Victor Arango-Quiroga
Hassaan Khan
Bao Vo
Haoxian Lin
12/24/2016

Design Document - OpenCrowd

1. Introduction

LettuceBuy is a web based application which provides convenience for customers as well as an opportunity for drivers to make extra money. Due to the limited amount of time that we have for this project, we will be limiting our functionality which will be explained further into this document. The purpose of our project is to allow customers to order online and maintain a text-based communication with the driver for more instructions/items. Both customers and drivers will have their profiles stored onto a server database which is also where the ratings for each driver resides.

LettuceBuy aims to improve time management of its clients by providing an easy-to-use grocery service. OpenCrowd wants to allow its users to order groceries online from the comforts of their homes or work in order to preserve time for more vital activities. Users will be able to manage their time more efficiently and still be able to get their desired item from the store with the LettuceBuy service.

2. Design Considerations

Describe the issues that need to be addressed or resolved prior to or while completing the design, as well as issues that may influence the design process.

In this project, the biggest issue that the whole team get into is communication and time management. Since not everyone has the same strength and time that work out with every other members of the team, communication issue is the priority problem which need to be solved. A technical issue that our team get into is to choosing the environment base for clients and drivers. Our team members don't have experience with mobile application platform. We have never built any app on Android or IOS platform before and our original idea was to make it a mobile app which would provide the most convenience.

We are assuming that the clients have an internet connection with decent bandwidth to ensure a decent connection at all times with the website. A big issue that needs attention is to ensure that the profile database is safe in case of a systems crash which Haoxian Lin will work on.

2.1 Assumptions

All payments and transactions are assumed to have been paid online prior to delivery since our project will most likely not deal with any type of banking. We are assuming that drivers can only select one list currently for the first prototype and clients can only post one active list. Drivers are assumed to be using a smartphone that can access the website using their mobile device. It is assumed that if either the client or driver presses the “Finish” button from their respective devices then the service was fulfilled with full satisfaction. We will implement a better functionality for this particular option in order to ensure security but for prototype one, we are focusing on our software’s core functionalities.

There are couple things that heavily impact to the design of our software. These are the internet connection and how the the system that is running the application and the system itself is been powered. We are assuming that the internet connection either for users or drivers will be stable and we will not run into any connections problems while doing any operation.

2.2 Constraints

We have a limitation of how long we can use the google cloud servers. Each student has $50 dollars available to use the servers which will be more than enough for our software purposes, but still we need to have this limitation into consideration.

We have six weeks to finish the final prototype. Time is a big constraint since multiple prototypes should be done prior the final one due to the process plan that we chose.

Drivers should use a smartphone in order to keep track of lists . We are not creating an app for this software, thus drivers need to be able to get into the LettuceBuy website from their cellphone to keep track of current lists. Reason for which drivers need to use a smartphone in order to be able to open the website from their cellphones.

There are some limitations on the string’s length from using Sqlite3 to interact with our servers, but it is not a problem for our software purposes.

2.3 System Environment

Application is running in web browsers environment. It requires users to have a comparable devices such that smartphone, tablet, laptop and computer which can open web browser apps.

Software (web browser) will interact with the hardware (google cloud) to do all the main function such as update information, send-receive orders by upload and download data from the server.

3. Architectural Design

Using a Ubuntu Linux based operating system, we provides our service through the internet. Graphical interface will be web-based, crafted with html and PHP. We will be using html for basic styling and PHP for database access and modification. The services we provided are simple and does not require large amount of computation, however, it requires fast and accurate update of information which has to be accessible on a mobile platform. This makes a web GUI a suitable choice. Software based GUI could be implemented for speed after the base is built. PHP integration to html will be responsible for most if not all functionalities. SQLite3 database will hold data such as account information, list content and status, etc.

Database: Sqlite3

Currently, a sqlite3 database resides on our Google Cloud server where three tables exist, one for clients, another for drivers and lastly list housing all of the lists submitted by clients. Name, address, password, and phone number are all columns of the clients and drivers table in the database.

