LettuceCOOK

Design Document

Team 30

Aditya Kotalwar
Alisha Lakkad
Priyansh Shishodia
Rushank Jhanjaria

INDEX

1. Purpose

People with very fast-paced life, neither have the time to manage their groceries nor the time to think for recipes they can make from the available ingredients. Even if someone comes up with a recipe, they might not have the ingredients to make it. LettuceCOOK provides a centralized platform to help college students and bachelors living independently, as it shows them what they can make with the groceries in their house and can help them obtain missing ingredients easily and efficiently.

This app will suggest recipes to you and lets you save the recipe you want to make, based on the ingredients you choose from your stock. If you're ever missing an ingredient to make a particular recipe, it will help you gather the remainder of the ingredients by either adding it to the common grocery list between the members of the household or by requesting ingredients from nearby friend households. It will also help you keep track of your groceries by displaying a common stock and maintaining a real-time, editable grocery list among the members of the household.

1.1 Functional Requirements

- User Account
  As a user,
  - I would like to register for a LettuceCOOK account
  - I would like to be able to login into my LettuceCOOK account.
  - I would like to have a "forgot password" button where I can reset my password
  - I would like to be able to log out.
  - I would like to ability to leave households

- Households
  As a user,
  - I would like to create a new household.
  - I would like to invite other users to join my household.
  - I would like to be added to other households if they send an invite

- Households
  As a member of a household,
  - I would like to see other nearby households as suggested friends so that I can later connect with them.

    - I would like to send connection requests to other households
    - I would like to accept connection requests from other households
    - I would like to invite other users to my house for a meal together.
    - I would like to request ingredients from households in my friend's list.

- Grocery List
  As a member of a household,
    - I would like to be able to update the grocery list to keep everyone in the household up to date with the latest stock in the house
    - I would like the application to add the ticked off items from the grocery list to go into the current stock so that it avoids the hassle of manually adding the items.
    - I would like to delete items from the grocery list so that I can undo any items which were added accidentally.
    - I would like to see the current stock of groceries so that I am aware of the groceries available.
    - I would like to send other households friend requests so that I can connect and talk with them.
    - I would like to request ingredients from households in my friend's list.
    - I would like the application to prompt me to update the current stock of the ingredients, when I select to make a recipe so that the stock is not outdated.

- Notifications
  As a member of a household,
    - I would like to receive notifications when other household members update to the grocery list so that I am aware of recent activities in the household.
    - The notification I will receive will contain information about what change has been made to the stock and who has made the change to it.
    - I would like to receive notifications when a friend household asks me for ingredients or invites me over.
    - I would like the application to notify me to add an item to the grocery list if there is none of it left in the stock.
    - I would like to be able to not receive notifications when updates have been made to the stock.

- Recipes
  As a member of a household,
    - I would like to search for recipes based on the items I have chosen from my stock.

- I would like to know the nutritional breakdown of the recipe so that I can maintain a healthy lifestyle.
- I would like to know if there are any missing ingredients for the recipes suggested to me.
- I would like the option to add the missing ingredient to my grocery list so that I have that ingredient the next time I try to make the recipe.
- I would like to save any of the suggested recipes so that I can refer to them in the future without having to go through the entire process.
- I would like to see all the recipes I have saved so that I can make them again.
- I would like to filter recipe search results based on its nutritional value and benefits.

1.2. Non-functional Requirements

- Server
  As a developer,
  - I would like the grocery list and stock to be updated for everyone in real time.
  - I would like the server to store data for users and households.
  - I would like the server to communicate with the database efficiently in order to display the common grocery list and stock correctly on everyone's device.

- Performance
  As a developer,
  - I would like LettuceCOOK to be able to run smoothly without crashing.
  - I would like the app to support multiple users from households and be able to handle multiple concurrent requests.
  - I would like the app to suggest a recipe without any time lag after the user chooses their ingredients.
  - I would like the app to handle errors efficiently from the server side.
  - I would like the app's response time to be around 500 to 600 ms given a good internet connection.

- Appearance
  As a developer,
  - I would like LettuceCOOK to be user-friendly and allow its users to navigate around the app easily
  - I would like the app to be fit to all the major screen size ratios for Android like 16:9, 18:9 and 19:9

- Security
  As a developer,
  - I would like the app to safely store all the user information and passwords in the database.
  - I would like to allow the user to be signed into only one household at a time.
  - I would like to make sure than non authorized client requests are denied.
  - (if time allows) I would like to let the user choose whether to give access to their location, so that we can suggest nearby household to them

- Usability
  As a developer,
  - I would like each user to have a unique username and will give an error message if the selected username while signing up is already chosen.

2. Design Outline

2.1 High Level Overview

This android app will allow users to keep track of their stock in their house and suggest them recipes based on the items they have in their house. We will access all the recipes by making calls to the Spoonacular API. Our implementation will include a client server model. Our server will store and access all the user and group data from the database. The server will also allow multiple users to use the app at the same time using a web API

1. Android Client
    a. Android will be the front end for the app. It will be the interface for our app on all the different phones that use it
    b. Android client will receive and send data to our server.
    c. Information that is retrieved by the android client will be then parsed and formatted in such a way that it looks appealing on a small smartphone screen.
    d. Information retrieved will show all the saved information in the app such as Saved Recipes, Current Stock, Grocery List and your Friend List.

2. Server
    a. Server will handle all the information which goes between the Android Client and the database.
    b. Any information that the client wants to save will go through the server and into the database where it gets stored. Server sends this information in the form of queries to the database.
    c. Any information that the client wants to receive or change from the database, the server sends queries to the database to retrieve that specific information which it later sends to the client.

