NixLord

Diksha
Saurabh
Shibasis

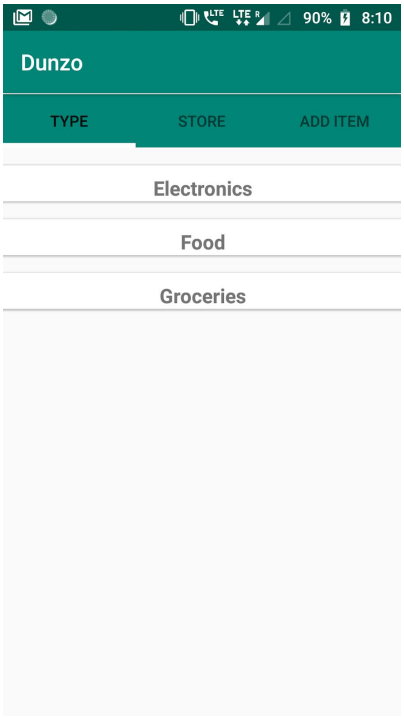
Dunzo Inventor

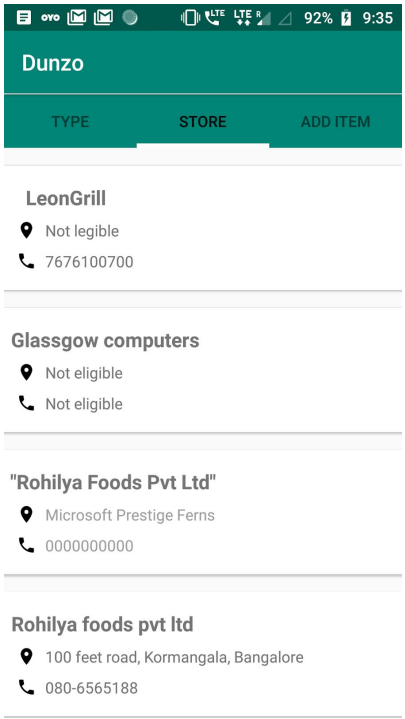
August 03, 2019

Product Overview

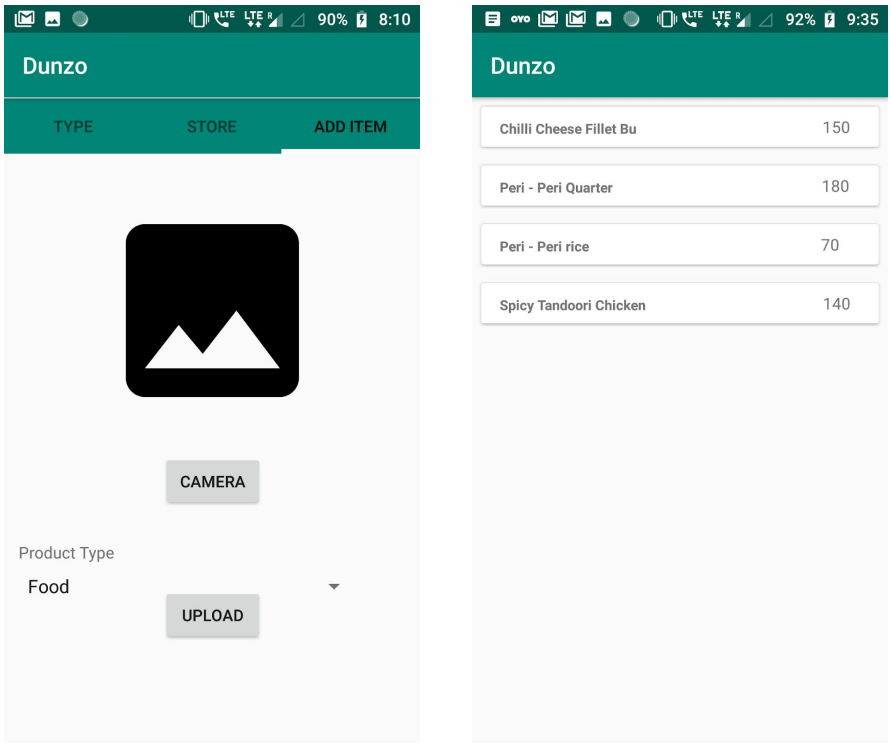
We are building an Android app capable of doing OCR which stores data collected from receipts in a cloud database categorized by stores and products. We use Firebase Firestore, a document-oriented database to store the inventory. The app can scan photos using the camera and then we get the text using onDevice Firebase ML Kit Vision APIs.

