Intelact - A cloud based, AI emergency responder

Daniel Keitley
dkeitley@gmail.com
University of Bristol

***Abstract*—Intelact is a web application that uses artificial intelligence and cloud based technology to rapidly detect and respond to emergency situations. Using Google's cloud infrastructure, the application is able to store uploaded videos in the cloud, identify the type of emergency situation within videos (e.g. medical, fire, physical threat) and subsequently provide tailored responses. Such a system could be used to autonomously store video and audio evidence, contact the relevant emergency services and provide useful safety information, with very low requirements for user interaction. The application can be run at https://intelact-186119.appspot.com/.**

***Keywords*—Intelact, cloud computing, AI, emergency response.**

I. Introduction

In an emergency situation there are multiple demands upon an individual. One must take care of their immediate safety whilst simultaneously act to alleviate the situation. For instance, if an individual is attacked on the street, the victim will be fully occupied either fending off their attacker, or escaping the scene, yet they would benefit greatly from additionally calling for help, collecting incriminating evidence and having access to information such as directions to a safe location.

In such a scenario, having access to cloud computing resources from a mobile device offers an effective way to outsource tasks without the need for user interaction. Here follows a proof of concept of an application that can be used to safely store uploaded videos in the cloud (to be used as video evidence); detect the type of emergency situation using machine learning techniques and subsequently respond in a situation-specific manner.

II. Functionality

The application proposed to overcome the challenges outlined above, is as follows. A victim in a dangerous situation clicks a physical button on their mobile device. This activates the application to continuously stream audio and video content to the cloud. As video and audio content is uploaded, the files are automatically analysed on the cloud using machine learning algorithms to identify the type of emergency situation and its level of severity. Following this, the application will complete a number of suitable tasks (e.g. contact the emergency services) and display or read aloud relevant information (e.g. first aid instructions) to the user.

Although the application proposed would work well as a mobile app, the proof of concept demonstrated here operates as a web application (see Appendix, Figure 9). At a high level, the prototype operates as follows:

1) A user triggers an event, representing an emergency situation, which notifies the application to create a database entry of the event and store information such as the date and time of the event and any information about the user.
2) The user then uploads a video file which is simultaneously displayed on the web-page and uploaded to Google Cloud Storage[5].
3) Once uploaded to the cloud the video is automatically analysed using Google's Video Intelligence API[6], which returns a series of labels representing items that can be detected in the video.
4) Using these video labels, the application classifies the event into either a physical threat, fire emergency, medical emergency, natural disaster or non-emergency situation.
5) Having determined the type of emergency, the application then lists some example responses which could be carried out by the cloud-based application.

In the following section the implementation details behind each of these steps are explained.

III. Implementational Details

Intelact was built using Google cloud products which are expanded upon below. The system architecture is also displayed in Figure 1.

A. Google App Engine

Google App Engine (GAE)[1] is an example of a platform-as-a-service (PaaS); a cloud computing system that enables developers to deploy applications on remote machines, without the need to manage their own computing infrastructure. In particular, it removes the need to manage low-level properties of physical or virtual machines, such as the operating system and run-time environments used.

Intelact uses Google App Engine to run multiple web servers which scale as demand increases or decreases. It simplified the development process significantly, since it removed the need to configure the Node.js[2] run-time environment.

B. Firebase Realtime Database

Firebase[3] is a mobile and web application development platform acquired by Google in 2014. Tts real-time database was used to store user and event information.

[1]Google App Engine: https://cloud.google.com/appengine/
[2]Node.js: https://nodejs.org/en/
[3]Firebase: https://firebase.google.com/

