How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

- Description
- Intended User
- Features
- User Interface Mocks
  - Screen 1
  - Screen 2
  - Screen 3
  - Screen 4
  - Screen 5
  - Screen 6
  - Screen 7
  - Screen 8
- Key Considerations
  - How will your app handle data persistence?
  - Describe any corner cases in the UX.
  - Describe any libraries you'll be using and share your reasoning for including them.
  - Describe how you will implement Google Play Services.
- Next Steps: Required Tasks
  - Task 1: Initial Project Setup
  - Task 2: Add Navigation Drawer, Login Dialog and Signup Screens.
  - Task 3: Create Layouts for Books, Setup recyclerview and adapter for fetching books.
  - Task 4: Configure the Database, Content Provider, for storing Book Details
  - Task 5: Setup Detail View for Books, Dialog for adding review and Reviews list
  - Task 6: Create List for Meetups and Forms for adding meetups
  - Task 7: Add a Maps fragment for ability to switch between list and maps while displaying book meetups.
  - Task 8: Add a dialog / sliding layer for providing sorting option the meetups and books.
  - Task 9: Configure the widget for displaying the Top Rated books

GitHub Username: https://github.com/sumeetmoray

Community Library

Description

Community Library is a book discovery application which helps user find new and interesting books. Users can discover new books with the help of ratings and reviews for the books organized according to various genres.
The application allows the user to Write reviews for the books they have read and also rate these books as per their liking.

The application will have its own backend API, which uses Postgres DB to store the data. The Backend API is built using Java Jersey Framework and can be hosted on AWS, or DigitalOcean cloud hosting service.

This application can also act as a public repository / public catalog of book reviews which can be shared and used by other services for marketing and publicity requirements.
This application is similar to “GoodReads” in concept. But still add and extend the existing concept by adding new features like “Book Meetups”.
The application will consist “Book Meetups” feature. This feature helps book lovers to create / organize offline meetups in their community. The meetups can be organized to share and discuss any particular book/ book’s or subject. The meetups can be sorted / listed by their distance from the user (as per users current location). This helps the user to discover the meetups which are nearest to the user.

The book Meetup Feature demonstrates the usage of Google play services which uses the two libraries “Location” and “Maps” from the play services.

Intended User

This application is intended for book lovers who wish to share their reading experiences with other books readers and also find / discover new and interesting books to read with the help of reviews and suggestions from other book lovers.

Features

List the main features of your app. For example:

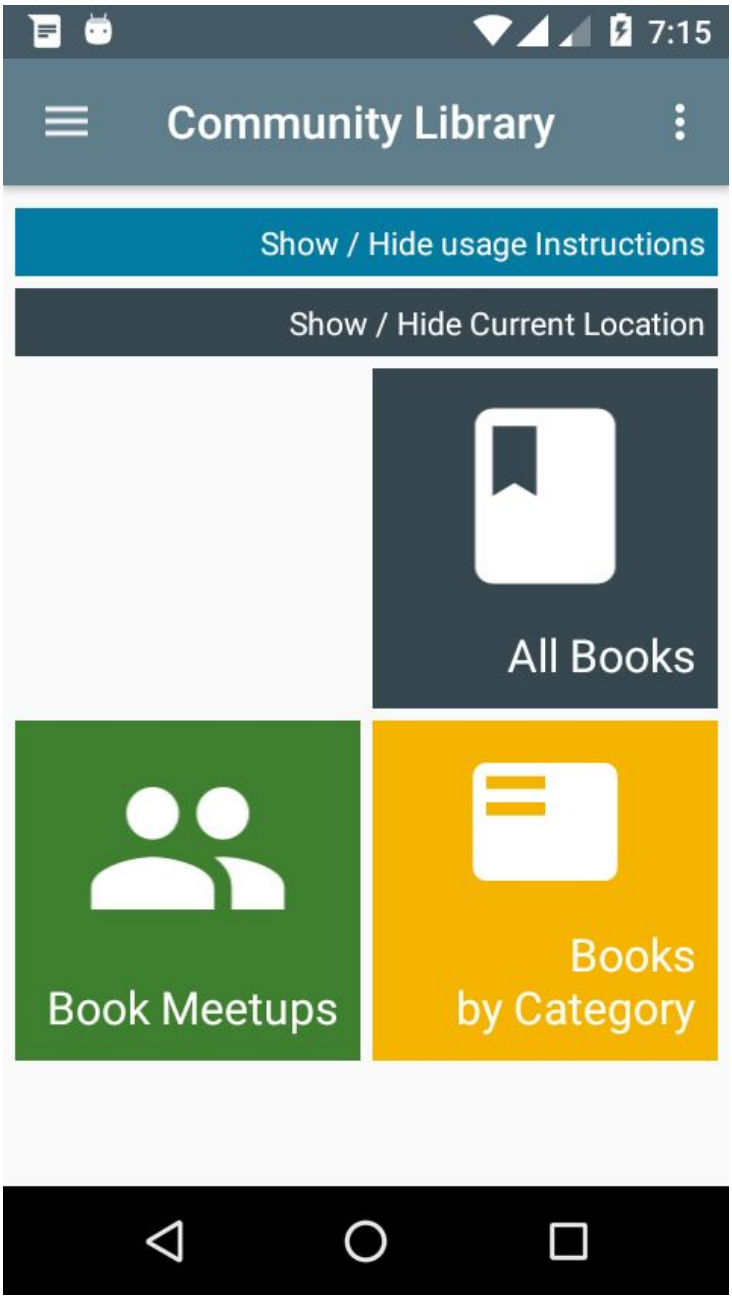
- Add new books to the Backend Server Database.
- Create new Genres / Categories for the books
- Sort Books by their title, rating
- Users can Mark books as Favourite and access all their favourite books in one place.
- Ability for a person to Register, login, write Review and rate any book.
- Ability to read reviews for any book
- Create Meetups for book lovers which can be assigned a location on the Maps.
- Meetups can be sorted by distance and the app has the ability to discover the meetups nearest to you.