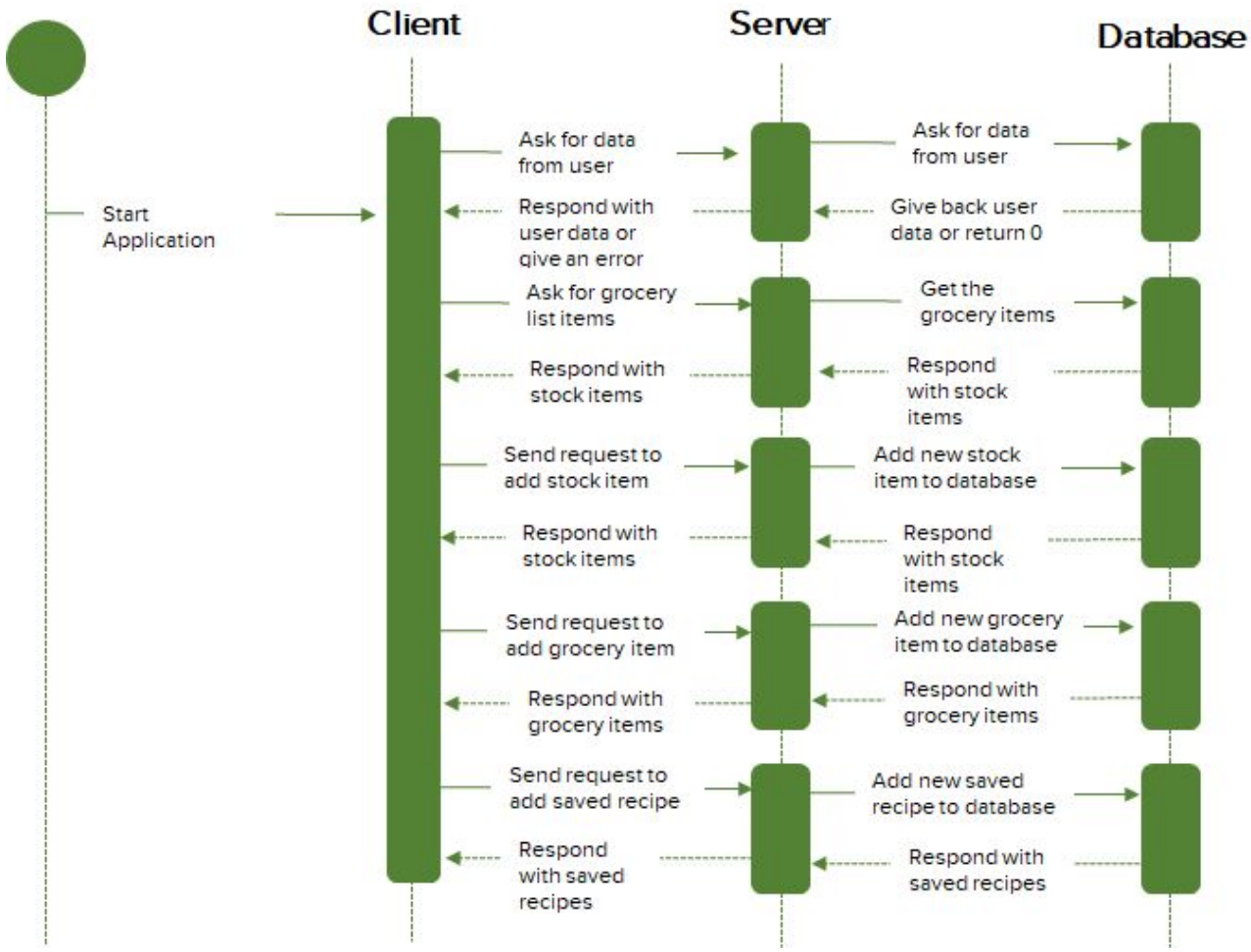
3. Database
    a. A non-relational database will be used to store all the information about the user and all the things they wish to store. The user information that the database will store is each user's username, password and the household they are part of. Information that

the database stores for each household are its different household members, its grocery list, its saved recipes and its stock.

b. Database sends JSON files with required information to server when client asks for specific information to be retrieved.

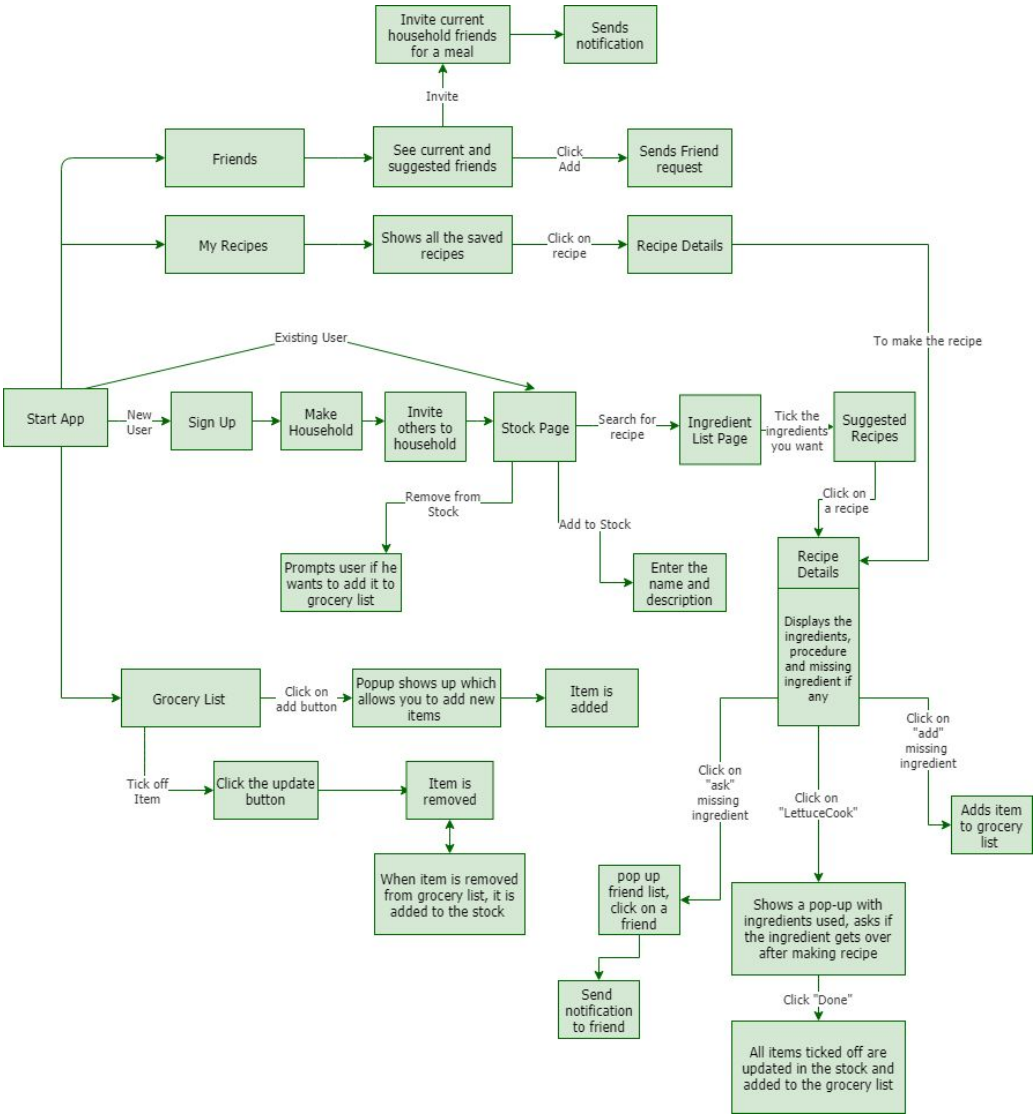
2.2 Sequence of Events Overview

All the requests made by the client will be sent to the server, which will interact with the database to retrieve and update data. This data will then be displayed to the client. When a new user is created, a request is made to the server, where ut error checks and add the new user to the database. When a member of the household adds to the grocery list, a new request is made to server, to add the item to the grocery list. The database then responds with the new updated grocery list and prints it for all the members of the household.



