

QuickShop

A faster way to shop

Kunal Malhotra

Sam Britt

Motivation

- User shopping behavior:
 - Adds items to an empty list
 - Categorizes them
 - Starts shopping !!!!
- Existing shopping list apps mostly categorize items into predefined categories, display available coupons, saving cards, display grocery stores in the vicinity, etc.
- Not much work done in optimizing the user's path through the grocery store
- Categories close together in the list can be in different parts of the store which makes the user go back and forth between aisles of the store.
- **What is needed? - A grocery list which dynamically sorts itself by category based on the location of items in the store.**

Functionality of existing shopping list apps

- Help users make a grocery list categorized under existing categories
- Offer free-form item input or have a large database of items for the users to choose from.
- Add frequently purchased items to new lists and keep track of items which have been checked out.
- Store specific coupon recommendation.
- Sync lists with friends / spouse.

QuickShop

- Android app to sort a shopping list by category to make the user pick up items in one pass without returning to a section
- Store layout stored in a SQLite Database
- Dynamic sorting of categories
- Helps saving user's time and effort
- List becomes a virtual guide helping picking up items in the right order.
- Great help in an unfamiliar store

Technologies used

- **Java** - Programming language
- **Eclipse IDE** - Development engine
- **Android SDK** - Application development
- **SQLite Database** - To record store layout information

Database Design

- Store names and location coordinates (Coordinates of the entrance door)

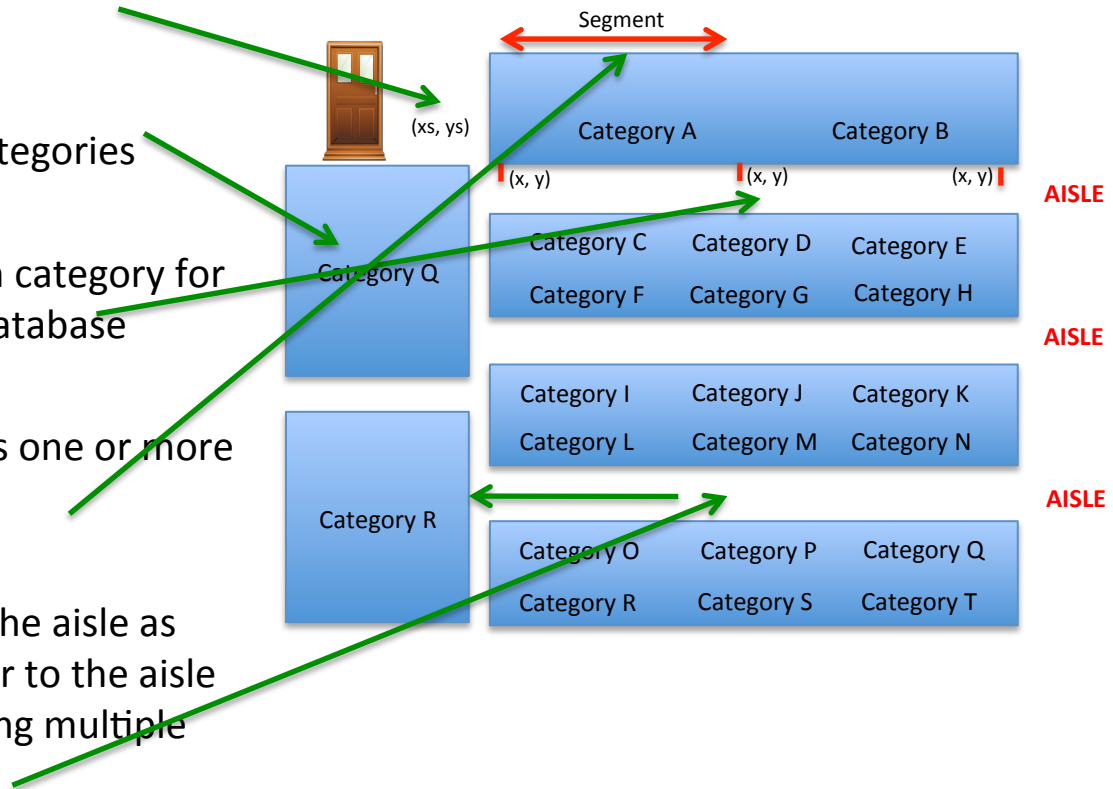
- System specified categories

- Coordinates of each category for every store in the database

- Each category spans one or more segments

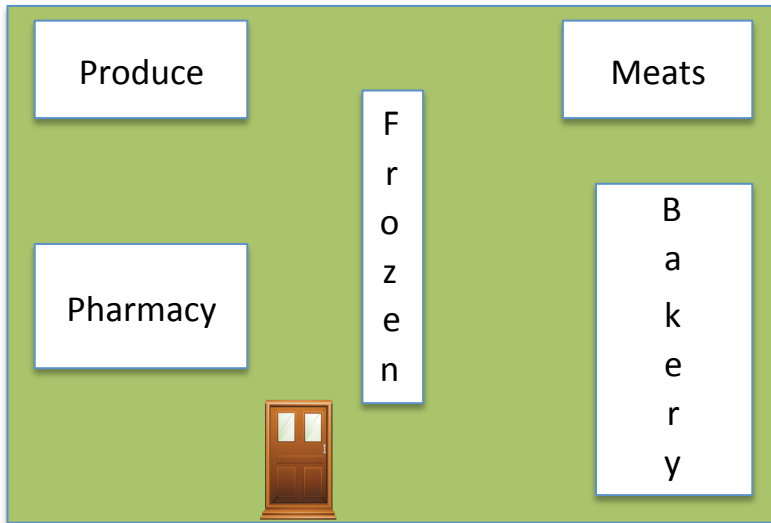
- Items can be along the aisle as well as perpendicular to the aisle (Coordinates spanning multiple aisles)

Store Layout

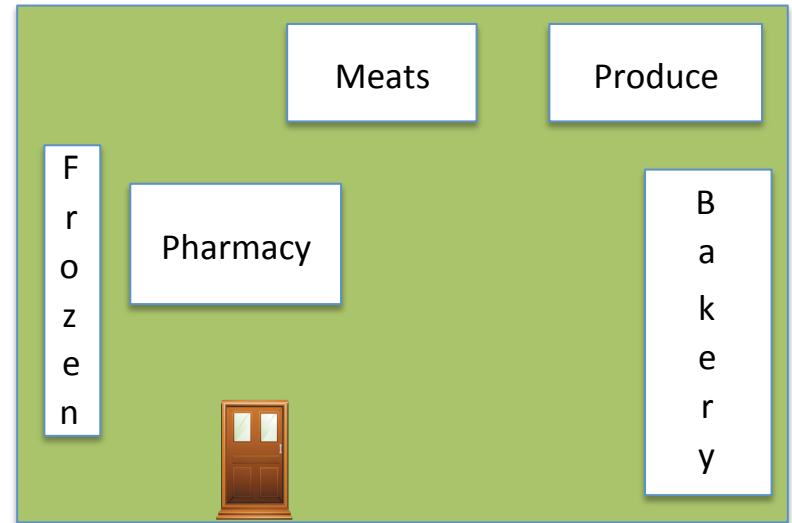


- The design is comprehensive since it plays a primary role in sorting the user's grocery list in a way that can optimize the time he takes to finish shopping.
- Topological component of the store was critical in designing the database
- Store floor plans of major chains are hard to get
- Manual collection of data - imported into our database