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1

This is a mockup for Home Screen. Which shows the options to navigate to the major portions of the Application.



Screen 2

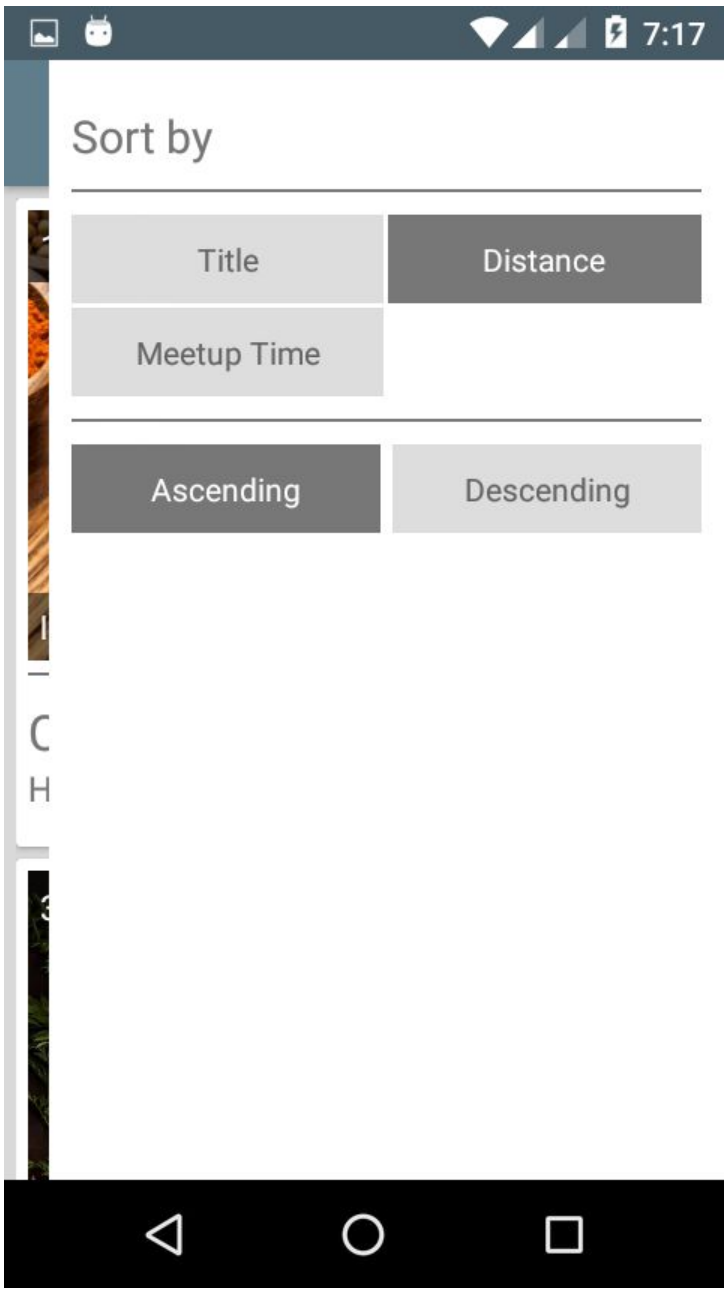
The Mockup for the Book Meetup Screen in a list form. The screen will have a toggle button on the action Bar. This toggle button will enable to User to toggle between the list or Map.

Screen 3

The Mockup which shows how the user can toggle between maps and list for book meetups. This shows the Map view and how meetups and their details can be displayed on the map.