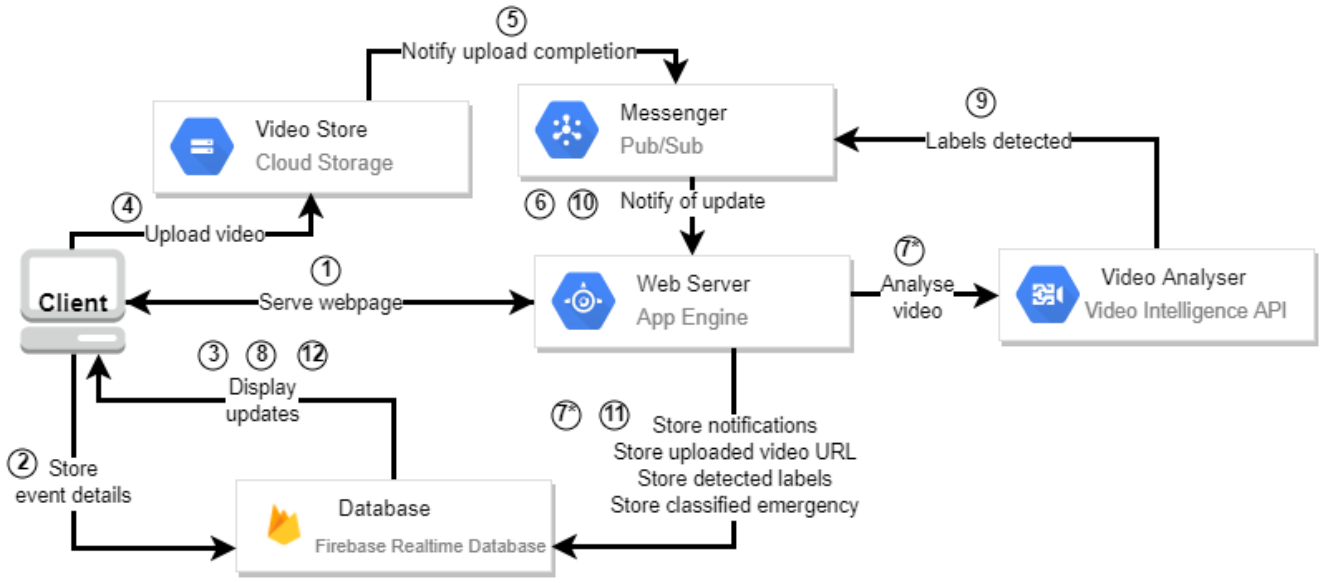


Fig. 1. System Architecture. The cloud technologies used in Intelact and the sequence of interactions (ordered from 1 to 12) between them as a user uses the application. Labels marked with an asterisk represent actions which occur simultaneously.

The Firebase Realtime Database[4] allows storage of information on the cloud without SQL and sync data across multiple devices in real time. This powerful feature allows Intelact to deliver fast updates to clients subscribed to particular events. For instance, if a user triggers an emergency event, they will automatically be subscribed to the corresponding database entry. Therefore, as soon as new information about the event is stored in the database, the subscribed clients are notified in real-time, and their web-page is updated immediately.

In addition, the Firebase Realtime Database provides convenient security rules, which allows Intelact to link users with their Google accounts. This means that the application is able to identify events relating to the same user and link them within the database.

Furthermore, the Firebase database provides forward compatibility. It integrates with both desktop and mobile devices and hence could be used in a practical Intelact mobile app. The database also supports offline caching. If a client interacts with the application while not connected to the internet, Firebase will store any changes to a local cache which is then synced with the cloud once the device reconnects. This feature would be vital in an emergency situation, where reliable internet access may not be available.

C. Google Cloud Storage

The central component of Intelact is video storage and analysis. If video footage of an emergency is to be used as evidence in court, it is vital that it is stored safely, and can be reliably accessed.

Google Cloud Storage[5] is a service that stores files across Google's large computing infrastructure. It is able to store files across multiple machines in different areas of the world, so they can be accessed rapidly regardless of geographical location. It also creates multiple backups of files and advertises a 99.9% availability guarantee[5].

For these reasons Google Cloud Storage is used to store video files uploaded by users. The upload button on the homepage sends a HTTP post request to a Google Cloud Storage bucket, which stores all event videos.

D. Google Cloud Video Intelligence

Google's Cloud Video Intelligence[6] provides application developers with access to Google's powerful computer vision models. It is able to analyse videos stored in Google Cloud Storage to identify entities and when they appear.

Intelact uses the video annotations provided by Google Cloud Video Intelligence to classify the emergency scenario in uploaded videos. For example, if combat is detected within a video, it is likely that the uploader is in physical danger. Similarly, the presence of blood in an uploaded video is highly suggestive of a medical emergency.

The benefit of the Video Intelligence API is that it can be used without any knowledge of machine learning or computer vision. It calls upon Google's models, pre-trained on thousands of YouTube videos, to provide fast and accurate annotations. The API allows developers to specify which geographical region to use for processing, so as to provide the fastest analysis possible.