3.1 Overview

Once the website loads, if it is a returning user, then he or she will be asked to provide two credentials, username and password. New users have the option to create new profiles which will require name, address, phone number, and a checkbox whether the user wants to be a driver or client. All of our interfaces will be web GUI using HTML and PHP.

After login occurs, if the user was a client then he will be taken to a web page designed specifically for clients. The client will be able to look at their profile credentials and their current active shopping list (if applicable). This is where the clients will be able to see if their list has been chosen by a driver (status). There will be an option that allows clients to create new shopping list that will allow the addition of items, store address, and delivery address. The client has the ability to “finish” a transaction which will automatically assume that the transaction is complete.

The drivers will also have their profile credentials displayed at the top right of the web page. They will have the option to take a look at the pool of lists currently active that will be retrieved from our google cloud server. Drivers can see and then choose who they want to go shop for (based on convenience). The pool of shopping lists that drivers see will contain number of items, address of the shop, address of the customer and time that the list was posted at (to ensure driver does not

pick a dead list). When they select the list, it will be loaded to the driver's main page and he/she will now see specific items on the list, address of store, name and address of the customer. After the driver has finished shopping, then the driver will have an option to "finish" this particular transaction which will assume that the job was completed. Later prototypes will improve upon this functionality to perform more thorough checks by confirming with both parties.

An option will later be implemented that allows both clients and drivers to check their previous shopping lists, customers and more transaction details.

3.2 Rationale

The system relies heavily on communication due to the nature of the system, we require a stable server and connection. Google cloud is being used to ensure a reasonable uptime as well as a reliable connection to the web. Our system also requires frequent access and update of data so we needed a database and for convenience therefore we decided to use SQLite3 for this project. Web based interface will be more accessible on a mobile platform since it does not require installation of softwares (assuming browsers are provided by the OS). We want the interface to be extremely easy to work with which is why we building the GUI in the web page which will prove to be difficult but it is a necessity due to html and php having useful GUI developing tools.

3.3 Conceptual (or Logical) View

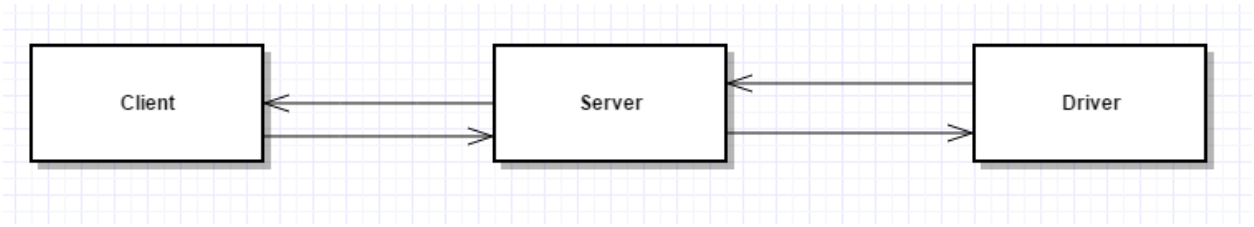


Figure 1

Figure 1 shows a very basic idea of how our software is supposed to function. A server that we will rent from Google Cloud will sit between our clients and drivers. Based on what each of them press, they will be communicating with the server to retrieve data stored in the database which we will use SQLite for. The server will be responsible of all communications in between until the list was chosen by the driver and both parties have phone numbers of one another. This server will be implemented and be programmed in PHP which will be in charge of storing data onto the database. An apache server will also sit between incoming traffic to allow easy redirection when multiple users are trying to access the same web page at the same time.