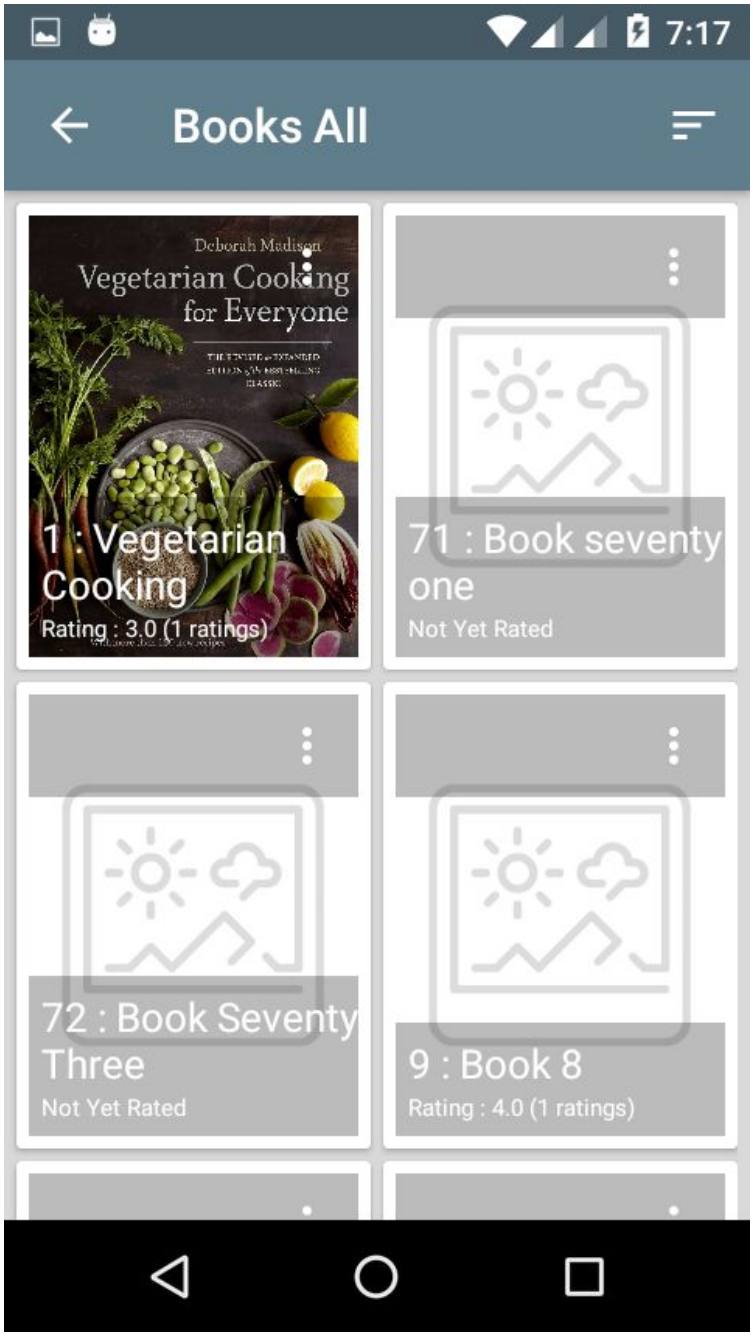
Screen 4

The Mockup for the same Book Meetups Screen described above. It shows the sliding layer which opens from the right and how user can select through various sorting options to sort through the list.

