zenitech

Case Study

# Keeping people connected through automation

Customer
**Inmate Communications Provider**

Location
**Northern Europe**

Industry
**Communications**

During the pandemic, video calls became an essential way for prisoners to stay connected to their friends and family.

However, the rules for these calls are as strict as if they were in-person visits to the prison. Only one caller is allowed per video call, and they have to match the visitor ID. Rather than having to manually review and monitor each call to ensure it complied with prison rules, Zenitech worked with a leading Inmate Communications Provider, the company responsible for communications systems in prisons, to automate this review process and keep people connected within the rules.

## Facial recognition and analysis

The Inmate Communication Provider develops, installs, and operates communications and media in over 650 correctional facilities, providing an essential service by keeping prisoners in touch with friends and family (which is crucial for rehabilitation).

The provider's systems have to adhere to the highest security standards. It operates in 20 countries and is used by more than 300,000 inmates.

Pandemic restrictions meant that visiting people in correctional facilities became almost impossible – this meant an increase in video calls and a sharp rise in demand for call reviewers (demand was outstripping availability).

zenitech

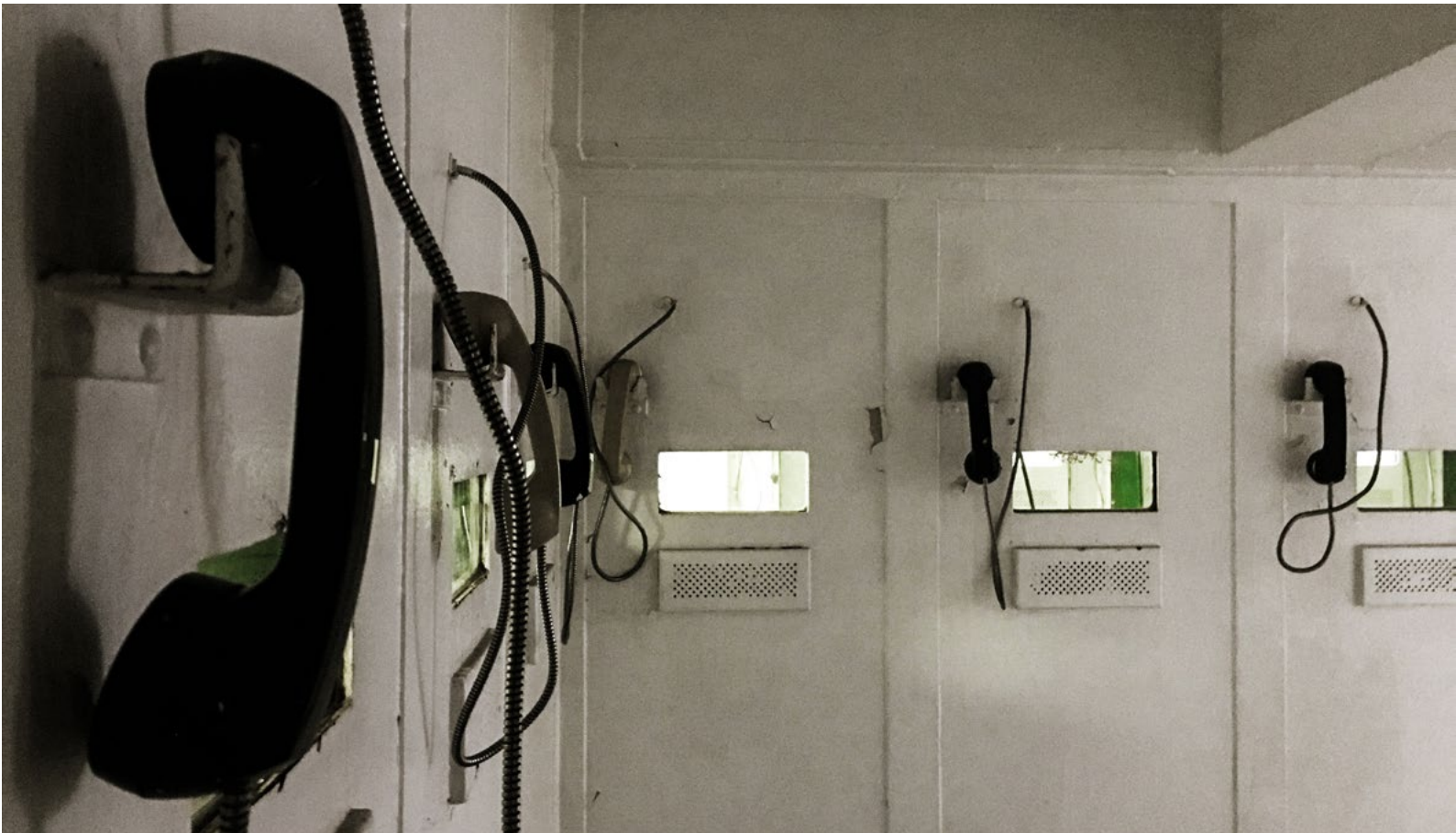
## Automated facial ID

The Inmate Communications Provider asked Zenitech to help create a PoC solution that would integrate into its current call system and could automate the review process to ensure communications stayed within the rules. This would increase the number of calls processed and save money by reducing the number of reviewers needed for each call.

Zenitech created a working PoC that automates the recognition and ID card-based validation of visitors' (callers) faces and detect if there were other faces present during the video call (as the rules specified that only one person could visit/be on the call at a time). The face identification solution used a neural network detection algorithm to find all the faces on the video stream. Then, it analysed each face and encoded them into a feature vector, which estimated the closeness of the face to the one on the ID card.

The quality of the video stream also needed to be analysed. If the quality was too low, the face matching would fail, and the call would terminate. To combat this, Zenitech created a solution to analyse the lightness and sharpness of images and create an image quality score, which didn't require reference images.

Zenitech bundled these algorithms into a Python package and built a facial recognition service around it. The service checked video quality and face detection and integrated it with the rest of the call system.



zenitech

## Creating connections for prisoners

Time was of the essence to ensure people stayed connected. The project was completed in just over two months, and has ensured prisoners can stay in touch with family and friends while adhering to strict visiting rules. It has been so successful that the solution is being rolled out across different services.

## Zenitech's technical approach

Zenitech used Linux as an operating system for development and deployment. Python was the primary project language, and we used Flask as a framework (with Gunicorn as the webserver) to create REST APIs. RabbitMQ was the messaging layer between different services, and OpenAPI described the REST endpoints.

Two of Zenitech's data scientists and one of our project managers worked with two backend developers and one backend lead from the client's team. The project was run in four sprints; the first sprint focused on developing the algorithm, and subsequent sprints focused on system integration.

At the start of the project, the client wanted to integrate image processing into the current system through a C++ API bundled into the call service. But Zenitech recommended a message queue-based approach with AMQP and a RESTful interface for control, which resulted in a more decoupled architecture, better security, easier deployment and improved testability.

As a result, during the development of the face recognition service, we used containerised solutions and built our service into a Docker container. We selected RabbitMQ as the message queue to handle the AMQP messages, which was also deployed using Docker.

www.zenitech.co.uk
info@zenitech.co.uk
linkedin.com/company/zenitechteam
facebook.com/zenitechteam
instagram.com/zenitech