

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](#)

[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: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Your Next Task](#)

[Task 4: Your Next Task](#)

[Task 5: Your Next Task](#)

GitHub Username: doublej89

Book Watch

Description

This is a book cataloging app (and not a book reader app!). It helps users keep track of the books they own, lend, intend to read, and are reading. The app comes with a default number of bookshelves where users can store references to the books they own and borrowed. These references include the title, author name, isbn number, cover image, page count, publisher,

publishing date, and genera of a book. One reference can exist in only one bookshelf, mimicking how real books can only exist in a single shelf. Additional bookshelves can be added according to the users needs.

Users can manually create a reference or search for a reference by the book's title, author name or ISBN number. The app also supports a barcode scanner which can retrieve the ISBN number of a physical copy and then perform a search. The app can search both locally and online using the google books api (locally means the bookshelves).

The app also allows users to access every book from all available shelves or access the contents of a single shelf (using the same list/detail flow). Books (references) can be modified or deleted, shelves can also be added or renamed.

Finally, users can keep track of the page number they're on of the book they're reading.

The purpose of this app is to make life slightly easier for anyone who reads daily (or frequently) and would like to keep track of the books they own, or have loaned. Readers who might be in possession of a large number of books may not want to repurchase the same books (as I can say from personal experience!). They also would want to remember to whom they've lend which books, and which other books they themselves have borrowed. Writing all this info down with physical pen and paper may be one solution, but that doesn't exploit the convenience of using a mobile app

Intended User

Anyone who reads and collects a lot of books, both physical and electronic.

Features

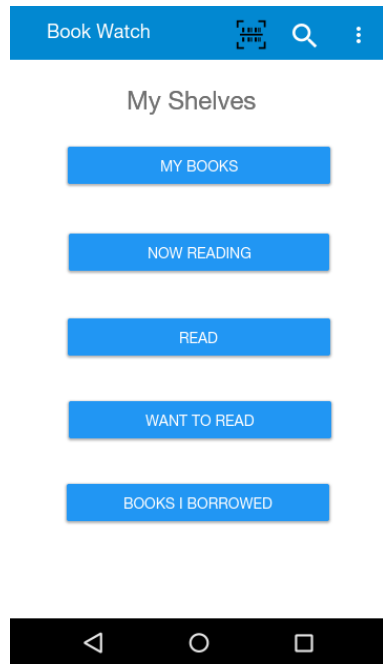
Main features of Book Watch:

- Stores book references
- References are grouped by shelves
- References in reference lists can be sorted by title, author or date of publish
- Searches for book references from Google Books by title, author, or ISBN number
- Scans barcode of a book and reveals information about that book from barcode

User Interface Mocks

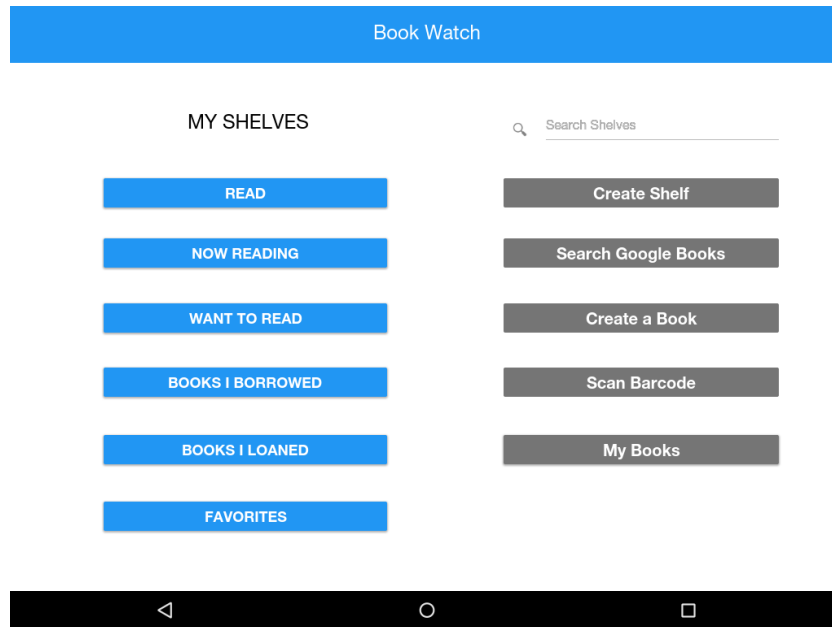
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1



This is the main screen of Book Watch. It shows all the shelves available to the user. A number of shelves are provided by default but the user can add more shelves if they choose. That option is available, along with other functionalities, as a menu item in the overflow menu.