User Interface

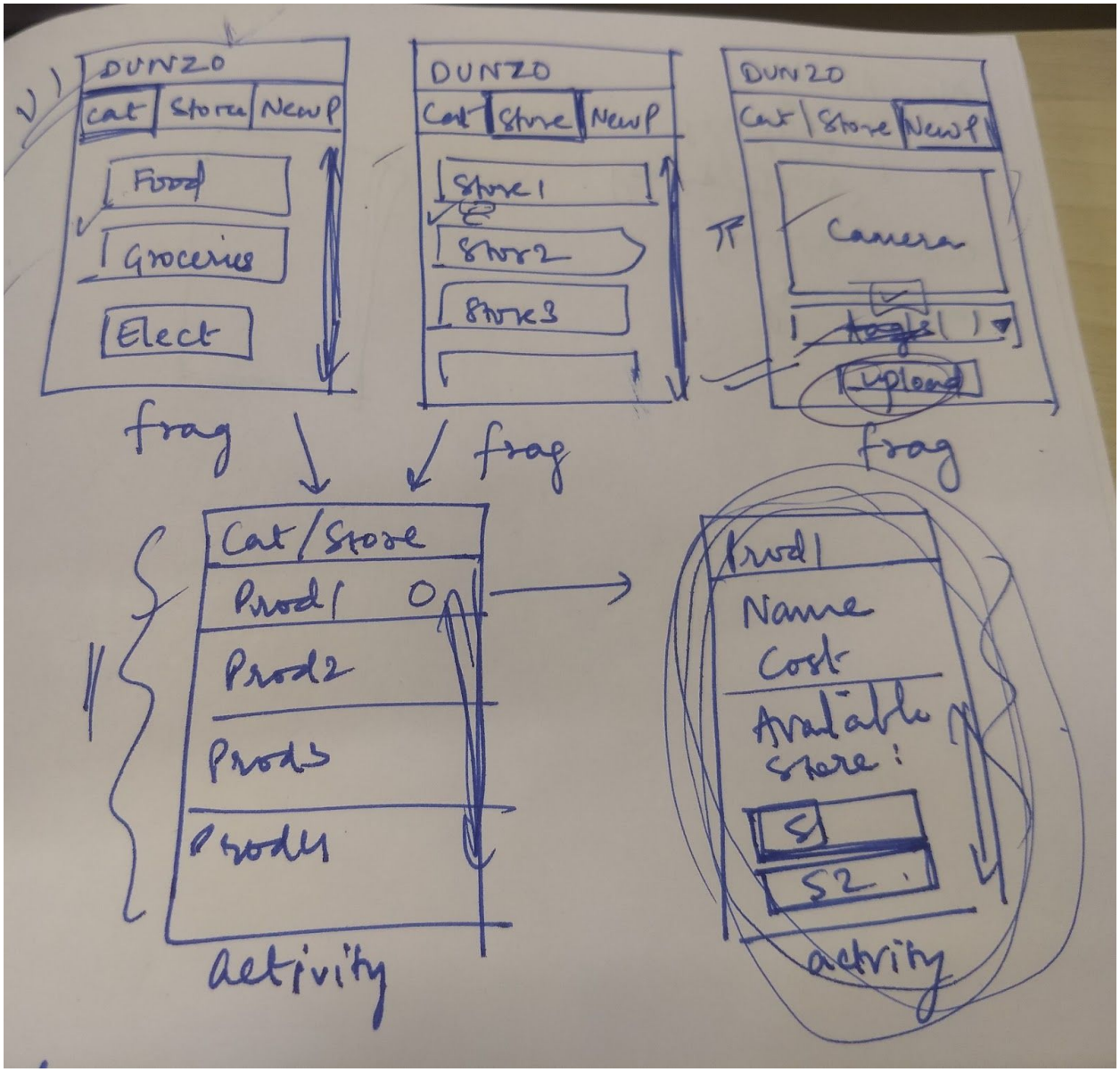




2



- ➔ We have a Tab Layout with three tabs.
  - ➔ TYPE: Products are categorized by Item types, i.e. Grocery, Electronics, Food, etc. Clicking on a category will show a list of items coming under that category.
  - ➔ STORE: Products are categorized by stores. Clicking on a store will show a listing of items available at that store.
  - ➔ ADD ITEM: In this tab, we can scan a receipt using the Android camera and upload Product and Store details to the Cloud database.



➔ The Products listing is shown in two of the tabs but sorted differently. Clicking on a Product will show a Product Detail page from where you can checkout the item.

Inputs

The input is currently a well-lit and properly taken photo. Currently the app is being improved to increase robustness and handle a wide range of image quality. The user also needs to specify the type of product being submitted.

Functional Requirements

- ➔ The Dunzo agent can browse products in the App to know which products are available in which stores.
- ➔ He/She can also browse stores and what products it has.
- ➔ Using the App's scanner functionality, the agent can update the inventory database by taking a photo of the bill.

Configuration

The app is built using Firebase Cloud Platform and Android Studio. It uses some open source libraries as listed in the dependency section. These dependencies need to be set up before building and using the app.

Milestones

- ➔ (DONE) Setup Firebase and Android App
- ➔ (DONE) Implement Image OCR Functionality using ML Kit.
- ➔ (DONE) UI
  - ◆ (DONE) Implement basic scaffold architecture of the App using Fragments, Viewpagers and so.
  - ◆ (DONE) Implement Add Item Tab
  - ◆ (DONE) Implement recycler view.
  - ◆ (DONE) Connect Firestore to RecyclerView
  - ◆ (DONE) Creating Product Detail Listing page.
- ➔ (DONE) OCR to Product/Seller
  - ◆ (DONE) Break OCR text into elements.
  - ◆ (DONE) Designate markers to segment the elements into Store and Product portions.
  - ◆ (DONE) Fuzzy search to find closest words to designated markers.
  - ◆ (DONE) Create Product and Seller from the portions.

- (DONE) Establish dataflow from Product/Seller detection to UI

➔ (ONGOING) Feature Enhancements

- (DONE) Display products from Firestore
- (DONE) Query the database as per use-cases.
- (DONE) Inventory updation from Add Item and Checkout
- (DONE) Final Integration and Testing.

Data Model

We use Firestore, a document oriented database.

➔ ADVANTAGES

- Simple to use but powerful enough for the usecase.
- Gives us a REST API automatically, allowing us to both have an API layer and App
- Firebase as a platform allows for much more utility like ML Kit which we also use.

➔ DISADVANTAGES

- Not as powerful as SQL for querying.
- Dependent on third party vendor. (Can be switched with some effort to Mongo).