2.3. Activity/State Diagram

The following diagram shows the flow of control of our application. We have designed this application in a way that is very user friendly and easy to navigate around. The flow chart moves from left to right and begins with the box which reads “Start App”. When the user starts the app, there are two ways ways the user can move forward. If it is their first time using this app, then, they will have to set up a username, password and household. If they are an existing user, then they will be taken to the household stock page right away. The stock page is one of the four main pages that this app will have. The other main pages of the app are the friends page, my recipes page and the grocery list page.



3. Design issues

3.1. Functional issues

1. What fields should we accept from the user while signing up?

   Option 1: First name, last name, email, username, password
   Option 2: Email, username, password
   Option 3: First name, last name, username, password

   Our choice: Option 2

   Reason: We chose Option 2 because we do not need the first and last name of each user as each user will be referenced by their households to their friends. Hence, the first and last name will be unnecessary information stored in our database as each user within the household will be referenced by their username. We need the email address of the user in case they forget their password during login. We will send them an email asking them to reset the password.

2. When should the items from the stock be sent to grocery and marked as “over”?

   Option 1: Immediately after the user presses the delete button
   Option 2: The delete button temporarily deletes it from the list and permanent deletion
   is made on pressing the update button

   Our choice: Option 2

   Reason: For the deletion, we chose option 2 because we want the user to be able to undo any changes made to the stock. If the uses presses the delete button accidentally and wishes to bring an item back to the stock then they won’t be able to if we just have one delete button that makes changes to the stock.

3. How should the user select the item to add to their grocery list or their stock?

   Option 1: We provide them with a list of grocery items from a database and they can search through and choose the ingredient they want to add
   Option 2: They shouldn’t be able to add to the grocery list once their household is set up as the grocery list updates the stock and vice versa.

Option 3: They should be able to input what they want to add manually.
Our choice: Option 1

Reason: We chose Option 1 as it is the best suited way to prevent any sort of spelling or grammatical errors for the ingredients in the stock or the grocery list. By providing them with the database of grocery items, we can make sure that the item they choose will be an ingredient that can be passed in our API call.

4. How should we suggest nearby household locations to other households?

   Option 1: We track the locations of all users who have the app in a given radius.
   Option 2: Users enter their address when they sign up
   Option 3: We ask the user to pin the location of their home when they first sign up and then show them all the households nearby in a given radius.

   Our choice: Option 3

   Reason: Asking the user to pin their location rather than manually adding their address is more user friendly and reduces the work to be done by the user. We should not track the location of the users of the app because we have to suggest households and not users to other households. Also, since the households are stationary, and users may not always be in their households, we will need the location of the household and not the user.

5. How will the recipes be saved in the app?

   Option 1: The recipes will be saved for the user and not the household, so a recipe saved by the user will not be visible to other members from the household
   Option 2: All recipes saved by a user will be visible to all the other members of the household under the tab ‘My Recipes’. Everyone can also see who has saved which recipe.

   Our choice: Option 2

   Reason: All the recipes saved by a user should be shown to all the members in the household because it also shows the other members the recipes that can be made by the groceries in the household. In case one member wants to make something another member has made, they can just access the common “My Recipes” tab rather than sharing the recipe within the same household. Also, the username of user who has

saved the recipe should be displayed near the saved recipe to keep tabs on who has saved which recipe so that every user can easily access the recipe they have saved.

6. Should we display the quantity of all the grocery items in the stock?

   Option 1: Yes
   Option 2: No

   Our choice: Option 2

   Reason: Everytime the user buys groceries ,it will be very inconvenient for the user to manually input the quantity of every item that was bought. Instead, the user can mark the item as “over” everytime the item gets over in the house. Also, the quantity of groceries that cannot be measured in numbers, such as milk, cannot be tracked. This is why it is most efficient to not display the quantity of every item.

3.2 Non-functional issues

1. What platform should we use?

   Option 1: Android
   Option 2: IOS

   Our Choice: Option 1

   Reason: Since not everyone in our group had OSX machines and most of our app development skills were limited to Android, we chose to develop our app for Android. Further, Android Studio can be downloaded on any type of machine, so everyone from our group can contribute to the app without having to go to lab machines. Therefore, the obvious choice for better app quality and ease was Android Studio.

2. What Android supported language should we use?

   Option 1: Java
   Option 2: Kotlin
   Option 3: C++

   Our Choice: Option 1

Reason: All of our group members have at least 2 years of experience in Java, but most of us are not familiar with Kotlin. Java is also more compact than Kotlin, which comes with an extra runtime size of around 800Kb. Furthermore, if all of us took the time to learn another language before developing the app, we would waste time that we could use to work on the app. In the interest of efficiency and quality, our choice was Java.

3. What server should we choose for backend services?

   Option 1: Firebase
   Option 2: AWS
   Option 3: Azure

   Our Choice: Option 1

   Reason: Firebase is a very versatile backend service which provides a system to store data in the database. Since it is owned by Google, it makes integration with Android very easy. Moreover, firebase comes with RealTime Notifications, and using the feature is easier than implementing them in either Azure or AWS. Real time updates in the app are integral features, and Firebase is very synchronous with this. Updating the grocery list and stock in real time will be easier than if we used AWS or Azure. Furthermore, Firebase can synch data across different clients effortlessly, which is the central idea of our app, letting everyone in the same household know what groceries need to be bought. These are some features that we needed to implement in the app anyways, so using Firebase helps us in the process. Adding all these features using AWS or Azure would not have been feasible in terms of the time taken to learn how to implement them and then actually implementing them. Among the various advantages that Firebase already has for our app, another one is that the free pricing plan for Firebase can support our app in its initial stages, so we will not have to buy a plan until later into the development stages of the app.

4. How should the users login?

   Option 1: Google
   Option 2: Facebook
   Option 3: Enter their details manually

   Our Choice: Option 3

Reason: Instead of connecting the app to other social media platforms, we decided that the user should input their information and create an account on the app. We came to this conclusion because we wanted our app to be a standalone platform where the users do not have to actually input their real name, date of birth or any other private information. The only piece of information that they have to enter is their email I.D. for password resets or household invites. Our main thought process for this decision was that we really did not have a need for all that information. In addition, manually adding information directly to the application makes it easier to input it into the database and access it. For these reasons, we thought that option 3 was best suited for our app.