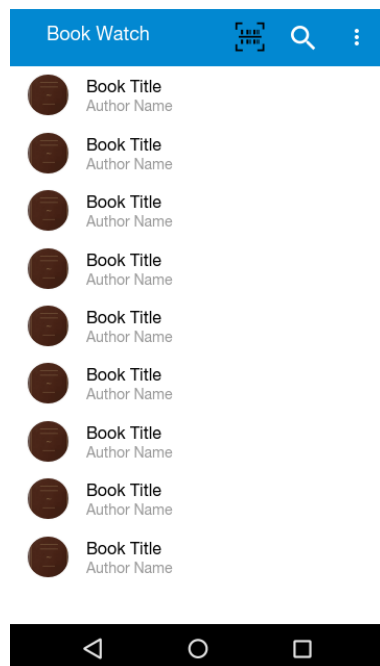
(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)



This is the tablet version of the main screen. The menu items in the overflow menu of the mobile version are placed on the right hand side of the screen.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)

Screen 2






This is the generic list that shows up whenever the user wants to see all books stored in all shelves (granted there are books in at least one shelf), books stored in a single shelf, or search results, either from the internet (google books) or locally.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)

Screen 3


Book Watch

Details

Use Status

Book Title



Author Name

genera

Page Count

AverageRatings


ISBN Number

Publisher




Publishing Date

Description:

+



Book Watch

Details

Use Status

Have you read this book?

Yes

☒

No

☐

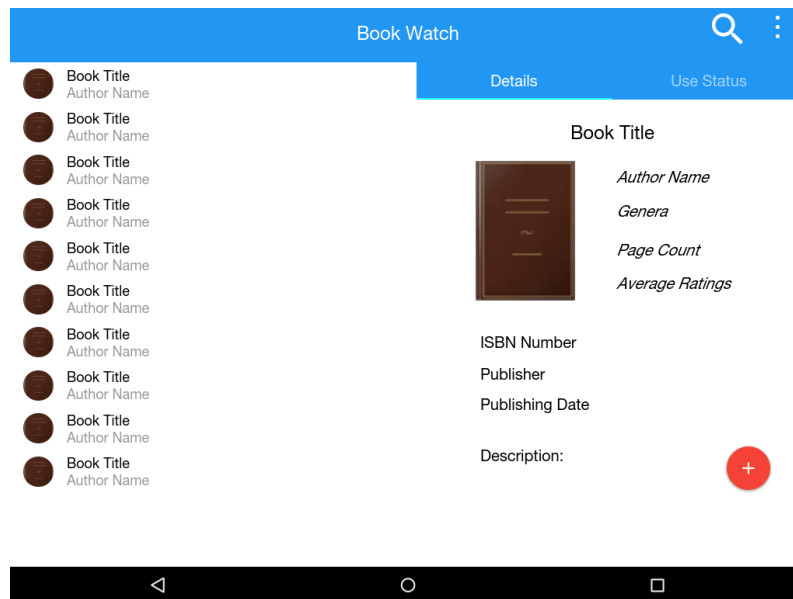
This book was loaned to :

Name of borrower

+

This is the screen that shows up when a list item is selected. The details tab shows the general details of the book. The use status tab asks whether the book is already read, and if the book was lend to someone. The floating action button is used to either store the book to a particular shelf if it hasn't been, or update the details if it has and the user chooses to change any detail.