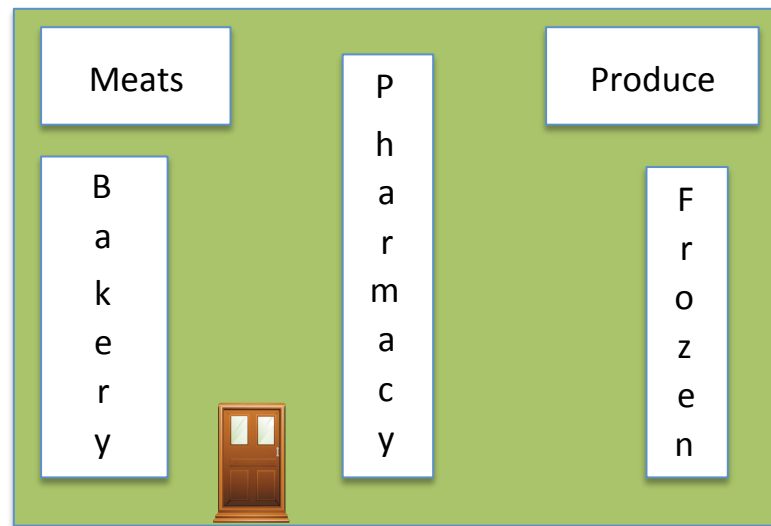
Various Publix Stores across Atlanta



Publix - Holcomb Bridge Road

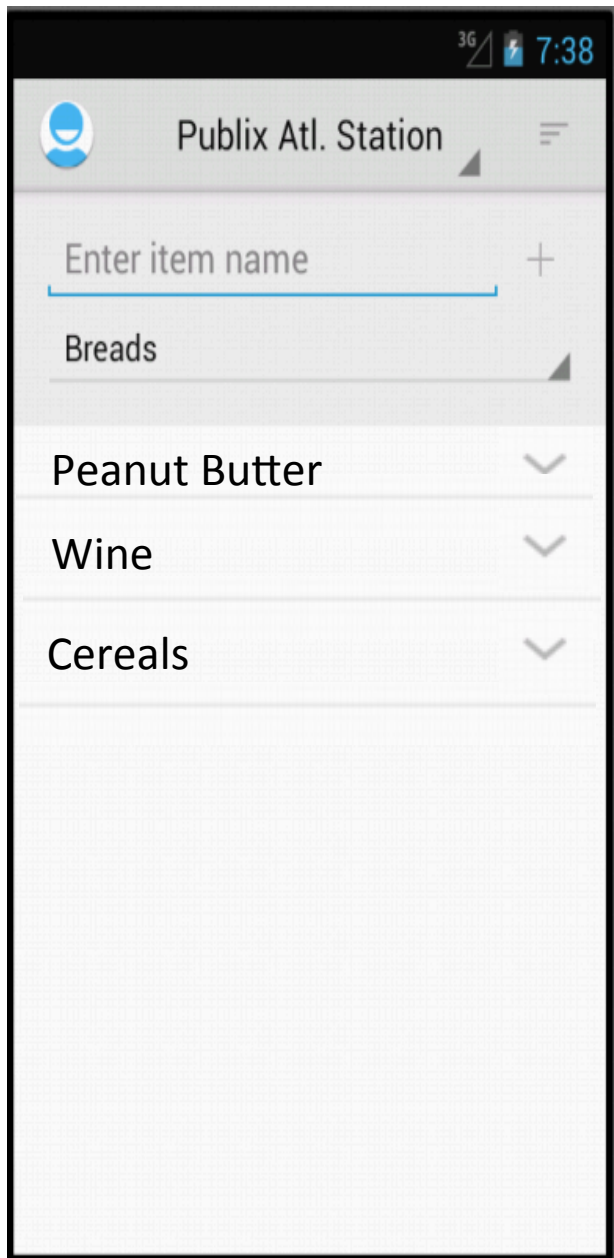


Publix - Atlantic Station

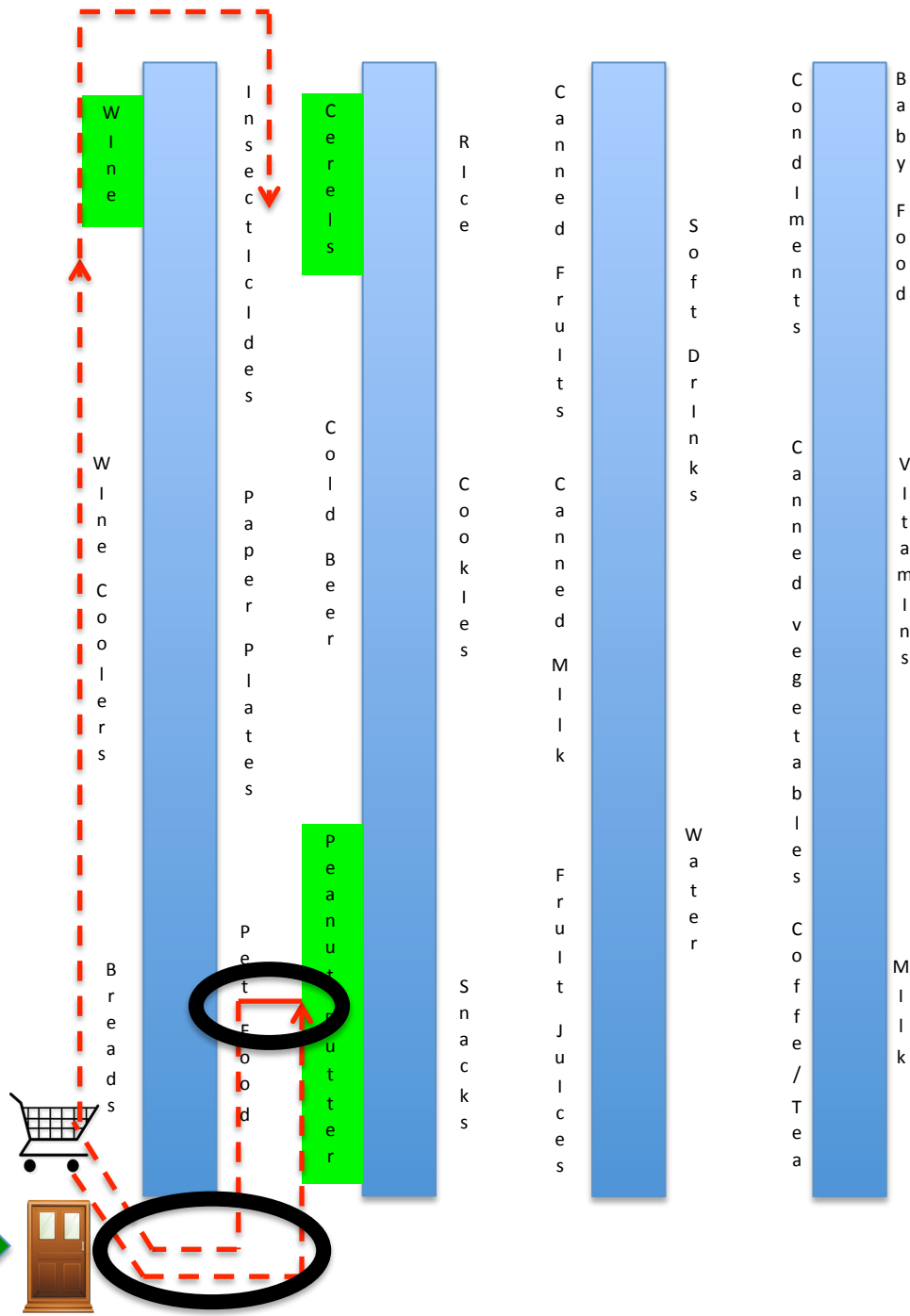


