

**Team 9 - Design Document: CS407**

**Team Members:**

* Ashwin Prasad ([prasad13@purdue.edu](mailto:prasad13@purdue.edu))
* Tomasz Parzadka ([tparzadk@purdue.edu](mailto:tparzadk@purdue.edu))
* Brian Rhee ([brhee@purdue.edu](mailto:brhee@purdue.edu))
* Han Wang ([wang2786@purdue.edu](mailto:wang2786@purdue.edu))
* Jaeyeon Kim ([kim802@purdue.edu](mailto:kim802@purdue.edu))

[**Purpose**](#_emzvvtpn6l6m) **3**

[**Design Outline**](#_mxvh9zjc6pwv) **4**

[High Level Overview](#_4ow82r18w6x0) 4

[Activity Flow Diagram](#_3an50tjzcumj) 5

[**Design Issues**](#_ne04ummbrej2) **6**

[**Functional Issues**](#_2oghaqmrezwz) **6**

[How should we add medications/consumables to the user’s profile?](#_cmayxa8r17hl) 6

[How should users navigate to their preferences, profile, and other application features?](#_ys54ugt6q47t) 7

[Users must be able to maintain an account](#_rwjkflm3t7lj) 7

[Users must be able to find a consumable/medication from a barcode picture, QR code scan, taking a or manual search of the name](#_o1qjl7ppauie) 7

[Users can see conflicts between consumables, medications and personal limitations](#_myzd3iefco5i) 7

[**Non-Functional Issues**](#_ydzjw9cnh3mm) **8**

[Should we develop the application for Apple or Android?](#_gt580rphl9k3) 8

[How can we secure the users account?](#_7q9i62tjiiwg) 8

[Which scripting language should we use?](#_uwuaw7sdokxe) 8

[What language should we use for front end development?](#_pw87oyqpse6e) 8

[What database should we use?](#_s1qk7nq85hr1) 9

[How should a user login to our application?](#_es7xjtj8dxcv) 9

[Which picture API should we use for the camera service?](#_x034mdyx3f3l) 9

[Do users need to create an account to use our services?](#_le80hz8h4pu2) 9

[**Design Details**](#_7yx2goscxxt0) **11**

[Class Level-Design Diagram](#_vhhw9e54ikbt) 11

[**Description of Classes and their Interaction**](#_c43qt9jyneam) **12**

[User](#_w8lnvxlff8q5) 12

[Medication](#_u32iate4nfz7) 12

[Allergy](#_3i7g6ijlvklt) 12

[Account](#_w5uk4n9ye9q) 12

[EventHandler](#_ytszk5ki13ij) 12

[CameraHandler](#_qe6g4xb9g2pb) 12

[CameraPreview](#_cf3mivper8hm) 12

[RequestHandler](#_z3xslsdljkrw) 12

[AllergyAnalysis](#_wil9ft6rattn) 13

[**Sequence Diagrams**](#_48jx60kdlgnr) **14**

[Sequence of events for Registration](#_3816owb0ghfr) 14

[Sequence of events when user wants information about a consumable/medication](#_afvk7lf2nq4o) 15

[Sequence of events when the user checks their conflicts](#_3i6crd23qm3v) 16

[The sequence to see conflicts begins with the user wanting to see their conflicts. The request then checks between the user’s medications and personal limitations for any conflicts. The conflicts are then displayed to the user.](#_dfpc7mcyjkk3) 16

[**UI Mockup  
UI Overview**](#_uvn7fu9faxct) **17**

# 

# 

# **Purpose**

Customers often find themselves struggling to find out whether particular foods or medicine, fit their dietary needs. It is important to make sure the product does not conflict with their allergy restrictions, and to find out fast. There are a few applications that have similar functionalities, but they are outdated and only supports old version of mobile devices with not user-friendly UX. Using camera functionality with Google Vision API, our app allows users to take a snap of a QR code, barcode or take a picture of the ingredient section from the product packaging to quickly identify whether the product is safe for them. AllergySnap aims to help users with allergies easily identify and avoid ingredients in potentially harmful products with a simple image recognition system.

