Table of Contents

1. Project Title and Authors

Team Number: 11
Team Name: Grindhouse
Team members:
Mallika Jain 6559234094
Stacy Tran 5305330579
Sandeep Suresh 7720774451
Ethan Smith 5910317497
Shayanna Ahuja 6061783107

2. Preface

Bean&Leaf is an Android application that provides customers with an easy way to discover local coffee shops and tea houses and merchants a platform to market their establishments as well as track the success of their business. In this document we outline the major system components, the high-level architecture explaining how those system components interact with each other, and an implementable detailed design in order to give a better picture of how this application is going to be created.

3. Introduction

This document details Bean&Leaf's architectural and design mappings and delves deeper into the technicalities of our application's structure. From a class standpoint, the application will consist of six classes and one interface: User interface, Merchant Class, Customer Class, Store Class, MenuItem Class, Trip Class, and an Order Class. Due to the shared functionalities and attributes of the customers and merchants, both classes inherit from user. The other classes are necessary to satisfy the trip, order, and menu requirements. The classes are distinguished in said ways to optimize functionality and establish a clean-cut organizational structure.

In order to accomplish the main features of Bean&Leaf, we use three APIs in conjunction with the aforementioned classes: Google Charts API (to display relevant information for the customer/merchant), Google Maps API (to get directions to nearby cafes and to display maps for the customer/merchant), and finally the Geolocation API (to notify merchants that a customer is currently in the store). The major components in our system are based off of two different starting points: when the user creates an account and when the merchant creates an account and adds a store to their profile. These APIs enable our core functionality and the nature of their usage will be further explained in the remainder of the document.

4. Architectural Design

Major Components

Component	***Input***	***Output***
User Edits Profile	User Creates Profile (takes current information)	Updates the user's profile by collecting the new input from the user and displaying it
Merchant Edits Profile/Store	Merchant Creates Account and adds Store (takes current information about the merchant and the store)	Updates the merchant's profile and store's menu items by collecting the new input from the merchant and displaying it.
Geolocation API	User Creates a Profile (user's location)	User Purchases an Item in Store (sends notification that the user is in the store)
Google Maps API	Merchant Creates Profile and Adds Store (inputs their store's location, adds menu items to their store)	The merchant's store gets added to the map with a location icon and the menu of that store is visible when the user clicks on the store in the map
User Purchases an Item in Store	Google Maps API (leads user to store), Geolocation API (tracks user's location)	Google Charts API, Recommendations, Caffeine Intake (display and notifications)
Google Charts API	User Purchases an Item in Store (user's purchase history)	Charts for the user to view in the app
Recommendations	User Purchases an Item in Store (user's purchase history)	Recommendations for the user to view in the app
Caffeine Intake	User Purchases an Item in Store	User can see their caffeine intake for that day/ gets notified when they hit certain caffeine limits