Publix -
Shallowford Road

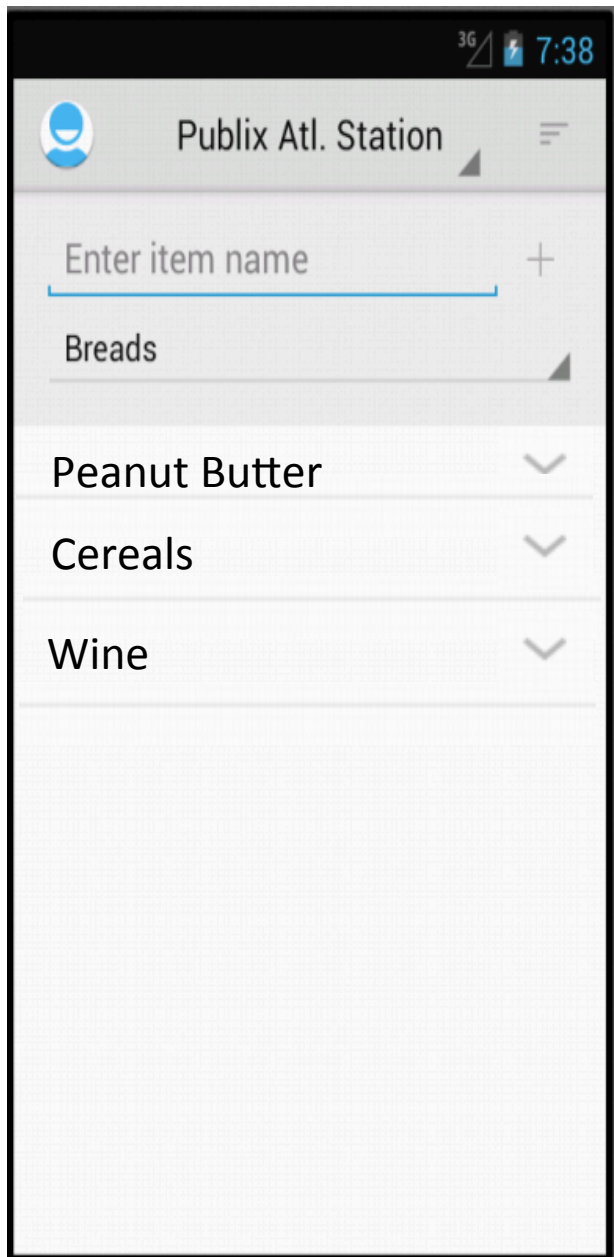
DEMO



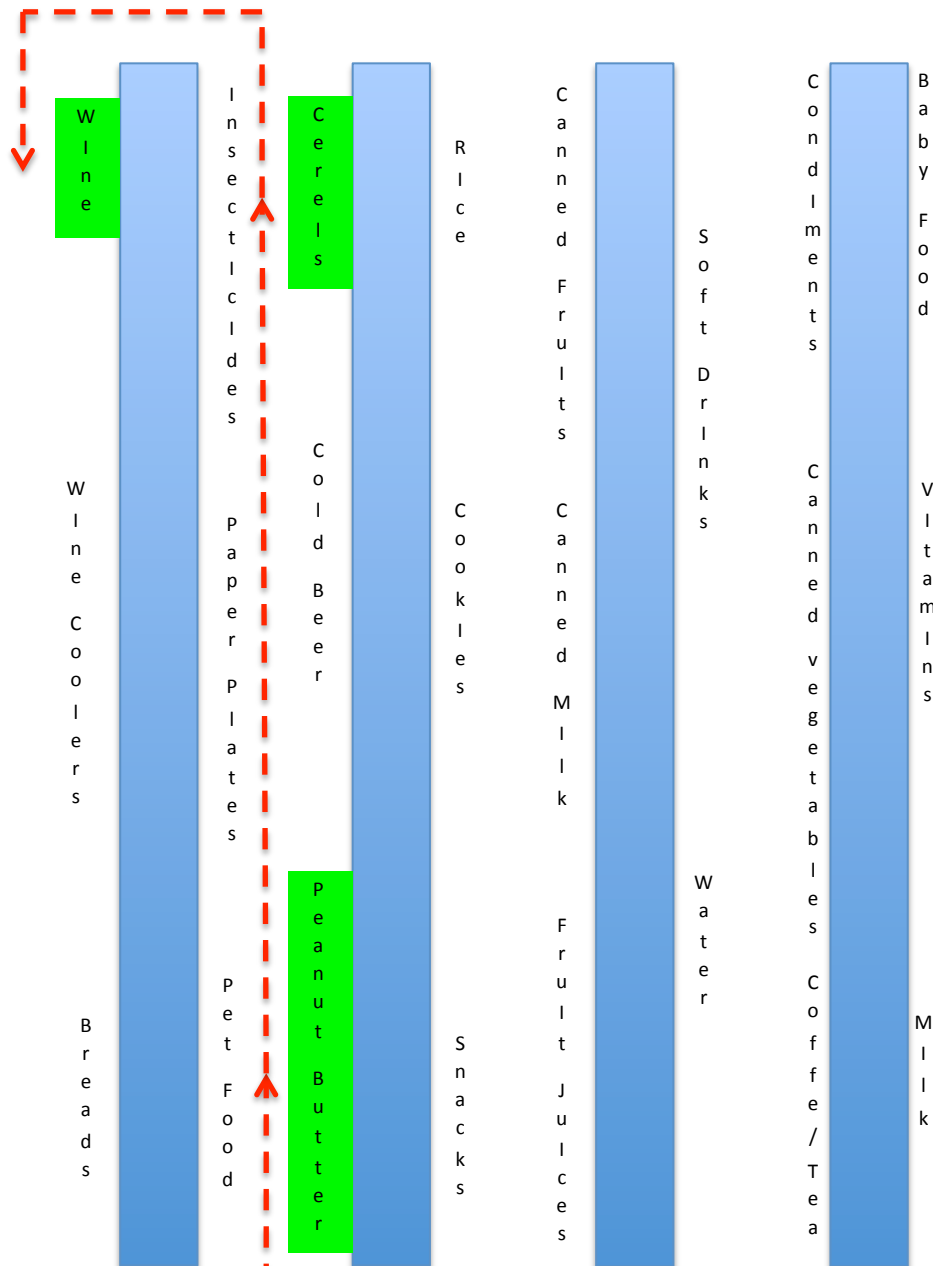
Start point →



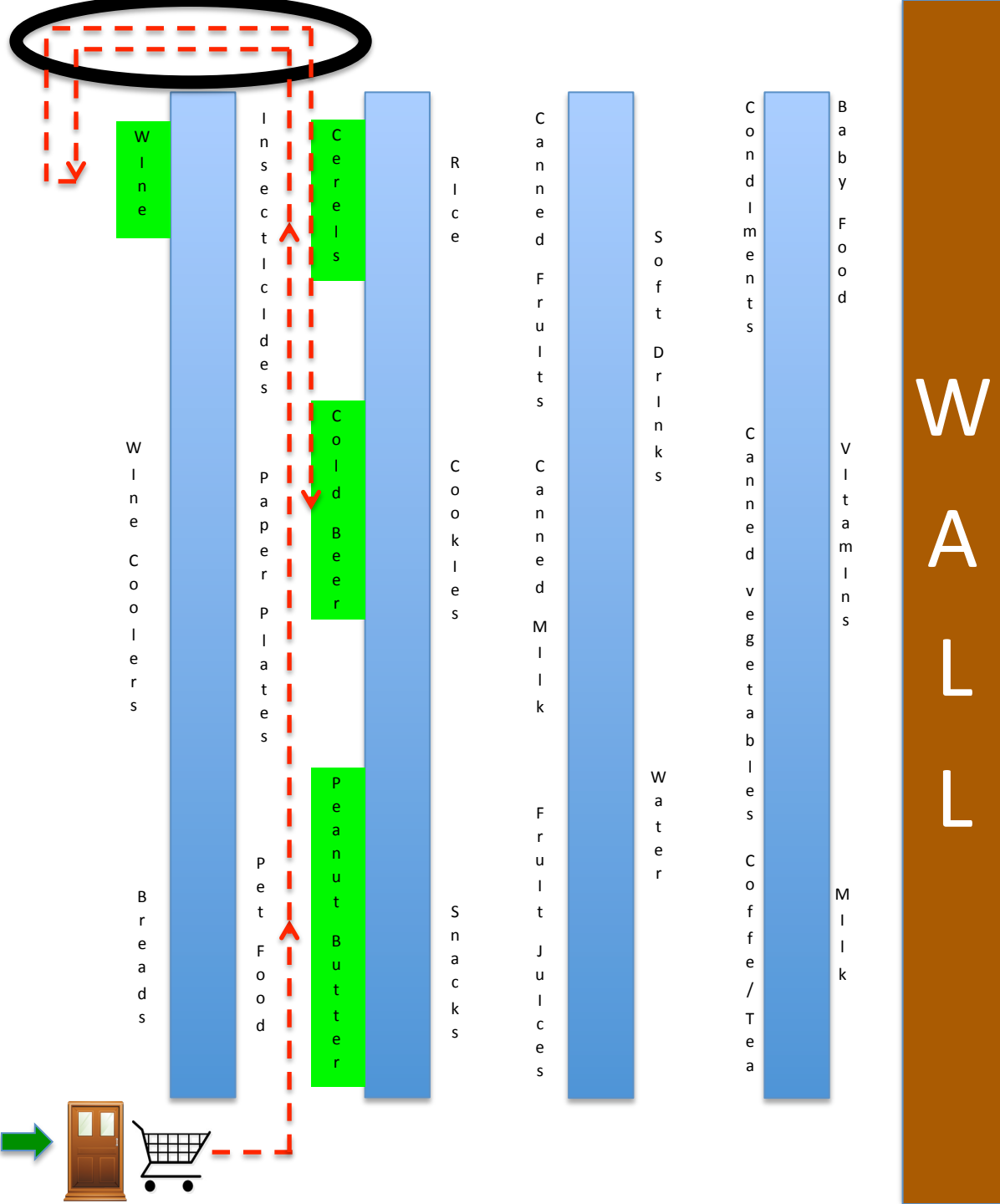
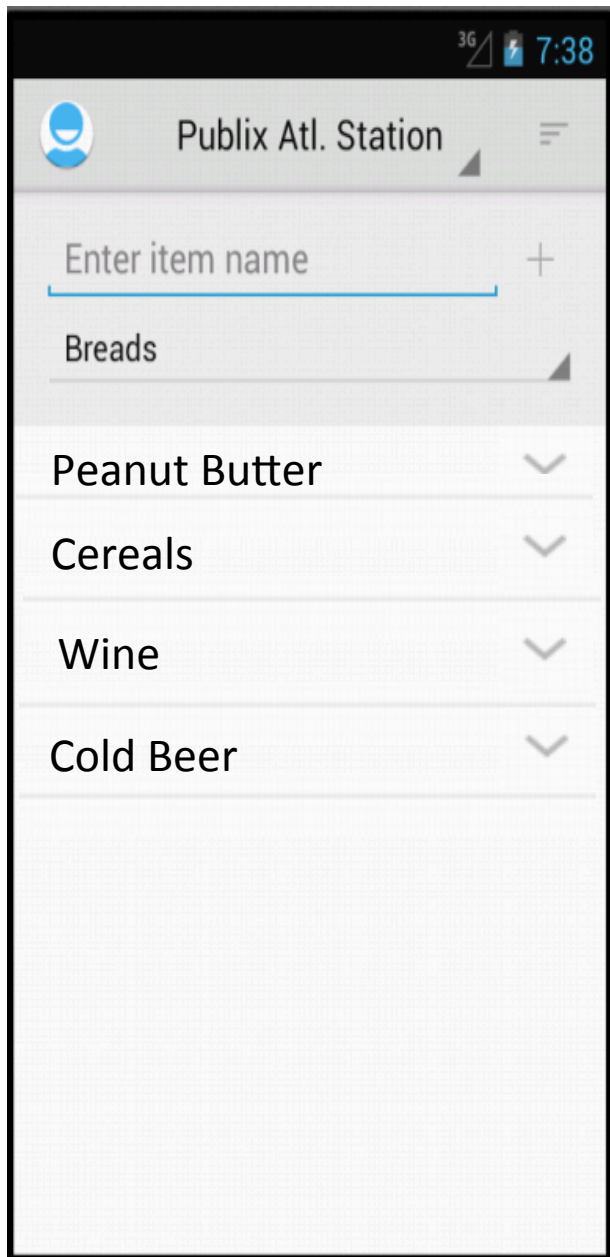
WALL

