audreywelch / **ios-lambda-timeline**
forked from LambdaSchool/ios-lambda-timeline

No description, website, or topics provided. Edit

Manage topics

9 commits **4** branches **0** releases **1** contributor

Branch: **master** New pull request Create new file Upload files Find file Clone or download

This branch is even with LambdaSchool:master. Pull request Compare


SpencerCurtis Update Comment.swift Latest commit `11fe31f` on Nov 5, 2018

LambdaTimeline.xcodeproj	Add starter project	4 months ago
LambdaTimeline	Update Comment.swift	4 months ago
.gitignore	Create .gitignore	4 months ago
Podfile	Add starter project	4 months ago
Podfile.lock	Add starter project	4 months ago
README.md	Update README.md	4 months ago

README.md

Lambda Timeline

Introduction

The goal of this project is to take an existing project called LambdaTimeline and add features to it throughout this sprint.

To begin with, you will take the base project which has basic functionality to create posts with images from the user's photo library, and also add comments to posts.

For today, you will implement the ability to add filters to images you post.

Instructions

Please fork and clone this repository, and work from the base project in the repo.

Part 1 - Firebase Setup

Though you have a base project, you will need to modify it. To begin, run `pod install` after navigating to the repo in terminal. Work out of the generated `.xcworkspace`

1. Create a new Firebase project (or use an existing one).
2. Change the project's bundle identifier to your own bundle identifier (e.g. `com.JohnSmith.LambdaTimeline`)
3. In the "Project Overview" in your Firebase project, you will need to add your app as we are using the Firebase SDK in

our Xcode project. You will need to add the "GoogleService-Info.plist" file that will be given to you when you add the app.

4. Please refer to this page: https://firebase.google.com/docs/auth/ios/firebaseui and follow the steps under the “Set up sign-in methods”. You will only need to do the two steps under the Google section. The starter project will have that URL type already. You just need to put the right URL scheme in. You can find the URL Type in your project file in the “Info” tab at the top.
5. In the Firebase project's database, change the rules to:

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

This will allow only users of the app who are authenticated to access the database. (Authentication is already taken care of in the starter project)

6. In the left pane of your Firebase project under "Develop", click the Storage item. Click the "Get Started" button and it will pull up a modal window about security rules. Simply click "Got it". It will set Storage's rules to allow access to any authenticated user, which works great for our uses.

Firebase Storage is essentially a Google Drive for data in your Firebase. It makes sense to use Storage in this application as we will be storing images, audio, and video data. If you're curious as to how Database and Storage interact, feel free to read Firebase's Storage documentation and look at the code in the base project. Particularly in the `Post`, `Media` and `PostController` objects. (Don't feel like you have to, however)

At this point, run the app on your simulator or physical device in order to make sure that you've set up your Firebase Project correctly. If set up correctly, you should be able to create posts, comment on them, and have them get sent to Firebase. You should also be able to re-run the app and have the posts and comments get fetched correctly. If this does not work, the likely scenario is that you've not set up your Firebase project correctly. If you can't figure out what's wrong, please reach out to your PM or Spencer.

Part 2 - ~~#NoFilter~~ #Filters

Now that your project is working correctly, you will implement the ability to add filters to the image(s) the user selects from their photo.

1. You must add at least 5 filters. This page lists the filters that you can use. Note that some simply take in an `inputImage` parameter, while others have more parameters such as the `CIMotionBlur`, `CIColorControls`, etc. Use at least two or three of filters with a bit more complexity than just the `inputImage`.
2. Add whatever UI elements you want to the `ImagePostViewController` in order for them to add filters to their image after they've selected one. For the filters that require other parameters, add UI to allow the user to adjust the filter such as a slider for brightness, blur amount, etc.
3. Ensure that the controls to add your filters, adjust them, etc. are only available to the user at the apropriate time. For example, you shouldn't let the user add a filter if they haven't selected an image yet. And it doesn't make sense to show the adjustment UI if they selected a filter that has no adjustment.

Go Further

- Clean up the UI of the app, either with the UI you added to support filters. You're welcome to touch up the UI overall if

you wish as well.
- Allow for undoing and redoing of filter effects.