4. Low Level Design

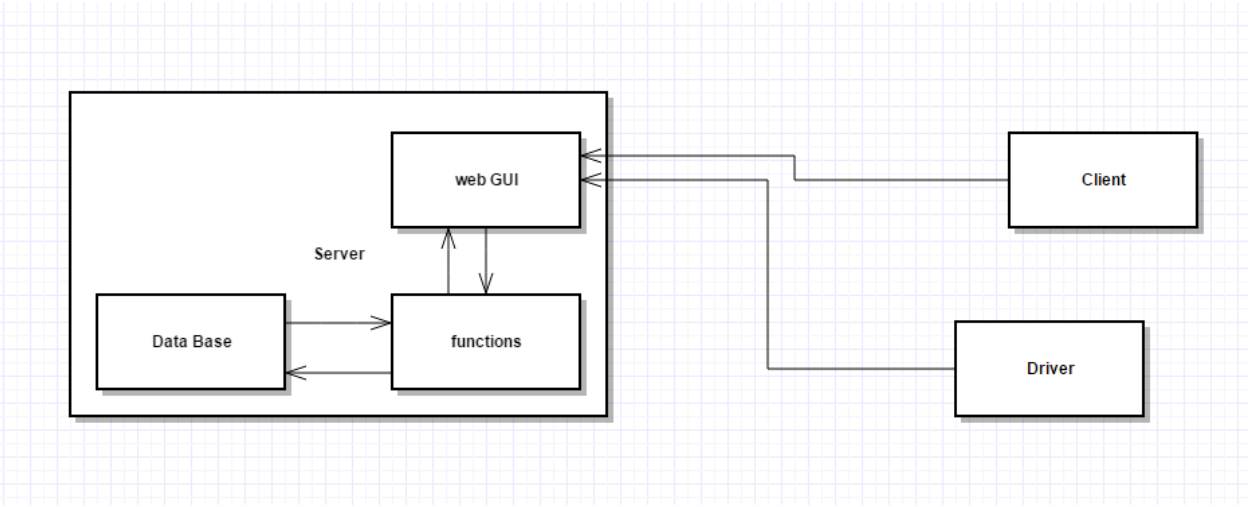


Figure 2

The figure above shows a more detailed view of how our software will be designed. The server will contain functions (listed below in 4.1) that will allow extraction of data based on the request made by either the driver or the client; both the client and driver will be communicating with the server through a simple GUI because convenience is our priority at all times. The GUI will allow access to functions which will directly interact with the database stored on the server. All functions will need to return the requested item back to the server which will then return it back to the caller. Server side functions is done in PHP while client side interface will most likely be made in Java since most of our teammates have prior Java experience. We will be using “dispatch” to enforce http rules on our site in a more organized fashion. It will allow us to match requests coming onto the server which will be based on profiles in our case.

Our interfaces will be implemented using HTML/CSS and PHP and will be on the web page itself. This will be an arduous task as all users will be interacting constantly with the interface we define therefore it will initially contain very basic functions to fit the functional requirements.