5. As our customer base grows, how do we plan to limit the calls to the API?

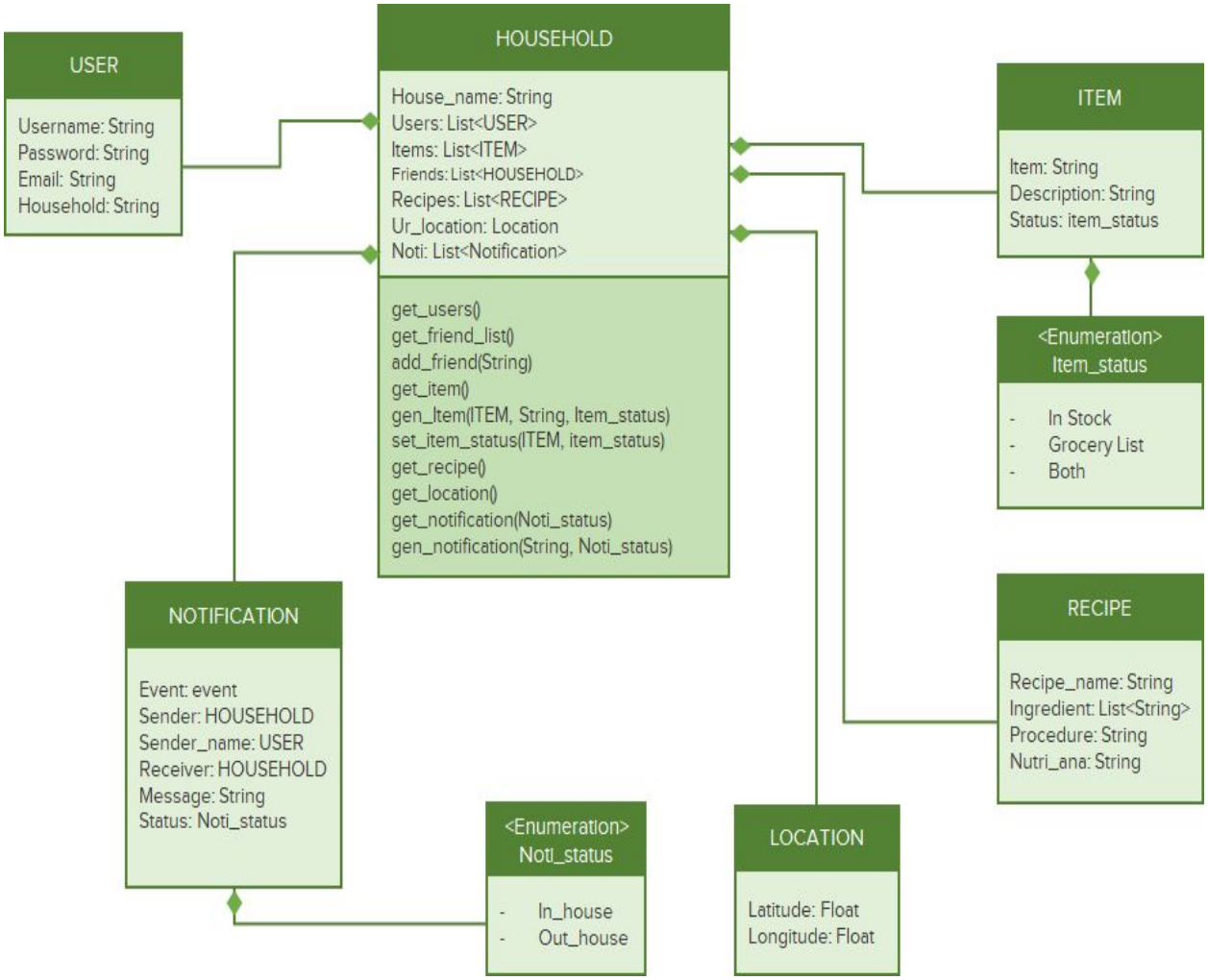
   Option 1: Automatically give a list of recipes from the current stock of groceries
   Option 2: Suggest recipes from selected ingredients only when user searches for the same
   Option 3: Give a list of recipes whenever a new ingredient is added or removed from the stock

   Our Choice: Option 2

   Reason: As our app grows, more and more people will start using it, which directly implies that the number of requests from the API will increase. Our best choice to reduce the number of API calls while still keeping the basic functionality of the app is only calling to the API when someone searches for recipes or ingredients for recipes. This way, we won't be sending too many requests and exceeding our quota. If we had chosen any of the other options, then the app would automatically send requests to the API, increasing the number of calls to it, which would inadvertently result in us exceeding our quota for the API.

4. Design Details

4.1. Data Class Level-Design



4.2. Description of Data classes and Interaction

These classes are designed around the various objects and variables needed in our application. Each class represents one aspect of LettuceCOOK and has its own attributes for each feature.

- User
    - Every new user will be a new object of this class.
    - Every user will have to have a unique username.
    - This class will store the username and password of the user.
    - The username of the user will be displayed next to every item they add or remove from the grocery list, or everytime they save a recipe.
    - The user's email address will be needed in case they forget their password and will need to reset it via email.
    - Each user will also be either invited to a household or will create a new household, so they will also have a respective household.

- Household
    - Every new household will be a new object of this class.
    - Every household name will have to be unique.
    - Every time a new user becomes a member of a household, they are added to the list of members in the household.
    - There is a list of items in the grocery list or the stock for that household.
    - Each household has a list of recipes saved by its users, so that they can be easily accessed at any time.
    - Once the household is made, a user who is in the household can pin the location of the house in order for it to be visible to other households which are in close proximity.
    - Each household has a list of notification sent by households that will be visible in the "Friends" tab

- Item
    - This class will store the name of the item that is chosen by the user.
    - Each item stored will be from a database of a list of grocery items.
    - The description of the item is optional for the user to fill and will give any additional information about the item such as the brand of the item, etc.

    - There will also be the status of an item which indicates whether the item is in the grocery list, in the stock, or in both.

- Recipes
    - Recipes got from the API and selected by a user will be an instance of this class.
    - The recipe name will be displayed in a condensed box which on clicking will expand to show details about the recipe.
    - The procedure and nutritional analysis will also be taken from the API and stored to show the user when they click on the recipe they want to see.
    - The list of ingredients will be compared to the list of ingredients in the stock of groceries in the household and the missing ingredients will be printed so that the user can either add that ingredient to the grocery list, or ask another household for the ingredient.