E. Google Cloud Pub/Sub

Google Cloud products integrate well with one another via an asynchronous messaging service known as Google

[4]Firebase Realtime Database: https://firebase.google.com/products/realtime-database/

[5] Google Cloud Storage: https://cloud.google.com/storage/

[6] Google Cloud Video Intelligence: https://cloud.google.com/video-intelligence/

Pub/Sub[7].

This service allows users to rapidly and reliably pass information between Google cloud applications. It is based on a publisher/subscriber model, where one application can post messages to a particular topic, which are then sent to other applications subscribed to that topic.

Since it is linked with Google's cloud infrastructure, messages sent through Pub/Sub are distributed uniformly across data centres and servers to ensure high performance, even as the number, size and rate of messages sent increases. Messages are also encrypted and stored in Google Cloud Storage so as to prevent message loss and guarantee secure communication between applications.

In Intelact, Cloud Pub/Sub is used to notify the web server when a video has been added to a Google Cloud Storage Bucket. Similarly it is used to notify the web server when a video has been analysed, so that it can classify the emergency. If the number of users uploading videos rapidly increases, Google Pub/Sub will maintain low latency and ensure that the web server is able to quickly respond and carry out the next task.

F. Implementation Summary

With each cloud component outlined above, the application can be now understood from a high level (Figure 1).

As a user enters the homepage URL into their browser, they connect to the Google App Engine servers running the Intelact web server. The server responds by delivering the user the application homepage (1). The user is then able to trigger an event through the user interface (UI), which creates a new entry in the Firebase Realtime Database (2). The client is automatically added as a listener to the database entry, so as soon as a message signifying the triggered event is added, the client receives the message and their UI is updated (3). The user is now able to upload a video through the HTML form. By clicking 'Upload', a HTTP post request is sent to a Google Cloud Storage bucket (4). Once the upload is complete, Google Cloud Storage sends a Pub/Sub message notifying the web server that a video file has been uploaded (5). The web server is automatically a subscriber to the Pub/Sub topic and therefore receives the uploaded video URL (6). With this information, the server simultaneously calls the Video Intelligence API with the Google Cloud Storage location and updates the database entry with the video URL (7). As soon as the database entry has been updated with the video URL, the client is notified of the change and updates its UI to inform the user that the video has successfully been uploaded to the cloud (8). After processing, the Video Intelligence API will return a JSON response of all entities detected in the uploaded video. It will send a Pub/Sub message with this information (9) that is received by the web server (10). The server then uses these annotations to classify the emergency and store the classification result in the database, along with some example responses (11). Finally, the user is automatically notified of the classification result and emergency responses as the database entry is updated (12).

[7]Google Cloud Pub/Sub: https://cloud.google.com/pubsub/

The final state of the application after these sequences of events is shown in Figure 9.

IV. Scalability Analysis

Apache JMeter[8], a load testing tool, has been used to emulate large numbers of users and demonstrate scalability. Each user was simulated by a sequence of HTTP requests which connect to the webpage and post a video file to cloud storage. The response times connecting to the App Engine servers, writing to the real-time database, calling the Video Intelligence API, and sending Pub/Sub messages, were measured during an initial period of normal use, followed by a testing phase. In the testing phase 100 users were simulated. In all following figures, the period of normal use starts from 18:15 until 18:30 and the testing phase lasts from 18:30 onward.

The volume of traffic arriving at the App Engine servers dramatically increased with the onset of the testing phase (Figure 2). However, the latency of app engine requests remained fairly consistent (Figure 3). The median latency of requests was maintained at around 4.5ms throughout the duration of both the normal-use phase and testing phase. This would suggest that the web server is responsive despite increases in demand.

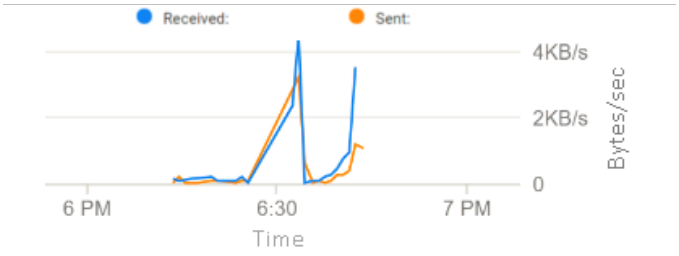


Fig. 2. Google App Engine Traffic. The rate of information received and sent to the web server running on Google App Engine during scalability testing.