4.1 Class Diagram

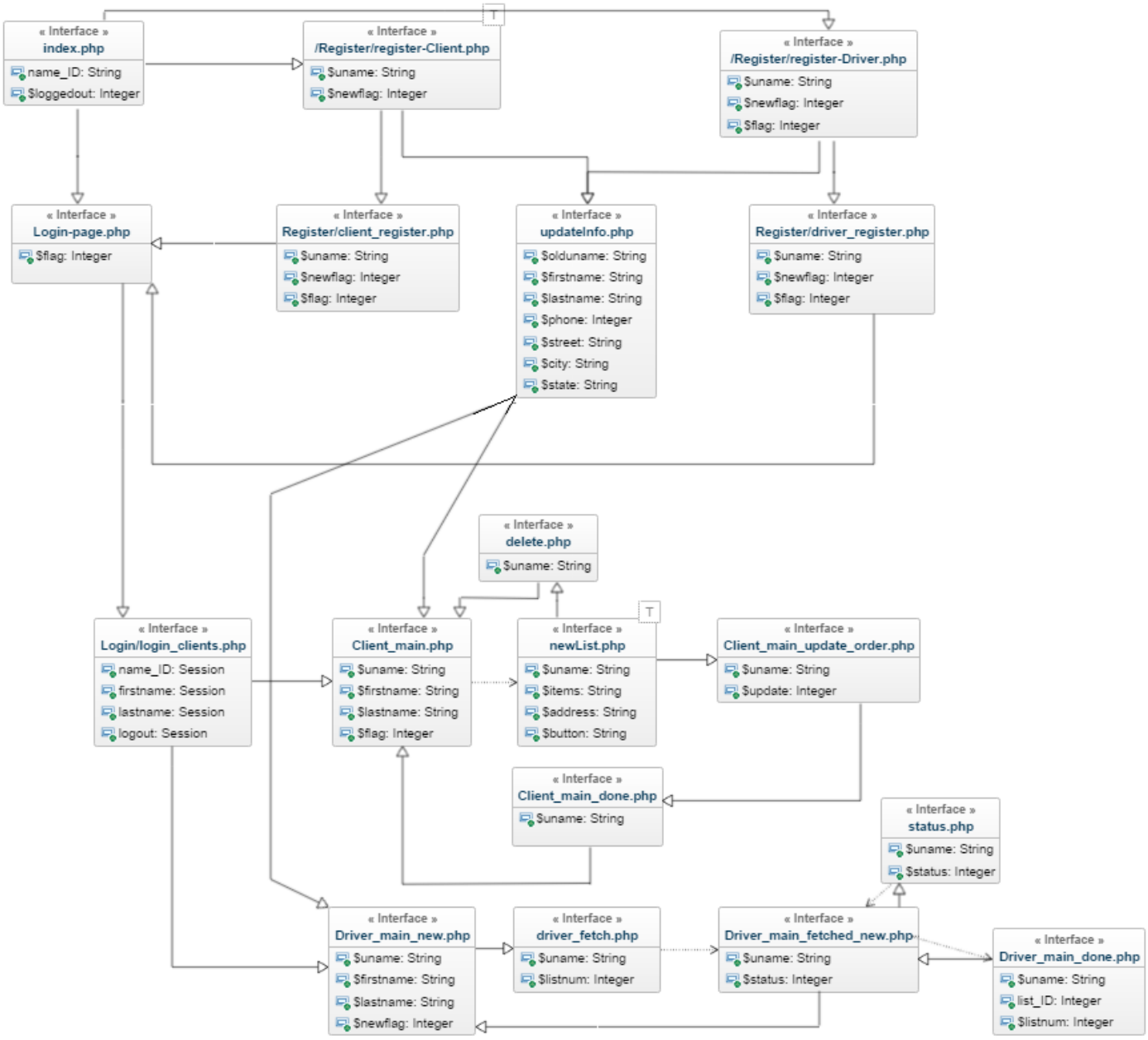


Figure 3

This diagram illustrates a small amount of functions that are to be included on the interfaces. The “New List” function will allow clients to create a new lists which will be added into the pool or the ‘list’ table in the database. The list will be created in SQLite3 or mysqli most likely. After a user logs in, the status of their current list should be automatically shown. After a delivery is completed client can mark it as complete and it will move to the history of the clients which will be implemented in later versions if given time. For drivers, the main page automatically displays unfetched lists, after deciding which one they like the most, they could use the fetch function to claim a list and the list will be removed from the pool, status of the list will be modified simultaneously.