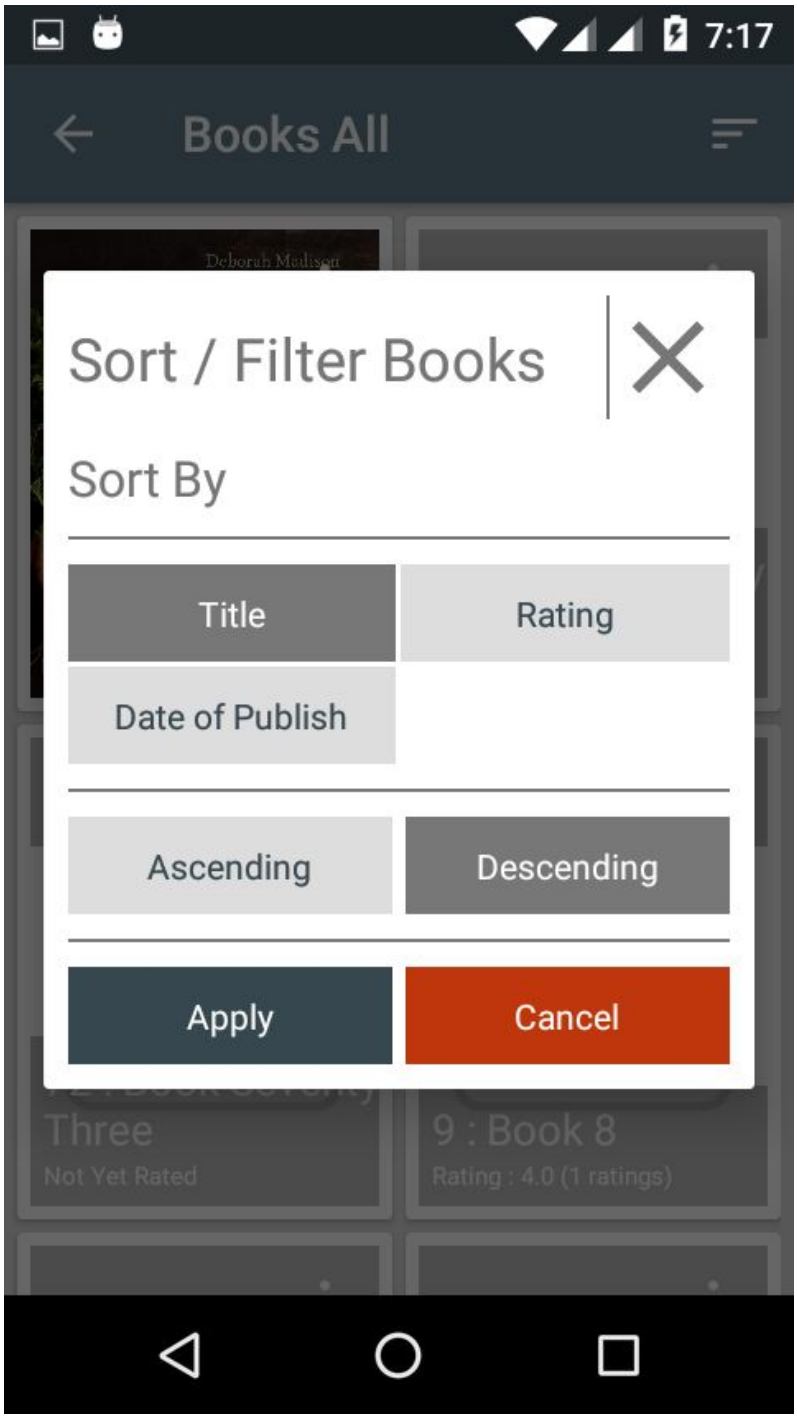


Screen 5

The Mockup for the screen which display all the books available in the database. These books can be sorted with the help of sort dialog triggered through sort menu option on the action bar.