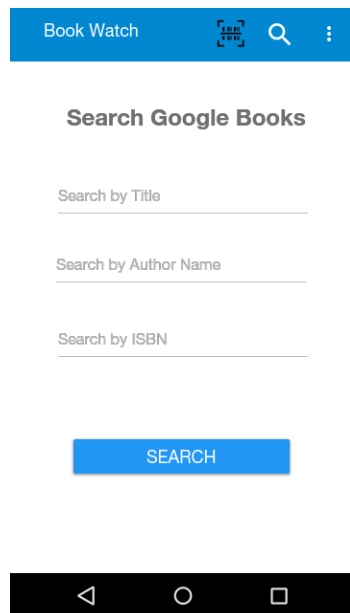
(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)



This is the tablet version of the master detail flow.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)

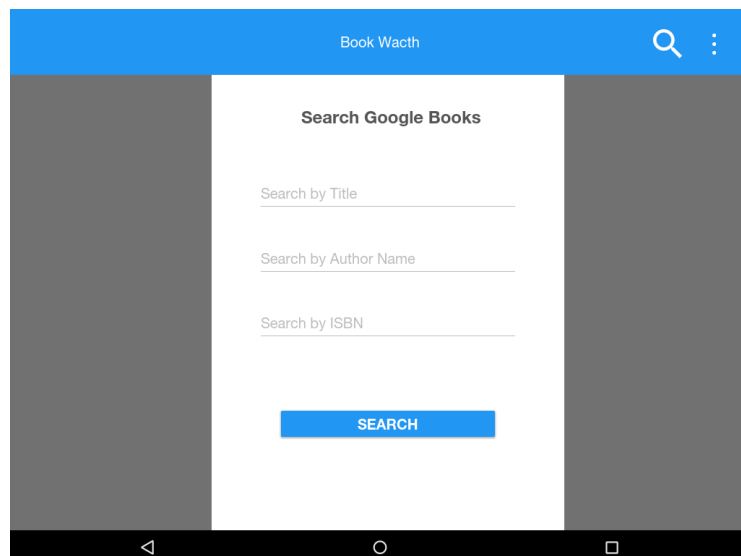
Screen 4



A mobile app search screen mockup. At the top is a blue header bar with the text "Book Watch" on the left, a book icon in the center, and a magnifying glass icon and a three-dot menu icon on the right. Below the header, the text "Search Google Books" is centered. There are three search input fields, each with a placeholder text: "Search by Title", "Search by Author Name", and "Search by ISBN". Below these fields is a blue button with the text "SEARCH". At the bottom of the screen is a black navigation bar with three white icons: a back arrow, a circle, and a square.

This is the screen that shows up if the user decides to search google books. The user can enter The title, author name or ISBN of the book.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)



A tablet app search screen mockup. It features a blue header bar with the text "Book Wacth" (note the typo) on the left, a magnifying glass icon, and a three-dot menu icon on the right. The main content area is white and contains the text "Search Google Books" centered. Below this are three search input fields with placeholder texts: "Search by Title", "Search by Author Name", and "Search by ISBN". At the bottom of the white area is a blue button with the text "SEARCH". The screen is flanked by two vertical grey bars on the left and right sides. At the bottom is a black navigation bar with three white icons: a back arrow, a circle, and a square.

The tablet version of the search page.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)

Screen 5

Book Watch

Details Use Status

Create a Book Reference
(Title, Author, and ISBN are mandatory)

Set Title

Add Thumbnail

Set Author's Name

Set Genera

Set ISBN

Set Page Count

Set Publisher

Set Publishing Date



Set Description

CREATE BOOK

This is the screen that shows up if the user wants to manually create a page. The user can set everything from the title to the description, publisher, etc. The use status tab is exactly the one shown above.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)

Book Watch



Create a Book Reference

(Title, Author, and ISBN are Mandatory)

Set Title

Add Thumbnail

Set Author Name

Set Genera

Set ISBN

Set Page Count

Set Publisher

Set Publishing Date

Description

Have you read this book?

Yes

No

☒

☐

This book was loaned to:

Create Book Reference

The tablet version.

(Note: The design presented here is by no means final. I will implement a more palatable design as soon as I have a functional app)

Key Considerations

How will your app handle data persistence?

The app will use a local Content Provider that will interact with a local sqlite database that will maintain at least two database tables. One will store shelf names while the other will store book references with each column representing a particular attribute (like title, author name, etc.)

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

I'll be using Picasso, for loading and caching of images; ZXing(zebra crossing) for barcode scanning. ZXing itself isn't a barcode scanner, but will interact with a scanning app on the host app's behalf.