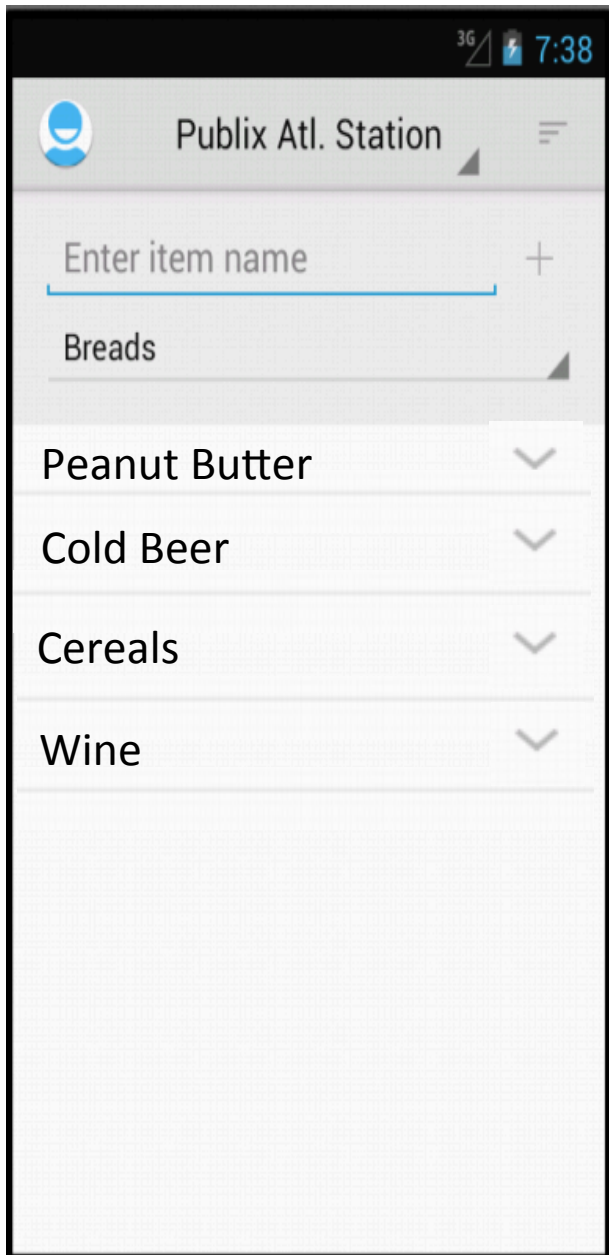


Start point

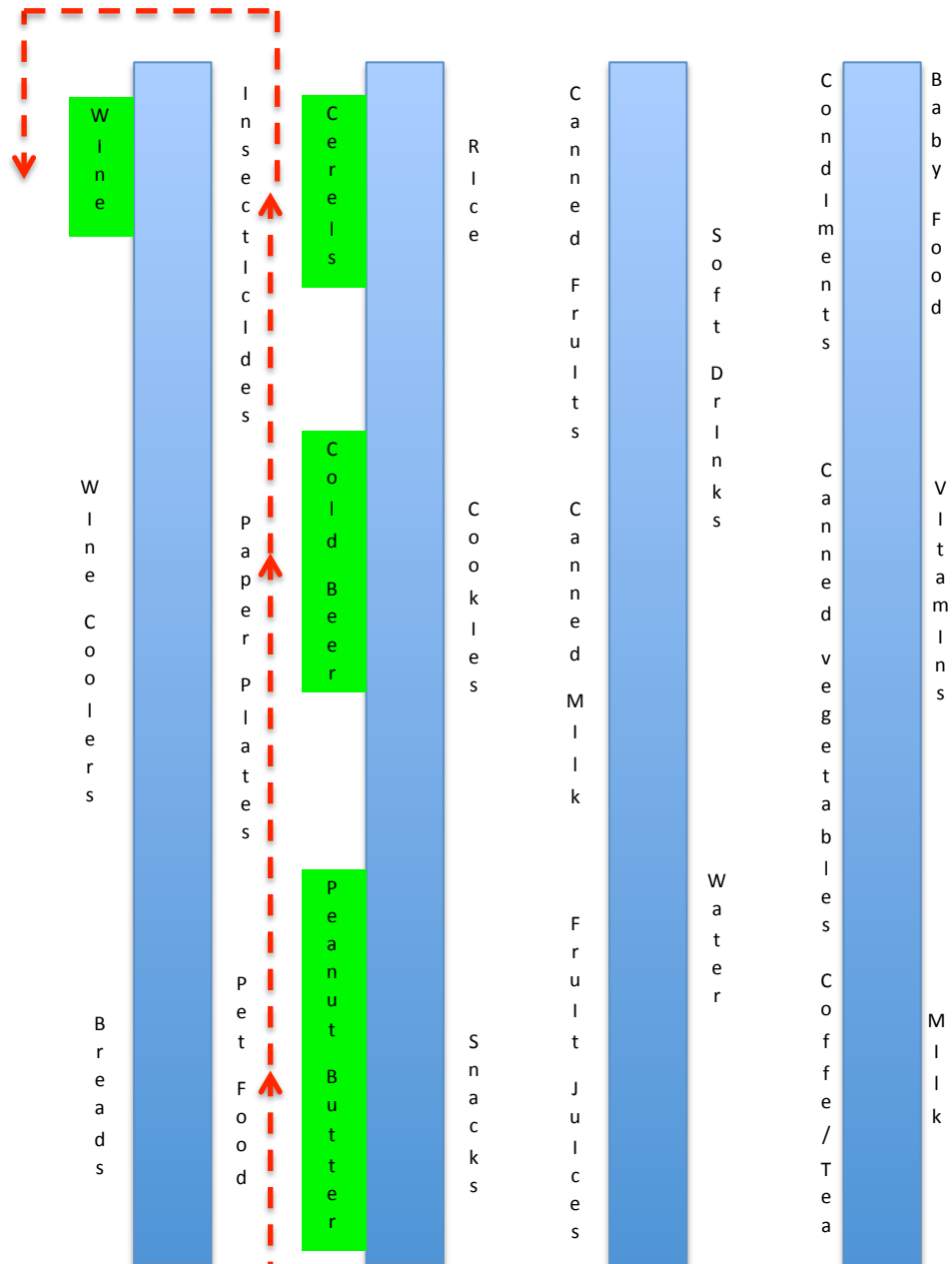


WALL

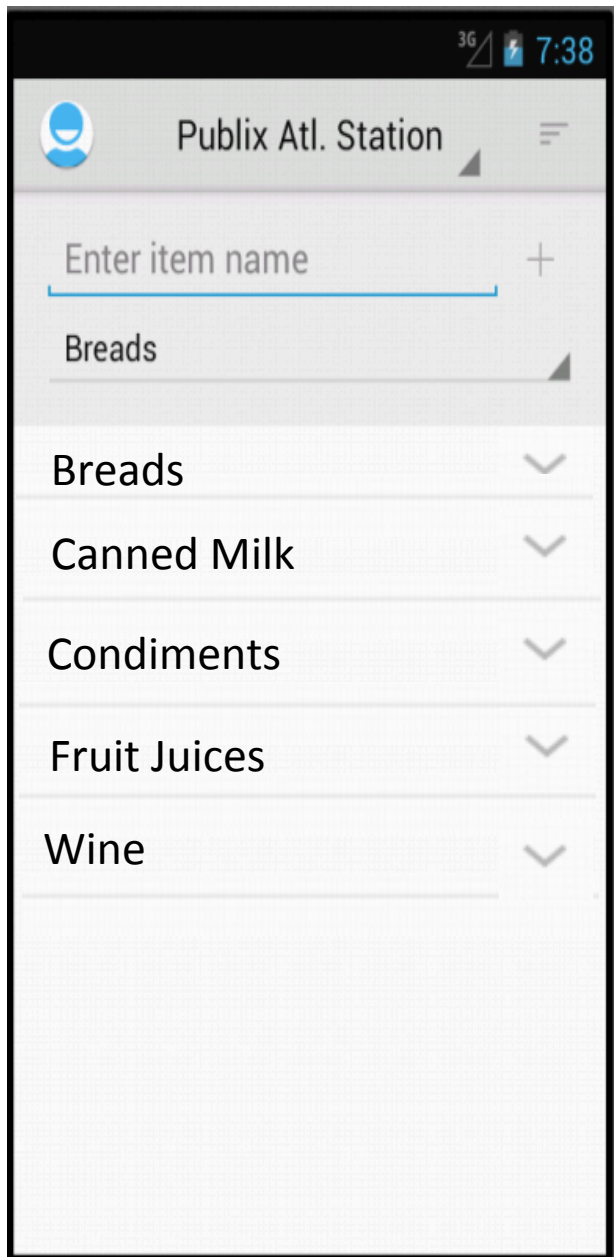




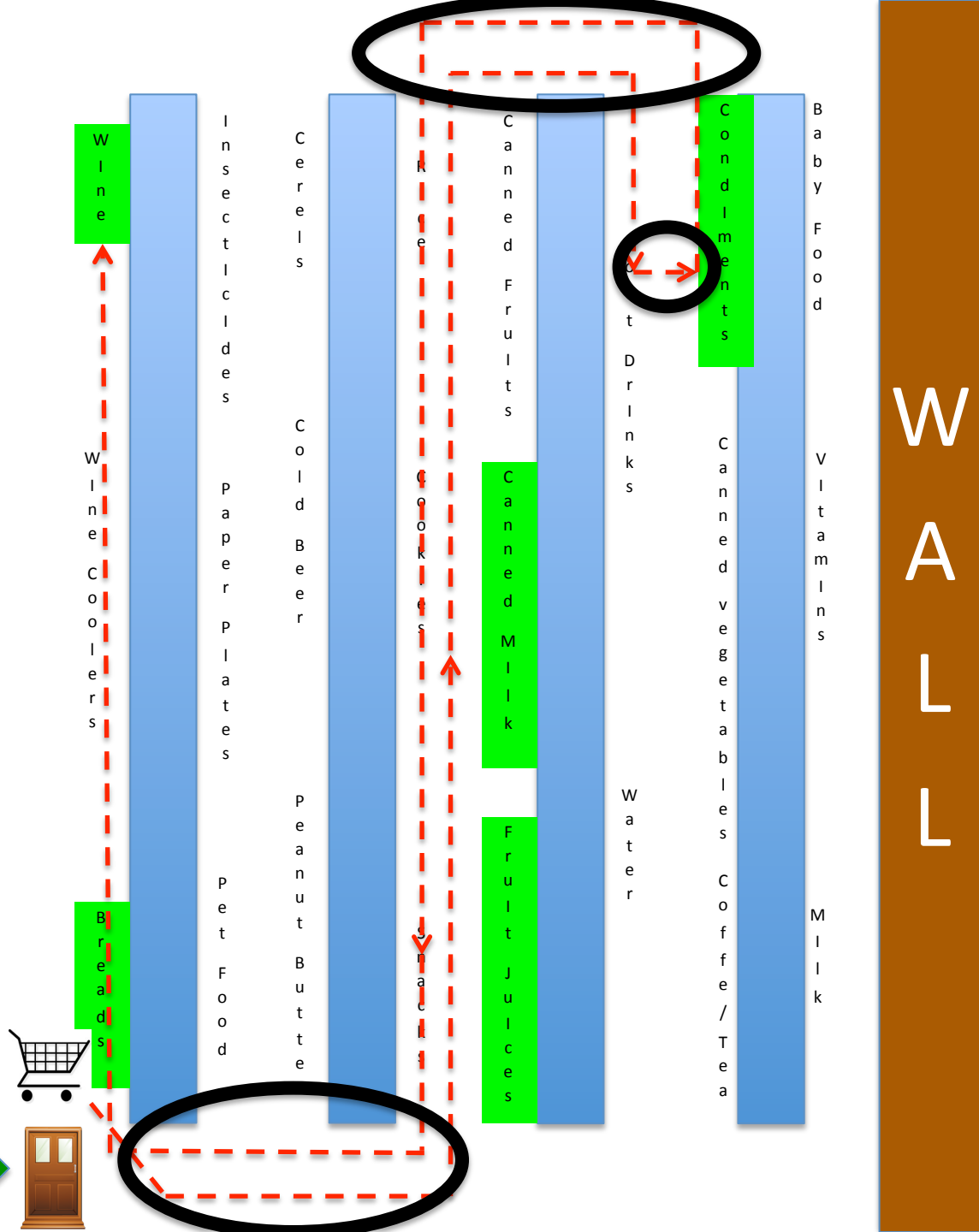
Start point

