Portfolio Camera Functionality

by Aleksandar Todorov

Table of content

Overview

This document provides explanation and design choices regarding the implemented taking picture evidence with the phone's camera functionality of the "Drieam Portfolio" mobile application. It explains in detail how to expand the existing camera functionality and also provides documentation for both Android and IPhone devices.

How to use

To use the image upload functionality, you must start your application. Click the "Add Evidence" button and select the "Take Picture" option from the list. The app will ask you to grant camera permissions and then it will automatically start. Taking a picture will redirect you to a preview screen where you need to name your evidence and you can then upload it. The app will automatically redirect you after to the home screen where you can see your newly added evidence.

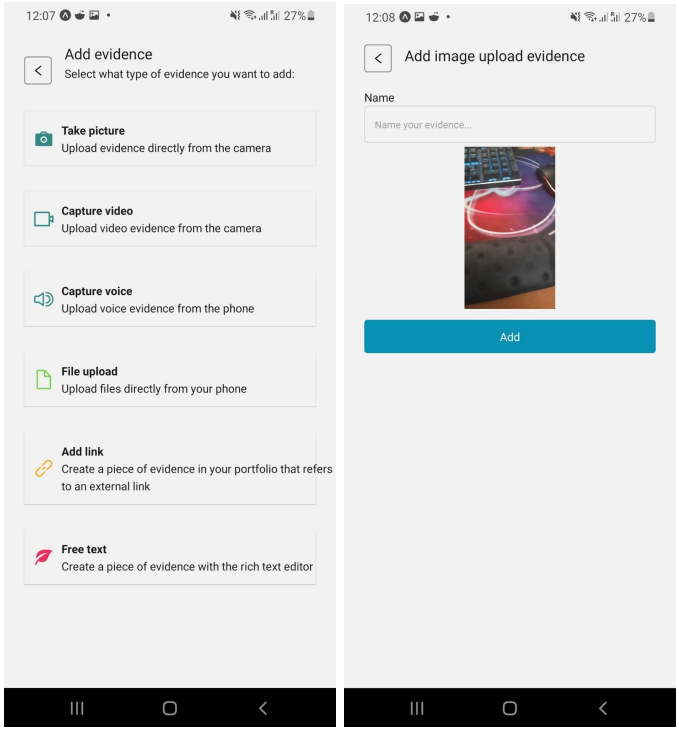

Figure 1: Add image evidence screen and preview

Camera implementation

Start debug environment

To open the development environment you must download Expo Go from either PlayStore for Android or App Store for IPhone in order to debug the mobile application. From then on you can connect your locally hosted project directly on your phone, where any changes in the code will automatically deploy on your phone.

As prerequisites you need to have Node.js installed on your machine. When you have it installed, you can start the project on your pc by running these commands in root directory:

npm i
npx expo start

Running these commands will automatically install any project dependencies and will also start the debug environment.
From then on you just have to use the Expo Go QR Scanner and scan the shown QR Code from the project.

Figure 2: Scan example QR Code

Image upload evidence

All of the functionality regarding the image upload evidence is located in the ImageUploadEvidence.tsx React component.

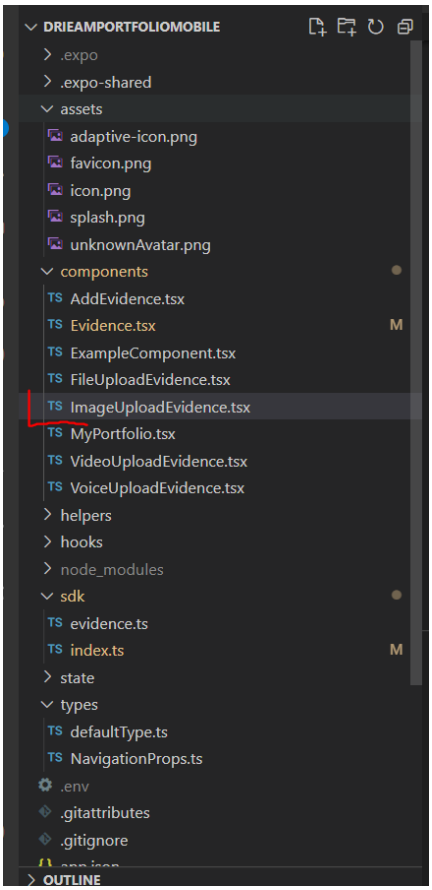

Figure 3: ImageUploadEvidence.tsx component location

When a user navigates to the image upload evidence screen, the system will prompt the user to give permission for the camera. If the user grants permission the camera will automatically start, however if the user refuses to give permission the camera functionality will not work.

