# Compositor

Prepared for: Mark Pavlidis, CTO / Co-Founder

Prepared by: Sivakumar Boju, iOS Developer

June 7, 2018

Developer Test

# COMPOSITOR

## Requirements

In the app's main view play source.mp4 composited with with still.png and mask.png such that the movie's motion blends with still.png relative the alpha channel of mask.png. That is, wherever mask.png is totally transparent, show still.png; wherever it's totally opaque, show the frame from source.mp4; wherever it's partially transparent, show a blend proportional to the alpha value.

## Guidelines

- use what ever frameworks you deem necessary to satisfy the requirements
- don't be afraid to ask questions
- take your time to do it well
- show your work (avoid monolithic commits)
- open a pull request when you are ready to have it reviewed

## Stretch Goals

1. Loop the video playback
2. Add a button to toggle bounce or repeat loop
3. Add a button that renders the composition to a video and saves the video to the Photo Library

# FLIXELPHOTOS

## Development Process

Below are the actual process that the developer went thru while completing tasks

- Added DEVELOPMENT.md to note down the development process
- Install cocoa pods and add SwiftGen 3rd party library to handle localization. Not sure at this time, what other libraries I will use to complete the task. (Personally I wanted to explore everything on my own code).
- Added AVFoundation and tested both the video and localization
- Added a custom view to handle the video related functionalities and tested with different orientations.
- Added a label to show the video progress and message related to all functionalities be it errors or successful message. This was not asked, but this later lead to unnecessary thread related issues. Surprised to learn return layer as AVPlayerlayer was treated as UI update and was expecting to use main thread.
- Added extension methods (from previous experience) to handle custom view initializations and custom notifications
- Added method to handle video on loop. Tested few of the CIFilter methods to see any of it can be used.
- Created the required permissions to handle Photos access to write and later on to record as well.
- Added required buttons to handle the required functionalities. Replay kit was added to handle recording start / stop.
- Added saving the video first locally on the device document/flixel folder and upon success added the asset to the photos.
- Tested most of the functionalities again multiple times.
- Image and Video overlay, I tried for sometime, but then didn't want to get stuck there. So moved on to completing all the functionalities before coming back to work on overlay.
- Learned a lot about Core Graphic, Core Images and other frameworks, some of which, I didn't had a chance to try before.

**Note:** Every time there was a simple progress in the project, the code was committed, along with development notes.

# ARCHITECTURE

### MainViewController

This is the main view to present the required functionalities.

### VideoView

- Custom View that handles everything to do with the Video, Still image and Mask image.
- Can be placed within any UIView. This parent view can manage size and placement of the VideoView.
- Has delegate protocol methods to notify  didStartRecording, didStopRecording, didFailRecording. This will allow the parent to decide, what to do for each event notification.

### Extensions

- UIView, UIImage and Notification Names were extended for easy reusability.

### Utiltitles

- A singleton to handle all the required document or file related functionalities

### ScreenRecorder

- An object that handles the actual start/stop screen recording
- Has protocol methods to notify the parent didFailRecording

### AlphaFilter, Filter

- Just trial objects, not used by the project anymore.