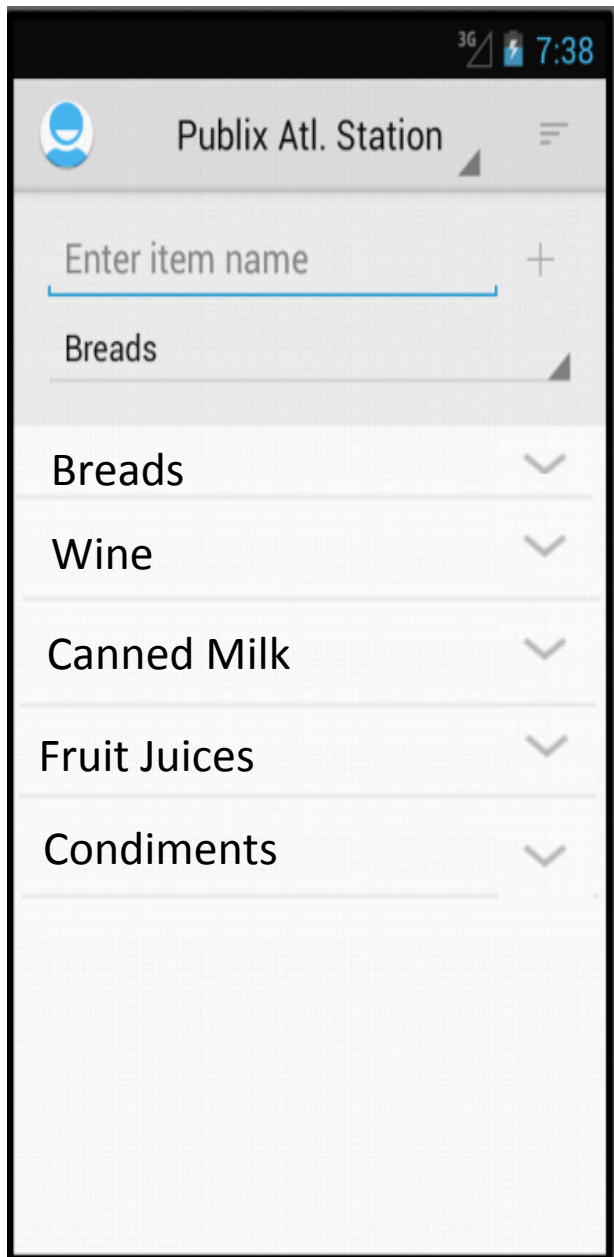


WALL

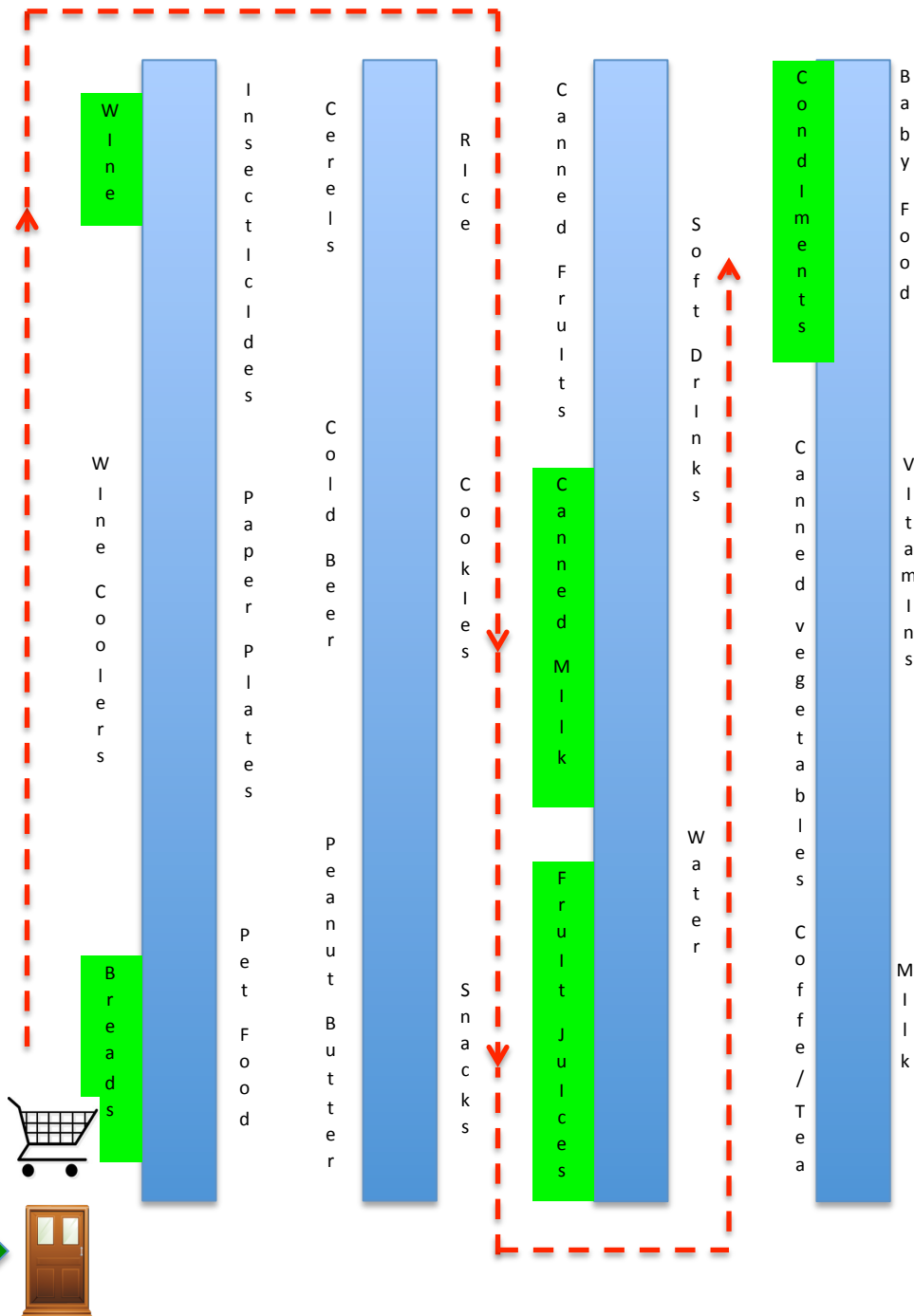


Start point →

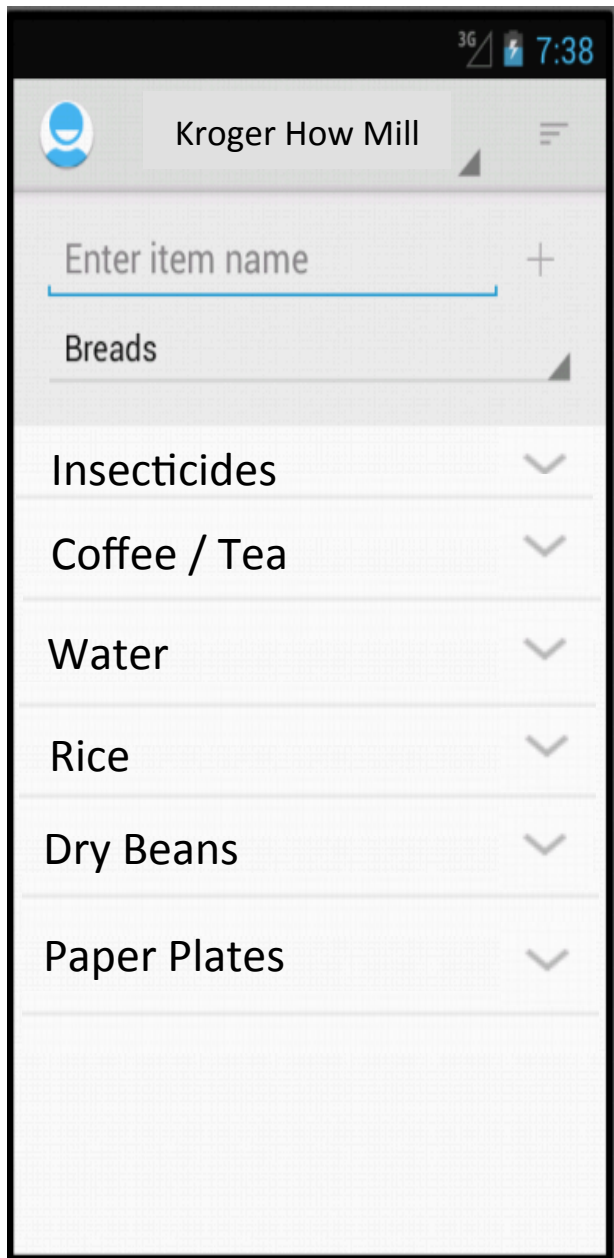




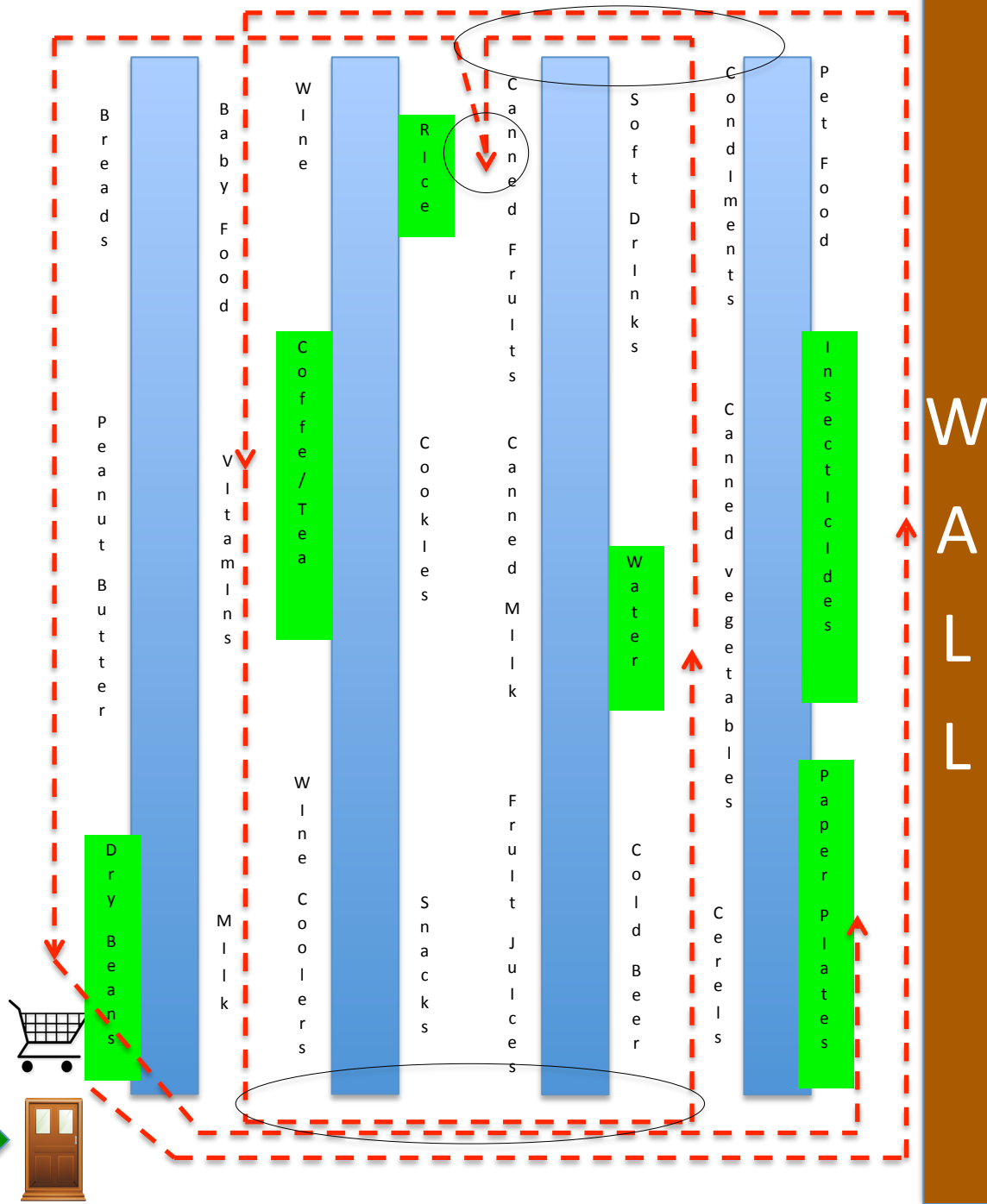
Start point →

