### *By Saulita Smith, AWS AI/ML Solution Architect/ML Data Engineer*

**Understanding Autism: A Unique Perspective**

Individuals with Autism Spectrum Disorder (ASD) experience life in ways that can be quite different from those who are neurotypical. Autism is a spectrum, meaning that the severity and impact on one’s ability to perceive the world can vary significantly from person to person.  
However, some struggles are more common among those with autism. For instance, prosopagnosia is a neurological condition that makes it difficult to recognise faces This condition can make it challenging for an autistic child to distinguish between different people, which can affect their ability to know who to trust, potentially jeopardizing their safety.  
But what if there were an app designed to help young individuals overcome this challenge? Below is the architecture of an application that aims to do just that.

Here's a detailed AWS architecture that allows an ASD user to log into an app, take a picture, label it as "family" or "friend," store it securely in S3 and automatically delete pictures of strangers if no permission is given within 30 seconds. No pictures of strangers are stored.

### Step-by-Step Workflow:

1. **User Authentication**:
   * The autistic user logs into the app using Amazon Cognito.
2. **Picture Upload**:
   * The user takes a picture using the mobile app.
   * The picture is uploaded to an Amazon S3 bucket managed by Amplify.
3. **Image Processing**:
   * An S3 event triggers an AWS Lambda function.
   * The Lambda function invokes Amazon Rekognition to analyse the image.
   * If the face is recognized as "family" or "friend," the user is prompted to confirm the label.
   * The user adds additional specifications if necessary.
4. **Storage in DynamoDB**:
   * The Lambda function stores the image metadata, including the label, in Amazon DynamoDB.
5. **Amazon Amplify:**
   * Retrieves the results and sends the response securely back to the user interface.

### AWS Architecture Diagram:

1. **Mobile App**:
   * User logs in via Amazon Cognito.
   * User takes a picture and uploads it via Amazon Amplify.
2. **Amazon Amplify**:
   * Facilitates the upload of the picture and manages storage.
   * Triggers an AWS Lambda function upon picture upload.
3. **AWS Lambda (Image Processing)**:
   * Invokes Amazon Rekognition to analyse the image.
   * Checks if the face is recognized (family or friend).
   * Prompts the user for confirmation and additional specifications.
4. **Amazon Rekognition**:
   * Analyses the image for face recognition.
   * Returns results to the Lambda function.
5. **AWS Lambda (DynamoDB Storage)**:
   * Stores the image metadata and labels in Amazon DynamoDB.
6. **Amazon DynamoDB**:
   * Stores metadata, labels, and additional specifications.
7. **Amazon Amplify:**
   * Retrieves the results and sends the response securely back to the user interface.

### Additional services:

1. **AWS CloudTrail**: CloudTrail is used to log all API calls and user activity across your AWS account. This includes activity from the API Gateway, Lambda functions, S3, Rekognition, and DynamoDB.
2. **AWS CloudWatch**: CloudWatch is used for monitoring and logging. It collects and tracks metrics, logs, and events from your Lambda functions, API Gateway, and other AWS services.

### Things to consider:

1. **API Gateway Placement**: The API Gateway remains outside the VPC. This setup allows it to be easily accessible by the user interface over the internet.
2. **Lambda Functions**: Lambda functions between S3 and Amazon Rekognition for preprocessing and invoking Rekognition, and between Amazon Rekognition and DynamoDB for handling the results.
3. **Data Flow**: Images being uploaded to S3, processed by Lambda, analysed by Rekognition, and results stored in DynamoDB and returned to the user interface via API Gateway.
4. **VPC Configuration**: Lambda functions have the appropriate VPC needed to access resources within the VPC. For example, if DynamoDB is in a private subnet, the Lambda function interacting with it will have the necessary VPC settings.
5. **Security**:
   * S3 bucket must have the appropriate permissions and policies to only allow access from authorized Lambda functions.
   * IAM roles used to give Lambda functions the necessary permissions to interact with S3, Rekognition, and DynamoDB.
   * API Gateway uses Cognito for user authentication, which is already in place as shown in the architecture.
6. **Error Handling**: Implementing error handling and logging within the Lambda functions to manage failures gracefully to ensure the debugging of any issues that may arise effectively.
7. **Cost Management**: Keeping an eye on the cost, especially with services like Rekognition and Lambda. Use AWS **Cost Explorer** to monitor and optimize expenses.
8. **Scalability**: This architecture is inherently scalable using AWS managed services, additionally I might consider implementing throttling in API Gateway to protect against spikes in traffic.

.