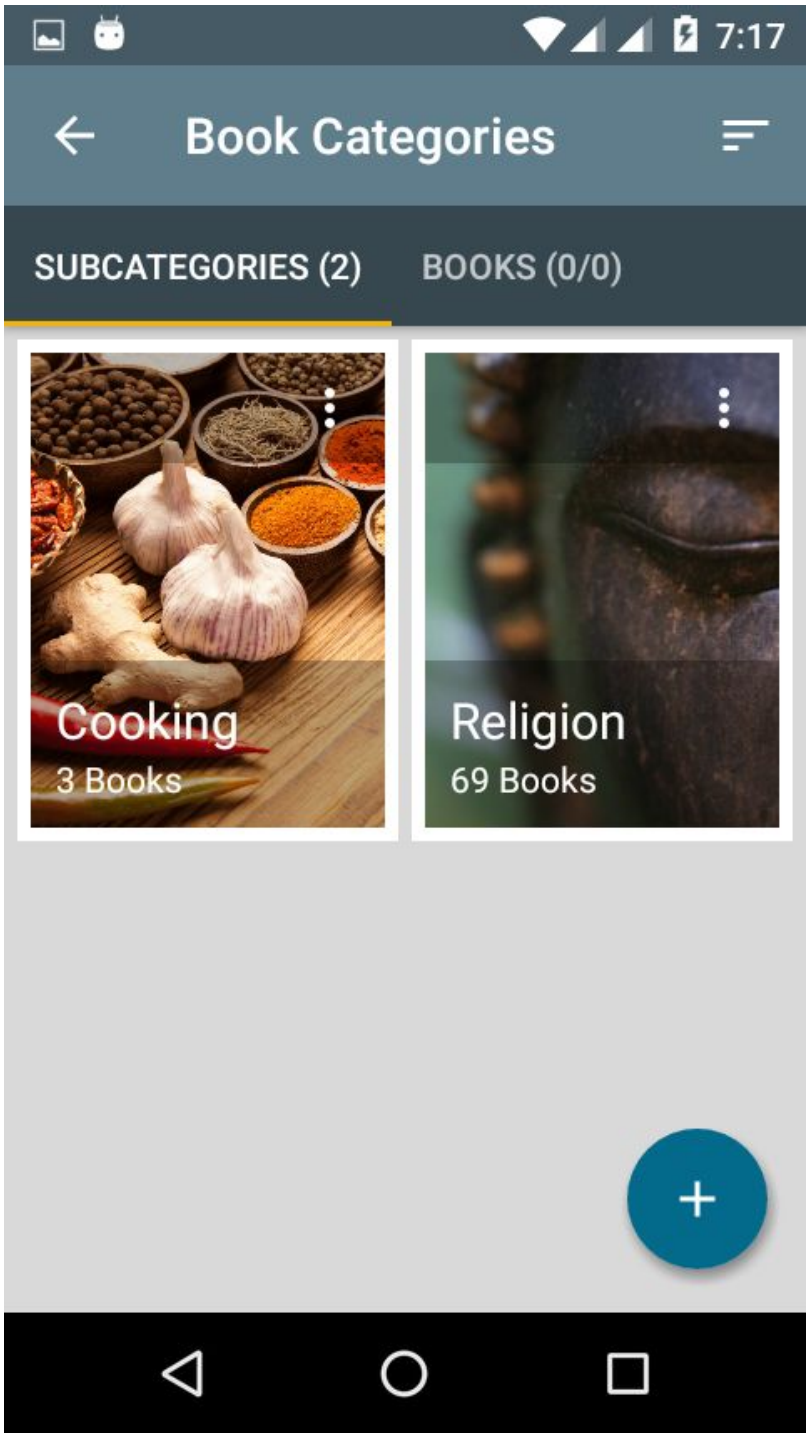


The following mockup shows the dialog used for sorting the books list in the same above screen.



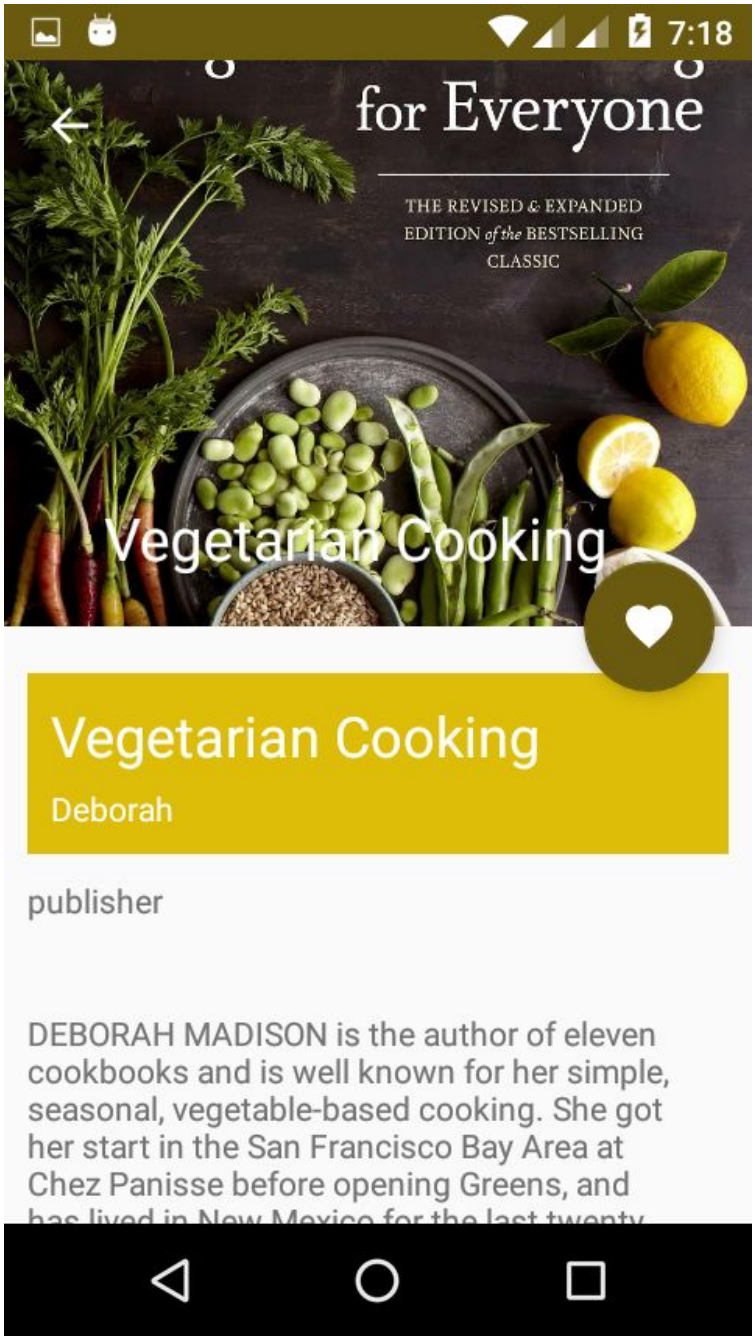
Screen 6

The Mockup for the screen where books are arranged according to the Genres or categories.

