Group 12: Final Project Proposal
iRecord

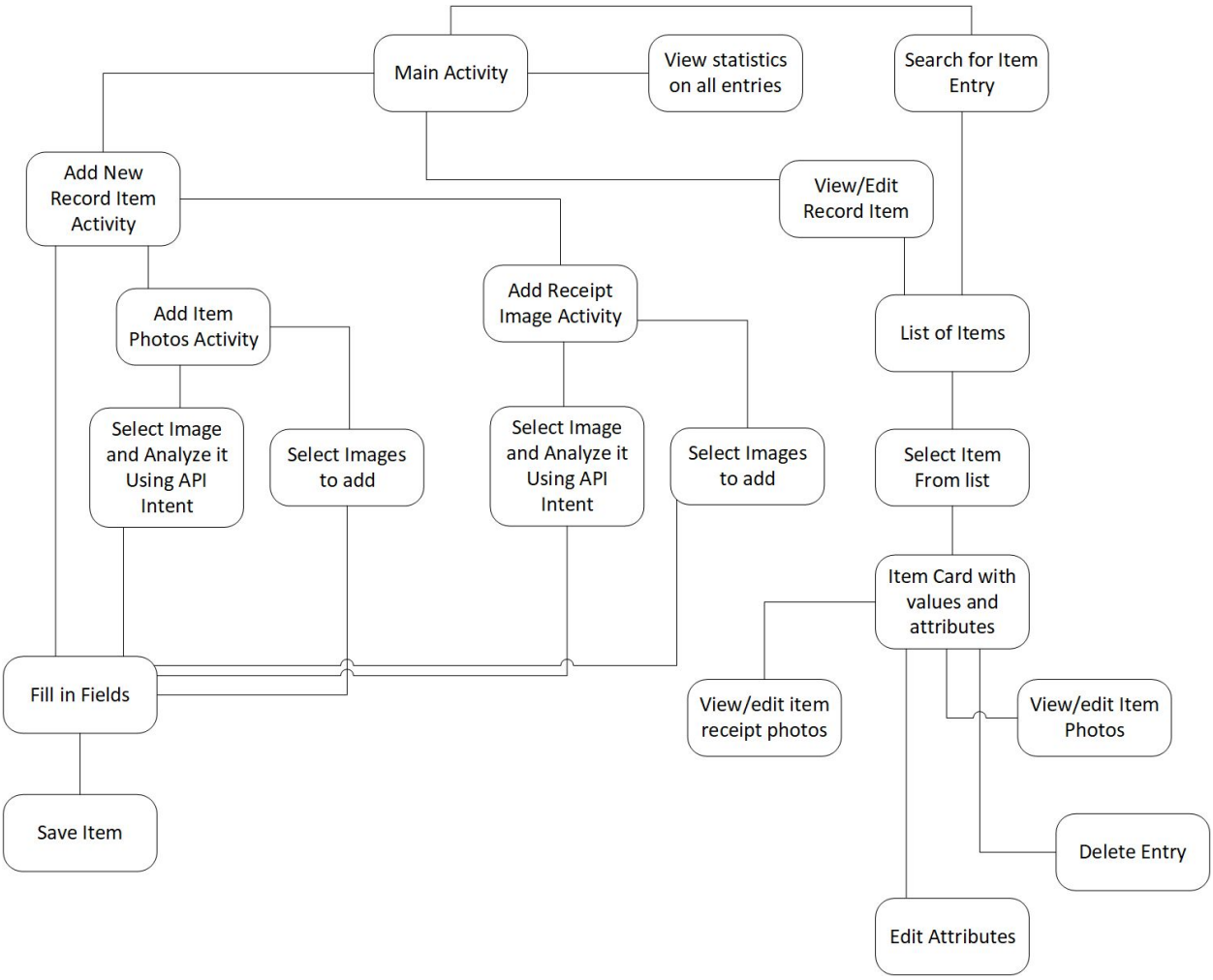
Members:
Daniel Domme (dommed@oregonstate.edu)
Ethan Patterson (patteret@oregonstate.edu)
Chendi Zhang (zhangch3@oregonstate.edu)

Description:
Collectively we have decided that a financial receipt recording application would be a useful and insightful mobile project. The objective of this application is to use a mobile devices camera to capture an image of a user's shopping receipt. After a successful image capture of a receipt, the application will use an intelligent API to identify the total purchase amount printed on the receipt. If this fails, the user will also have the option to manually enter their total. The images will be statically stored on the user's device. The user will also be prompted to enter a subject they wish to identify the receipt with. This could include things such as gas, groceries, online purchases, etc. This categorized information will then be available to the user to see the total amount spent for a given month and the total by each of the categorical receipt options available to the user. The target audience for this application are those that wish to automate the process of their spending habits with aims to improve their financial budgeting or identify spending habits.

