Design document for: Inventory Application ("Invii")
Authors:
Arshdeep Padda
Bryce Schooling
Jason Lim
Nguyen Nam Pham
CS 401 03 Group 04
Date: 14 Feb 2022
Status: Planning, Draft

Concept Summary

Invii is an inventory management application for representing and managing physical inventory spaces. At its base, it represents an *item* that can carry attributive data such as name, description, images, location, cost, quantity, etc. Further, these items can be organized into *collections*, which serve as folders and can contain several items, and be made to represent physical locations, containers, or even abstract categories. Collections themselves can contain attributes such as: name, description, location, images, QR codes that scan to themselves, and computed values such as unique item count and total item counts, etc. Because Invii serves to accommodate multiple use cases and application domains, it does not impose a particular organization style on its users.

Audience / Customer

We wish to produce an application that gives several features for inventory representation management, yet imposes few, so as to be flexible for several applications. We anticipate Invii to be usable for the following: stock / inventory management for commerce businesses (consider eBay sellers), IT departments (any other department or organization in need of tracking inventory), personal storage and archiving, residential or commercial moving.

Background

Consider you are moving, either a small commercial office or residential unit, and you pack away your items into moving boxes. Invii can be used to document what items are involved in the move, their notable descriptions (important for commercial moving), and their location. Boxes can then be labeled with QR codes that can be scanned to show what items are within them in the app. This serves both to organize and document inventory.

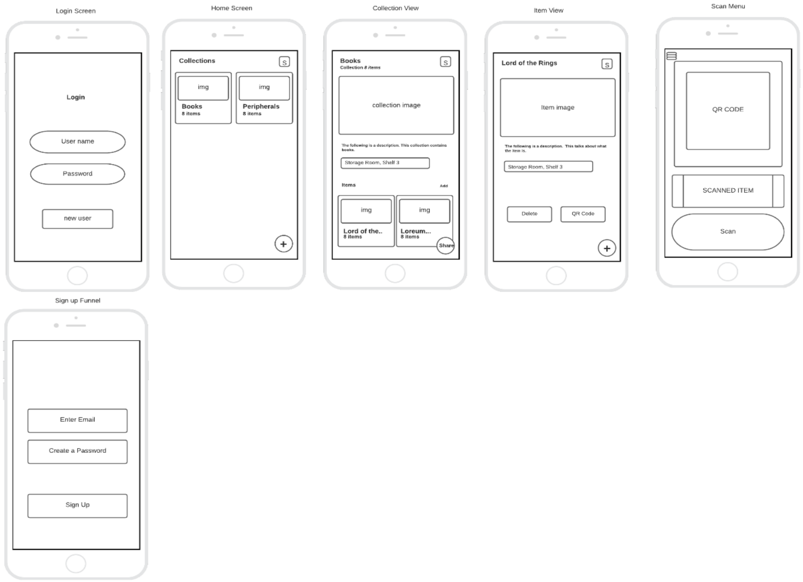
Consider you are a small business selling on an e-commerce platform. You maintain your inventory in Invii, with each sale item being represented as an item in the app, complete with description, price, and stock count. These items are further organized into collections, which are abstract categories in this case (recall the flexibility of the application), such as: accessories, electronics, cables, cases, etc.

The design for this app arose as a need to track what items exist and where they are located, and information regarding said items. From there, it was extended for use in more robust inventory management applications. QR codes were used to quickly pull up what items are present in what containers without the need to physically open the container, and an in-app search function to determine what items exist and where they are located. Items, for various purposes, can also be tracked via QR codes with information pertaining to it, for documentation / tracking purposes. Thus, its application broadened from initially being required for amature home inventory management to more flexible and commercial purposes.

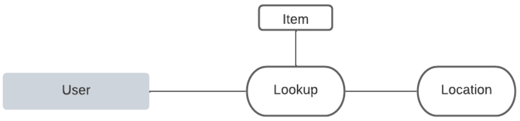
Application Cost & Projected Success

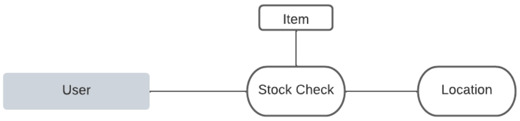
We have no license fees to contend with as the project is an open source public project. It is estimated to take about 10 weeks to complete starting from the early stages of planning, all the way to the early prototype phase. Assuming 10 hour weeks, with 4 collaborators working on the project, it would be estimated to take a total of 400 work hours. Assuming an hourly rate of $X, then it would be roughly $400X for just the manpower. In addition there are no fees for listing the app on the google play store for Android, however we would need a developer account. Developer accounts cost $25 to register as a one time payment. Assuming the project succeeds, google will take 15% of the first 1 Million made.