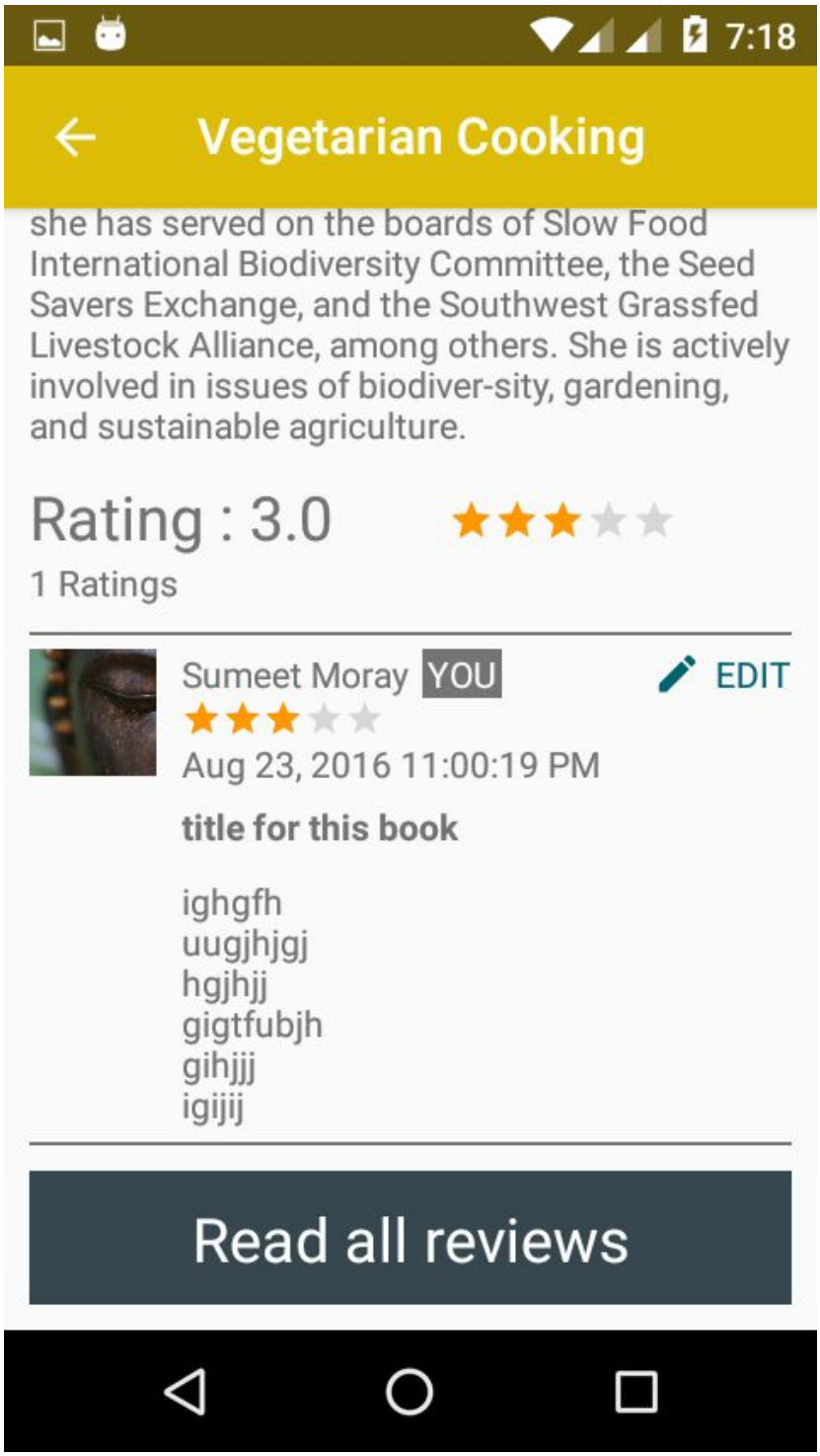


Screen 7

The Mockup for the Book Details screen which shows the various details about the book like its description, Author Name, Reviews and Rating. It also provides the option to favorite the book. Which is implemented by using FAB as an anchor with toolbar.