API to be used:
We will be using the Azure cognitive science text recognition API to analyze receipts. This will be used to identify the total amount recorded on the receipt. This information is intended to auto populate the price field of the application. An Azure object recognition API will also be used to recognize the object in the inventory pictures that a user adds.

Text Recognition API:
https://westus.api.cognitive.microsoft.com/vision/v2.0/recognizeText?mode=Printed
Object Recognition API:
https://westus.api.cognitive.microsoft.com/vision/v2.0/detect
Receipt Analyze API:
https://westus2.api.cognitive.microsoft.com/formrecognizer/v1.0-preview/prebuilt/receipt/asyncBatchAnalyze

UI Hierarchy:



The application will initiate on the home screen where the user will have the option to view or edit their inventory items, add a new inventory item, or review inventory statistics. A quick inventory report will be displayed below these options. See “Main Activity” mock for an example.

If the user clicks add new inventory items from the home screen, they will be directed to a new instance providing them the option to add a new image of a receipt and or pictures of items recorded on that receipt. See the add new inventory items mock for an example. Adding images of purchased items is intended for high value items for proof of ownership or for insurance purposes. The user will also have the option to add a receipt without storing an image.

If the user chooses to add an image of a receipt they will be taken to a new instance allowing them to use the AI API to auto-analyze their image, returning information back to the add new inventory items instance auto populating the fields. They will also have the option to just capture an image of the receipt without the use of the AI API.

If the user chooses to add pictures of items themselves they will be taken to a screen allowing them to use an AI to identify the item in the picture. This information will be taken back to the

add new inventory instance auto populating related fields. The user will also have the option to capture an image without using the AI API allowing them to fill out the information fields themselves on the add new Inventory items.

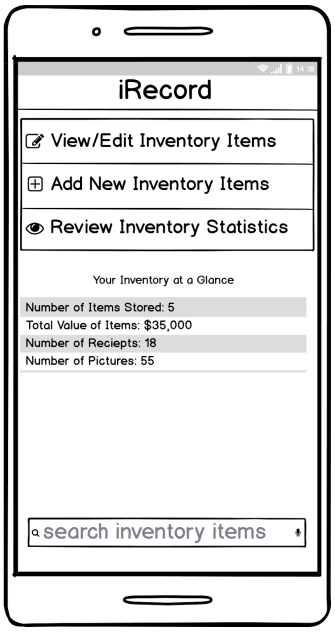
From the home screen, the user can view and edit inventory items by clicking the view/edit inventory items button. This will direct them to a recyclerview list of inventoried purchases the user can select from. By selecting a purchase they will be directed to an instance of detailed information about that purchase. In the detailed instance item the user will have the ability to view and edit information regarding that purchase.

Additional features not covered in class:
File storage and camera usage:
When the users try to add a new record of receipt,

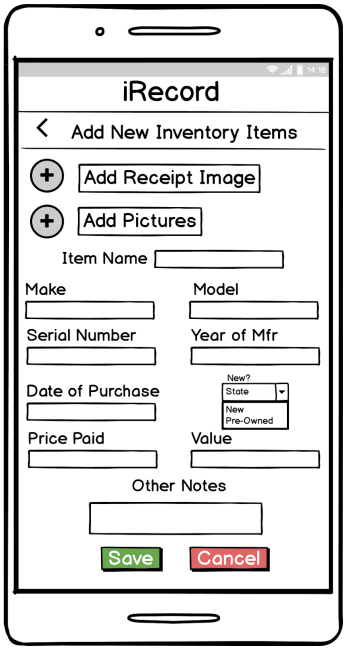
1. They can take a picture of the item and/or the receipt, which will use the camera. (Camera Usage) After taking the pictures, the background process will store the pictures into the
2. They can select photos from the Gallery. The back-ground process will automatically copy the selected photos and store them in the application's own directory. (File Storage)

Mocks:

1. **Main Activity: Ability to select activities to view/edit or add new item record. Ability to review inventory statistics. Ability to search for item records. Overview of statistics.**



a. **Add New Inventory Items Activity: Ability to launch adding receipt or item pictures activities and edit information text fields.**



i. **Add Receipt Image Activity: Ability to launch explicit tasks to get pictures from camera roll to either send it to be analyzed by API or just add them. Analysis would attempt to process receipt text auto-fill information.**