4.2 Sequence Diagrams

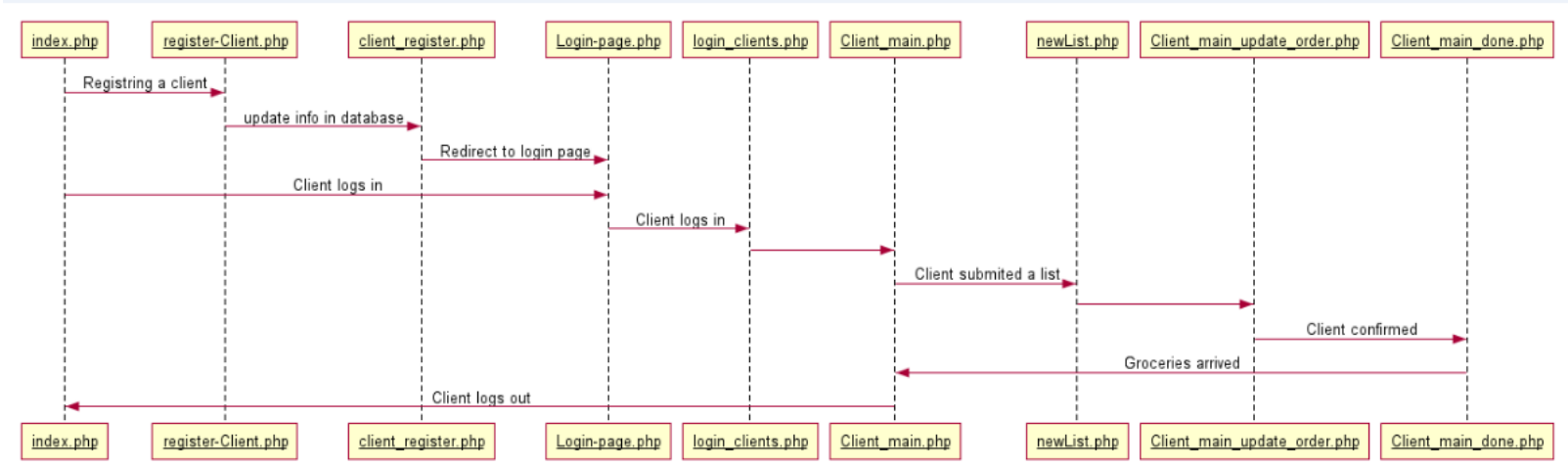


Figure4.1

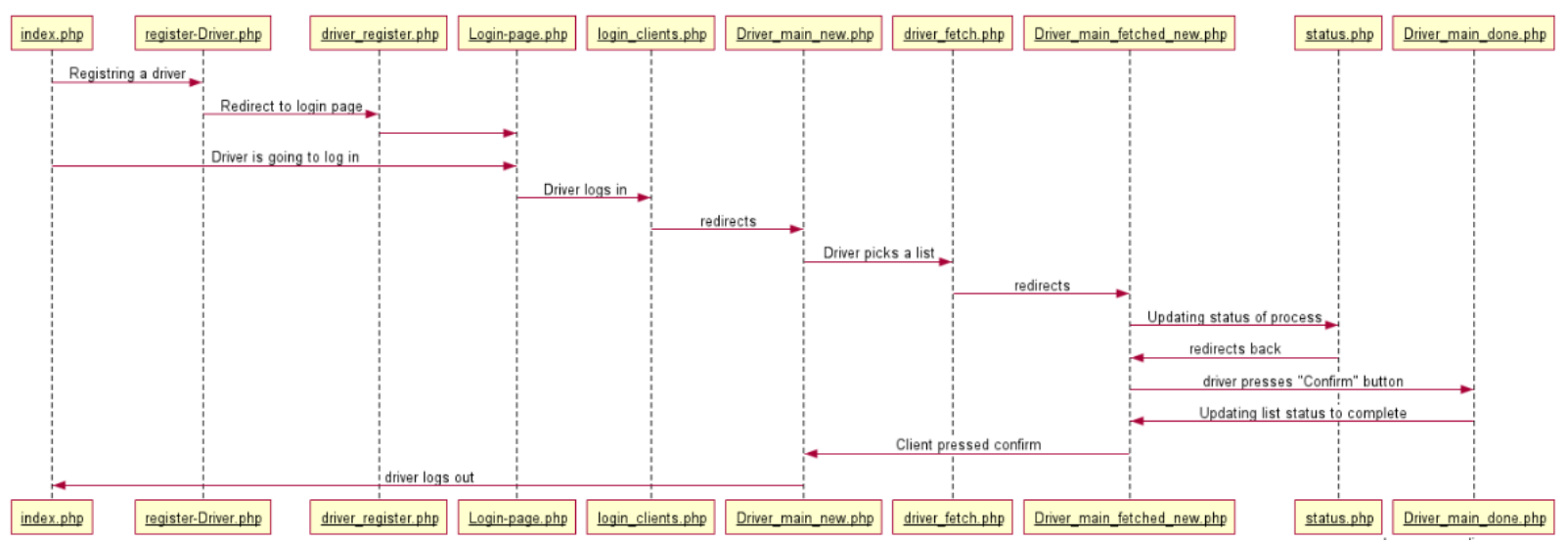


Figure 4.2

When a user access the server through the browser with the correct domain name, apache2 server will redirect the traffic from coming into port 80 to the www folder in our server. In the folder, it contains our index page as well as other need components to provide required functionalities. The files will be written with html as well