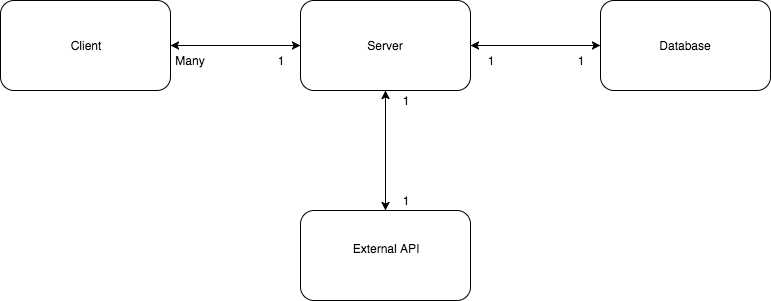
#### 

# 

# **Design Outline**

#### High Level Overview

The Android client and our server form a many-to-one architecture to allow a large number of users to use our application concurrently. The server queries the database to fulfill requests from the client. The server first validates the request made by the client, and if a valid request is made, the server queries the database and then returns an appropriate response to the client.

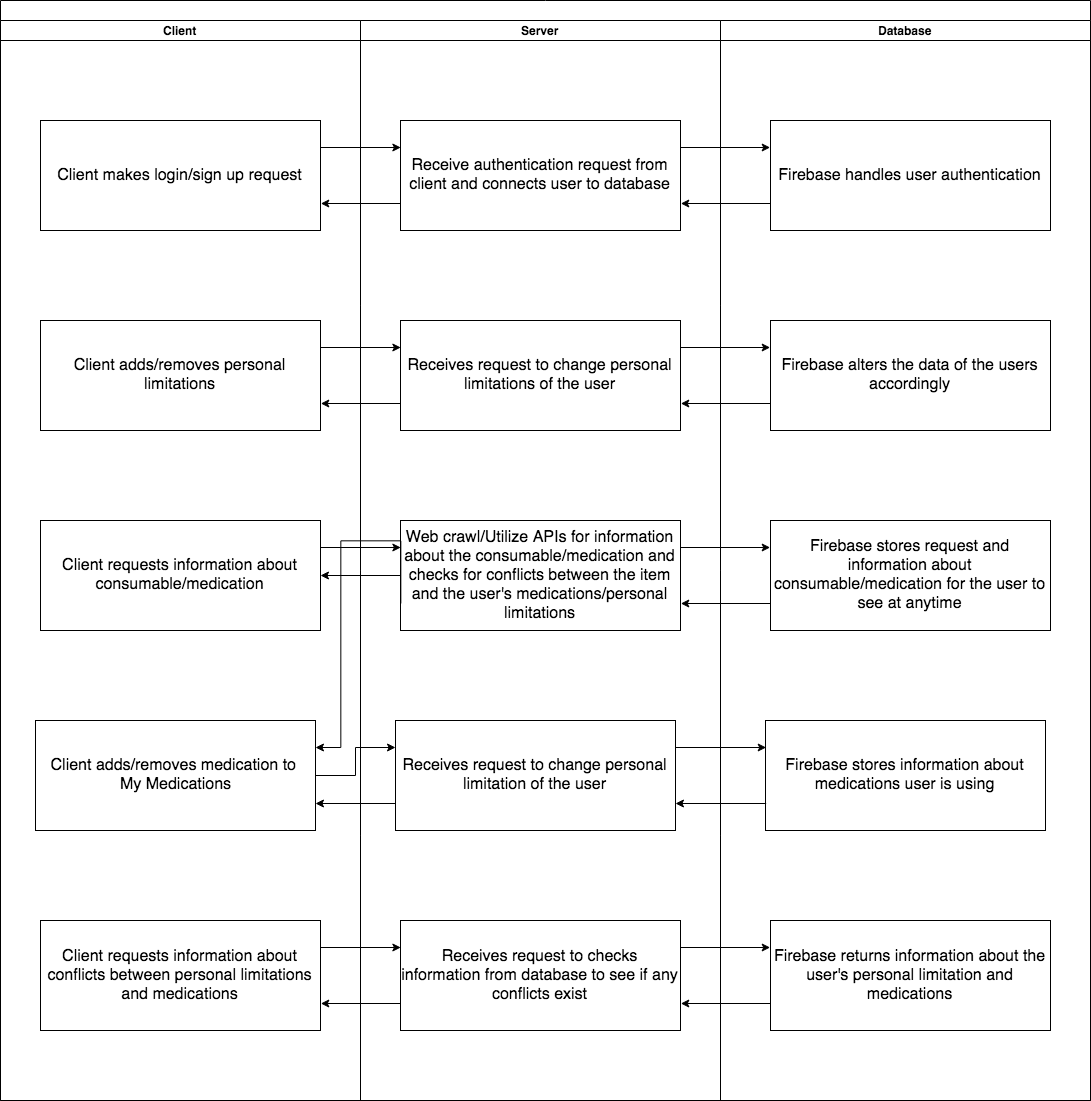


1. Client:
   1. Client sends a request to the server using URLSession.
   2. Client receives request from the server as a URLResponse.
   3. Client processes the response and displays the response appropriately.
2. Server:
   1. Server receives a request from the client and sends queries to the database to collect the appropriate data.
   2. Server receives the data from the database, process the data for an appropriate response, and sends the response to the client using JSON.
   3. If appropriate, the server sends requests to external APIs (e.g. Google Vision API) to process the client’s request.
   4. Server will process the response returned from external APIs and sends a JSON response back to the client.
3. Database:
   1. Database receives a query from the server and responds to the server with the queried data.
   2. Database will also process user authentication through Firebase’s services
4. External API:
   1. External APIs receive requests from the server and responds to the server with the requested information.