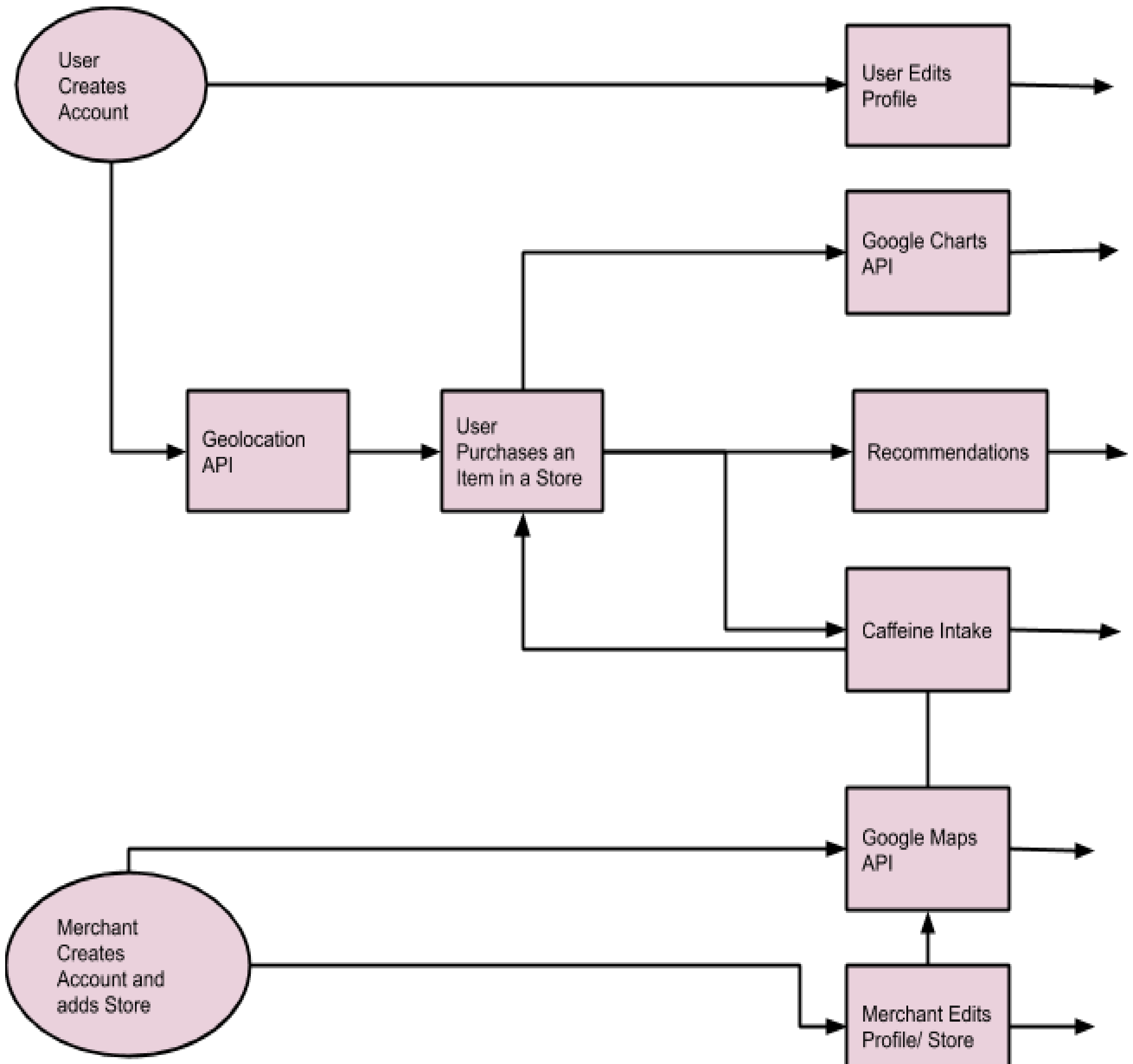
Components to Requirements

Component	***User Stories it Relates to***	***Explanation***
User Edits Profile	4	**4**: Displays the user's profile after any edits or changes are made
Merchant Edits Profile/Store	9, 11, 12	**9**: Displays the merchant's profile after any edits or changes are made **11**: Displays the store's menu after any edits or changes are made **12**: Displays all of the merchants stores and their respective menus/other information after any edits or changes are made
Google Charts API	6, 10	**6**: Displays the user's purchase at a store **10**: Displays all user's purchase history at a given store
Google Maps API	1, 2, 3	**1**: Allows user to view nearby coffee shops and tea houses on a map once merchant adds them. **2**: Gives directions to a coffee shop from the list of shops available on the map. **3**: When the merchant adds items to their store when creating their profile, it will be displayed when the user clicks the location icon of the store created by the API
Geolocation API	13*	**13**: The app recognizes the user is in a coffee shop or tea house using the API and a pop up occurs encourages the user to input their purchase.
User Purchases an Item in Store	5	**5**: Records the user's purchase at a store
Recommendations	8	**8**: Recommends drinks to the user's using their purchase history
Caffeine Intake	7	**7**: Calculates the user's caffeine intake from their purchase history that day and alerts the user when they hit certain limits

* This is an addition to the listed user stories in 2.1. Below is the added user story:
13. Title: Notify the user when they are in a coffee shop or tea house by having a pop up encouraging them to input their purchase at the store.

