Project Proposal for COSC 499 - Option 3

Team Number: 09
Team Members: Soren Stenback 56957228, Muhammad Bakar 89639025, Jan-Yaegar Dhillon 61173076, Baz Sivakua 17000217

---

1 - Overview

The MVP (minimum viable product) of our project follows a process going from a professional user sending out a video request to the professional user being able to view the processed video.

Firstly, a professional user submits a request through the web app to some number of client users. If the client user(s) have an account on the web app, they will receive a message in their web inbox about the request. However, if the client user(s) does not have an account on the web app, they will receive an email from the web app asking them to register an account. Once the client user(s) does this, they will receive the request in their web inbox. Next, the client user(s) will then be able to accept the request, leading them to the requested webpage, or will be able to reject the request and fill out a reason why that the professional user will be able to view. If the client user(s) accepts the request, they will be able to upload a video to the request web page that will be processed and have identifying information removed. In the event that the upload fails, the client will be presented with an appropriate error message. Once the client user has successfully uploaded their video, the web app will process it and save the processed video. If the web app fails to process, the client user will receive an appropriate error message. Lastly, the professional user will be able to view the video. The professional user will be able to either download the video to their own computer or stream the video through the web app. This is dependent on the client user's privacy decision. If the download or stream fails, the professional user will receive an appropriate error message.

The purpose of this software is to facilitate communication between professionals and clients through the medium of videos while ensuring the client's privacy. This will enable clients and professionals to interact with confidentiality. This solution is unique and valuable for the ease of use it provides to both client and professional users, as well as the reduction of potential visual bias. This solution has advantages over other solutions due to its general usability and the use of video applications and responses instead of text-based applications and responses.

1.1 - Envisioned Usage

Scenario 1: Robert Smith is a Spanish teacher who teaches Spanish to people around the world. He wishes to test two of his students' speaking abilities: Jon and Bert. Robert wishes to keep the identity of his students private, for there is no need to know their faces for the current lesson. To accommodate his needs, Robert decides to use our client's web app for it offers the special ability to keep the identity of his students safe. Robert logs on to the website. After logging on, Robert makes a new request. Robert types in the instructions for his students and their respective email. Additionally, toggles on the blur face option – which will blur any faces in the video. Finally, Robert clicks send after selecting the

due date, and the website asks Robert for confirmation. After confirming, Robert's request is sent to Jon and Bert. After a few days, Robert receives a notification via email that two videos have been sent to him. Robert hops onto the website and navigates to the newly sent videos. The videos are presented to Robert with blurred faces. Robert watches through the video sent by Jon and Bert.

Scenario 2: Sam Hunter is a 19-year-old baseball player living in Oregon. She has been asked by Tim Beck, a scout from Johns Hopkins University, to send a video of her batting since Tim would like to keep Sam's identity safe – and because Tim is across the country, thereby making the transfer of video in person difficult. Tim has sent a request through our client's web app. Since Sam does not have an account on the website, she receives an email asking her to make an account to upload a video. Sam clicks the link and is asked to sign up. After inputting all the information and agreeing to the terms and services, her account is made. She signs in and navigates to the request made by Tim. She accepts the request and is shown instructions and information about what processing will occur on the video. Sam uploads the video. However, it is too large. An error message is displayed to Sam, informing her that the video is too large. After some work, Sam is able to reduce the file size. She uploads the video and is asked to confirm her choice. Sam confirms her choice, and the video is sent to Tim.

Client users can directly upload to professional users video footage answering specific requests, as well as have their personal information processed out of the video to keep their confidentiality.

Professional users can make requests for video footage in response to the needs of the professional users and have direct access to the footage once it is processed.

---

2 - Major Milestones

Deadline	Item	Deliverable
Term 1 - Week 4 Sep 24 - Oct 1	Project Plan	● Complete a clear project plan that explains where the progress of the app is headed. ● Allocate roles after in-depth planning based on skills and preferences and ensure every team member knows what they are to complete and their responsibilities for the project. ● Set up and prepare the GitHub and associated project pages ● Research tech stack
Term 1 - Week 9 Oct 30 - Nov 5	Mini-Presentation	● Create designs in Figma ● Understand and create plan for AWS Step Functions ● Configure AWS Amplify + Tailwind + Next ● Create plan for structure of AWS CodePipeline ● Complete a homepage for the web app that