as php scripts. After the commands are being process with php5, operation to the sql server will be made.

4.3 State Diagram

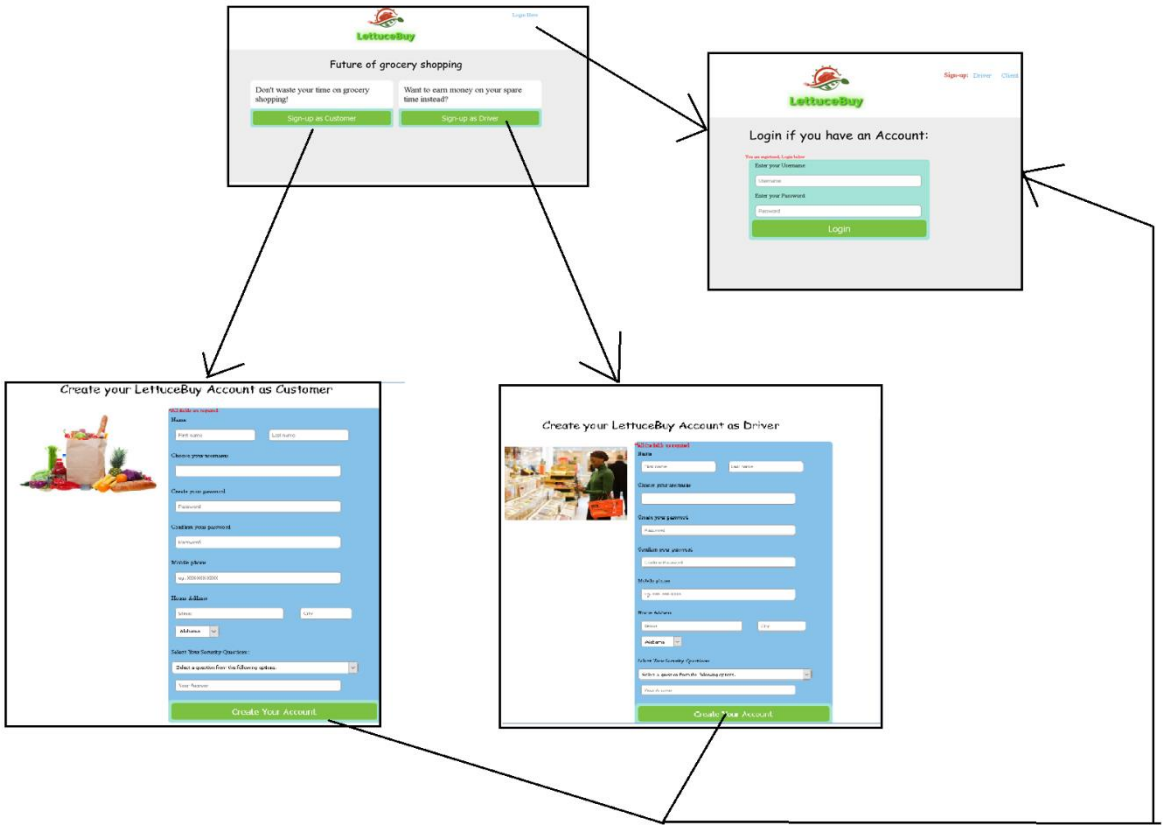


Figure 5.1

In figure 5.1 we show the state diagram for new clients that want to create an account in Lettucebuy and how clients can log into their account. Here, the user gets redirected to the corresponding register pages. After they submit valid information in every field, then an account