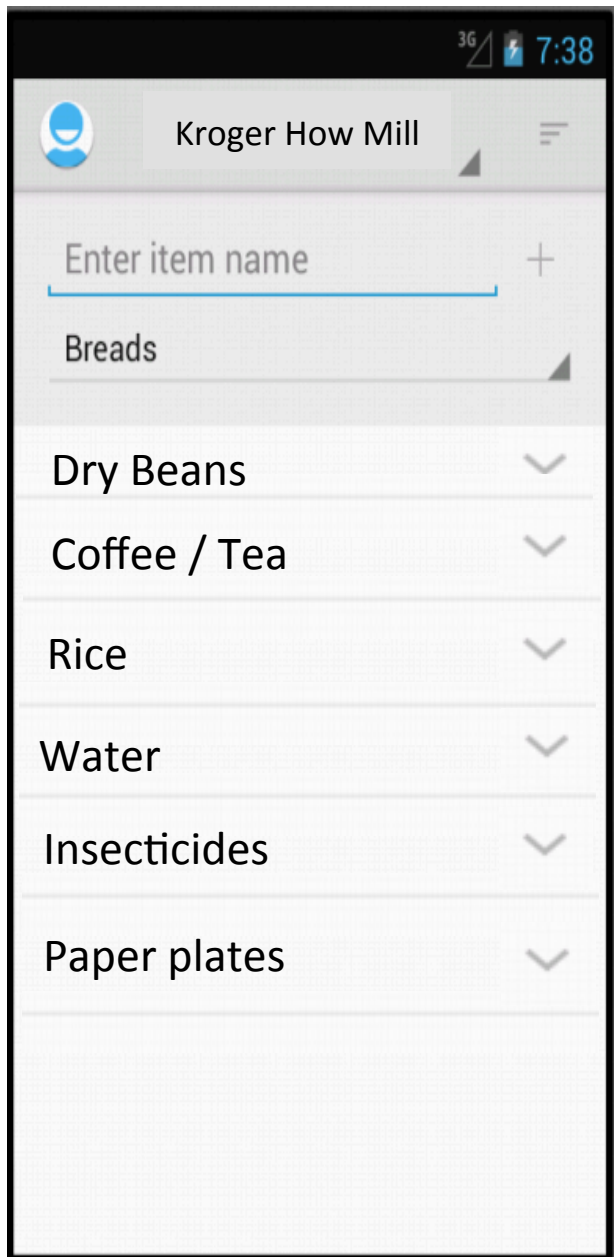


WALL

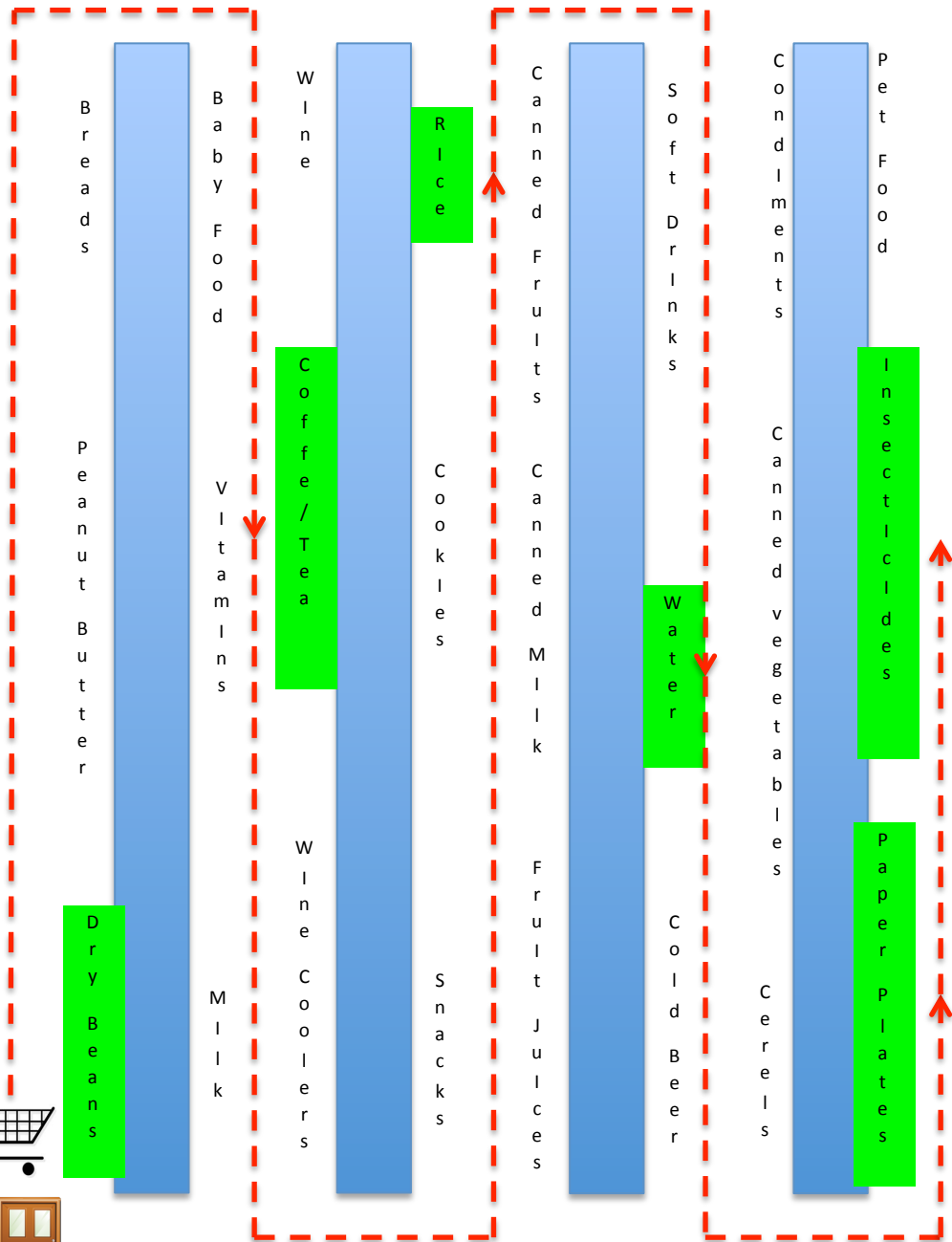


Start point →



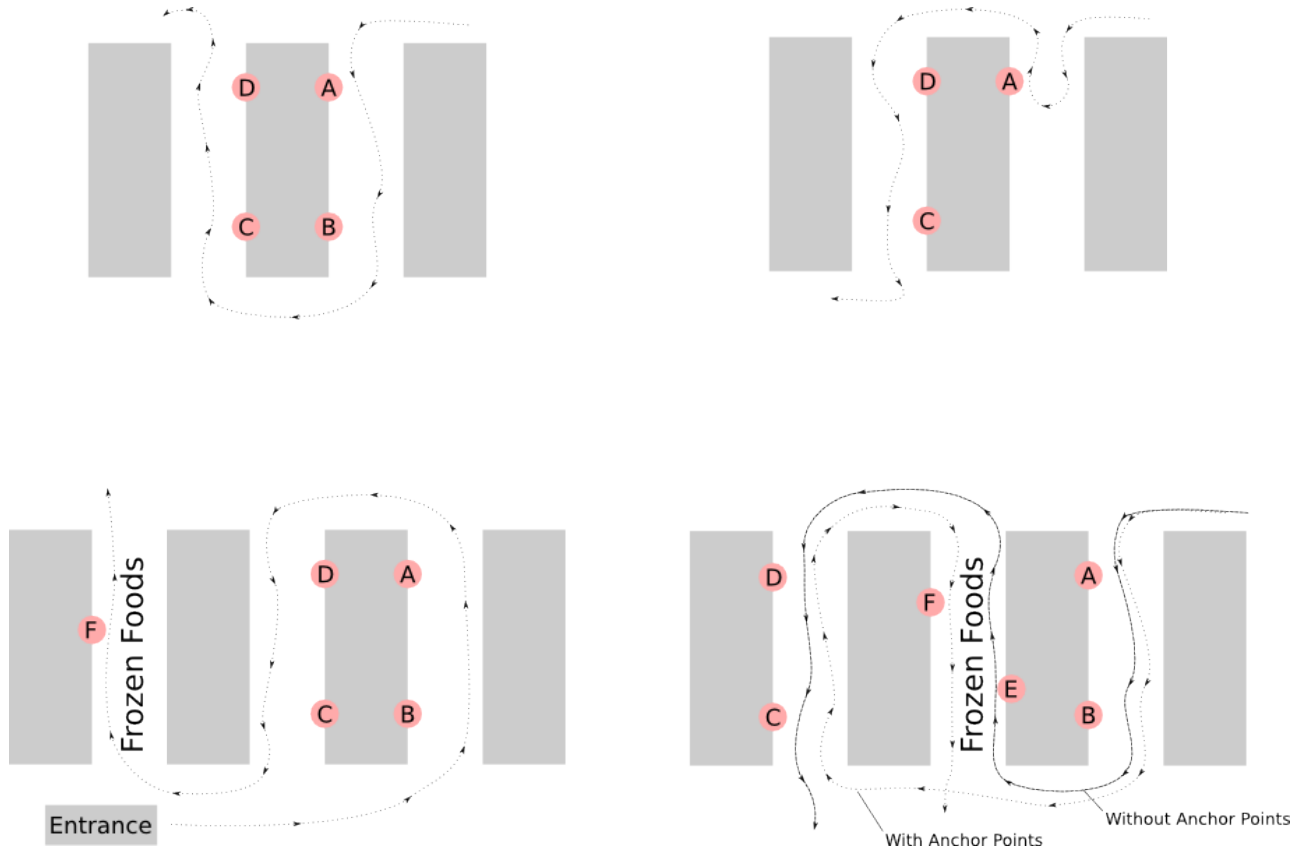


Start point →

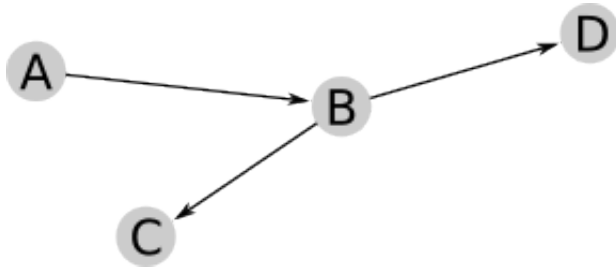