Description: The user should be immediately prompted to record their purchases when they enter a coffee shop or tea house. Once the user makes a purchase, they can record it through the pop up.

Block Diagram

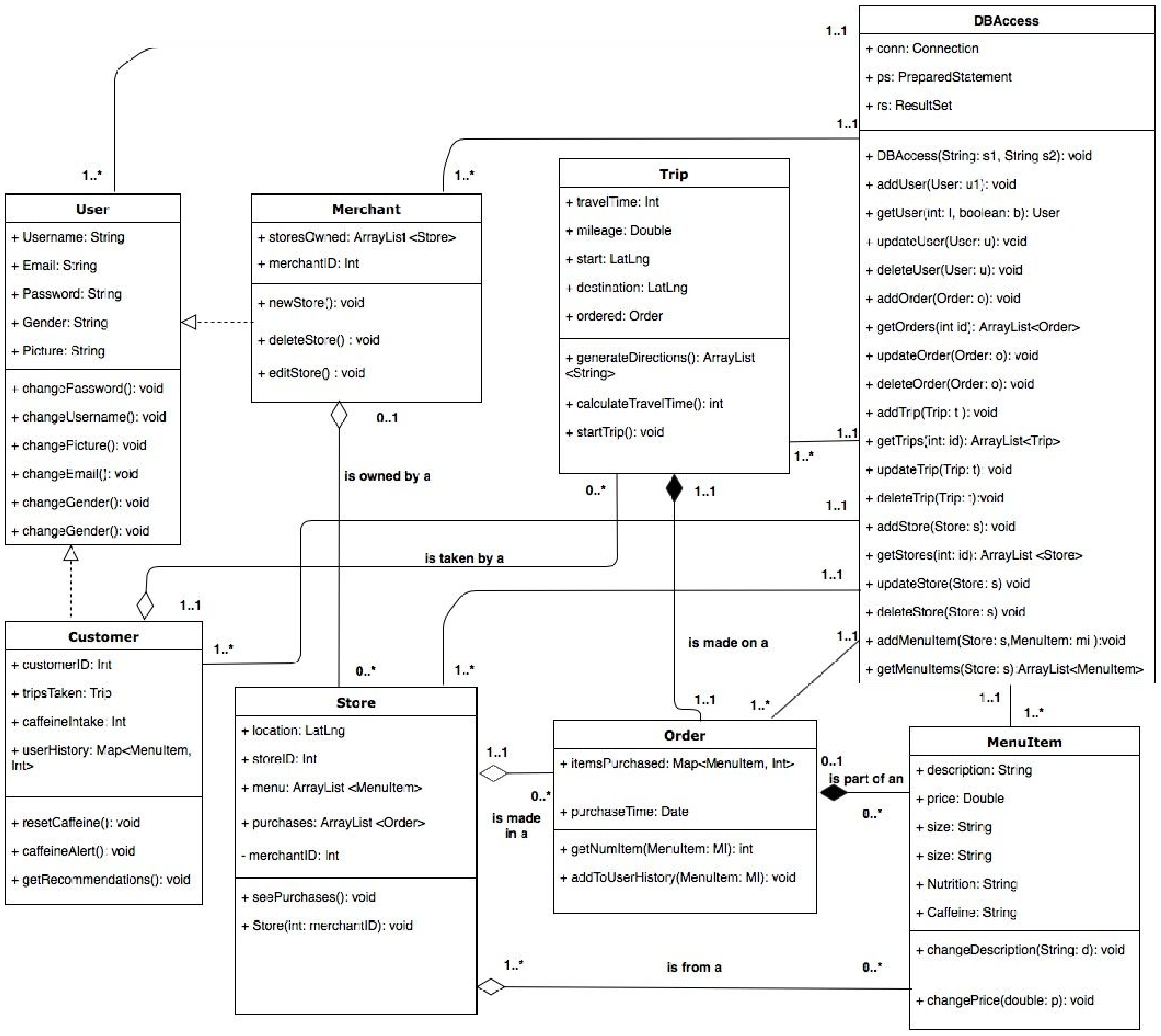


Rationale

The major components in our system are based off of two different starting points. The first starting point where data originally gets generated is when the user creates an account. The second starting point is when the merchant creates an account and adds a store to their profile. At both of these points, the rest of our components are triggered based on certain actions by the user and merchant. When the user or merchant first create their profile, they have the opportunity to edit the information they initially inputted. The User Edits Profile and Merchant Edits Profile/Store components get triggered, take in the initial input, records the changes made by the user and merchant, and outputs the updated information. After the creation and modification of a merchant profile with the store location information, the Google Charts API takes the given location and outputs a chart for both the user and merchant that displays the stores location. Once the user creates an account, their information is sent to the Geolocation API that captures the user's current location and notifies the user when they arrive at a coffee shop or tea house. This notification is the input for the user to add their purchases during that trip to the coffee shop. The information collected when the user inputs their purchase at a certain shop gets passed to three different components: Google Charts API, Recommendations, and Caffeine Intake. The Google Charts API displays the user's purchasing history in a variety of charts. The Recommendations component uses the user's purchasing history to display drink suggestions, unique for the user. The Caffeine Intake component uses the user's purchasing history for that day to calculate their total daily caffeine intake and alert the user if they have surpassed certain thresholds.

