
GitHub Username: victoraldir

BuddyBook

Description

BuddyBook provides a quick and simple way to catalogue and inventory the books you want to read later. Easily add new books with a simple search, or by scanning a barcode. Once your books are in, arrange them by folders, mark them as read, and even view the cover art and descriptions. All your data is going to be stored in your account and automatically synchronized when you login the app. BuddyBook also helps you to keep in track with books you lend. It will regularly remember about your lendings so that you can remember asking them back after while.

BuddyBook uses Google Books APIs to fetch book information

Intended User

This is an app for book readers. People who love to read either physical books or digital and for those who are always eager to get the next book

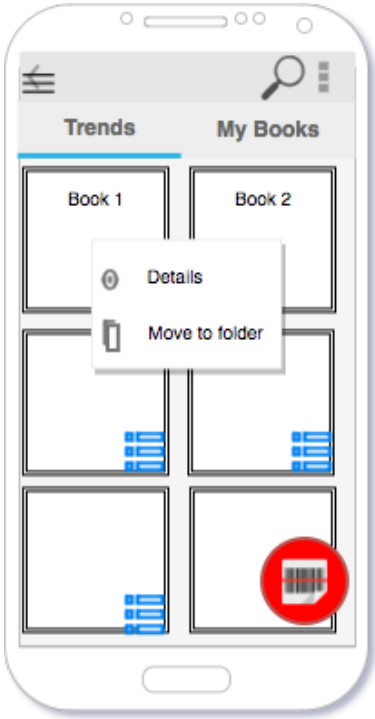
Features

- View book descriptions, author, and cover
- Saves information on cloud
- Add custom books
- Capture barcode for book searching
- Create wishlist
- Create custom folders
- Manage lendings