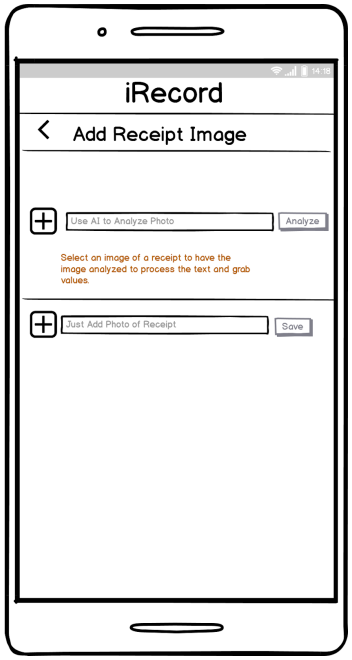
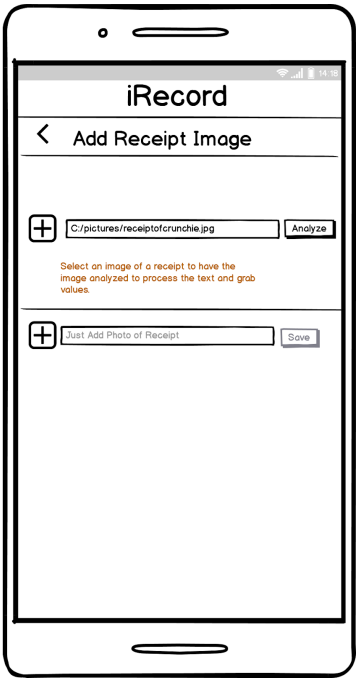


Image selected for analysis.



API POST sent and awaiting results

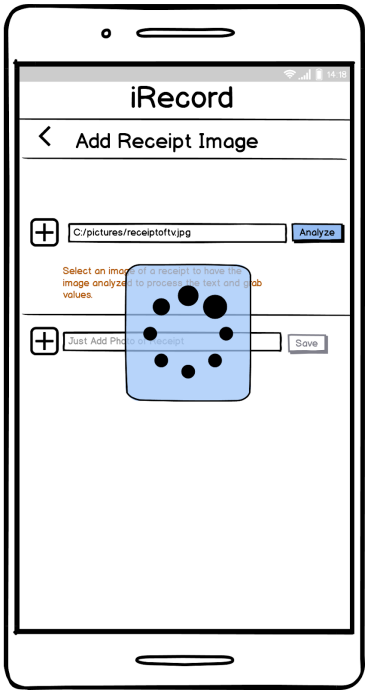
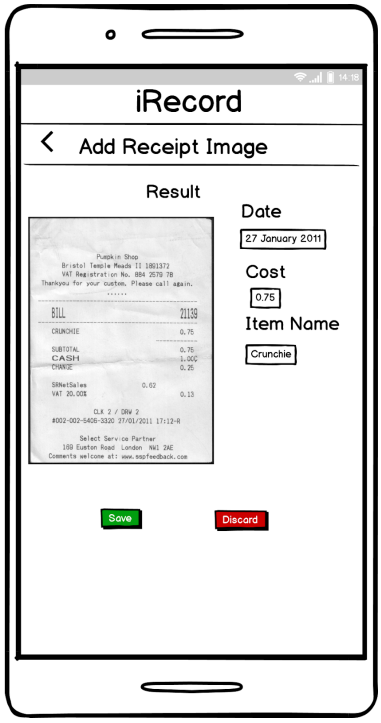
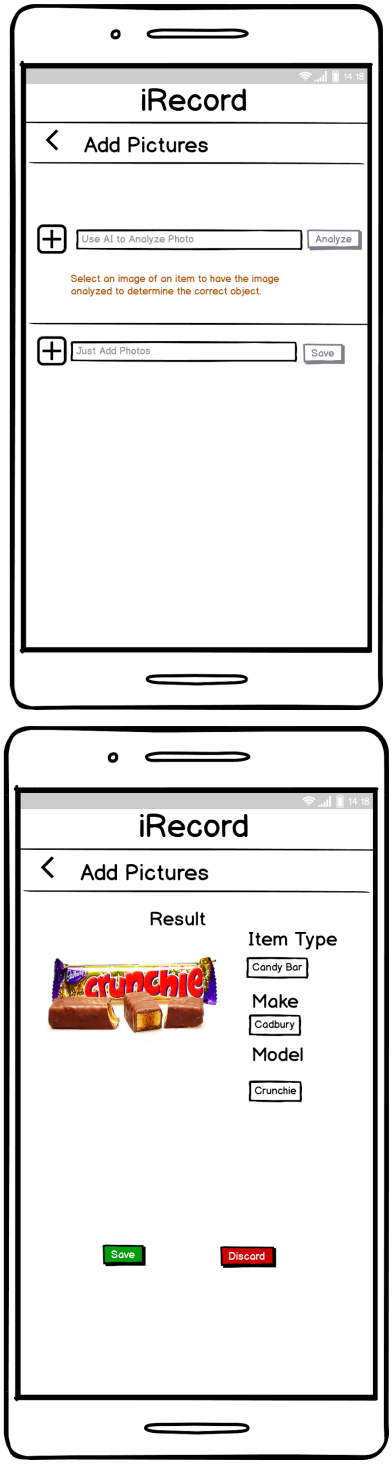