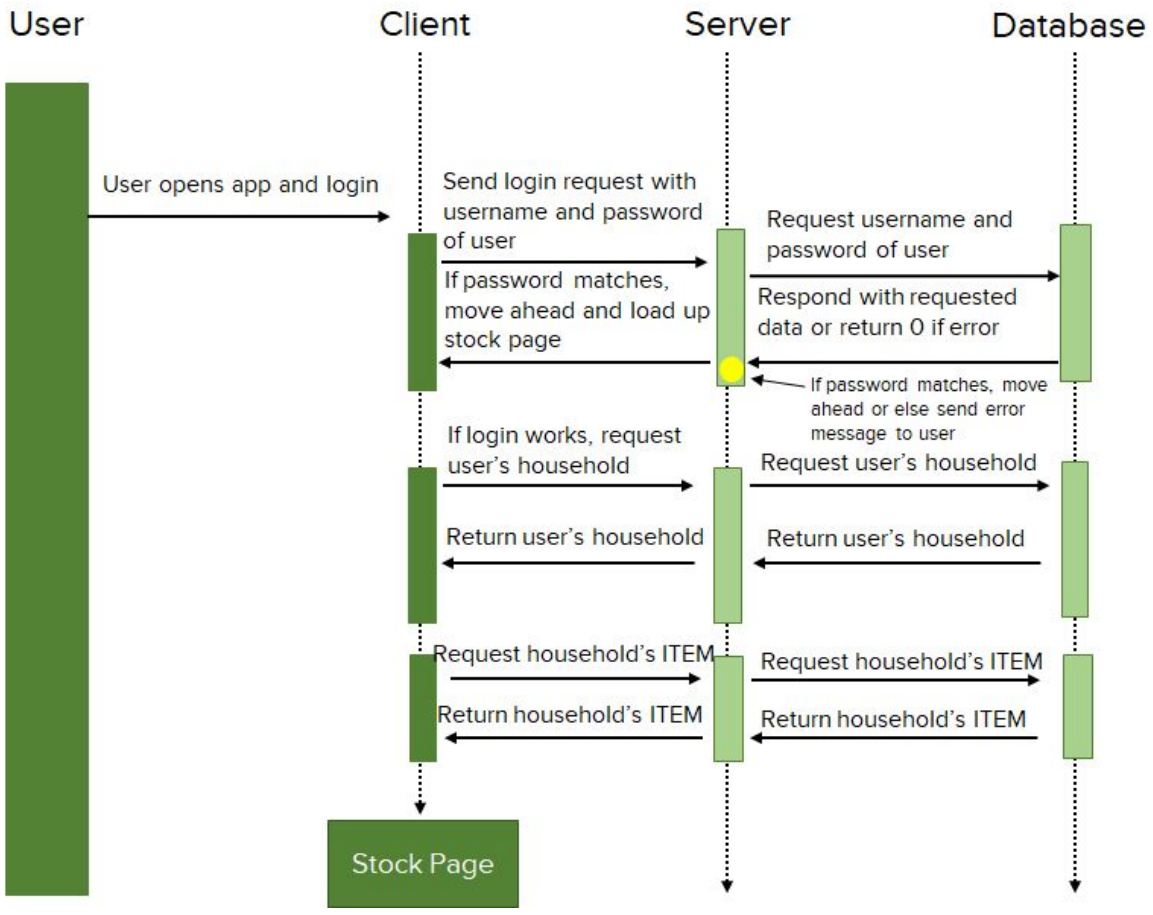
- Notifications
    - There will be two types of notifications: in-house and out-house.
    - In-house notifications will be sent to every member of a household when one member updates the stock or grocery list. Each change will be documented by showing the username of the member who made the update.
    - Out-house notifications will be sent to members of other households. If a member from one household wants to ask for ingredient from another household or invite them over for a meal, a notification is sent to all the members from that household.
    - The sender of in-house notifications will be a member.
    - The sender of out-house notifications will be a household.
    - The message sent will be a pre-written message, to reduce the inconvenience the member will face in typing a message that will be the same every time a user asks for an ingredient.

- Location
    - Everytime a member of a household pins the location of a household, a new object of this class is created.
    - The latitude and longitude will be used to find houses using this app within a given radius, so that members can see and add new households as friends.

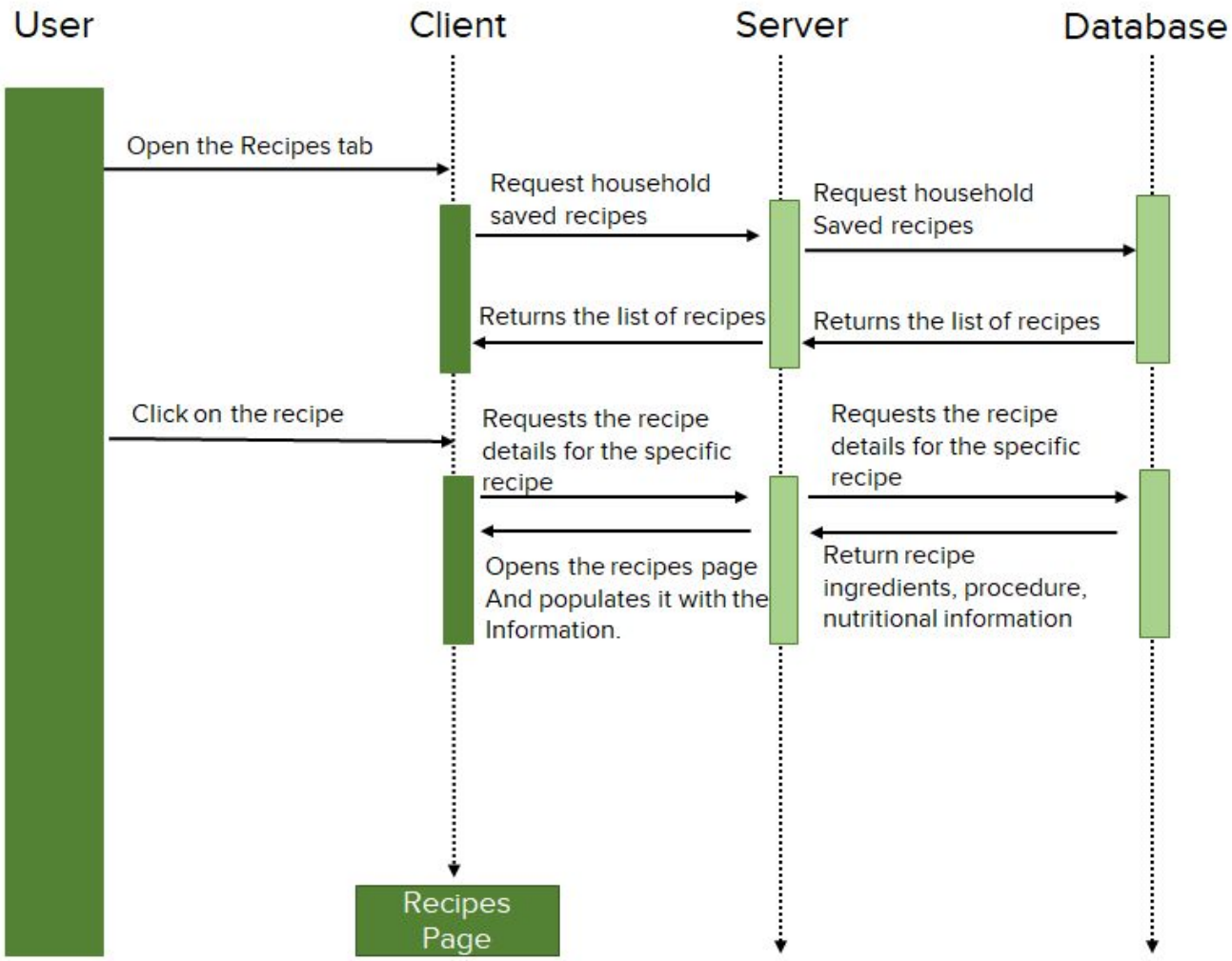
4.3. Sequence of Events

The following diagrams depict the sequence of the major events in this application, including user login, adding items to grocery list, showing saved recipes, and updating the stock. The diagrams depict the interactions in the client server model. Whenever a user performs an action on the frontend user interface, the client will send a request to the server and the server will send a query to the database to acquire the data needed or to update the data. The client will process the data and update the UI display accordingly.