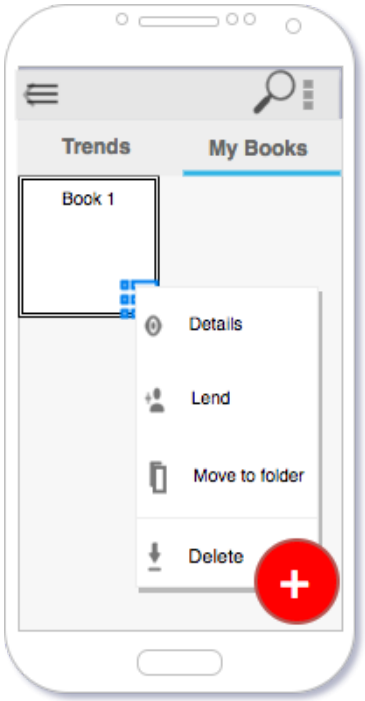
User Interface Mocks

Screen 1 - Login

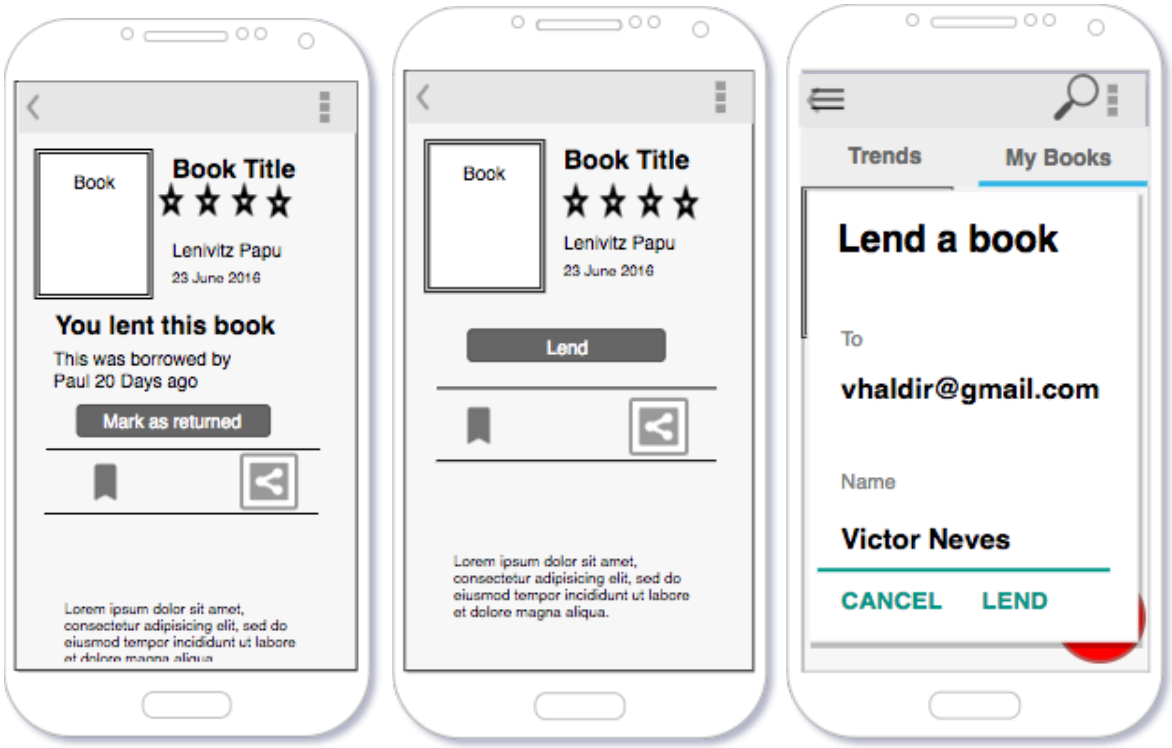
Screen 2 - Top books



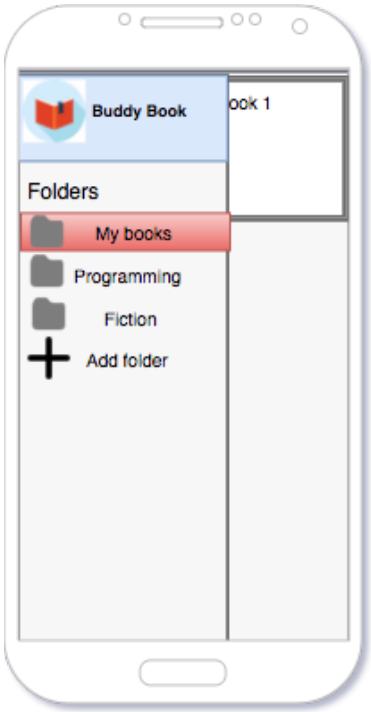
Screen 3 - My books



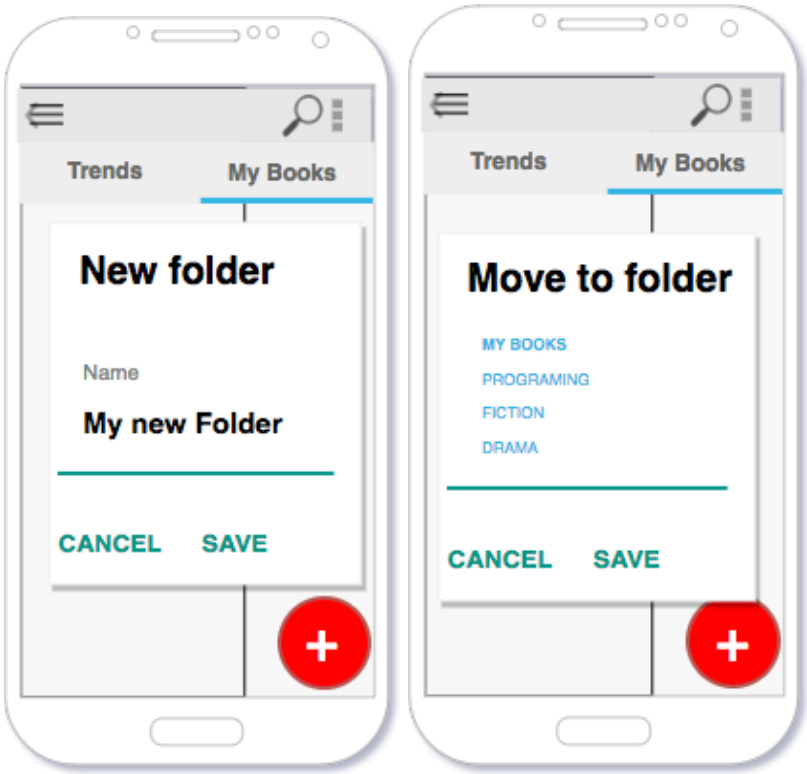
Screen 4 - Details from my books (Book lent, book no lent, lend dialog)



