As part of our project, we provide an API allowing third party applications to upload media. When a user retrieves media through a website or mobile app, we want to be able to show a thumbnail before they download the full file, the issue we face is, how do we generate thumbnails for a wide variety of filetypes.

The first decision is to determine what should be generating the thumbnails, there are three obvious options. First, the third-party application could provide the thumbnail, however this means that each developer has to replicate effort, though they would know what filetypes they will need to support; this also reduces flexibility, for example if we ever wanted different resolution thumbnails for different devices. Second, the media API server could generate the thumbnails, this prevents any duplication of effort as thumbnail generation is only performed at one point. One potential pitfall with this approach is that the media API needs to support a large set of formats so as not to limit the functionality of the third-party applications. Performing thumbnail generation here also allows for the requirements to be changed in the future, without third party applications being affected. Lastly, thumbnail generation could be done on the user's device, this however would require the full file to be downloaded and so the experience would be worse, and in the web case, possibly slow even once downloaded. However, this does give the most flexibility with changing how the thumbnail is displayed as they are only generated when the requirements for the thumbnail are known.

Considering the various approaches, it is easy to rule out generating the thumbnail on a user's device. Whilst this is most flexible from a developers point of view, it provides the worst experience for the user, wasting data and potentially being slow. Ultimately, we decided to have the media API server generate the thumbnail, generating the thumbnail here allows for a similar level of flexibility, whilst ensuring the user has a good experience. We ruled out having the third-party developer generate the thumbnail to remove the burden from them, and reduce duplicated effort and room for bugs.

Having decided to generate the thumbnail on the media API we then needed to determine the formats we would support. It would be preferable to support every possible format, but this is impossible, instead we decided to come up with a supported list of formats for third party developers. Considering that the current applications generate video, images and text, we decided to start there, selecting the formats in each category that we believed were most suitable.

For images we decided to support at least PNG (Portable Network Graphic), JPEG, and SVG (Scalable Vector Graphic) files. We decided to support PNG as it provides a lossless raster image format and provides reasonable compression ratios. JPEG was selected as it is a very common format for photographs, it is a lossy compression algorithm however the artifacts produced are less visible in more natural images. JPEG also produces very small images in terms of file size. Finally SVG was selected as it is a vector graphic format, allowing for storage of vector images that can be scaled without loss in quality.

For video we decided to support at least H264 in the MP4 container with MP3 or AAC audio. This is also known as MPEG-4 Part 10, and is a very common video format on the internet. Whilst the H264 format is unfortunately patented, it is supported by most modern phones and web browsers, and so is a good fit for media consumption by users. Free alternatives considered were WebM and Ogg Theora, however both WebM and Theora have lesser support and so could be problematic for users, especially those using older devices.

For text we decided to limit our support to just plain text files and PDF (Portable Document Format) files. Whilst many other formats are used for documents such as Microsoft's docx, we believe that the third party applications should be able to provide a PDF equivalent through libraries available. We decided to support txt as a simple format for those exhibits that wish to provide only basic text to the user.

Having created our list of essential formats, being PNG, JPEG, SVG, MPEG-4 Part 10, plain text and PDF we began to look at methods for generating thumbnails. One option was to use a pre-existing library for NodeJS, and there are some that claim to support 450+ formats such as filepreview. We ultimately decided against using such a library as if in the future it was necessary to support a format that the library did not support, it could be problematic to add support if using a library. Instead we decided to look at how we could generate thumbnails for files ourselves.

For image formats, ImageMagick was chosen as it is a useful library for transformation of wide selection of file formats, including PNG, SVG and JPEG. In fact, ImageMagick also supports PDF and plain text files. ImageMagick can then output an image output to be used as the thumbnail on any of the target platforms. For video formats, FFmpeg was chosen to capture individual frames from the video. FFmpeg was selected for its thumbnail filter, allowing heuristic detection of a meaningful frame within a video file automatically. Both FFmpeg and ImageMagick are free software, and we are able to bundle them with our final software solution.

Alternatives to ImageMagick and FFmpeg exist, however as ImageMagick and FFmpeg are the most common tools for image and video editing respectively, we believed that they were the best choice for the task. Additionally, for ImageMagick to support PDF files Ghostscript is also required, fortunately this is also free software that we can include in our solution.