It is notable however that the most latent App Engine responses, tend to worsen during the testing phase compared to normal use. The latency of the 99th percentile responses became more erratic, reaching highs of 40ms. Further analysis could reveal whether this pattern holds for longer testing periods.

The database is also scalable to large numbers of write operations. The write speeds during testing were roughly 1-2ms, which is as fast as during normal use (Figure 4).

The traffic and latency of Video Intelligence API calls was also monitored during testing. As with the App Engine server, traffic dramatically increases during the testing phase, although with a larger time offset (since the API is only called once a video has been uploaded). Nevertheless, the latency remains fairly consistent throughout. At 18:39, the API traffic is at its peak (1.8 requests per second), yet the API latency is 5.7ms, comparable (if not competitive) with the latency during normal use.

Despite positive results for 100 users, the application's scalability was impeded by the quotas set for the Video Intelligence API. The default quotas are set to 100 requests per 100 seconds and 300 seconds of video content per 100

[8]Apache JMeter: http://jmeter.apache.org/

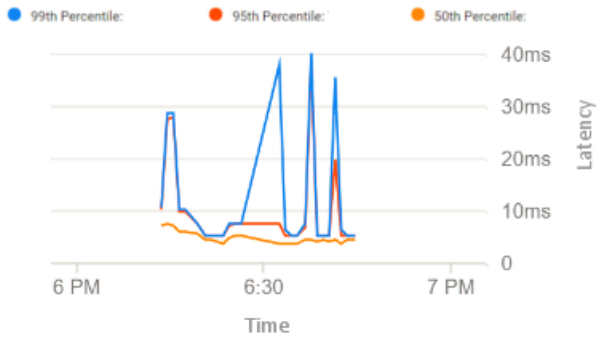
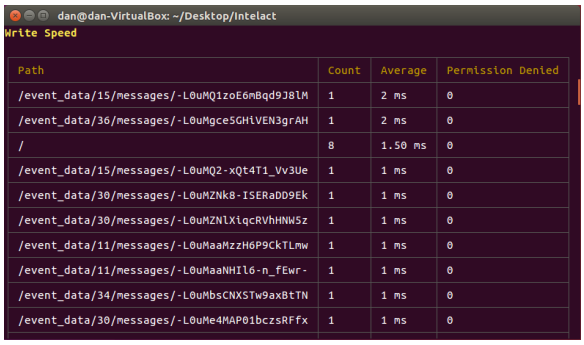


Fig. 3. Google App Engine Latency. The latency of App Engine responses during scalability testing. The orange, red and blue lines represents the 50th, 95th and 99th percentile of responses respectively.



Path	Count	Average	Permission Denied
/event_data/15/messages/-L0uMQ1zoE6mBqd9J8lM	1	2 ms	0
/event_data/36/messages/-L0uMgce5GHiVEN3grAH	1	2 ms	0
/	8	1.50 ms	0
/event_data/15/messages/-L0uMQ2-xQt4T1_Vv3Ue	1	1 ms	0
/event_data/30/messages/-L0uMZNk8-ISERaDD9Ek	1	1 ms	0
/event_data/30/messages/-L0uMZNlXiqcRVhHNW5z	1	1 ms	0
/event_data/11/messages/-L0uMaaMzzH6P9CkTLmw	1	1 ms	0
/event_data/11/messages/-L0uMaaNHIl6-n_fEwr-	1	1 ms	0
/event_data/34/messages/-L0uMbsCNXSTw9axBtTN	1	1 ms	0
/event_data/30/messages/-L0uMe4MAP01bczsRFfx	1	1 ms	0

Fig. 4. Firebase Database Write Speeds. Results from the Firebase database profiler tool which show the average speeds writing to the database during scalability testing. The speeds are comparable to those seen during normal use.

seconds. Testing greater than 100 users and videos longer than 300 seconds resulted in video analysis errors, which meant the application failed to classify events and provide situation-specific responses. The API quota thus acts as the application's scalability bottleneck and must be lifted to improve performance to more than 100 simultaneous users.

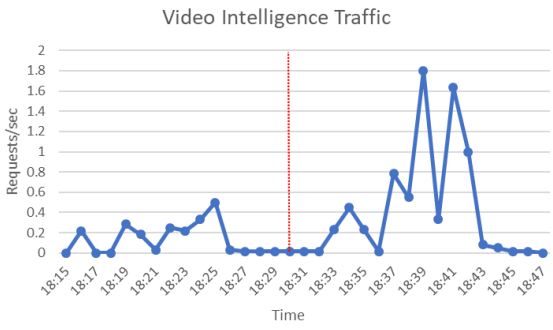


Fig. 5. Video Intelligence API Traffic. The rate of function calls to the Video Intelligence API during scalability testing. The red line indicates the time at which the period of normal use ends and the testing phase begins (18:30).