WALL

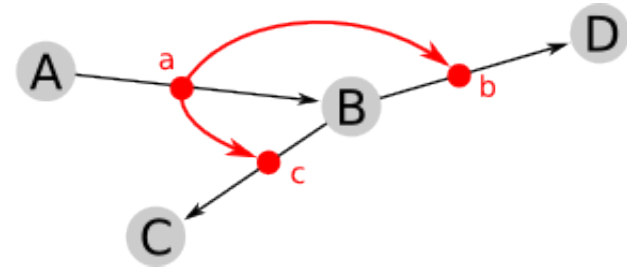
Issues in finding shortest path



Modeling Turns



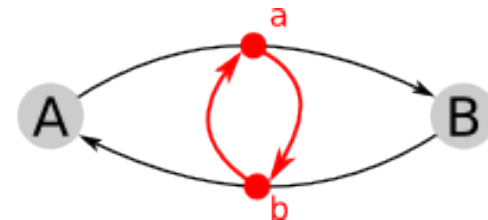
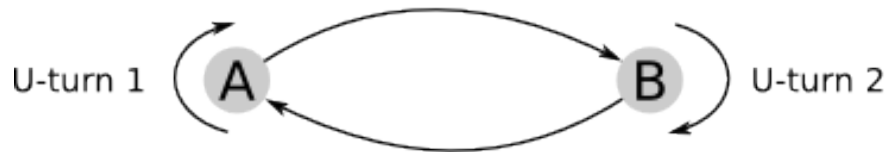
Primal Graph



Dual Graph:

