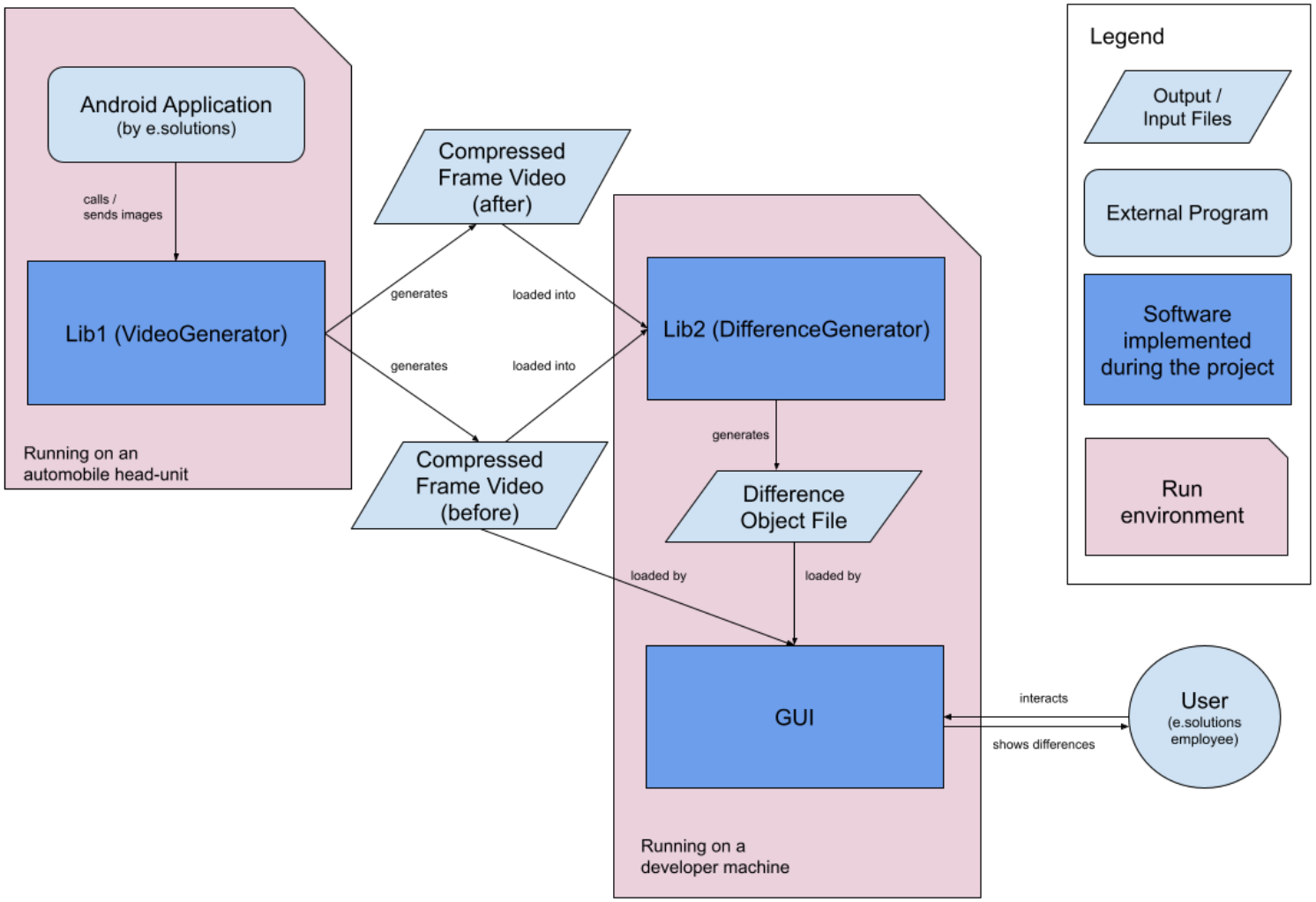
Software architecture for GUI-Frame-Diff

Runtime Components

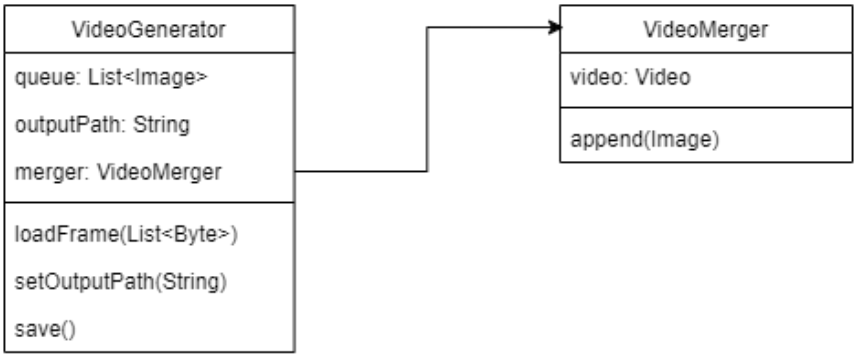
https://docs.google.com/drawings/d/1CmRSXvGgmkh3ppcD1VME0LlOeNEKh4k9CvAr5rYAkNc/edit