The Pub/Sub messaging service is a critical component of the Intelact system. It alows the storage, database and video analyser compoents to work together. If there is high latency in sending messages between any of these components and the

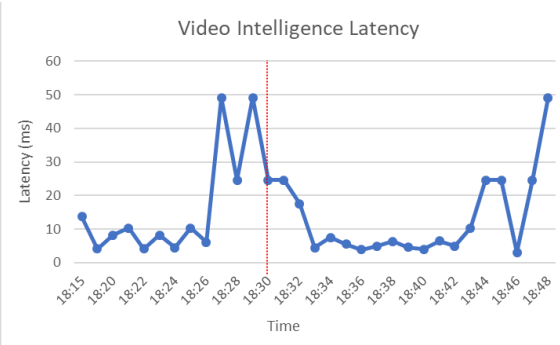


Fig. 6. Google Cloud Infrastructure

web server, the web server may sit idle waiting for information and deteriorate the user's experience.

As with the components discussed previously, the traffic (Figure 7) and latency (Figure 8) of requests to Pub/Sub was monitored during normal use and testing. Pub/Sub traffic increased sharply with the onset of the testing phase while the latency remained relatively constant. This demonstrates that the speed at which event messages are sent and received between components also scales effectively with the number of users.

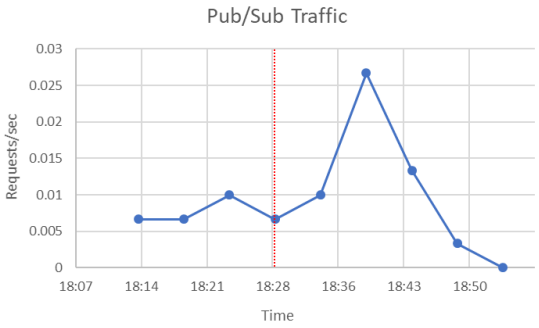


Fig. 7. Video Intelligence API Latency. The latency of responses to the Video Intelligence API during scalability testing. The red line indicates the end of normal use and start of the testing phase (18:30).

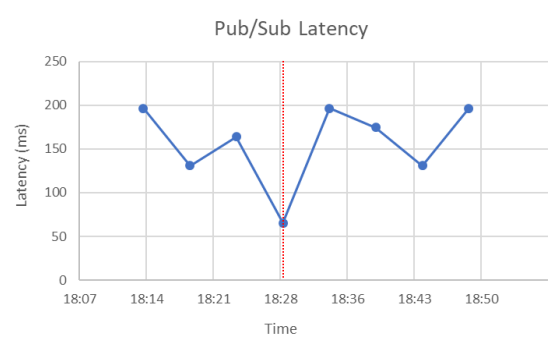


Fig. 8. Pub/Sub Traffic. The rate of requests made to Google's Pub/Sub messaging service during scalability testing. The red line indicates the time at which the period of normal use ends and the testing phase begins (18:30)

Presently, the cloud storage remains the only component of Intelact to lack quantitative data on scalability testing. Since upload times are dependent on internet bandwidth,

the upload times during simulations were heavily biased. As more threads simultaneously uploaded video files from the same network, internet upload speeds dramatically decreased. In a real-world scenario where multiple users connect from multiple computers, the effect upon upload speeds may be different.

Furthermore, it was not necessary to monitor storage access times, since the application displays videos directly from the client's local machine. The video file stored on the cloud is only accessed via the Video Intelligence API, which is challenging to monitor. Nevertheless, the storage bucket used is hosted regionally in Europe, meaning that access speeds should be sufficient from the UK. Should it be desirable to improve access times in other parts of the world, the storage bucket can be distributed to multi-regional locations.

V. Future Directions

The prototype described in this paper provides a proof of concept of the middleware components of a cloud-based emergency response application. However, further steps are needed to fully implement the front-end and back-end components of the system.

Intelact would better serve its purpose as a mobile application which can continuously stream information to the cloud. As a mobile application, the system could have direct access to GPS/location information, the user's contact directory, their messaging applications and other data. With these mobile services, Intelact could perform more effective emergency responses.