Interface Mockups



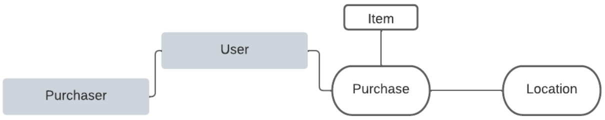
Design-Use Case Diagrams



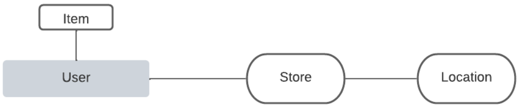


Use Case: Lookup
Participants: User, Item
Pre-condition: collections.contain(item)
Post: return location

Use Case: Stock Check
Participants: User, Item
Pre-condition: collections.contain(item),
item.quantity > 0
Post: return location



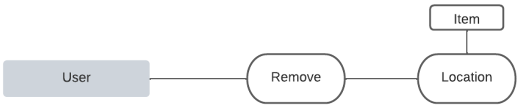
Use Case: Purchase
Participants: User, Requestor, Item
Pre-condition: collections.contain(item),
item.quantity > 0
Post: decrement quantity, return location



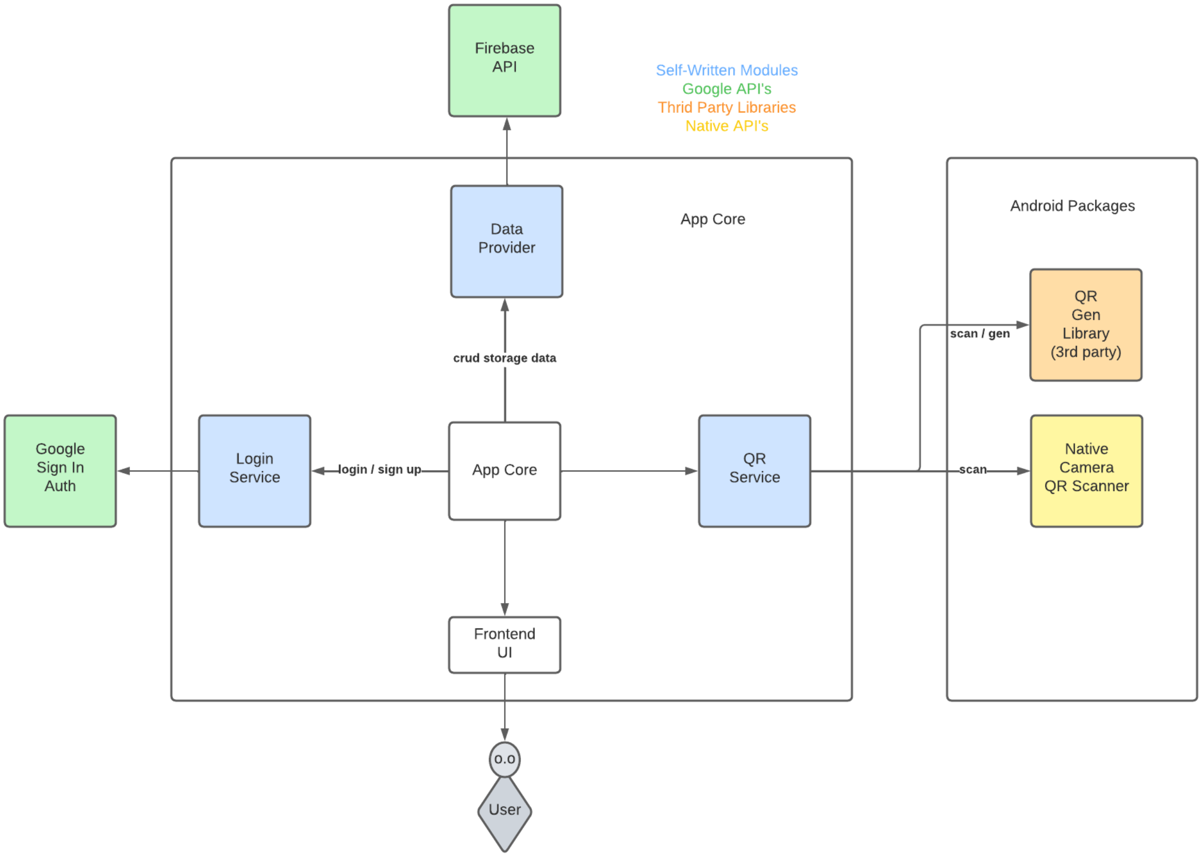
Use Case: Storing item
Participants: User, Storage Location, Item
Pre-condition: User has item

Post: collection.addItem()