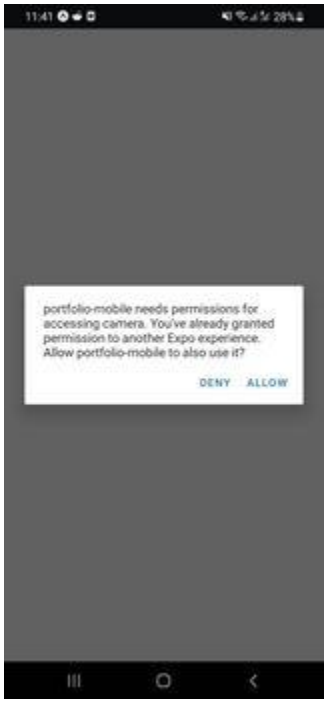

Figure 4: Asking user for phone permissions

Camera image ratio

By default the Android image ratio of the camera is 4:3. However this causes distortions when used by a camera which supports a different aspect ratio such as 16:9 or 18.5:9. Thus a custom function was created to dynamically handle various aspect ratios. The function is called **prepareRatio** and full documentation of it can be found in the ImageUploadEvidence.tsx file.

```
// set the camera ratio and padding.
// this code assumes a portrait mode screen
const prepareRatio = async () => {
    let desiredRatio = '4:3';  // Start with the system default
    // This issue only affects Android
    if (Platform.OS === 'android') {
        if (!camera) return
        const ratios = await camera.getSupportedRatiosAsync();

        // Calculate the width/height of each of the supported camera ratios
        // These width/height are measured in landscape mode
        // find the ratio that is closest to the screen ratio without going over
        let distances: any = {};
        let realRatios: any = {};
        let minDistance = null;
        for (const ratio of ratios) {
            const parts = ratio.split(':');
            const realRatio = parseInt(parts[0]) / parseInt(parts[1]);
            realRatios[ratio] = realRatio;
            // ratio can't be taller than screen, so we don't want an abs()
            const distance = screenRatio - realRatio;
            distances[ratio] = realRatio;
            if (minDistance == null) {
                minDistance = ratio;
            } else {
                if (distance >= 0 && distance < distances[minDistance]) {
                    minDistance = ratio;
                }
            }
        }
        // set the best match
        desiredRatio = minDistance!;
        //  calculate the difference between the camera width and the screen height
        const remainder = Math.floor(
            (height - realRatios[desiredRatio] * width) / 2
        );
        // set the preview padding and preview ratio
        setImagePadding(remainder);
        setRatio(desiredRatio);
        // Set a flag so we don't do this
        // calculation each time the screen refreshes
```

Figure 5: Aspect ratio calculator function

Taking pictures

Taking pictures is handled natively by the expo-camera package, which then returns a URI (location string) of the newly created image and other metadata such as file name, extension and date.

```
78    async function takePicture() {
79        if (!camera) return
80        const photo = await camera.takePictureAsync()
81        console.log(photo)
82        setCapturedImage(photo)
83    }
```

Figure 6: Taking picture functionality

From then on we save the URI and metadata in-state in order to then send it via Dream's API to the Portfolio database of the currently logged in user.

Uploading image evidence

Image uploading is handled by the **addPictureEvidence** function located in the ImageUploadEvidence.tsx file.

```
async function addPictureEvidence() {
    if (!capturedImage) return

    let localUri = capturedImage.uri;
    let filename = localUri.split('/').pop();

    if (!filename) return

    let match = /\.(\w+)$/.exec(filename);
    let type = match ? `image/${match[1]}` : `image`;

    let formData = new FormData();


    //@ts-ignore
    formData.append('files[]', { uri: localUri, name: filename, type });
    formData.append('evidence_type', 'file');
    formData.append('name', evidenceName);
    //formData.append('collection_ids[]', configFiles.portfolioId);


    const res = await fetch(`https://portfolio.drieam.app/api/v1/portfolios/${configFiles.portfolioId}/evidence`, {
        method: 'POST',
        body: formData,
        headers: {
            'content-type': 'multipart/form-data',
            Authorization: `Bearer ${configFiles.bearerToken}`,
            'X-CSRF-Token': configFiles.XCSRF
        },
    }).catch(ex => console.log(ex))

    console.log(res)

    navigation.navigate("Home")
}
```

Figure 7: addPictureEvidence function

The function extracts the filename and type of the stored in-state image and attaches it to a FormData file. From then on we call the Drieam API with the evidence, where it is stored in the database and the file can be viewed on the Desktop app. The system then automatically redirects the user to the evidence dashboard where he can see the newly added evidence.

Expanding camera functionality

Currently the functionality of the camera is fairly limited as it allows only taking pictures with the base preset, which includes taking a picture without any additional hardware or software features provided by the phone. However, additional features can be expanded to include geolocation, flash mode, optical-image-stabilization and more.

Depending on the needs of the user the functionality of the camera can be expanded by using the **expo-camera** package. It provides a feature rich SDK to allow various customization options. The library offers developers to interact with the phone's camera and supports flash, barcode scanner, video stabilization options, face detection, autofocus and more.
You can find the full documentation for the SDK from the Expo website: https://docs.expo.dev/versions/latest/sdk/camera/