App Name: Bookworm
Team Members: Raify Hidalgo, Yash Jalan
Language: Objective - C

Overview:
Our app is a social platform specifically for book enthusiasts and users who are interested in connecting with others who have similar interests in literature and who want to trade or exchange books with nearby users. We want users to be able to register and login to our app with the knowledge that they will be interacting with people that want to exchange books and are willing to be contacted for trade. The user will be able to upload information about books they own, which will be stored in a database within Firebase, and the user will be prompted to add a location to be added through the use of MapKit. Other features include contacting other users through email, seeing a map view of pinned locations of a user's choice, and viewing the full library of all owned books by users.

Implemented Features:
- Firebase database
- Firebase authentication through email/password and through Google sign in
- Gradient overlay on landing page
- User registration with firebase
- Animated book opening segue using animateWithDuration, UIViewAnimationOptionCurveEaseIn, and CATransform3DIdentity
- Book page flip sound using AudioToolbox
- UI Tableview for library collection
- MapKit and CoreLocation for map view plus adding location of users books
- MessageUI for using Apple's email service to send emails to contacted users
- FIRStorage (Cloud Storage, for images)
- UIImagePickerController for use of camera

Team Member Responsibilities:
Raify:
- Firebase implementation & data architecture
- Login - Registration - Posts view pages
- Book opening animation
- Reading/writing data from database
- Joint-control over design of app
- UI Scroll View
- Proposal and final document writing
- Logo editing

Yash:

- MapKit implementation
- Camera/photo taking implementation
- Book collection - book detail - user detail pages
- UI Table View
- Send email to contacts: “share with friends”
- Joint-control over design of app
- Flowchart display
- Sound

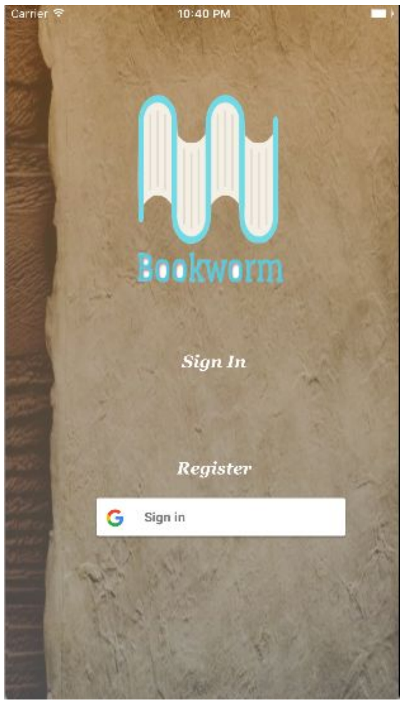
Technical Challenges:

There were multiple instances when we had to decide between working on an idea that sounded cool to integrate and working on what was feasible at the time. For example, we wanted to add smaller features for personalization, such as a book enthusiast category per user and a book rarity score but those features were thought of near the end of the process, too close to commit to. We also found ourselves debating on integrating features that seemed like most apps had versus using a feature that was more integral to the use of our app. The main example here is our decision to do contact by email and user posts instead of chat messaging. We didn’t believe that the app ultimately is about instant messaging but about the use of the contact and map to find other users to trade with.

In other aspects, we found technical challenges with establishing the proper database. Multiple times revisions had to be made to the database when we figured out what was the best way we could access the data and display its content. Going in we wanted to establish a hierarchy as flat as possible to avoid looping through but instead we found that certain nesting was needed to avoid querying through multiple tables in the database.

We also had difficulty at one point with animation between segues. We wanted to make a curl animation that would act as a page flip. There was an integrated animation that did a page curl within Xcode but it added a touch point and would return the user to the previous page. By doing this it would also keep us from interacting with text fields so we had to get rid of it. We continued to attempt other animations that were relevant to the act of reading a book until we settled on animating a book opening for a view that was about to segue to another. This can be seen in the segues from sign in view controllers and the registration view controller.