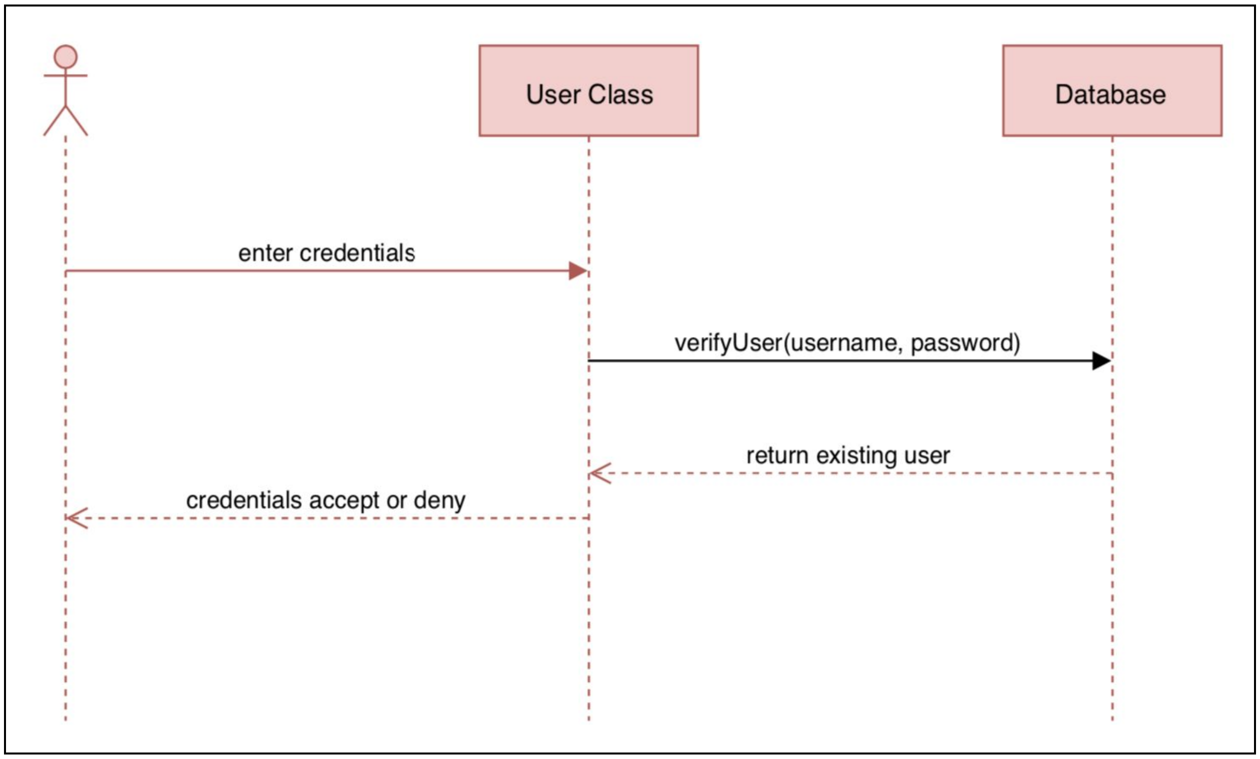
5. Detailed Design

Class Diagram

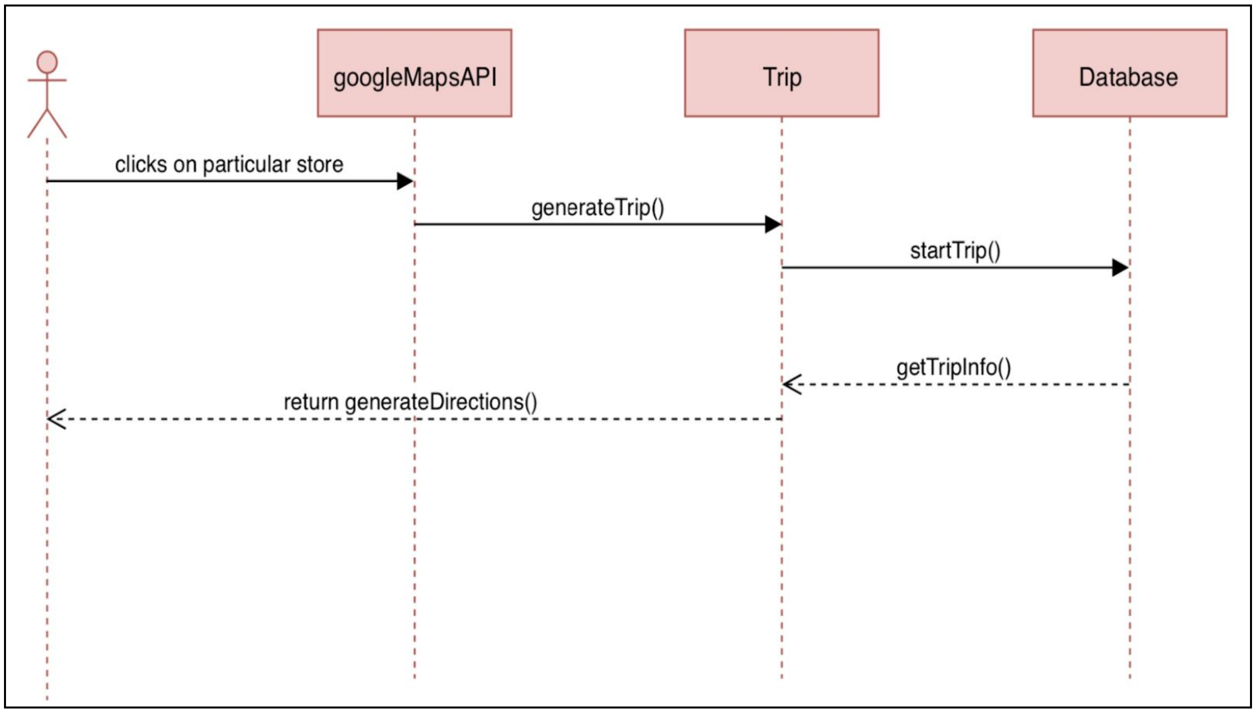

DBAccess
+ conn: Connection
+ ps: PreparedStatement
+ rs: ResultSet
+ DBAccess(String: s1, String s2): void
+ addUser(User: u1): void
+ getUser(int: I, boolean: b): User
+ updateUser(User: u): void
+ deleteUser(User: u): void
+ addOrder(Order: o): void
+ getOrders(int id): ArrayList<Order>
+ updateOrder(Order: o): void
+ deleteOrder(Order: o): void
+ addTrip(Trip: t): void
+ getTrips(int: id): ArrayList<Trip>
+ updateTrip(Trip: t): void
+ deleteTrip(Trip: t):void
+ addStore(Store: s): void
+ getStores(int: id): ArrayList <Store>
+ updateStore(Store: s) void
+ deleteStore(Store: s) void
+ addMenuItem(Store: s,MenuItem: mi):void
+ getMenuItems(Store: s):ArrayList<MenuItem>
User
+ Username: String
+ Email: String
+ Password: String
+ Gender: String
+ Picture: String
+ changePassword(): void
+ changeUsername(): void
+ changePicture(): void
+ changeEmail(): void
+ changeGender(): void
+ changeGender(): void
Merchant