will be created and they will be redirected to the login page. The figure also illustrates the path that a user will follow if they went to our url. It is a state machine implementation because a user will be redirected to different pages and pages will display different messages based upon the input that is given to the pages. This figure also shows a more in-depth visual to how the whole process will occur with server and database involved. If it is a returning user, then he/she will be redirected onto the main pages' GUI where they can interact with the different functions on the website which will be shown in the next figure. If the user is new however, then he or she will be redirected onto a profile creation page where they will enter their credentials to have a profile stored onto the database which will then allow access onto the webpage.

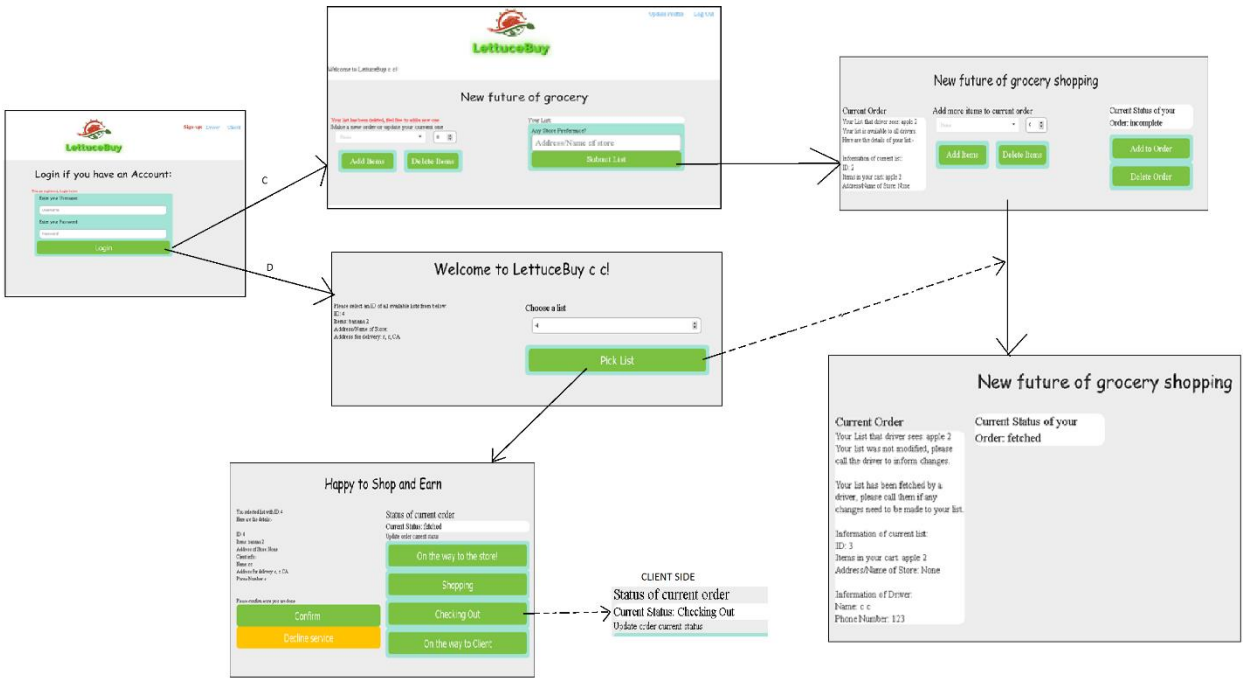


Figure 5.2

Once a driver is able to pick a list from the pool of available lists, they will be redirected to a new page where they have a chance of dropping the entire list which will then bring you back to the previous page and put client's list back into the available pool of lists. Driver is urged to use the buttons to let the client know of the status of their list and where in the process is the driver. The status of the list will change according to what the buttons say and the client will be able to see the status change.