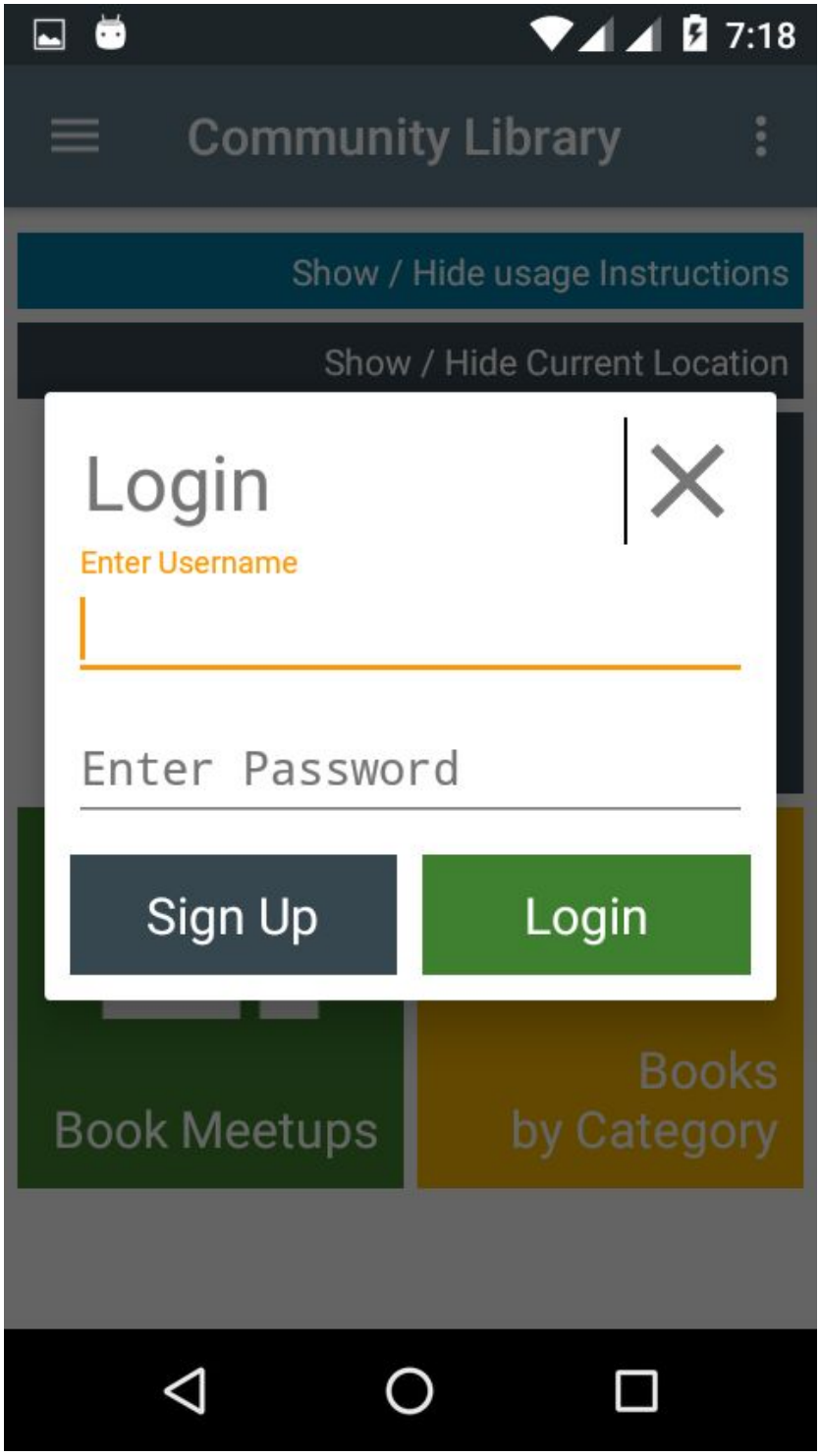


The portion of the same screen which shows how users can write and edit their reviews.



Screen 8

The Mockup for the Login Screen which shows the Login Dialog.



Key Considerations

How will your app handle data persistence?

The app will use internal SQLite database to storing the book details. We can also build a content provider to access the database.

This will help enable us to use loaders while populating the Books List.

Describe any corner cases in the UX.

CASE 1:

Providing the ability to hide the Floating Action button while scrolling down.
The Floating action button obscures the views behind it. This prevents the users from accessing the options behind the FAB. This problem can be solved by hiding the FAB while scrolling down. The default library does not provide features to automatically handle hiding the FAB while scrolling down. Therefore this needs a custom implementation.
The feature can be developed by implementing the recyclerview scroll listener. This will provide the value of dx, dy as per scroll magnitude. The dy value can be used to obtain the direction of scrolling and decide when the hide the fab and when to show the fab. The negative dy value indicates scrolling down which positive dy value indicates scrolling up. The listener can use this logic to show and hide the fab when required.