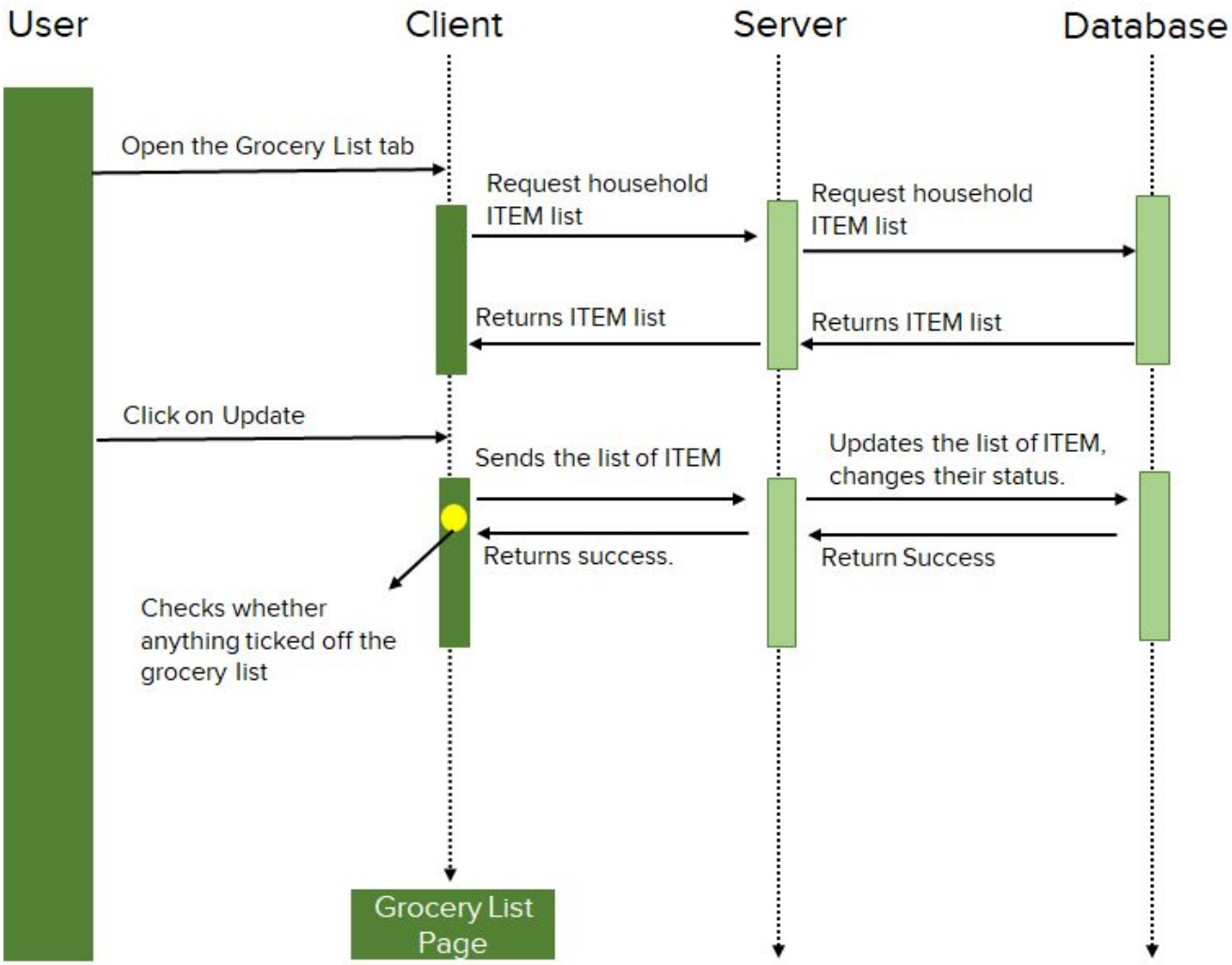
- Sequence of Events when user opens the app and the Stock page is opened