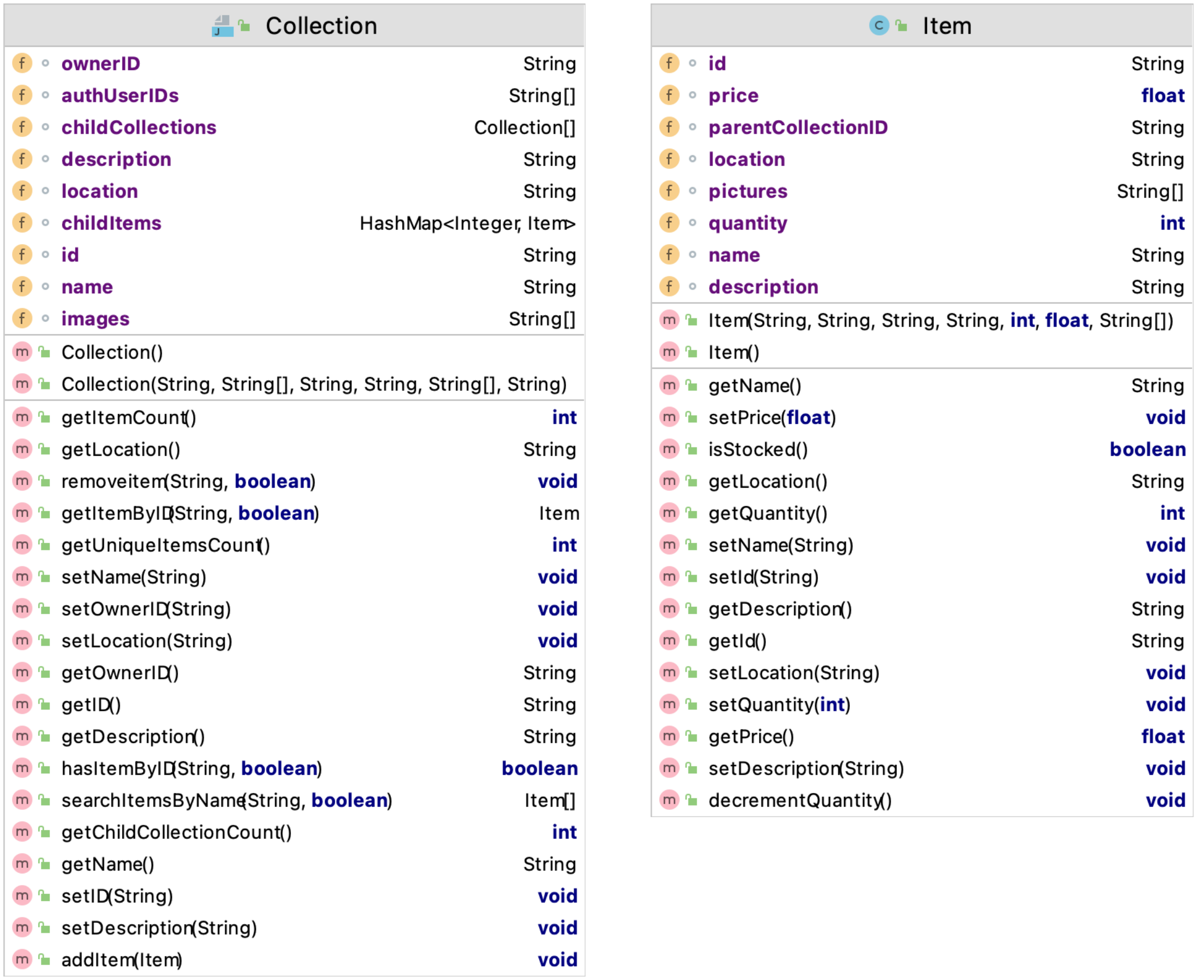
Use Case: Removing item
Participants: User, Storage Location, Item
Pre-condition:
Collection.hasItem()
Post: collection.removeItem()



Design-Detailed Design



UML Diagram



Collection	
ownerID	String
authUserIDs	String[]
childCollections	Collection[]
description	String
location	String
childItems	HashMap<Integer, Item>
id	String
name	String
images	String[]
Collection()	
Collection(String, String[], String, String, String[], String)	
getItemCount()	int
getLocation()	String
removeitem(String, boolean)	void
getItemByID(String, boolean)	Item
getUniqueItemsCount()	int
setName(String)	void
setOwnerID(String)	void
setLocation(String)	void
getOwnerID()	String
getID()	String
getDescription()	String
hasItemByID(String, boolean)	boolean
searchItemsByName(String, boolean)	Item[]
getChildCollectionCount()	int
getName()	String
setID(String)	void
setDescription(String)	void
addItem(Item)	void

