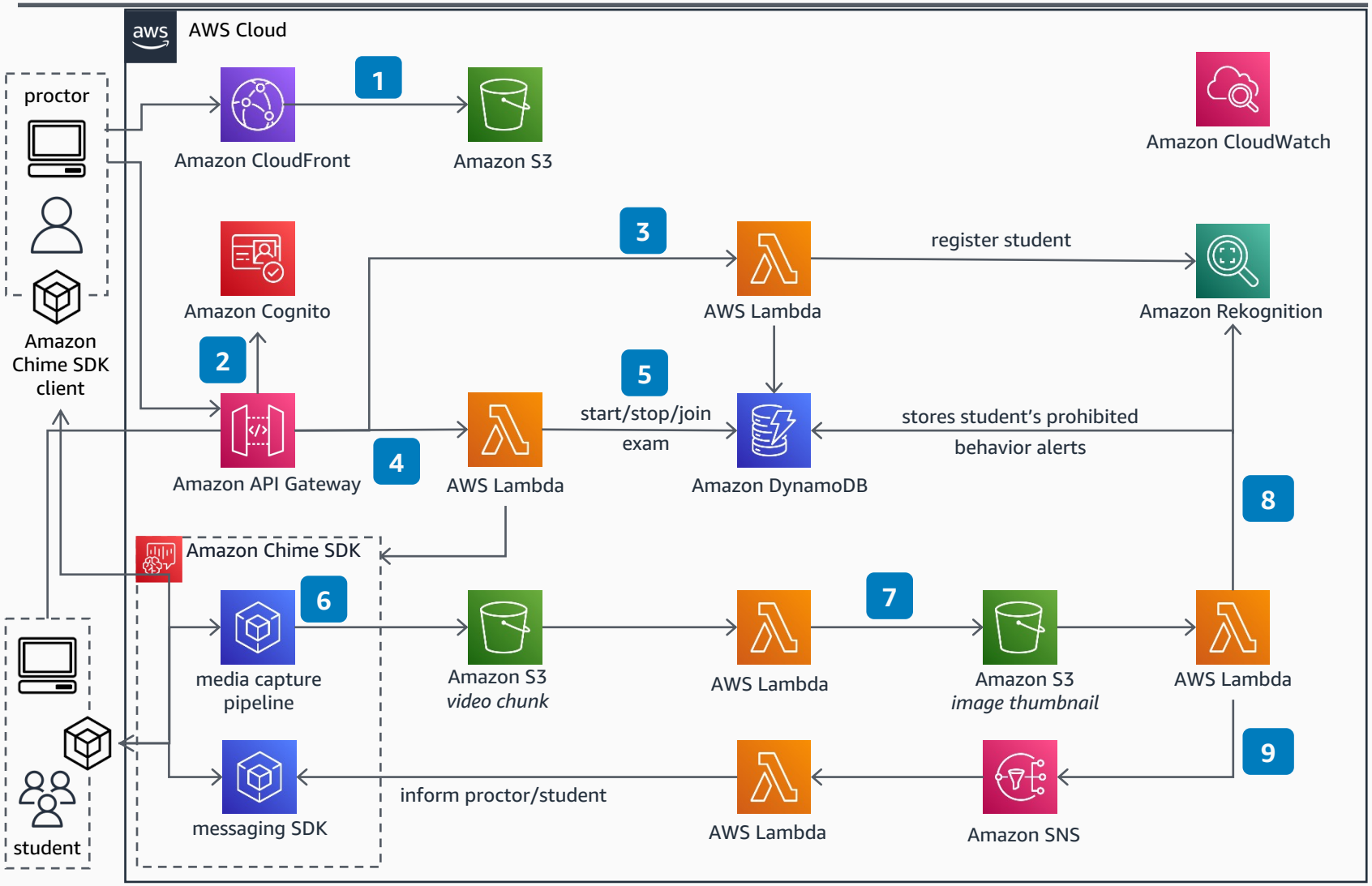
Hybrid Remote Proctoring Solution

Hybrid remote proctoring is for education use cases where students can be monitored live by a human with artificial intelligence (AI) assistance to highlight prohibited objects and behaviors. This solution can be used for in-person exams as well as remote exams.

Note: this is a technical diagram only and does not account for possible privacy implications, laws, and regulations that may apply to this scenario, such as just-in-time privacy notices, user consent, and data usage, retention, processing, and deletion of personally-identifiable data/information, including biometric data/information.



1. This architecture hosts static webapp code for the solution in **Amazon Simple Storage Service** (Amazon S3) bucket which is served behind **Amazon CloudFront**, providing low latency access to content while serving cached content from edge locations spread across the globe. Content in the **Amazon S3** bucket can be secured against unintended access using CloudFront Origin Access Control.

2. Server side REST APIs can be served behind **Amazon API Gateway**, using serverless **AWS Lambda** functions. **Amazon Cognito** can be used to provide secure authenticated access to these APIs and managed identities for proctors and students.

3. For registration flow, students are registered with an API request that passes the student's photo to an **AWS Lambda** function. This function internally calls an **Amazon Rekognition API** to get a unique FaceId based on extracted facial feature vectors. The student's FaceId, is then stored inside an **Amazon DynamoDB** serverless database for future matching needs, based on the customer's policy.

4. The proctor starts the exam by calling an **Amazon Chime SDK Messaging API**, which starts an **Amazon Chime SDK** session as well as an **Amazon Chime SDK Messaging Channel** for chat-based communication between the proctor and each student participating in the exam. The student's consent is captured at the start of the exam with relevant policies.

5. This information gets stored inside **Amazon DynamoDB**. To stop the exam, a similar request gets called which updates the exam status within the database.

6. Once the exam starts, the webapp students use to join connects with a separate **Amazon Chime SDK** session using the **Amazon Chime SDK** client. The webapp is also configured to capture camera video stream and screen capture using **m**edia capture pipelines, with consent from the user**.** These are saved to an **Amazon S3** bucket in up to five second file chunks for internal audit and archiving purposes.

7. For every new video chunk saved, an **Amazon S3** PutEvent calls an **AWS Lambda** function, which pulls one image frame from the video chunk and stores that image frame inside another **Amazon S3** bucket as a .jpeg file. This is done to help optimize cost; the solution can use the full video chunk as needed.

8. This cues another **Lambda** function for that .jpeg file, which validates it using **Amazon Rekognition** with various API calls for verifying the student's face, checking how many faces are inside the frame, whether no face is detected, and so on. These response alerts are saved inside **Amazon DynamoDB** for internal audit purposes.

9. If **Amazon Rekognition** detects prohibited actions, it sends an alert to the proctor using **Amazon Simple Notification Service** (Amazon SNS) alerts. This cues an **AWS Lambda** function to send an alert to the proctor (and student, if needed) using the **Amazon Chime SDK m**essaging channel.

Reviewed for technical accuracy March 30, 2023

AWS Reference Architecture