

 **IvanCaldwell / ios-project-week-challenge-1**
forked from [LambdaSchool/ios-project-week-challenge-1](#)

No description, website, or topics provided.

Edit

[Manage topics](#)

1 commit

1 branch

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

 erica initial commit


Latest commit 0659504 on Oct 29

 [README.md](#)

initial commit

a month ago

README.md



Unit 1 Project Week: Books

Congratulations for finishing the first four course weeks of Lambda's iOS program!

It's time to take the skills you've learned and apply them to an independent project. This is the first time you're asked to work on a project of this scale so we'll give you some guidance to get you into the groove rather than making you select and design a project entirely on your own.

Your project spans an entire week. Your project instructions this week are closer to a Sprint Challenge than an afternoon Challenge in that they are sparse.

Treat this week like you're building this application at a company. Your client is your instructor. Your project managers (including your section lead) are your main support throughout the week.

The main objective of this week is to develop the minimum viable product (MVP) feature set listed below using the techniques, skills, and technologies you've studied here at Lambda.

Instructions

This project is open-ended. After you finish the requirement set listed below, spend your remaining time debugging and testing your product and implementing extra features, either suggestions listed below or those you come up with yourself. Run your ideas through a PM or your section lead and have them approve it before you get too creative. They're there to help you stay on task, on time, and deliver a product.

Remember: it's more important to deliver a working well-tested product than it is to have flashy but incomplete features.

Your API

Your project uses the Google Books API and Firebase data stores. Read this [Google Books Getting Started overview](#). It explains how the API works and how you can use it. API documentation discusses data formats and the operations you can perform. The more you understand the API, the better you'll be able to use it.

Requirements

You must implement each of these requirements in your application:

- Your model processes records that represent a book volume. Each record contains the following properties at a minimum:

1 of 3

12/10/18, 12:16 PM

- A title (sourced from Google's API)
- An image representing the book cover (sourced from Google's API) that is scaled down to a reasonable size so it does not take up too much space.
- A `String`-based book review sourced from your application user
- A property that represents whether or not your user has read this book.
- Your user can search for books with a text query. Matching books are displayed to the user.
- Your user can tag books to categorize them in "bookshelves". A single book may exist in more than one shelf. For example, your user might have a "Favorites" bookshelf as well as a "Thriller-Suspense" one.
- Your user can view a list of all their bookshelves.
- The book records including review, etc, are stored at Firebase and are read into the app whenever the user launches your app.
- Your user can insert, update, and delete books in their bookshelf or move them from one bookshelf to another. This information is stored on their behalf at Firebase and is read in whenever the user launches your application.

Extra Requirements

After finishing your required app elements, you are not finished with your project. We expect you to implement at least two extra features. You're at liberty to choose the features you'd like to implement.

These extra requirements (or others you come up with on your own) are not necessarily things you have learned throughout the course. Treat them as if your client adds a feature at a late date that you don't already know how to implement. How will you go about learning how to bring that feature into your product?

A few ideas are as follows:

- Allow more than one user to use the same database without conflicting or overwriting each other's data. The same information must be accessible at each launch of the program, so be sure to use good Firebase hygiene.
- When you mark a book as read, add it also to a pre-made "Already read" bookshelf. Books later marked as unread must be removed from this bookshelf.
- Share a link to a book (there is one to the Google Play store in the JSON), or your review of it to Twitter, Facebook, etc.
- Create and display a QRCode with a link to the book. Friends with mobile phones can read the QRCode and follow your link.
- Find the book on Amazon and open it in the Amazon app, or on a web browser.
- Print out your review including an image of the cover.
- Add a button to read out your review using a speech synthesizer.
- Create a "Recently read" or "Recently reviewed" screen that uses the modification time for the "didRead" or "review" properties to select items to showcase.

Commit History

Commit your code often and meaningfully. This helps both you (in case you ever need to return to old code for any number of reasons) and anyone you're working with.

Commit regularly when you've made a complete and quantifiable change to your code. There is no set amount of time that should happen or amount of code written between commits; simply use your best judgement. Adding new features. Fixing a bug. A good rule to follow is if what you've done can't be summarized in about 50 characters, you **may** have gone too long before committing. If you are committing more than one change at once, you're not committing often enough.

Your PMs will check that you are committing regularly, with clear and understandable commit messages.

Work Alone

Your project week project is an individual effort. Your work is your own and your classmates are responsible for their own project time as well. Your application reflects your proficiency as a developer and your command of the concepts and techniques in the first iOS Unit.

The twenty minute rule continues to apply. Seek help from your PM or Section Lead if you run into problems that are keeping you from moving forward or leading you towards a mire you'll get bogged down in. Your team is there to help you. Just make sure you give your fellow students the space they need to succeed, too.

Planning

Design your interfaces and schedule your time carefully. Keep your scope narrow and focused. Even though you have a week to work in, there's a lot of work to do and you need to make sure you're moving forward with good progress with an end in sight and will have a completed application to present.

Thoroughly test and debug your application well before it's time to present it. Stick to your schedule and make sure that testing and refinement is part of that schedule.

Remember: Simpler and working is better than fancy and unfinished.

Progress Dashboard

-

Download checked items