This section is dedicated to detailing the high-level architecture and design of the project. It will discuss the purpose of each component and the software engineering principles applied to make design choices.

# Software Design

The project’s design began with understanding the MLaaS threat model described in §\ref{sec:threatModel}. Figure \ref{fig:abstractNetwork} depicts an abstract layout of the project’s core components. There are two categories of components: \textit{online} describing inference performed directly in response to a user’s request to emulate the MLaaS model, and \textit{offline} describing inference results generated on batches of data, independent of the frontend. The offline components are used during the implementation and evaluation stages to develop and refine the system (labelled \textit{testing}), and to collect and analyse the results presented in Chapter \ref{chap:evaluation} (labelled \textit{evaluation}).

An overview of the project’s repository is given in Figure \ref{fig:filetree}. Excluding SEAL, all code was written for the project. An object-oriented design methodology was used to allow separate components to be implemented independently and isolate the layers depicted in Figure \ref{fig:abstraction}. Another advantage is that it allows straightforward substitution of different inference methods and encryption schemes.

The application’s online components can be split into four abstract layers, from high-level interface to low-level implementation.

1. The \textit{graphical user interface} component allows users to configure the encryption scheme and inference method, and upload videos.
2. The \textit{client} and \textit{server} components are responsible for managing the network communication to allow videos to be transferred to the server for inference, and the results returned to the client.
3. The \textit{encryption} component provides an API for the cryptographic primitives to allow videos to be encrypted and decrypted in the client, and operated on by the server in the \textit{inference} component.
4. The implementation of the cryptographic primitives provided by the CKKS scheme through the \textit{MeKKS} and \textit{SEAL-Python} libraries.

Figure \ref{fig:abstractInference} provides insight into the composition of the inference component. The scope of this project only considers the layers above encryption primitives. However, lower layers do exist. Another layer relevant to this investigation may be the hardware implementation. The IoT devices used for surveillance cause significant constraints on computational performance. Accelerators, such as GPUs, could be investigated to improve running times of cryptographic operations on these devices [BADAWI].

# Class Structure

This section provides a more thorough insight into the project’s structure. Figure \ref{fig:clientUML} details the arrangement of the client-side, Figure \ref{fig:serverUML} contains the classes composing the server-side, and Figure \ref{fig:mekksUML} depicts the structure of the MeKKS library – the bespoke CKKS implementation. While overlaps between diagrams exist, they have been separated into three figures for clarity.