CASE 2:

The application provides the ability to toggle/ switch between list and maps for book meetups. This can be complicated. The list and maps are implemented as two different fragments. The fragments are replaced when the toggle button on the action bar menu is clicked.

CASE 3:

The screen also provides the sliding layer for sorting the results. The sliding layer is a very convenient way to sort the results obtained in the list. We need to implement the custom UI logic for creating the options and modularize them in a fragment which can be reused whenever required.

Describe any libraries you'll be using and share your reasoning for including them.

For example, Picasso or Glide to handle the loading and caching of images.

The application will use the following libraries.
Picasso for Loading images.

Butterknife for easy binding of views it saves time on binding views and assigning click listeners to the UI views with the help of easy to understand android annotations.

Retrofit for making network calls. - Retrofit is a type safe library for making network calls. This library simplifies the process of making network calls and fetching and parsing the results on the background thread and displaying them on the main thread. Retrofit makes the process of making network calls type safe and simple and saves time by cutting out lot of boilerplate code involved in the process.

Dagger 2 for Dependency Injection : Dependency Injection simplifies and modularize the application code. Dagger 2 is one of the best and most popular dependency Injection libraries currently available.

UCrop Image Cropping Library - When adding new books the users are also expected to add book covers. This requires the ability to crop the images as per requirements. This library helps the user to crop the images before adding it to to database.

Icepick : Icepick simplifies the task of saving and restoring activity instance state in the activity lifecycle with the help of simple annotations.

Play Services Location : This library Provides a fused location provider api. WHich helps a battery efficient way to fetch user location in the form of latitude and longitude.

Play Services Maps : This library provide the maps fragment which can be used to show meetups as markers on the maps.

Describe how you will implement Google Play Services.

Describe which Google Play Services you will use and how.

The application uses the following libraries as a part of google play services :
compile 'com.google.android.gms:play-services-maps:9.4.0'
compile 'com.google.android.gms:play-services-location:9.4.0'

The play services Location library is used for obtaining the current location latitude and longitude of the user. This latitude and longitude can be passed as query parameters in the network call in order to fetch the book meetups nearest to the user.

The maps library will help the user to views the meetups as markers on the map. This will provide a more fascinating and aesthetically pleasing way to discover the meetups.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Initial Project Setup

Setup basic Libraries, Dependency Injection, Model Classes and create Retrofit Contract Interfaces for making network calls

In the initial Setup we configure the very basic tasks in the project. Without these tasks the development cannot begin.

In this initial project setup we will add gradle dependencies for basic core libraries like butterknife, Retrofit, Picasso, and google play services libraries.
We will also add android v4, v7, appcompat and design support library. To provide the basic material design components and other components for the app.

We will configure dependency Injection for Dagger 2 by overriding the application class and creating modules and components for basic modules for Injecting retrofit classes.

Task 2: Add Navigation Drawer, Login Dialog and Signup Screens.

In this task we will add Navigation Drawer for the Home screen. And also add login and Sign up Screens.

We will setup validation and error messages for sign up screens.

Task 3: Create Layouts for Books, Setup recyclerview and adapter for fetching books.

In this task we will add Navigation Drawer for the Home screen. And also add login and Sign-Up Screens.

We will setup validation and error messages for sign up screens.

Task 4: Configure the Database, Content Provider, for storing Book Details

Build the Content Provider

- Create Contract Classes for the SQlite Database Helper
- Implement Content Provider Class and Override the four basic methods
- Use content resolver to insert and store the book Information obtained from making a network call.

Task 5: Setup Detail View for Books, Dialog for adding review and Reviews list

In this task we will design the detail view for displaying book Information. This view will have a FAB to favourite the book and it will also have an option to write and Edit the Rate and Review the book by the user which is logged in.

Task 6: Create List for Meetups and Forms for adding meetups

We will create this list in a same way we created the list for books.

- Add a Floating action button to provide an option for creating meetup.
- Build a Create Meetup Screen.
- Validate all the form Inputs and show errors by using setError Method on the EditText

Task 7: Add a Maps fragment for ability to switch between list and maps while displaying book meetups.

- Create action bar menu for displaying the toggle action view
- Create Map Fragment and implement the callback onMapReady()
- This callback would contain the code for adding markers to the map. The map would also have a horizontal list for displaying various options.

Task 8: Add a dialog / sliding layer for providing sorting option the meetups and books.

Add a sliding layer which will contain sorting options in order to sort the meetups. It requires to us to implement a custom UI logic for highlighting and dimming the buttons when clicked.

Task 9: Configure the widget for displaying the Top Rated books

Describe the next task. List the subtasks. For example:

- Create app-widget provider configuration in xml file
- A class that extends a remote Views Service
- A class which extends App widget provider.
- A Remote Views Factory Class

Add as many tasks as you need to complete your app.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"