Exploration of the back-end components of the system could also vastly improve the applications functionality. For example, the system could benefit greatly from analysing audio content more deeply. A user who is not able to physically interact with the application, could instead feed information to the application through voice control. In times of ambiguity, a user could directly instruct the application to complete specific tasks (e.g. contact a family member), provide information about the emergency, indicate the level of severity by the tone of their voice and so forth.

Another limitation of the current prototype is its mechanism for classifying emergencies. The current classification rules work by identifying key words in video annotations of sample emergency videos. However, due to the small number of test videos collected, these classification rules are unlikely to generalise well. To improve classification accuracy, one could obtain more sample footage or implement more intelligent classification rules; perhaps looking at combinations of key words.

VI. Conclusion

Intelact provides a scalable prototype for a mobile application able to help individuals in emergency situations. Using cloud technology, the application developed is able to carry out complex computations without the need for user interaction and has the potential to deliver intelligent responses in an emergency. The technology used has been proven to scale well with increased numbers of users and increased amount of video content, demonstrating that it could reliably provide guidance in emergency situations, even with fluctuating demands.

Acknowledgment

The author would like to sincerely thank their family for providing turkey sandwiches and Christmas chocolates throughout the writing of this report.

APPENDIX A
APPLICATION SCREENSHOT

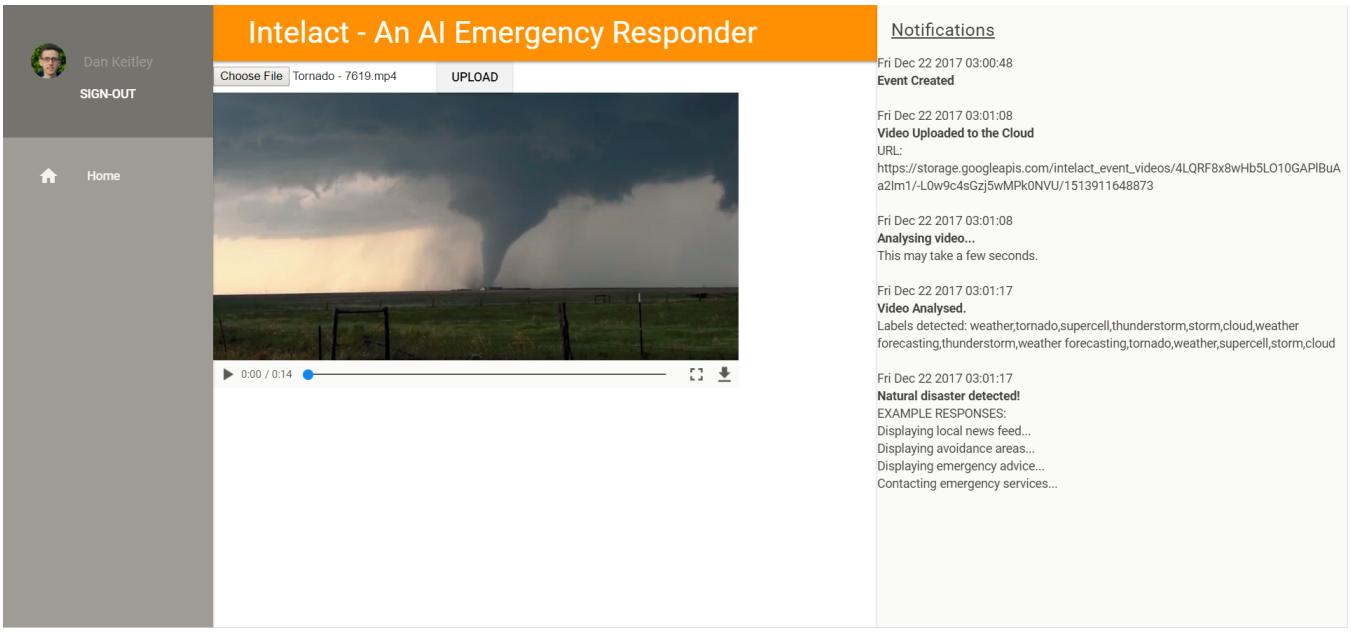


Fig. 9. A screenshot of the Intelact application. Here, the application has correctly identified a tornado in the uploaded video and subsequently classified the event as a natural disaster.