

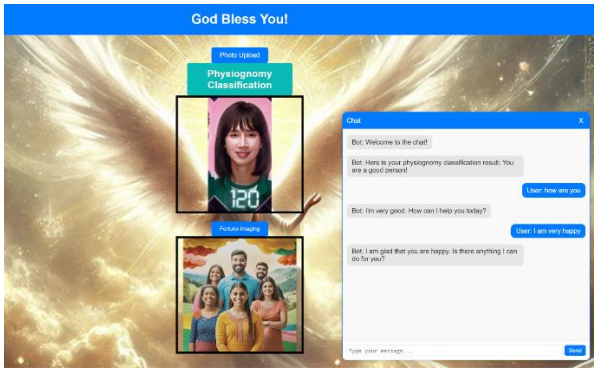
   When users utilize the Visualized Fortune feature, the system organizes the results from Facial Interpretation and the chat history from Fate Consultation to generate a personalized visual representation of their fortune.

   First, the frontend transmits the classification results and chat history to a Lambda function. Within the Lambda function, these two pieces of data are sent to the Amazon Titan text model provided by Amazon Bedrock via an API call. The model processes the data to summarize and generate a prompt suitable for image generation. Finally, the generated prompt is passed to the Amazon Titan Image Generator G1 model on Amazon Bedrock to create the image. The resulting image is then sent back to the frontend through the Lambda function and displayed on the website.

5. Deployment and Integration through code

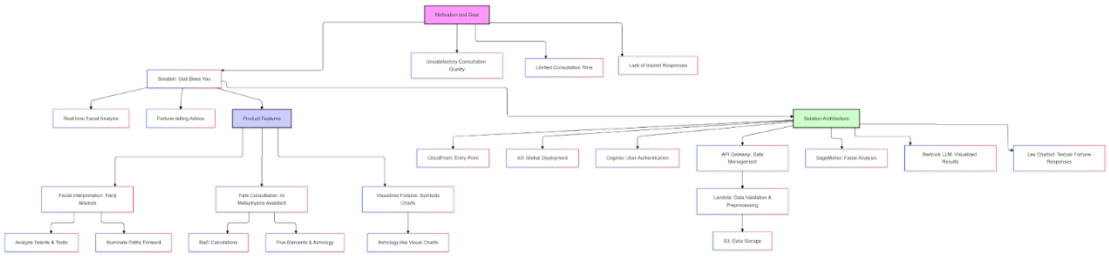
   To ensure smoother collaboration within our team, we documented all configurations and infrastructure settings directly in Python files using AWS CDK. This approach allows us to centralize and standardize our infrastructure definitions, making it easier for team members to understand, modify, and review changes. By using Python, a language familiar to most developers, we reduce the learning curve while enabling dynamic and programmatic infrastructure management. This not only improves team efficiency but also ensures that all changes are version-controlled, traceable, and easily replicable, fostering a collaborative and transparent development environment.

4. Project Outcomes



In our project implementation, users are welcomed to our platform with an engaging and interactive experience. Upon accessing the page, they can start by uploading a selfie. Our model then analyzes their facial features and classifies them as either a "good" or "bad" character based on predefined criteria. Following this, users can enter a chatroom to interact with a fortune teller, sharing their thoughts, questions, and uncertainties. The fortune teller uses the conversation, combined with the classification results, to generate a personalized fortune chart that visualizes the user's future prospects. This end-to-end process provides an entertaining and immersive journey for users, blending technology with creativity.

5. Mind Map



6. The Most Time-Consuming Part

The development process presented multiple challenges that required extensive time and effort to resolve. Two major issues consumed the most time: configuring access control across AWS services and determining the optimal serverless backend architecture for our application.

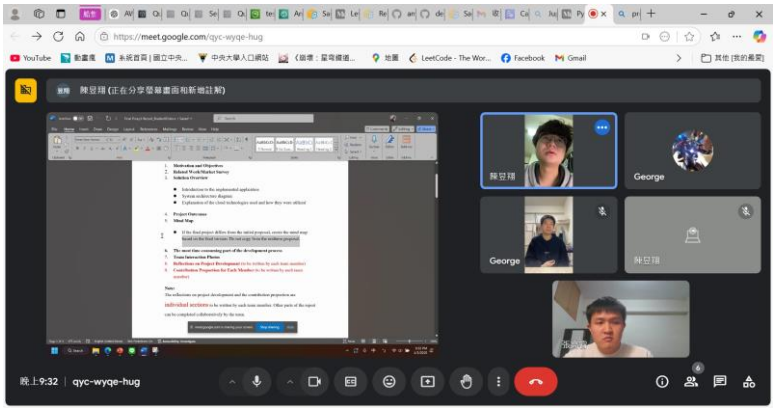
Firstly, configuring permissions between the web frontend and Lambda function required a deep understanding of CORS and IAM roles. Each service's permission settings had to align precisely. The Lambda function also needed additional permissions to interact with the Bedrock model APIs. These tasks were further complicated by unfamiliarity with Amazon's permission settings, resulting in significant delays.

Secondly, selecting the appropriate serverless architecture for the backend posed its own challenges. Initially, we experimented with using Lambda layers to manage facial similarity computations. However, Lambda's constraints—such as a decompressed layer size limit of 250MB and a pre-upload file size limit of 50MB—proved problematic. Libraries like NumPy exceeded these limits, making it impossible to deploy our code through simple environment partitioning.

Our next approach was SageMaker, but our unfamiliarity with AWS's image URIs led to repeated deployment failures. We initially believed that any AWS-provided image URI would suffice, only to encounter discrepancies between our runtime scripts and the container's architecture. These bugs were difficult to diagnose because error messages did not explicitly indicate mismatches. Furthermore, SageMaker's multi-step deployment process—model creation, endpoint configuration, and endpoint setup—added complexity. Each test run required over 10 minutes to reveal errors, leading to a cumulative debugging time of approximately 35 hours.

In summary, both challenges highlight the importance of deep familiarity with AWS's infrastructure and limitations. Addressing permission issues required careful coordination of roles and policies, while overcoming SageMaker deployment hurdles necessitated iterative experimentation and extensive error analysis. These experiences underscored the need for meticulous planning and thorough understanding of tools, ultimately enhancing our team's problem-solving skills and system design expertise.

7. Team interaction Picture



8. Reflections on Project Development

During the brainstorming phase, I am glad we chose an interesting topic, which made the development process much less stressful. Initially, we thought the project would progress smoothly as planned. However, once we started working on it, we realized that some features were not fully supported by AWS. This led to a lot of trial and error as we experimented with various configurations before finally achieving success.

I am also deeply grateful to my teammates. Even amidst the intense pressure of final exams, everyone supported each other and kept pushing forward. Together, we

managed to complete the entire project, which is an achievement we can all be proud of.

9. Team Members and Task Allocation (Indicate Contribution Proportions)

Name	Student ID	Job Description	Contribution Ratio
陳昱翔	111403525	Iac Document Demo	33%
薛耀智	111502026	Frontend Lambda Bedrock Document	33%
張高睿	111502517	Fortune Classification Lambda Document	33%