Bookworm Flow and More Comments

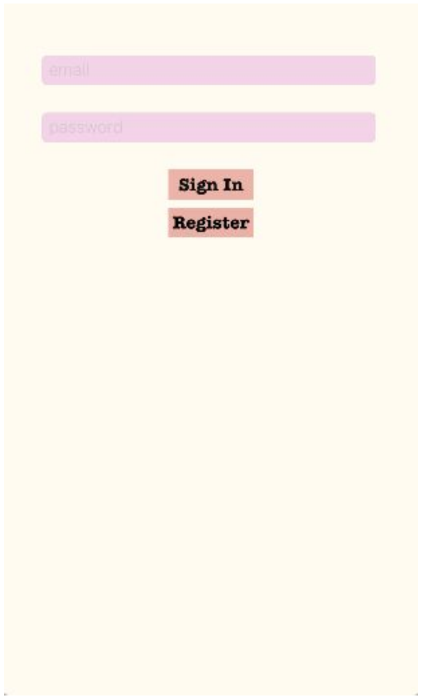


Our landing page (Signin view controller).. Logo was initially designed by Freepik. Through a free license we were allowed to use the framework he had which was the book design, as long as we provided this attribution and the license along with our submission. We added the title and changed the color and font.

Sign in button connects to our email sign in view controller

Register connects to our register view controller

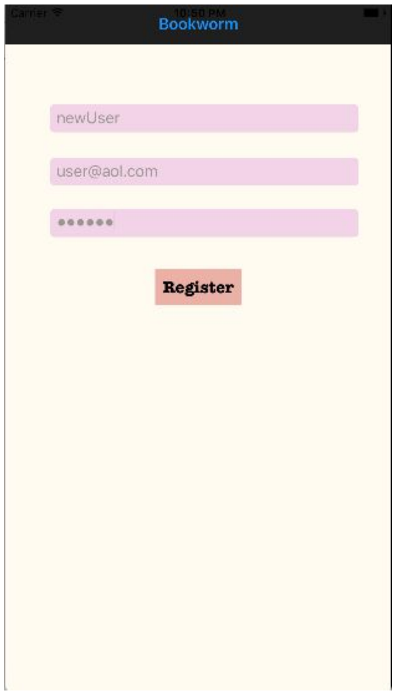
Google sign in button implemented through Firebase and Google's property



Email sign in view controller. Once a user clicks on 'sign in' in the previous page they are taken here. Authentication is handled by Firebase and we also do some checking for empty values in the fields.

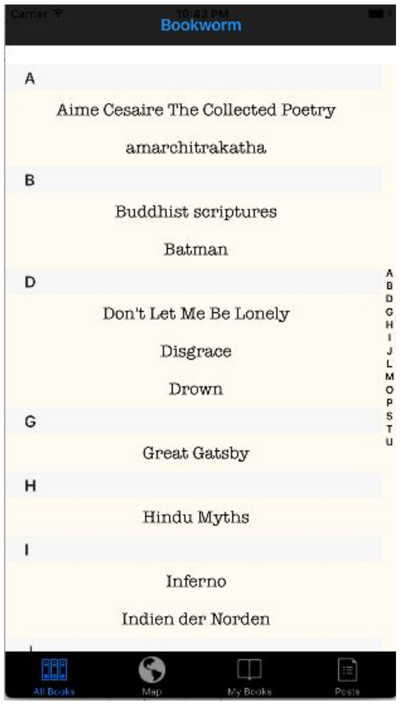
Initially we wondered whether or not to use anonymous sign ins or not to include it at all but we decided it was best to include it since the main point of contact was through email, so users should at least have some layer of protection there

Fun fact: Color scheme was changed over 10 times in an attempt to find something we both agreed upon