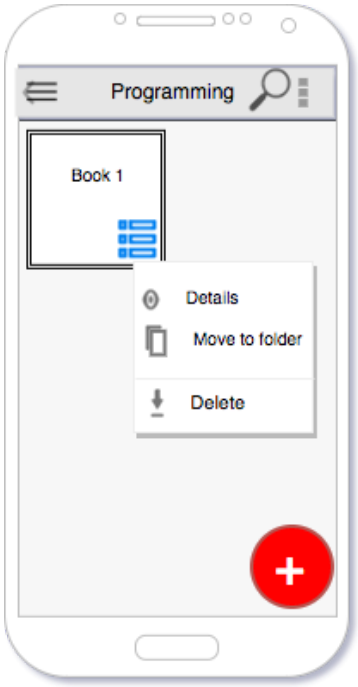
Screen 5 - Implement Drawer Folders



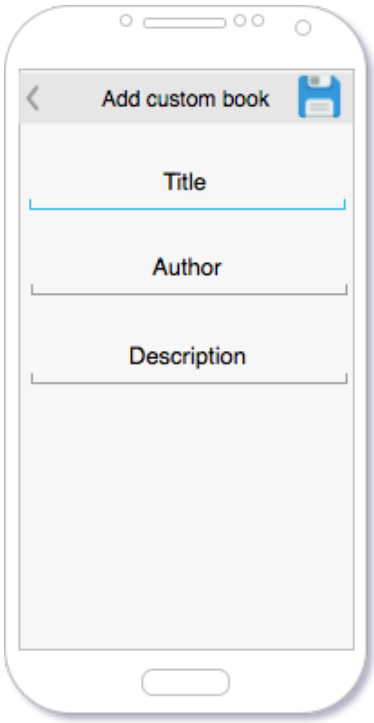
Screen 6 - Add new folder and move to folder



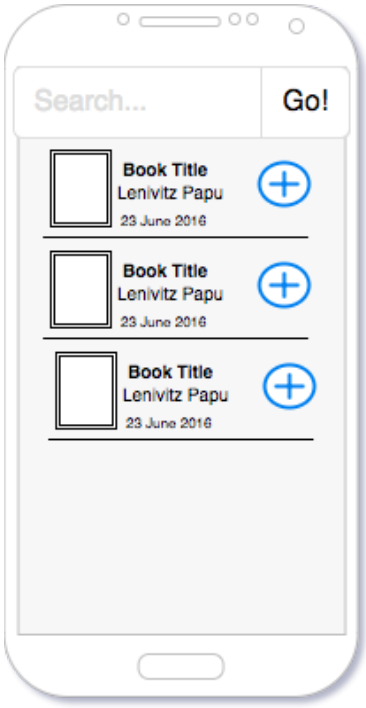
Screen 7 - List books folder



Screen 8 - Add custom book



Screen 9 - Search book



Screen 10 - Lent books widget

Key Considerations

How will your app handle data persistence?

I will build my own content provider. All data will be persisted on SQLite and synchronized with Firebase Real Time Database

Describe any corner cases in the UX.

If the the user has no internet connection, operations like search will only fetch books already persisted in local database.

If the user clear the data of the application through menu settings and try access the application without internet connection, as message will be shown on the trend section informing that the app could not fetch data from the API due to lack of connection.

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

- Firebase Auth - To authenticate users
- Firebase Real Time Database - Persist Folders and books
- Firebase Crash - For crash reporting
- Glide - Load book folders and caching
- Retrofit - To manage network connections
- ButterKnife - For view binding and reduce lines of code
- Setho - For database debug and network analysis
- Junit - To drive unit tests
- Timber - Logging
- AboutLibraries - Create section with some information of libraries used on this project
- AdMobi - Monetization
- ZXing - To scan barcode of physical books in order to get ISBN

Describe how you will implement Google Play Services.

I will be using using 3 features of firebase, they are: Firebase Auth (to authenticate users), Firebase Stores (to cloud persistence) and Firebase Crash (to monitor crash of my app). Also I will be using AdMobi which also depends on Google Play Services, to monetize my app.

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

- Configure libraries
- Create repository
- Commit skelethon

Task 2: Implement content provider

- Implement helper and contract to create tables folder and book
- Implement URI matcher.
- Implement Content Provider
- Validate operations with setho
- Cover with junit tests

Task 3: Create Login

- Create Firebase Project
- Configure Firebase UI
- Implement Firebase authentication workflow

Task 4: Implement “Trends” tab

- Set up retrofit to fetch data from Google Book APIs
- Map java entities from Google Book APIs Json
- Setup RecyclerView, Adapter and Item Layout

Task 5: Implement “My books” tab

- Implement “Move to folder” operation
- Implement RecyclerView filtered by books flagged as “My book”

Task 6: Set Up drawer of folders

- Implement RecyclerView of folders in Drawer component.
- Implement add folder by dialog
- Implement folder deletion

Task 7: Implement book list from folder perspective

- Implement RecyclerView listing books from folder

Task 8: Create book details activity

- Design coordinator layout to show Image, Book description, book author and score.
- Implement URI to fetch book by ID from the content provider
- Implement Book image using Glide

Task 9: Implement operation lend and Mark as returned

- Create database column flagging book as lent
- Implement dialog with form to inform details of the person who's going to take the book
- Save dialog information in the database

Task 10: Implement book searching

- Implement search interface of Android to find books by typing its information on actionbar.
- Implement search by barcode (ISBN)

Task 11: Implement add custom book activity

- Create insert book operation in Content Provider
- Place 3 edittexts on xml layout to get title, author and description of the custom book
- Insert book out of UI thread

Task 12: Implement lent books widget

- Implement AppWidgetProvider and RemoteViewsService
- Create list item XML to present the data

Task 13: Implement AdMob

- Implement InterstitialAd to be shown when user select folder

Task 14: Design

- Design logo
- Review font
- Review material design guidelines