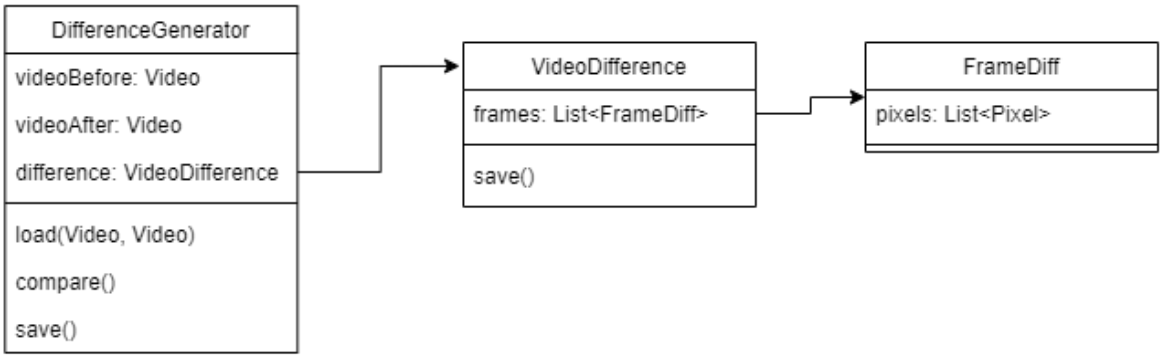
Code Components

https://drive.google.com/file/d/1bGFoUJJ4K7U_BCFub4c1iO3WYdtgBCY-/view?usp=sharing

Lib1
(VideoGenerator)



Lib2
(DifferenceGenerator)



GUI

Placeholder
(Currently, the design focus lies on the above libraries)

Technology stack

During the project, we will be using the following technologies to form our stack:

- Kotlin (as the main programming language)
- Gradle (for building)
- Android SDK (for Android library development)
- javaCV (for video and image manipulation)
- OpenJDK (for common java implementations and helpers)

For the GUI:

- Has not been decided on
- If a requirement is to use lib2 directly in the GUI, a java framework might work best
    - e.g. the Swing or JavaFX for desktop application development

Explanation of diagrams and choices

Runtime components

The three main components for this project and their interactions are pictured in the first diagram.
The first library is meant for generating a compressed video file out of various images. The library is getting called and used by a not further known Android application of the customer which also supplies the images.
The second library takes two of the videos created by the first library and compares them for their differences. The library then generates a new video with highlighted pixels for the found differences.
The full capabilities and requirements for the third component, the GUI, are not quite specified yet. The application should provide the possibility to select two video files for comparison and let the user review these videos side-by-side. There is also a possibility for stepping forwards and backwards of the videos required.

The application will either use the library for generating differences directly or just let users view the differences, but regarding the requirements a direct integration of the library seems to be more reasonable at this time.

Code components

For the technical implementations of the libraries some additional points need to be considered.

The first library has to process two different situations. It has to both generate a new video if requested and also be able to append one or more files to an existing video.
Therefore, the library needs at least two major components, one generator and one merger.

The generator processes the basic functionalities of the library. It works with an OutputPath, a list of images and uses the Merger component.
The OutputPath contains the filename for the to be created or edited video file. If the file does not exist, the library has to create a new one.

The Merger then takes the existing video and appends the list of images in a lossless way.

The second library has to compare two input videos frame-by-frame. It has to load the videos correctly, identifying the used compression codec. Afterwards, it loops through the videos and compares corresponding frames for differences. The found differences are then bundled into a new video, where each frame is the pixel-wise difference of two corresponding frames. Matching pixels are marked black, while differing ones are marked with the difference value.

Tech Stack

As lib1 runs an automotive head-unit, it will have to be in the form of an Android library to be used by an Android application developed by our stakeholders. On that head-unit, resources are limited and thus performance matters. That is why we choose to use Java wrappers for performant image and video manipulations tools which are optimized for running time and memory usage.

We use gradle for building and dependency management as it is a standard in the java/kotlin ecosystem and use the Java and Android SDKs as standard implementation libraries.

We are currently focused on the development of the two libraries, therefore not much is yet decided for the GUI. We do not have a concrete vision for our technology stack, but we possibly need to integrate the second library into the application and therefore use a technology with good compatibility with Java/Kotlin libraries.
The GUI will be therefore probably implemented with a Java GUI toolkit or framework.