#### Activity Flow Diagram

Several examples of flows of events are shown in the diagram below.

* A flow of user authentication includes a request from the client that is sent to the server. The server validates and sends this request to Firebase, which handles user authentication. Once user authentication is complete, Firebase returns a completion status to the server and ultimately to the client.
* A flow of how information for a consumable/medication that was either searched manually or taken a picture of is shown. We web crawl the information or utilize APIs that have information about foods and medications. Firebase will store this information, so users can go back and see what they’ve previously scanned/searched.
* A flow of how conflicts between medications and personal limitations. The server processes this request and sends requests for information about the user’s medication and personal limitations and utilizing APIs and cross-examining the items, the conflicts are found.



# **Design Issues**

# **Functional Issues**

#### How should we add medications/consumables to the user’s profile?

* Option 1: Add it via QR Code/barcode
* Option 2: Add it via manual search
* **Option 3: Both Option 1 and Option 2**

For users, we want to make it simple and easy to gain information about medications and consumables. Thus, we want to allow multiple avenues of checking medications and consumables. If a user has the item in person, they can take a snap of the item’s barcode/QR code. If not/they just want to manually type the name of the item, they can go about checking their item that way.

#### How should users navigate to their preferences, profile, and other application features?

* Option 1: Paginated browsing with a central page
* Option 2: Central page with icons corresponding to features
* Option 3: Unified sidebar overlay
* **Option 4: Bottom bar with icons corresponding to features**

Based off our decisions to have multiple features , we have decided that having a unified bottom bar with icons corresponding to features as our primary means of bringing context to our application’s main screen. It provides for a simple UX where users can switch over from features of our application with ease. This method leaves the most screen space for when the user wants to navigate between menus.

#### Users must be able to maintain an account

* The user should be able to create their own account, via sign-in / login from email, username, Facebook, Twitter, or Google accounts.
* The user should be able to view their saved allergies and medications that are synced to one’s account.
* The user should be able to securely recover their login information in case they forget (send via login email, Google, Facebook, Twitter account).

#### Users must be able to find a consumable/medication from a barcode picture, QR code scan, taking a or manual search of the name

* While the user is typing in the search boxes for a medication or consumable, possible options for medications/consumables should pop up based on relevance.
* The user should have the option to save a medication to their account and input how frequently they use it.
* The user should be able to see past consumables that they searched or scanned via QR code/barcode.

#### Users can see conflicts between consumables, medications and personal limitations

* The user should be able to check if they have a conflict with a consumable via a barcode/QR code scan, taking a picture of the ingredients of the consumable or manually searching the item. Users can see previous consumables that they snapped via their profile’s feed.
* Conflicts between medications the user has inputted they are taking as well as the inputted personal limitations will be displayed on the “My Conflicts” tab

# **Non-Functional Issues**

#### Should we develop the application for Apple or Android?

* Option 1: Apple
* **Option 2: Android**

The most optimal solution would be to create the application for both Android and iPhone using Xamarin. Xamarin allows for the development of Android, iOS, and Windows apps with a single shared .NET code base, however, due to time limitations, we want to gain familiarity with Android and therefore want to stick strictly to just an Android application.

#### How can we secure the users account?

* **Option 1: Have the user create a strong password.**
* **Option 2: Hide the users information when accessing personal settings. (asterisk)**
* **Option 3: If the user is signed-in, the user must re-enter their current password to be able to change their email or password.**
* **Option 4: If the user wants to recover their account, they will need to have access to email for recovery.**
* **Option 5: Use firebase that provides Firebase CLI.**

Since we want to ensure consumer security, we are taking extra precautions to various scenarios. Firebase provides tools that will allow us to authenticate, authorize, and validate data the user inputs. We are also taking precautions to ensure that none of the users personal information is displayed without validation.

#### Which scripting language should we use?

* Option 1: Node.js
* Option 2: Python
* **Option 3: Java**

We choose Java as our scripting language for the product. The reason for this was because out of all the options, more people had experience working in Java than in the other 2 options. Moreover, Android Studio heavily uses Java and will be our primary work tool in our application development.

