Multiplatform

SimpleX supports Android, iOS and desktop (in terminal mode only). The idea is to have a desktop application with UI that will act like a separate client without linking to any mobile application. Same architecture as in mobile (UI + backend via Haskell library). The UI will be built using already known Jetpack Compose but with it's multiplatform version that is named Compose Multiplatform and is developed by Jetbrains team (that is behind Kotlin and Intellij IDEA, for example). By using the multiplatform version of Compose the app can be run on any platform that supports JVM (initial iteration will be based on JVM-only, no native parts). Pros of using JVM:

- current libraries that is used by the app can be supported there as well (except Android-only libraries that will not be used in desktop version at all)
- code needs to be changed only a little in terms of calling platform-based methods like getting a time. Having native version would require to use multiplatform library that provides such data
- it has support of Swing that can give us more flexibility of displaying some non-ordinary widgets like video player
- so many libraries are in JVM-world only
- good performance.

Cons of using JVM:

- have to put JVM runtime inside a binary or provide documentation for how to install JVM to the end user of the app
- higher memory usage in comparison to native version.

Moving to multiplatform has the following steps:

- move files into Compose Multiplatform project structure by using `.kts` files for Gradle projects
- divide the code to Android only, desktop only and common parts. Same common part will call methods that will act like interfaces and be executed on a current platform without common part to worry about implementation
- create stubs for hardware features like video/audio/microphone/camera in desktop part while having them as is in Android part
- make the project to build successfully without using hardware features in desktop
- implement video support like here. Under the hood it uses vlcj library which is using VLC native library
- implement audio support using the same `vlcj` library
- implement camera support using webcam-capture that is using `vlcj` too
- since we have a working camera on this stage, QR Code scanner will be implemented too by the lib that is already used on Android
- rewrite the whole WebRTC stack for using Google's WebRTC lib with a help of webrtc-java helper but with our own build of WebRTC lib since

it has our changes to encryption/decryption possibilities. This has to be used on Android as well
- ensure the app can be run on all platforms with JDK
- prepare distribution of binaries per platform.