Describe how you will implement Google Play Services.

I will use AdMob and Analytics.

To implement AdMob, first I'll add the google play services ads dependency in my app level build.gradle file. Then I'll add the AdView View to whichever layout where I want ads to display. In

the necessary Activity class corresponding to the layout, I'll inflate the AdView by its id and instantiate an AdView object. An AdRequest object will be instantiated to load the AdView, and depending on my needs I will either instantiate a bannerAd or an InterstitialAd object. Calling a bunch of methods on that object will display the ad. One of these methods is setAdUnitId(string) which receives a string corresponding to the ads unit id that can be generated in my AdMob account where I would already have registered my app.

To use Google analytics first I'll add the internet and the network access state permissions in the Manifest. Then I have to add the google play service analytics dependency in the app level. Then I'll have to generate the configuration file at the Google developers site which I'll then have to add to my project folder. Then I'll add the google services classpath to the project level build.gradle file and the google services plugin and the google play services analytics dependency to the app level gradle file. To track screens, I'll create a subclass of the Application class that will provide a Tracker object that will send the screen view of the screen(s) to Analytics. To track a screen I must initiate a tracker object via the Application subclass I already created in the Activity or Fragment corresponding to that screen. Then in the onResume method I have to write the code that will send the screen name to Analytics

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: Project Setup

Setting up project

- Create new project in Android Studio.
- Add all libraries and plugins including Picasso, ZXing, Google Services, etc.
- Populate Android Manifest with the necessary permissions. At the very least INTERNET and NETWORK_STATE permissions, since the app will communicate with internet.

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity and Fragment

Setting up database interface:

- Build the database Contract where the names and column names of all the database tables used are declared, and the interface that generates the necessary Uris to interact with the content provider is created
- Build the Database Helper that has the necessary code to create the database and all necessary tables based on the table names and column names declared in the database Contract.
- Build the Content Provider that contains all the code to facilitate querying, inserting, updating, deleting, and bulk inserting to the database tables using the Uris defined in the Contract.

Task 3: Your Next Task

Creating internet search interface:

- Create a new project in google cloud console and generate an api key to be used to fetch data using Google Books api
- Build an Intent Service to make Google Books api request in background thread.
- Create the search layout that receives user input and communicates with the Intent Service in order to fetch data from Google books.
- Create two more layouts corresponding to a list/detail flow where the list view (recycler view) will be populated by a list of book data approximating the user inputs.
- Allow the user to store a particular book reference to a particular shelf from the book's details view.
- Add menu item to the main activity that would trigger the search layout.

Task 4: Your Next Task

Enabling List/Detail layout to load from database:

- Equip the List/Detail layouts with Cursor loaders so that when a local search request is made, they can fetch book data from the database. The onCreateLoader() methods in the List/Detail flow must also have conditional statements to handle cases for when the user wants to see all available books and also when the user wants to see books of a particular shelf.

Task 5: Your Next Task

Main Activity interface, Create book, and create shelf:

- Equip main activity with the ability to display a list of all the available shelves (a list of buttons, not a list view). The shelves are loaded via Cursor loader

- Create a layout with a number of text fields corresponding to the number of book attributes, and a button that can pick a photo from the device storage. Once the mandatory text fields are filled, the book can be stored in the database. This activity can be accessed from the main activity menu.
- Another main activity menu item can open a dialog that will ask for a shelf name to store

Task 6:

Creating a barcode scanner and finalizing the app:

- Create a menuitem in Main Activity that uses a barcode scanning app if one is installed, or downloads one if it's not installed. Sends the scan results to the Intent Service where the ISBN is used to fetch book data from Google books.
- Create Multiple versions of the same layouts and put them in different drawable, layout folders corresponding to different screen sizes. Add necessary code in Activities and Fragments to account for different versions of the same layouts.
- Move all strings to the strings.xml file.
- Implement material design to bolster the look of the layouts
- Implement AdMod so that the user sees a banner ad whenever he opens a details page of a book.
- Create signed apk for the app

(Note: The tasks described are an assumption of what I may need to do during the construction of the app. Additional tasks may become necessary if I come across technical issues I haven't yet considered.)

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**"