+ storesOwned: ArrayList <Store>
+ merchantID: Int
+ newStore(): void
+ deleteStore() : void
+ editStore() : void
Trip
+ travelTime: Int
+ mileage: Double
+ start: LatLng
+ destination: LatLng
+ ordered: Order
+ generateDirections(): ArrayList <String>
+ calculateTravelTime(): int
+ startTrip(): void
Customer
+ customerID: Int
+ tripsTaken: Trip
+ caffeineIntake: Int
+ userHistory: Map<MenuItem, Int>
+ resetCaffeine(): void
+ caffeineAlert(): void
+ getRecommendations(): void
Store
+ location: LatLng
+ storeID: Int
+ menu: ArrayList <MenuItem>
+ purchases: ArrayList <Order>
- merchantID: Int
+ seePurchases(): void
+ Store(int: merchantID): void
Order
+ itemsPurchased: Map<MenuItem, Int>
+ purchaseTime: Date
+ getNumItem(MenuItem: MI): int
+ addToUserHistory(MenuItem: MI): void
MenuItem
+ description: String
+ price: Double
+ size: String
+ size: String
+ Nutrition: String
+ Caffeine: String
+ changeDescription(String: d): void
+ changePrice(double: p): void
is owned by a
is taken by a
is made on a
is made in a
is part of an
is from a
1..1
1..*
0..1
0..*