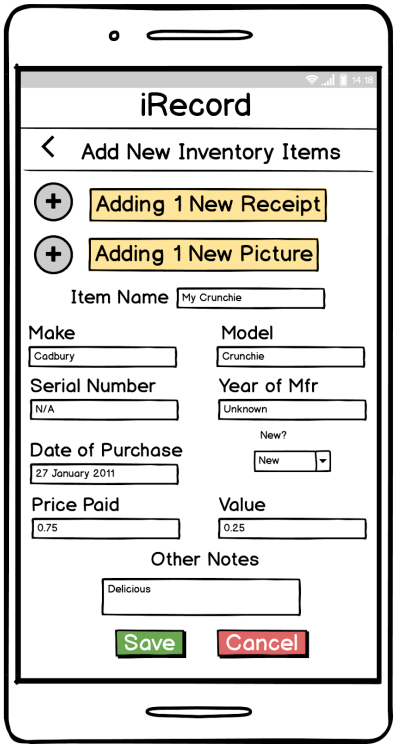
Image displayed and fields auto-populated with best guess by API. Users can edit and correct.



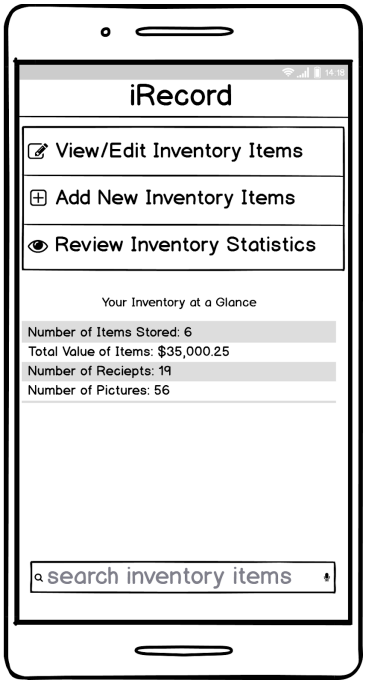
ii. **Add Item Pictures Activity: Similar to Receipt. Uses different API end-point for object recognition to try to populate information about the item being recorded. The user will be able to edit information.**



b. Add New Inventory Items Activity Submittal: a new item is ready to be added to the list of records



2. **Main Activity: the new item has been added to the records. The statistics are updated to reflect this.**



a. **View/Edit Inventory Items Activity: The list of all item records with general information is presented to the user. The items are clickable to start a new activity to view the items.**

    i. **View Individual Inventory Item Activity: An individual record is viewable. All photos of the item and receipts are available. The user will be able to edit the record.**


iRecord
View/Edit Inventory Items
Pontiac Fiero
Photos
Make
Pontiac
Model
Fiero
Year of Mfr
1985
New?
New
Receipts
Serial Number
JH4TB2H26CC000000
Value
$5000.00
Date of Purchase
3 September 1985
Other Notes
Cool Car, Dude!
Price Paid
$9599.00

Database Tables

Table: item

Column	**type**	**description**
itemID	String	**Primary key**
itemName	string	Item name-- not null
itemType	string	
make	String	
serialNumber	String	
model	String	
value	double	
pricePaid	double	
newItem	boolean	If the item is new -- true else -- false
serialNumber	string	
purchaseDate	Date/String	
manufacturedDate	Date/String	
note	String	

Table: Item_photoes

Column	**Type**	**description**
itemID	String	**F-key** Uuid of item
photoPath	String	**P-key** Path stores
photoType	boolean/String	receipt/object