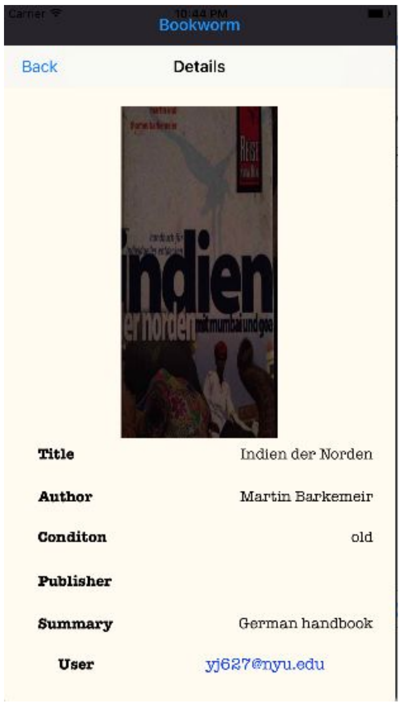


Register view controller. A user can land on here either through the landing page or the sign in page.

For future projects we would like to implement multiple steps for registration. There have been recent trends in apps (and web apps) that include 2 - 4 steps in the registration flow. A lot of this is for analytics and making sure the user can give as much information about the type of user they are without alienating them with many fields at once.

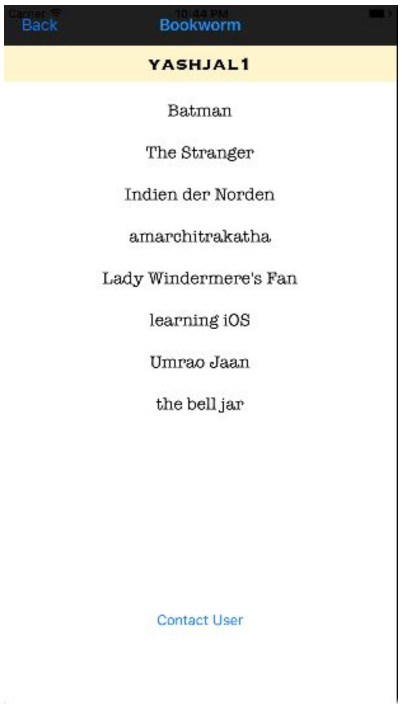


Book listings. Here we have a UI Table that contains the collection of all user uploaded books and their details. One of the more difficult decisions we had to make was whether or not to commit to file sharing and uploading actual book files. This is still something we are interested in but it most likely shifts the focus of the app away from trade and into something closer to iBook with strangers. Maybe not a bad idea but we didn't feel comfortable we could fully implement file sharing and test out book sharing in time. Instead we have users add books by their details and they are retained in our database as owned by that user.



Title	Indien der Norden
Author	Martin Barkemeir
Conditon	old
Publisher	
Summary	German handbook
User	yj627@nyu.edu

Clicking on any book within the UI Table in leads to the book detail view controller. The page is updated for every book. The last thing we added was images so only a small sample of books actually have images but this shows the type of template we were thinking of.