Item	
id	String
price	float
parentCollectionID	String
location	String
pictures	String[]
quantity	int
name	String
description	String
Item(String, String, String, String, int, float, String[])	
Item()	
getName()	String
setPrice(float)	void
isStocked()	boolean
getLocation()	String
getQuantity()	int
setName(String)	void
setId(String)	void
getDescription()	String
getId()	String
setLocation(String)	void
setQuantity(int)	void
getPrice()	float
setDescription(String)	void
decrementQuantity()	void

Related Work

Sortly (https://www.sortly.com/) for iOS is a robust competitor in this space. It has evolved in recent years from a simpler inventory management application that advertised to amature (relative to now) inventory management (e.g. residential moving) to focusing more heavily on robust inventory management for commercial domains. It offers summaries of inventory, collaboration on management (sharing access), tracking users and their activities (e.g. changes to items), generating reports, cost summaries of inventory spaces, etc. One of its original ideas was generating QR codes for its folders. Now, items can be *checked* out, a most vital feature for departmental inventory management.

Suffice to say, Sortly is a mature, established, and gold standard form of an inventory management application to strive for.

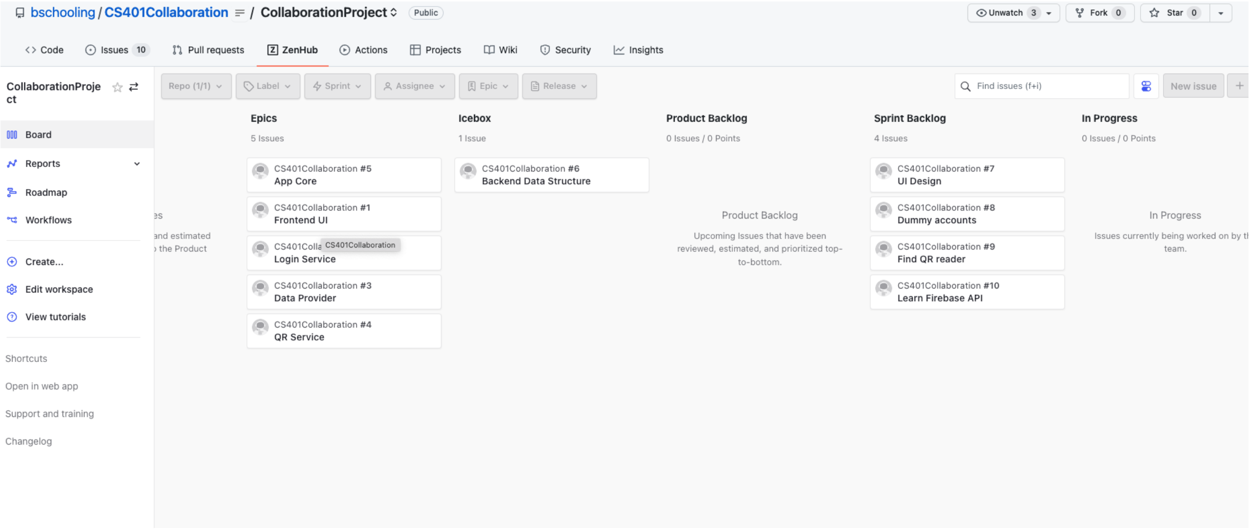
Frameworks / Services / Cloud / Backends

Backend: Firebase for document based database storage, local sqlite instance otherwise.

Testing

Unit testing of module components using JUnit to ensure proper core and edge case functionality.

Schedule



Epics
5 Issues

- CS401Collaboration #5 App Core
- CS401Collaboration #1 Frontend UI
- CS401Colla CS401Collaboration Login Service
- CS401Collaboration #3 Data Provider
- CS401Collaboration #4 QR Service

Icebox
1 Issue

- CS401Collaboration #6 Backend Data Structure

Product Backlog
0 Issues / 0 Points

Product Backlog
Upcoming Issues that have been reviewed, estimated, and prioritized top-to-bottom.

Sprint Backlog
4 Issues

- CS401Collaboration #7 UI Design
- CS401Collaboration #8 Dummy accounts
- CS401Collaboration #9 Find QR reader
- CS401Collaboration #10 Learn Firebase API

In Progress
0 Issues / 0 Points

In Progress
Issues currently being worked on by t team.

.

Dependencies

Generating QR codes and scanning will be outsourced to third party libraries or native scanning.

Firebase library for communicating with backend service API.

Presently exploring native components for UI elements, with consideration to research alternative UI framework libraries as design needs change.

Tentative: Login service. If we use a remote backend, we can look into using Google’s native account management (ie: “login with Google”). Else, just use local storage and forgo the usage of user accounts. Such API’s include Google Sign-In or One Tap sign in and sign up.