		explains what the web app is along with buttons to select different options in the web app, such as viewing your account, viewing your inbox, etc. ● Complete a web page to create a request from a professional user to a number of client users where the professional user is able to fill in client users to receive it and what the request is for.
Term 1 - Week 13 Nov 27 - Dec 3	Design Submission	● Complete system architecture design and have placeholder pages for all needed webpages. ● Configure AWS Cloudwatch for monitoring and analytics ● Set up user authentication and access management with AWS IAM and Cognito (through Amplify) ● Complete general user interface design with standardised colours, fonts, and layouts. ● Complete a working registration page where users (both professional and client) are able to sign up for an account. ● Complete ability for the web app to send emails to users who have not registered for an account when a request is made. ● Complete account information page to view user's email, status as either client or professional, and history of requests. ● Complete inbox page where users will receive notice of any requests sent to them. ● Complete documentation on all working components of the web app. ● Complete peer review and assessment of team members' skills and progress. ● Implement end-to-end testing and Jest ● Set up AWS Fargate environment to run Batch jobs on
Term 2 - Week 4 Jan 29 - Feb 4	Peer Testing	● Complete hotfixes and bug fixes for previous milestones. ● Configure testing/building/packaging with Jest and AWS CodeBuild ● Complete page routing code, ensuring information is saved when needed. ● Complete user authentication for users who sign in with their accounts. ● Configure AWS API Gateway to expose endpoints that can be called by the rest of the app ● Configure S3 to store videos in buckets ● Create docker image for processing that can be run in parallel with AWS Batch for efficient video processing

		• Write function to implement AWS Rekognition to detect faces in videos stored in an S3 bucket • Write OpenCV function to blur faces in extracted frame/video • Complete configure DynamoDB and connect with lambda • Complete documentation on all working components of the web app. • Complete peer review and assessment of team members' skills and progress.
Term 2 - Week 8 Mar 25 - Mar 31	Peer Testing	• Complete hotfixes and bug fixes for previous milestones. • Complete implementing a chat functionality for users to communicate through the web • Complete implementing security to protect users' data that is stored. • Complete profile information for professional users to advertise to client users. • Complete CSS and web layout implementation. • Implement AWS Cloudfront for content delivery • Add AWS Elasticache for data caching to improve performance • Complete error handling and applicable error messages to users. • Complete documentation on all working components of the web app. • Complete peer review and assessment of team members' skills and progress. • Configure app security with AWS WAF and Shield
Term 2 - Week 13 Apr 1 - Apr 7	Final Project	• Complete hotfixes and bug fixes for previous milestones. • Test thoroughly • Complete the implementation of video streaming through the web app and download functionality for professional users to save videos. • Complete documentation on all working components of the web app. • Complete peer review and assessment of team members' skills and progress.

Table 1 - Proposed Project Milestones. Documents in detail what is expected at each milestone.

3 - Technology Stack

User technology: (Supported) web browser (basically just not IE, use Playwright for compatibility testing) The user will interact with the platform through their web browser as this is a web app. We will use **AWS Simple Email Service** to communicate with users through email.

Programming languages: With the exception of using **Python** to process the videos and blur detected faces, we will be using **TypeScript** to create our web app and its functions.

Front-end/UI: To create and manage the application's front end, we will use **React** and **TailwindCSS**. We will use **NextJS** for the React app as it offers routing and streamlined development. These tools will allow us to build a modern, interactive website in a streamlined fashion. FIgma will be our primary tool for prototyping and planning designs.

Connecting with AWS: As AWS services are a key part of the client's needs, **AWS Amplify** allows us to connect our web app to various AWS services easily. For example, the Storage module allows videos to be uploaded to **Amazon S3** from the client side/browser, which is secure and scalable storage for the videos. We will be closely following the AWS Servless Application Model.

Server-side logic: We would be using **AWS Lambda** functions to run serverless functions that make up our backend and connect various services. We will also use **AWS API Gateway** for API management and for creating API HTTP endpoints. Our lambda functions with be orchestrated and managed into a workflow using AWS Step Functions. Our app will be using the AWS SAM (serverless application model) toolkit, including the CLI.

Data storage: We will be using **Amazon DynamoDB** as our app's database to store structured data and **Amazon S3** to upload and store videos. Both are secure and scalable and easily integrated with the rest of our stack. We will use **AWS Elasticache** to cache frequently accessed data and improve performance.

User authentication: We will be using **AWS Cognito** to manage user authentication (through Amplify), allowing for easy and secure sign-in.