- Sequence of Events when Recipes Page is opened

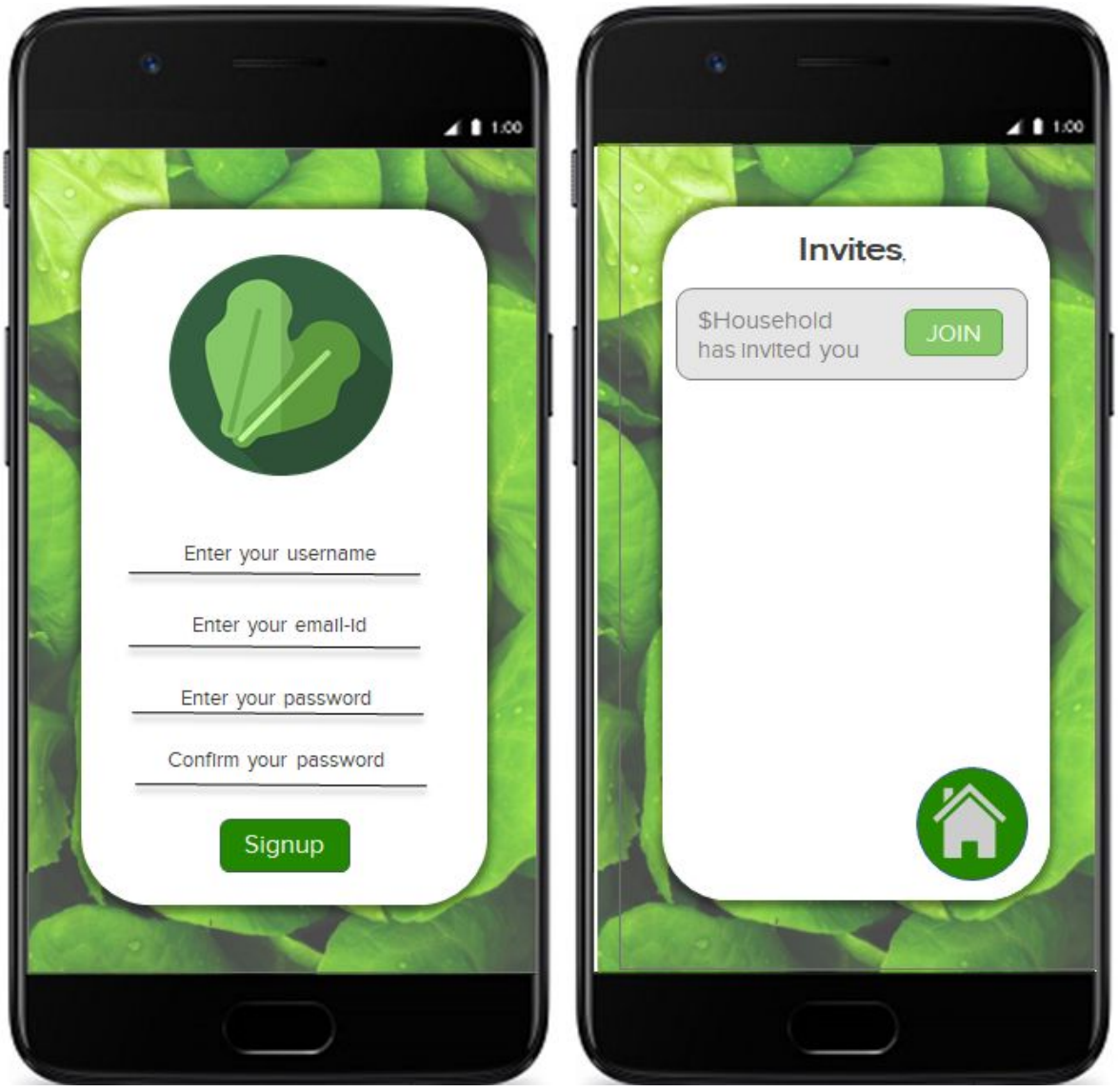


- Sequence of Events when Grocery List Page is opened

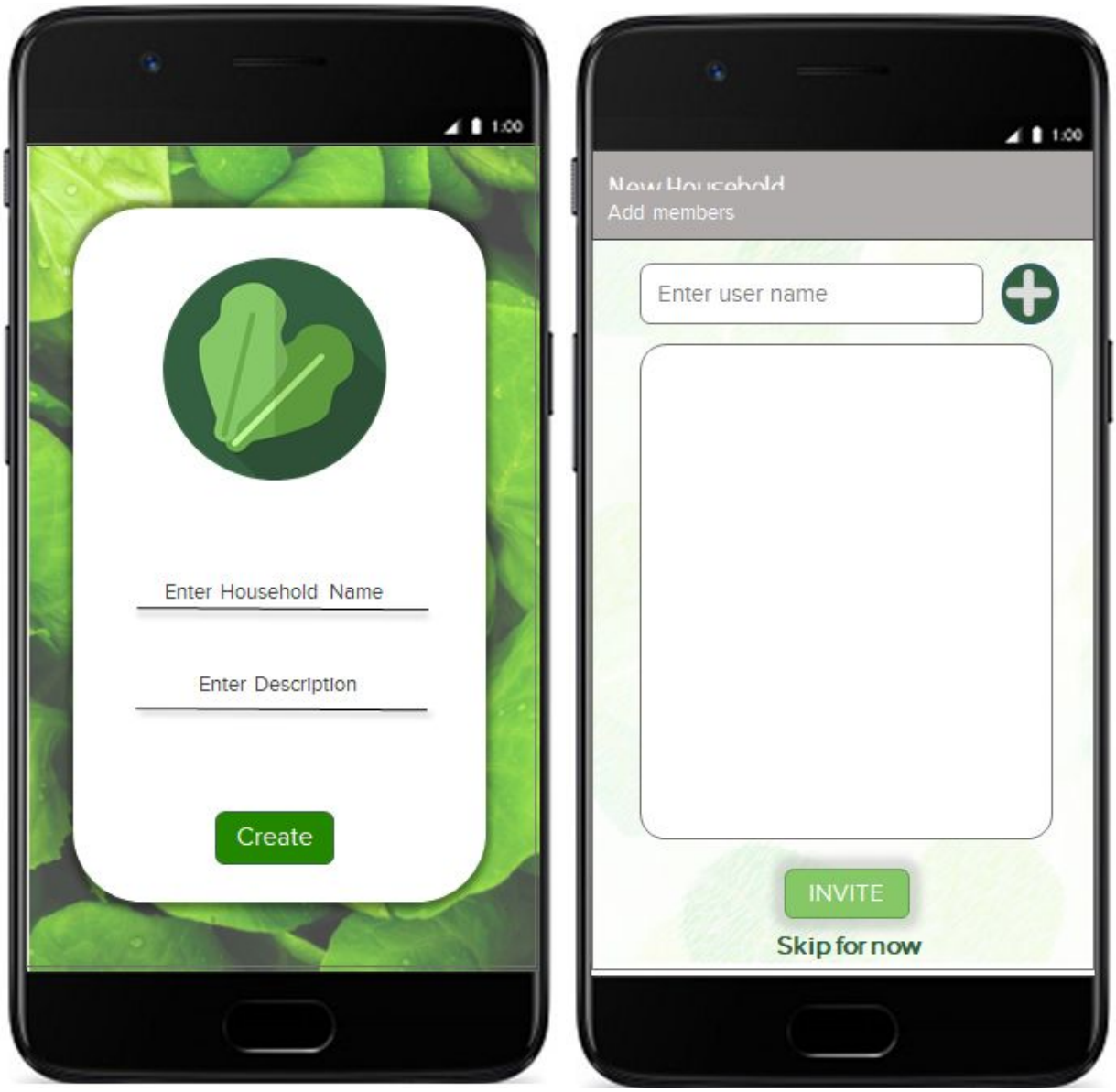


4.4. UI Mockup

- Sign up page and the Invites Page

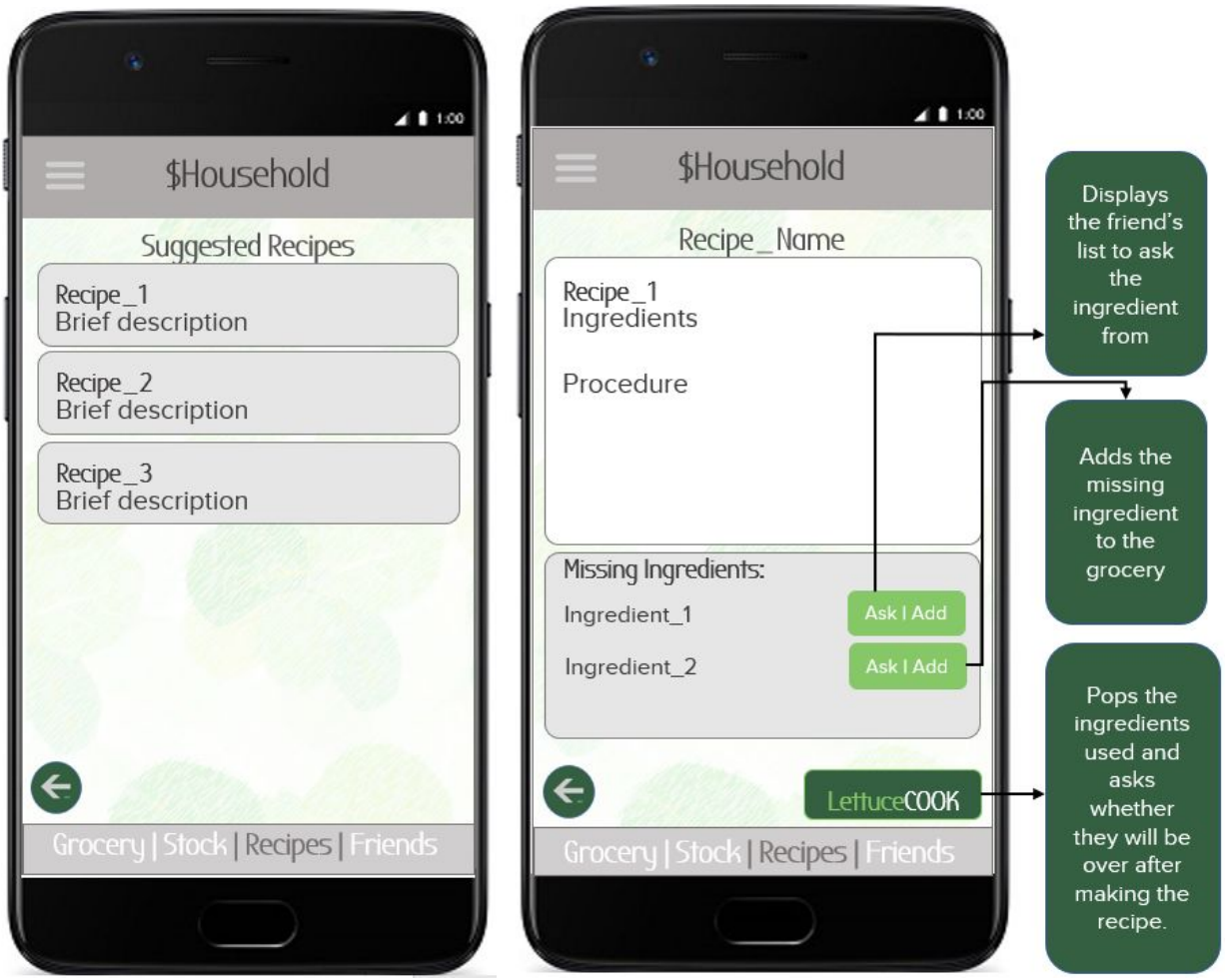


- Create Household and invite members page

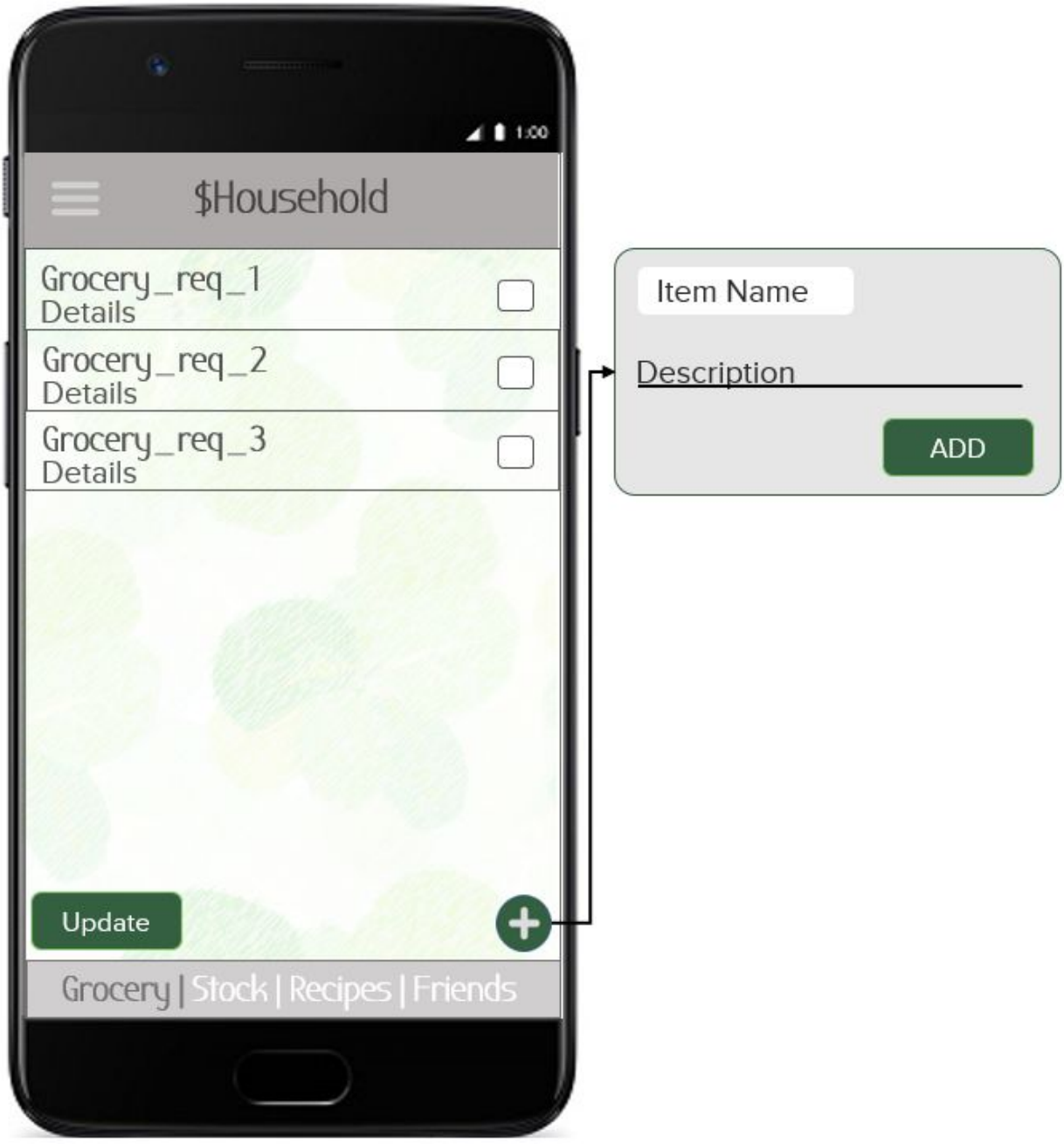


- If logged in with a household opens the **Stock** page, when you search for one pops the ingredient list

- Displays the suggested recipes, clicking on one displays the recipe details with a list of missing ingredients.



- The Grocery tab, displays and maintains the grocery list, the add button adds new item to the list



- Friends Tab


1:00
$Household
Notifications
Message
Inhouse/Outhouse
Suggested Friend Households
Suggested_1
ADD
Suggested_2
ADD
Friend Households
Friend_1
INVITE
Friend_2
INVITE
Grocery | Stock | Recipes | Friends