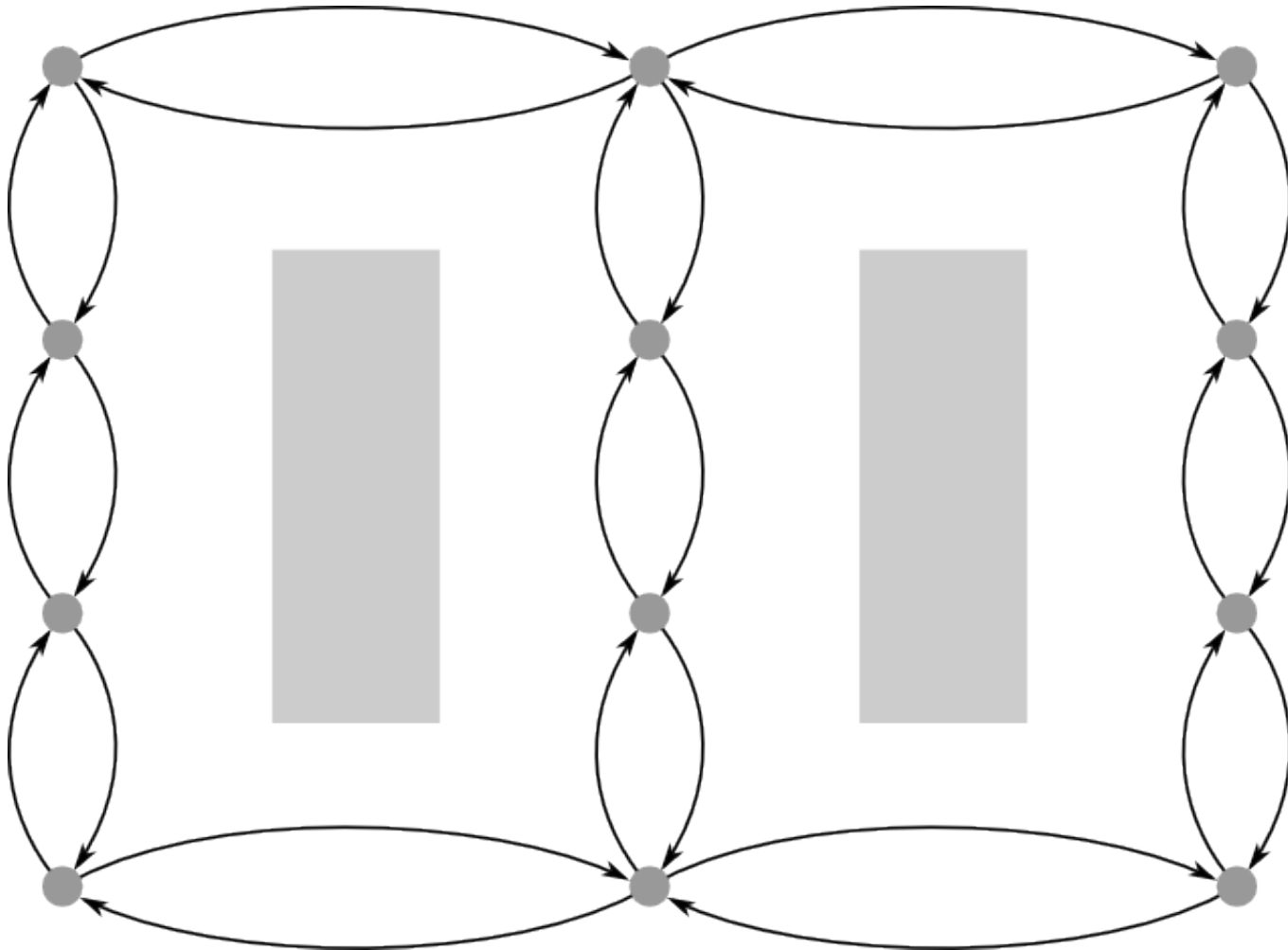
Nodes: Primal edges

Edges: Turns in primal graph

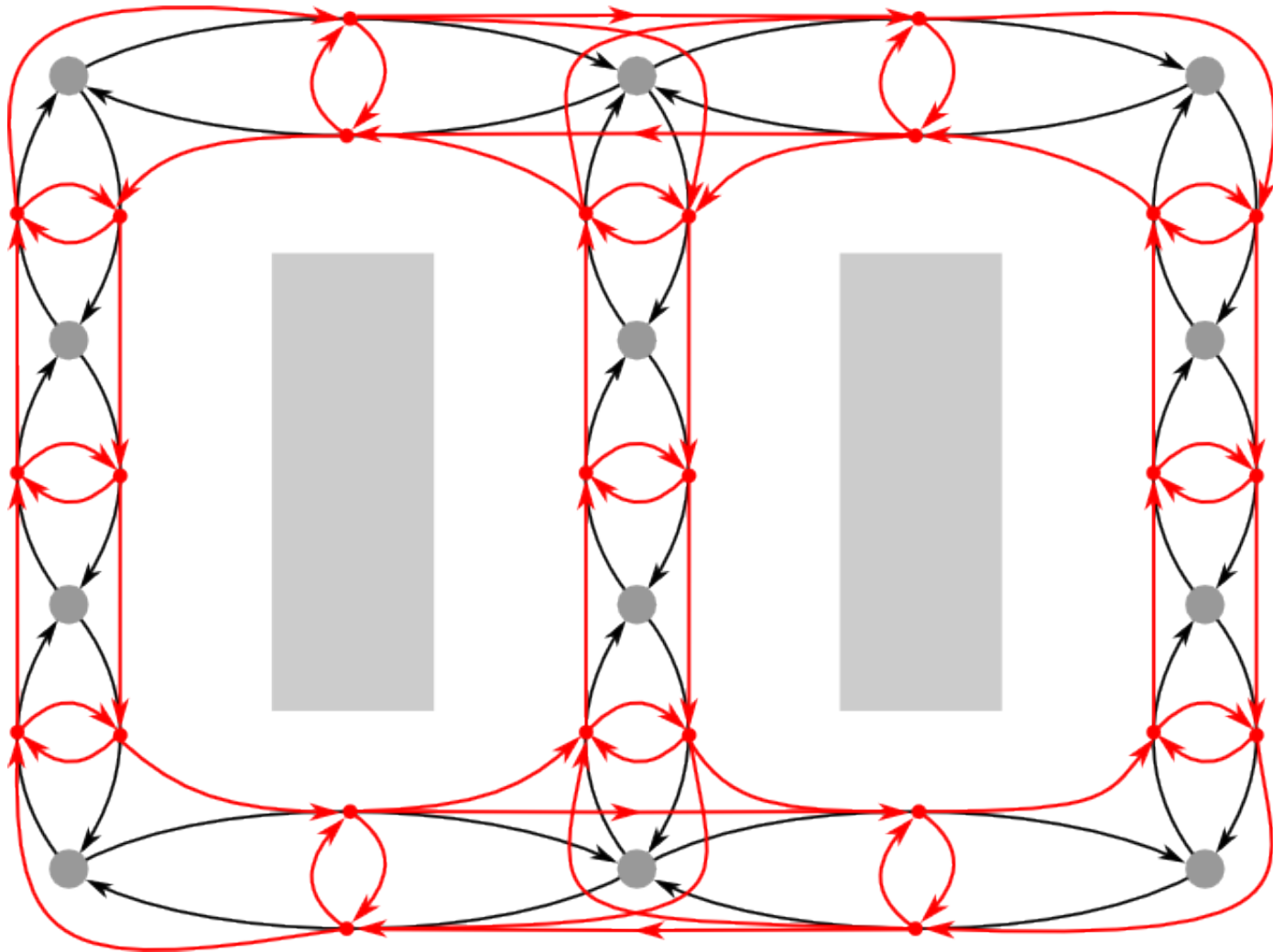


U-turns modeled naturally

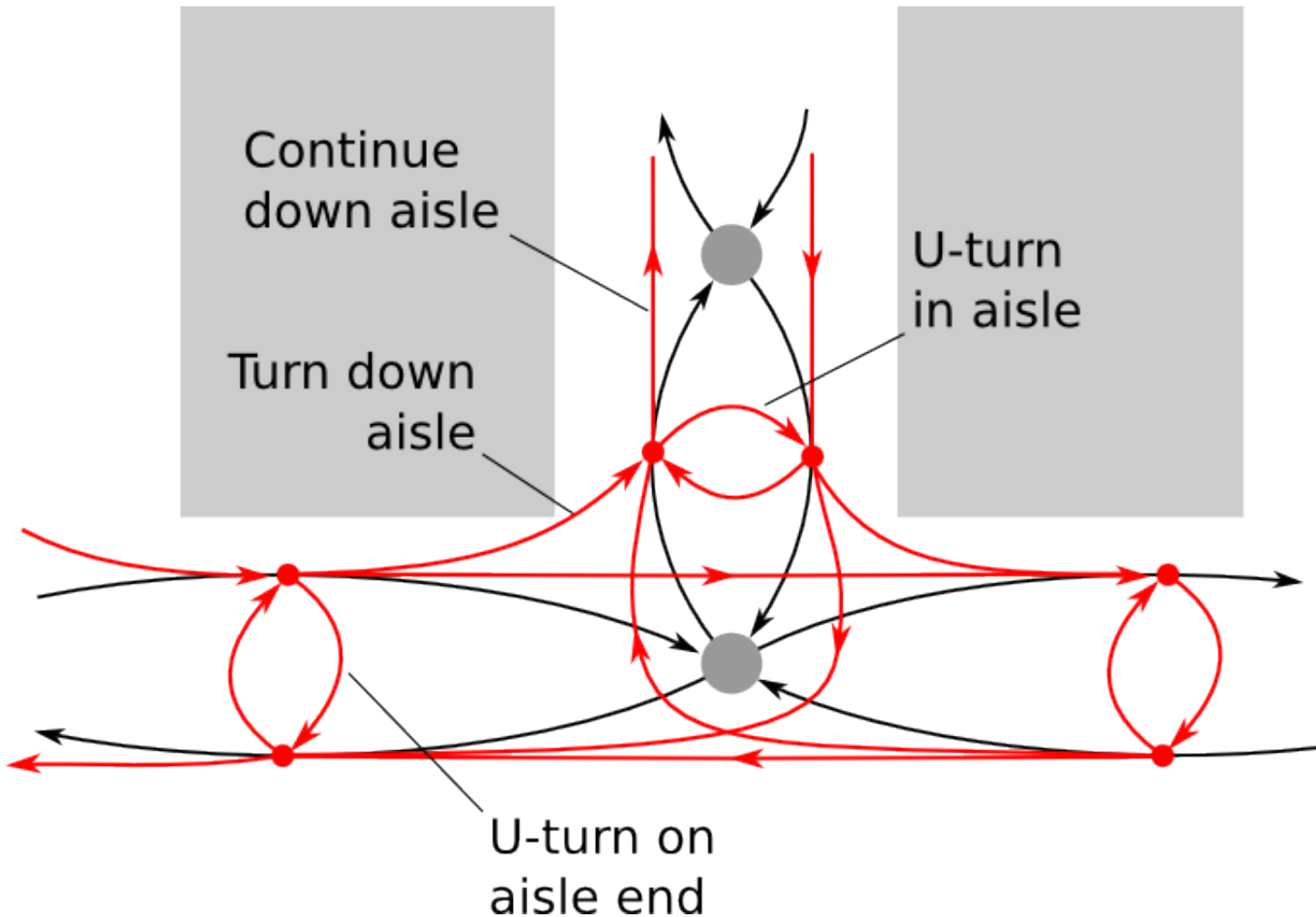
Layout of the store



Layout of the store



Turns in detail



Algorithm

- Start with fully characterized Dual graph
- Use all-pairs shortest path finder (Dijkstra's algorithm on every node)
- Each category associated with
 - a set of **primal edges**
 - a set of **dual nodes**
- Consider distance from a dual node to a category as:

$$\text{dist}(u, C) = \min_{v \in C} \{\text{dist}(u, v)\}$$

- Greedy algorithm to walk from category to category

Future Work

- Sync the app to automatically gather store layout data into the database.
- Integrate Google Maps in the app to automatically get the store location.
- A useful additional feature for QuickShop would be an easy way for a user to map out a store himself; such data could then be made available to other users in the same area