Video Processing: AWS Lambda and **AWS Elemental MediaConvert** to transcode the video for optimal performance and delivery. Due to the limitations of AWS Lambda's compute and the potential large file sizes of the videos we would be processing, we have opted to run our **Docker** images with **AWS Batch** on **AWS Fargate**. This would allow us to run several images in parallel and would allow for automatic scaling while remaining serverless. To detect faces we would use **AWS Rekognition**, which can accurately detect faces in uploaded videos so we can blur them. We would use **OpenCV** and **Python** (running on the Docker images in Batch on Fargate) to blur detected faces. We would use **AWS Step Functions** to orchestrate these various services together to ensure the reliability and function of the video processing.

Content delivery: We will use **AWS Cloudfront** to deliver the web app's content and static assets quickly and reliably. We will use AWS Route 53 to manage traffic and DNS routing for end users.

Monitoring, logging, and analytics: AWS Cloudwatch will monitor our web app and allow us to access logs and information about the performance easily.

CI/CD: We will be using **AWS CodePipeline** to easily automate the updates and features we release.

Testing: We will be using **AWS CodeBuild** to test and package our code. We can use it in conjunction with **Jest** for unit testing. For end-to-end testing, we can use **Playwright** to ensure compatibility and that everything works together.

Security: To ensure the security of our website and its data, we will use the **AWS WAF (Web Firewall) and Shield** services to protect us against common exploits and DDoS.

---

4 - Teamwork Distribution and Anticipated Hurdles

4.1 - Team Experience, Expertise, and Areas of Learning

Category	Jan	Baz	Muhammad	Soren
Experience	Web and mobile app development, mostly frontend	Web development, database design	Web development	Web development, database design and OS design
Skills	JS/TS, Python, Java, HTML, CSS, React, SQL. Dabbles in Kotlin and Dart.	Java, Python, React, SQL, Front end CSS and HTML, Web design and development	Java, Python, SQL, MongoDB, HTML, CSS, JavaScript, PHP	Java, C, SQL, MangoDB, HTML, Python, JavaScript
Expect to Learn	Serverless and cloud technologies	Use of API and frameworks, AWS, Tailwind project and time management	AWS, Tailwind, React, and TypeScript	AWS, CSS, React, Tailwind and TypeScript

Table 2 - Team Experience, Expertise, and Areas of Learning. Documents each member's reflection on their own skills and areas of improvement that would be useful for task assignment.

4.2 - Expected Areas of Contribution

Category of Work/Features	Jan	Baz	Muhammad	Soren
Project Management: GitHub Project Board Maintenance	✓	✓	✓	✓
Technical Direction: Time Estimation, Making Programming Choices	✓			
Technical Help: Finding Technical Solutions	✓		✓	
Troubleshooting: The Go-To When Others Are Stuck				✓
System Architecture Design	✓	✓	✓	✓
User Interface Design	✓	✓		✓
Workflow design/lambda orchestration/API management	✓		✓	
Front-end Development (Next, React, CSS, Amplify)	✓	✓		
Credential System: Account permissions (IAM), User Log-in, and Sign-up (Cognito)			✓	
Email Functionality: Search Emails in System, Generate and Send Automatic Emails				✓
Inbox/Messaging Functionality		✓		
Error Handling System				✓
Request System: Create, Delete, Manage Requests Made by User		✓		
Security (WAF/Shield) and traffic management (CloudFront)				✓
Building, testing, and deploying (Jest, Playwright, CodeBuild, CodePipeline)	✓		✓	

Category of Work/Features	Jan	Baz	Muhammad	Soren
Analytics, app performance monitoring, and optimization				✓
Video Upload and Storage System (S3, Elasticache, MediaConvert)			✓	
Optimize for Multiple Devices		✓		
Video Processing: Facial Recognition and Blurring	✓			✓
Container management (Docker, Batch, Fargate)	✓			✓
Database Setup and Configuration (DynamoDB)			✓	✓
Presentation Preparation	✓	✓	✓	✓
Design Video Creation		✓		
Design Video Editing				✓
Design Report			✓	
Final Video Creation			✓	
Final Video Editing				✓
Final Team Report		✓		
Final Individual Report	✓	✓	✓	✓

Table 3 - Expected Areas of Contribution. These tasks were divided amongst the team based on each member's experience, strengths, and weaknesses. They decided to make sure that the members were given an equal amount of work that they felt they could complete during the project. Each category of work/feature(s) has been grouped together for convenience, a more detailed list of deliverables is provided in Table 1.