YASHJAL1

Batman

The Stranger

Indien der Norden

amarchitrakatha

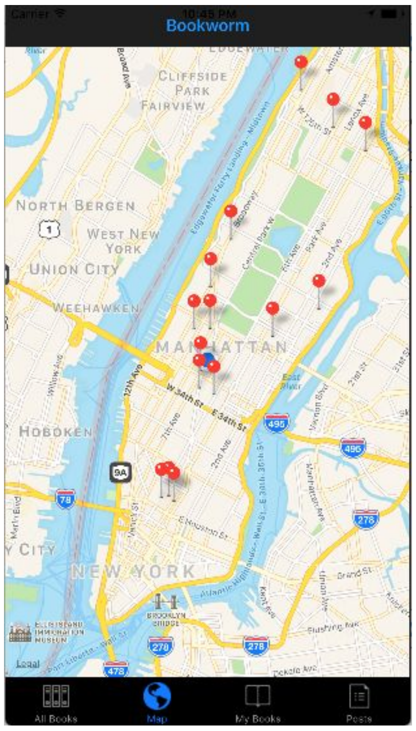
Lady Windermere's Fan

learning iOS

Umrao Jaan

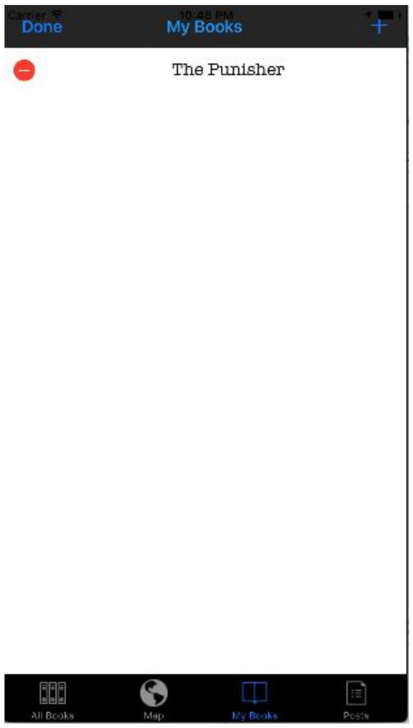
the bell jar

Clicking on the blue user field in the book detail page leads here. This is the user detail page which shows the username of the user that was clicked along with the collection of books which they have listed. The contact user button brings up the create email view that Apple has for emails, populated with the user’s outbound email and an initial message. This is meant to be the main point for a user to contact another for any interests in their books.



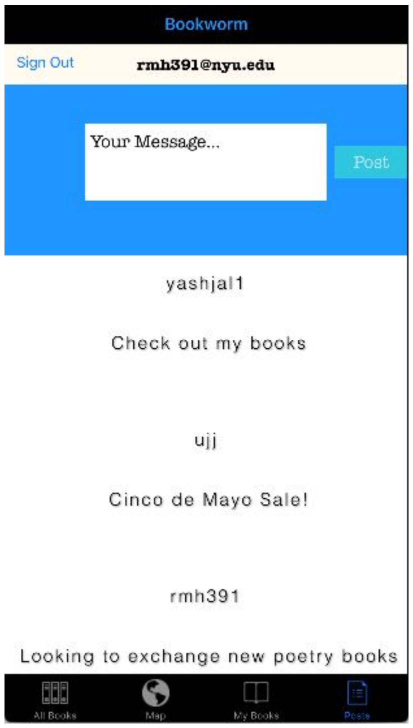
Map view. We used MapKit to create this view and Core Location to help us find our locations and pin according to it. However we can also pin according to where we place the pin on the map. This was pretty fun to learn about but it did force us to tackle some interesting technical difficulties.

The focal point of the difficulty was a difference in Xcode and iOS versions between us where in one environment the code would crash on access to the map but would work perfectly on the more recent version. Goes to show, always have to stay up to date.



The Punisher

This is my books page. Here we show the current logged in user's collection of books. This user right now only has one book sadly, but we can either delete the book if need be or add a new book to the collection. Once the book is added the database updates immediately, which is good since Firebase markets themselves heavily based on their "real time database".



yashjal1

Check out my books

ujj

Cinco de Mayo Sale!

rmh391

Looking to exchange new poetry books

Speaking of real time, we had some difficulties with the posts section because of the timing with updating and downloading data from the database. Our main concern with posts was to provide a place where users could indicate to others what they have or are looking for. We wanted to read from the database to display all the posts, update the database if a message is posted, and read from it again and update the scroll view. To do this we needed to dispatch a delay before reading from the updated database or else we would read before it was updated.

This is an area that we wanted to flesh out more as well but first we needed to get this down to turn in. It would be fun to implement a “For Trade” or “Asking For” flow for posting based off of the needs of the users

Looking Ahead

This was a fun project to work on and we learned a lot about the workings of Apple and the properties of their products. It was a bit of a steep learning curve since we both came into the class without any prior knowledge of Objective-C, iPhone development, or app design but we are both glad we took took it.

As far as what we want to do next, Swift seemed like a fun language to learn more about as well so that is on the horizon. We definitely want to use what we learned for this app and apply the good (and learn from the difficult) onto new ones. I personally (Raify) want to work on an app for poetry which brings the often collaborative but not digital environment of poetry writing and sharing to the space of the iPhone.

With that said, thank you for the time spent teaching us this semester Prof. Hull - Raify & Yash