Sequence Diagrams

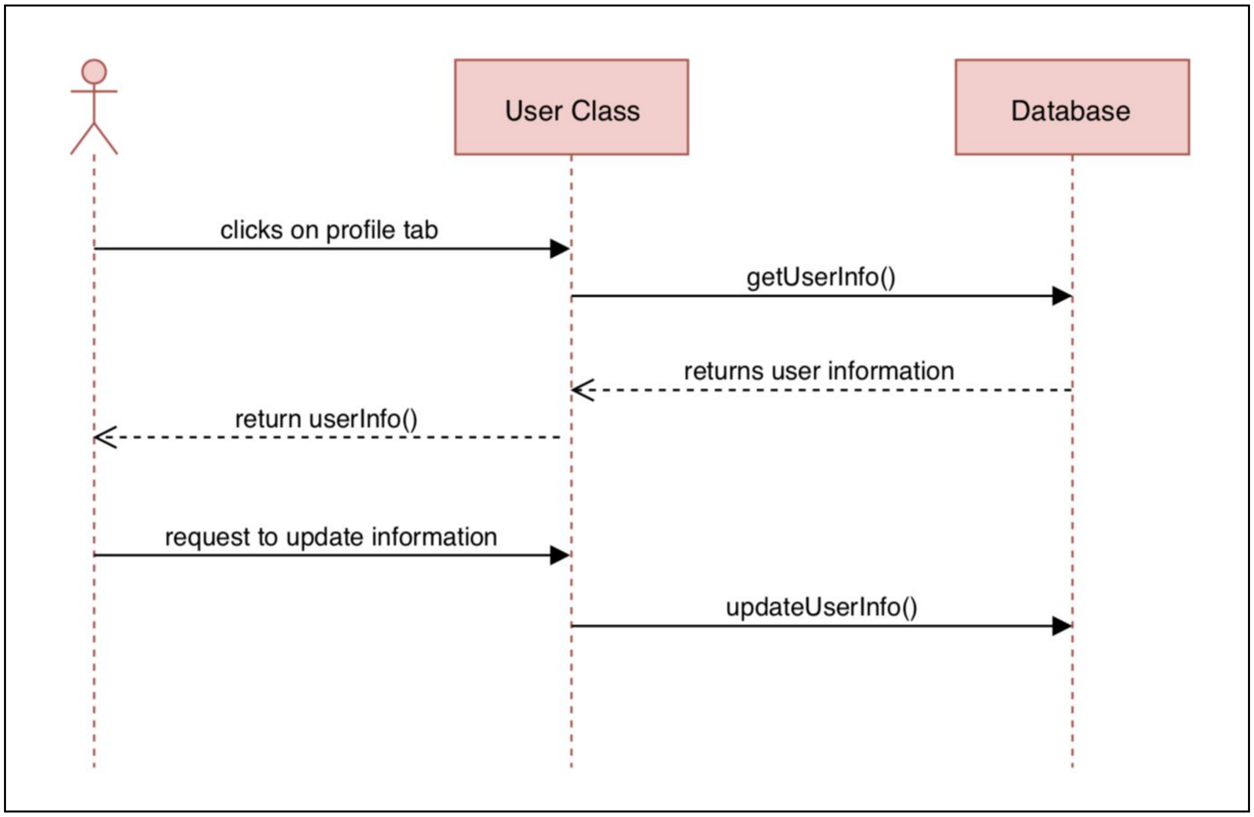
1. Merchant or customer login



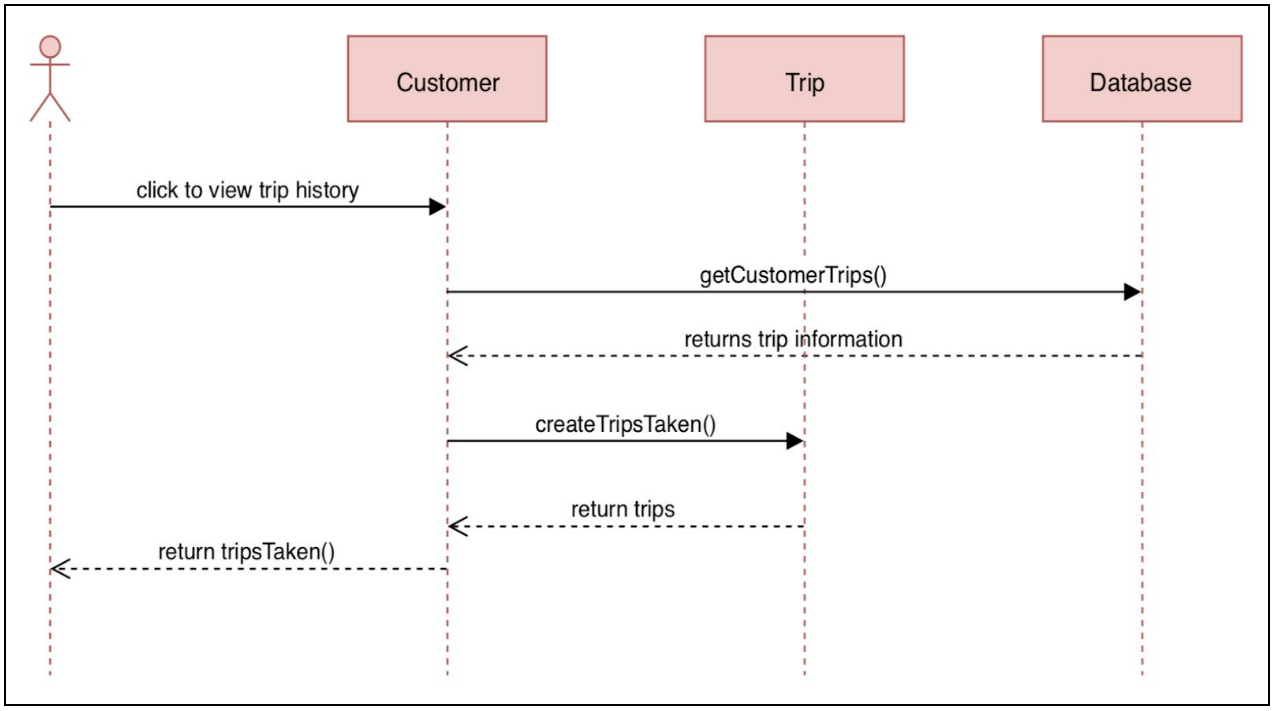
2. Retrieving directions to store



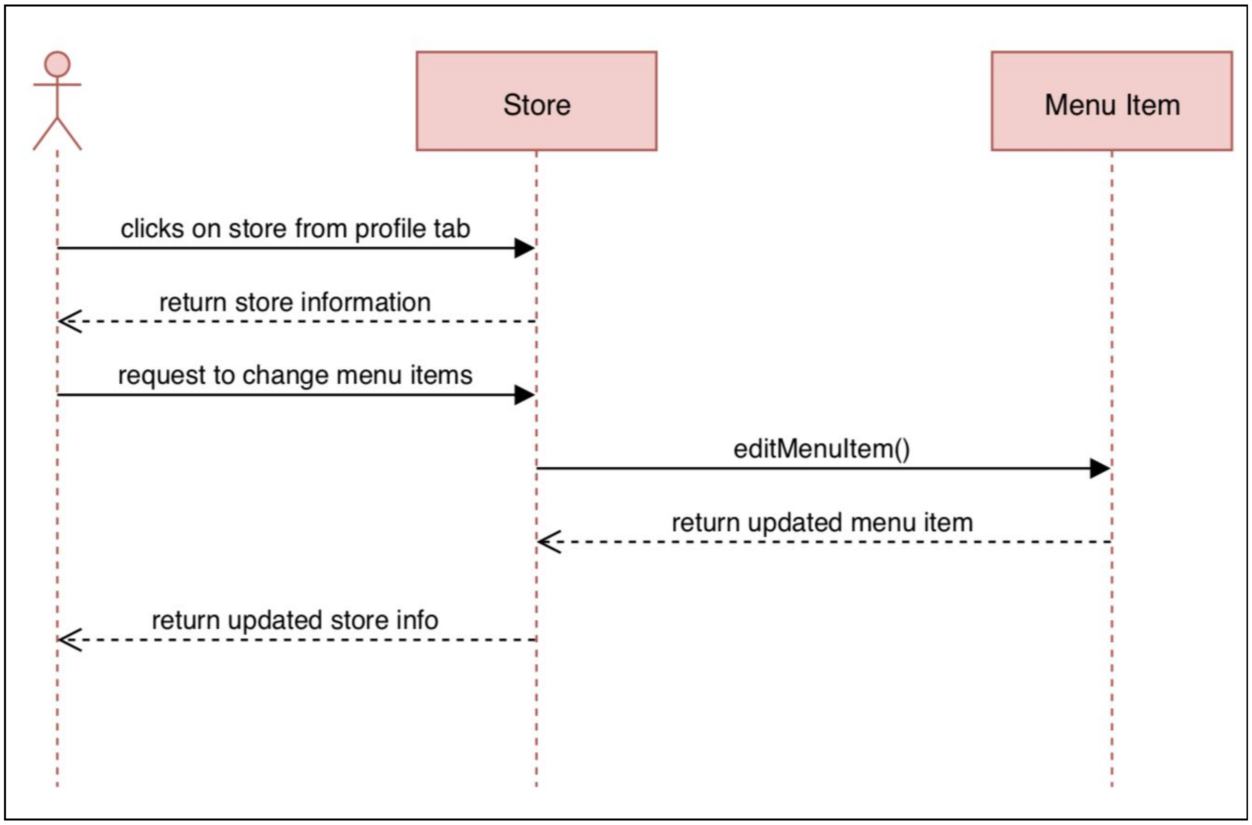
3. Merchant or customer updating profile information



4. Retrieving customer trip history



5. Updating menu items for merchants


Store
Menu Item
clicks on store from profile tab
return store information
request to change menu items
editMenuItem()
return updated menu item
return updated store info

Relationship of Java Classes to Block Diagram Components

For most of the components in the block diagram, the relationship to the detailed design is self-explanatory. For example, a user creating an account is associated with the User and DBAccess classes. A user getting directions to a store would use the Store, Trip, and DBAccess classes. Alerting a user to too much caffeine intake would use the Store, MenuItem, User, and DBAccess classes. However, for some components in the block diagram the relationship to the classes is less clear. One example is when a user purchases an item and records the order. In this case, the Store, MenuItem, Trip, Order, and DBAccess classes are all involved in recording the transaction within the app. Finally, the recommendation component will use the Order, Trip, User, and DBAccess classes. Note that all components will use the DBAccess class.