#### What language should we use for front end development?

* Option 1: C#
* Option 2: C/C++
* **Option 3: Java**

Since we’re working on an Android application in Android Studio, Java is going to be our front end language. It’s recommended by Google for use for Android development, and it is well documented with detailed, official tutorials on how to get started and build with it. This will be beneficial given our limited timeframe to develop this application.

#### What database should we use?

* Option 1: Hosting with services such as AWS
* **Option 2: Firebase**
* Option 3: Utilizing Firebase along with another DBMS

After thorough research through our database hosting options, we have decided that utilizing Firebase as a standalone database is the most optimal solution for us. Not only is Firebase a free option, it also deals with authentication details out of box, which relieves many future headaches we may face. Firebase has all of the features we need and none we don’t; therefore, it is the most efficient and secure database for our use case.

#### How should a user login to our application?

* Option 1: Login with our proprietary login system
* Option 2: Login via Facebook, Twitter, or Google
* Option 3: No login required
* **Option 4: Combination of options 1 and 2**

We would like to cater our application to all sorts of users, and the large majority of our target demographic has either a Facebook or a Google account. This makes signing up a painless process and will motivate more people to sign in and store their data to their AllergySnap accounts. This provides a more seamless and convenient experience for our users. Logins are preferred for our application because with a login, we can provide user travel history, customized user route suggestions, and saved destinations for users that transfer over across different devices.

#### Which picture API should we use for the camera service?

* **Option 1: Google Vision API**
* Option 2: Amazon Rekognition
* Option 3: Clarifai

When researching different image analysis API’s we came to the conclusion that the Google Vision API supports our efforts the most for this application because it has a simple REST API and does not include irrelevant tags. We did not go with Amazon Rekognition API because it is known for the facial analysis, and therefore, is not required in our application. When researching Clarifai we noticed that is was strong for concept modeling but also provided irrelevant tags.

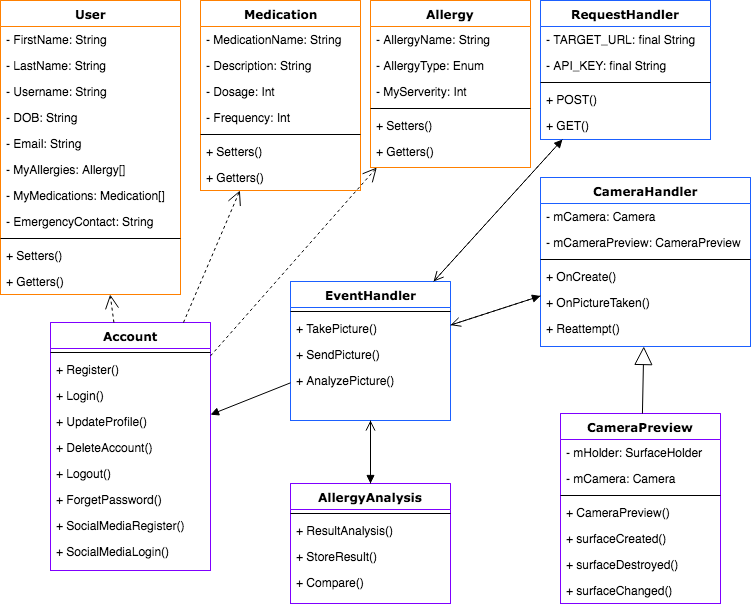
#### Do users need to create an account to use our services?

* Option 1: We will not offer the ability to have an account in our application.
* Option 2: Users can choose whether or not they want to create an account.
* **Option 3: Users must create an account.**

We are requiring users to create an account for security purposes. The user is entering possibly confidential information that should only be displayed for the account it was intended for. Creating an account also allows the user to be able to possibly use the same account on another Android device.

# **Design Details**

#### Class Level-Design Diagram



#### 

#### 

# 

# **Description of Classes and their Interaction**

#### User

* User is a model class that represents the currently logged in user.
* It contains Name, username as well as other user specific fields like My Allergies and My Medications, etc.

#### Medication

* Medication is a model class that represents a certain medicine.
* It holds the the medication name as well as the dosage and frequency for user who are taking them.

#### Allergy

* Allergy is a model class that represents a specific allergy.
* It represents a allergy the user might have, the type of the allergy and how severe for this user.

#### Account

* Account class is a controller class that contains all functions that relate to user account manipulation, including registering a account, updating, etc.
* This is a class for developers to concentrate their functions for account changes.

#### EventHandler

* EventHandler is a controller class that represents the main function of the app, to take a picture and analyze it with allergy information user already entered.
* There is the orchestrator for camera functionality, HTTP Request and result analyze algorithm.

#### CameraHandler

* CameraHandler is a controller class that represents all functions that control camera behaviors in AllergySnap app.
* The main task of this class is to levages Android Camera API including preview pictures, taking pictures and saving pictures.

#### CameraPreview

* CameraHandler class extends CameraPreview class, CameraPreview represents how image is displayed on the screen.

#### RequestHandler

* RequestHandler is a controller class for making HTTP Requests to Google Vision.
* It contains variables like API\_TARGET\_URL and API\_KEY, and everytime there is a API call needed from user, it will route through this class and make the request, then return the HTTP Response back to the caller.

#### AllergyAnalysis

* AllergyAnalysis represents the functions that digest and process the responses from API calls with User’s profile.
* It also takes the final result for analyzing picture and returns it back to User Interface.

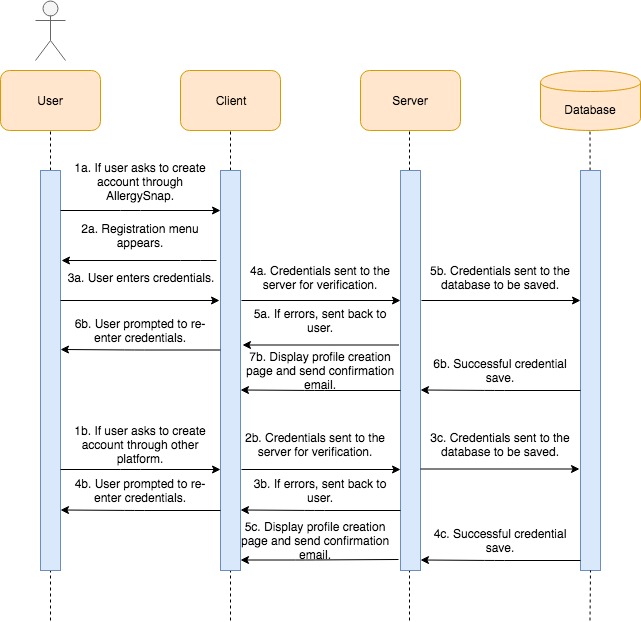
#### 

# 

# **Sequence Diagrams**

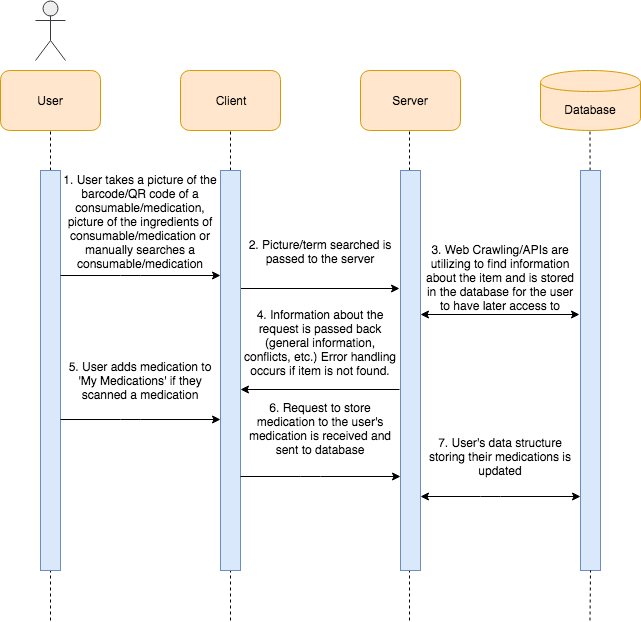
### Sequence of events for Registration

The registration sequence begins with the user opening the registration menu. The user then fills out the menu with basic account creation information or creates an account with Twitter, Google, or Facebook. Then basic credential validation will occur. Once the inputted data is successfully saved to the database, the account will be created and a confirmation email will be sent to the user and then send the user to create their profile.



#### Sequence of events when user wants information about a consumable/medication

The sequence of finding information about medication/consumables starts with the user defining what they want information about. They either take a picture of the QR/barcode of an item, a picture of the ingredients of the item or by searching the name of the item. Then, the item is passed through and information about the item is web crawled/found using APIs. The general information and conflicts is then displayed to the user. Then, the user is then prompted to add the medication to their medications if they requested information about a medication.



#### Sequence of events when the user checks their conflicts

#### The sequence to see conflicts begins with the user wanting to see their conflicts. The request then checks between the user’s medications and personal limitations for any conflicts. The conflicts are then displayed to the user.

#### 

# **UI Mockup** UI Overview