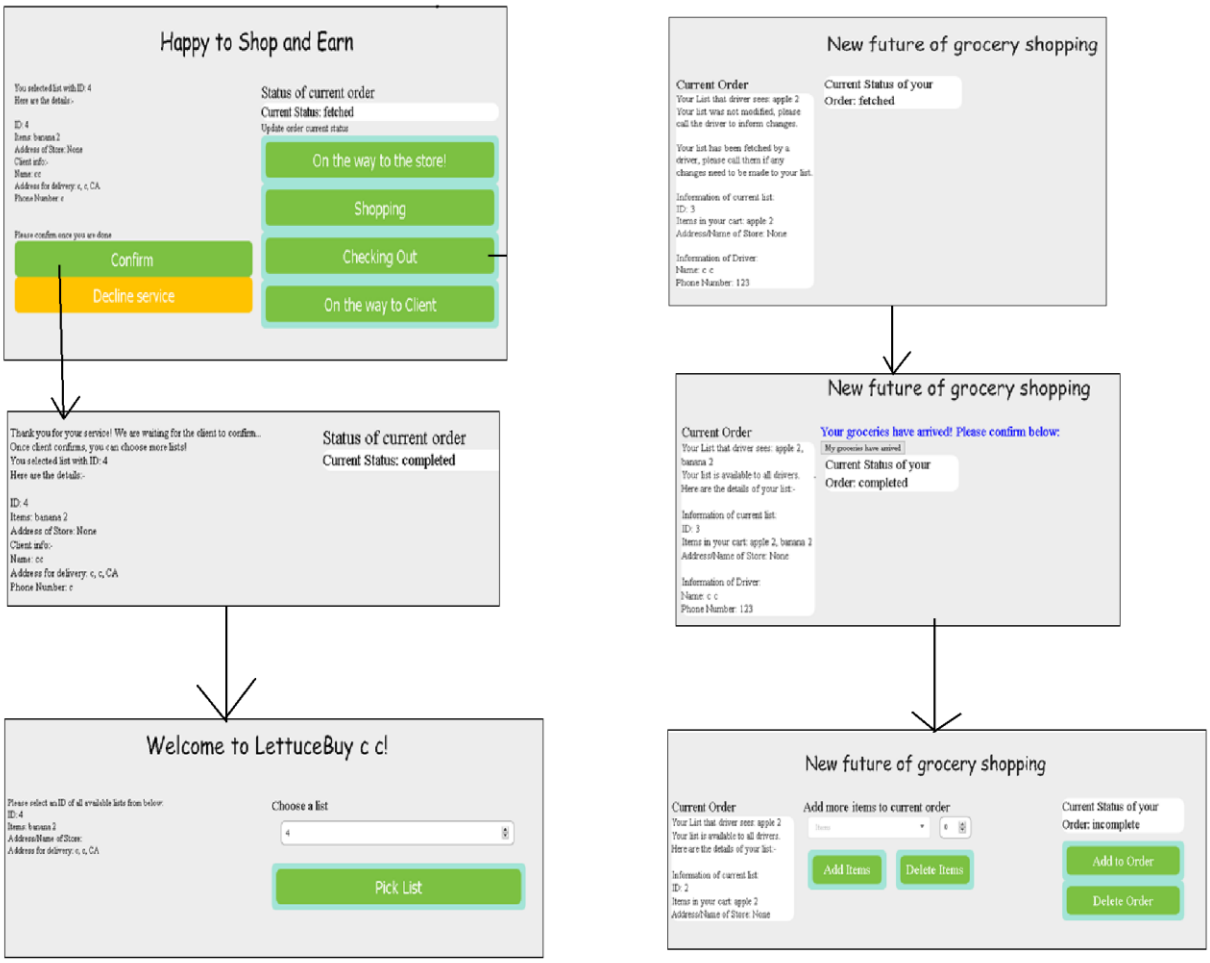
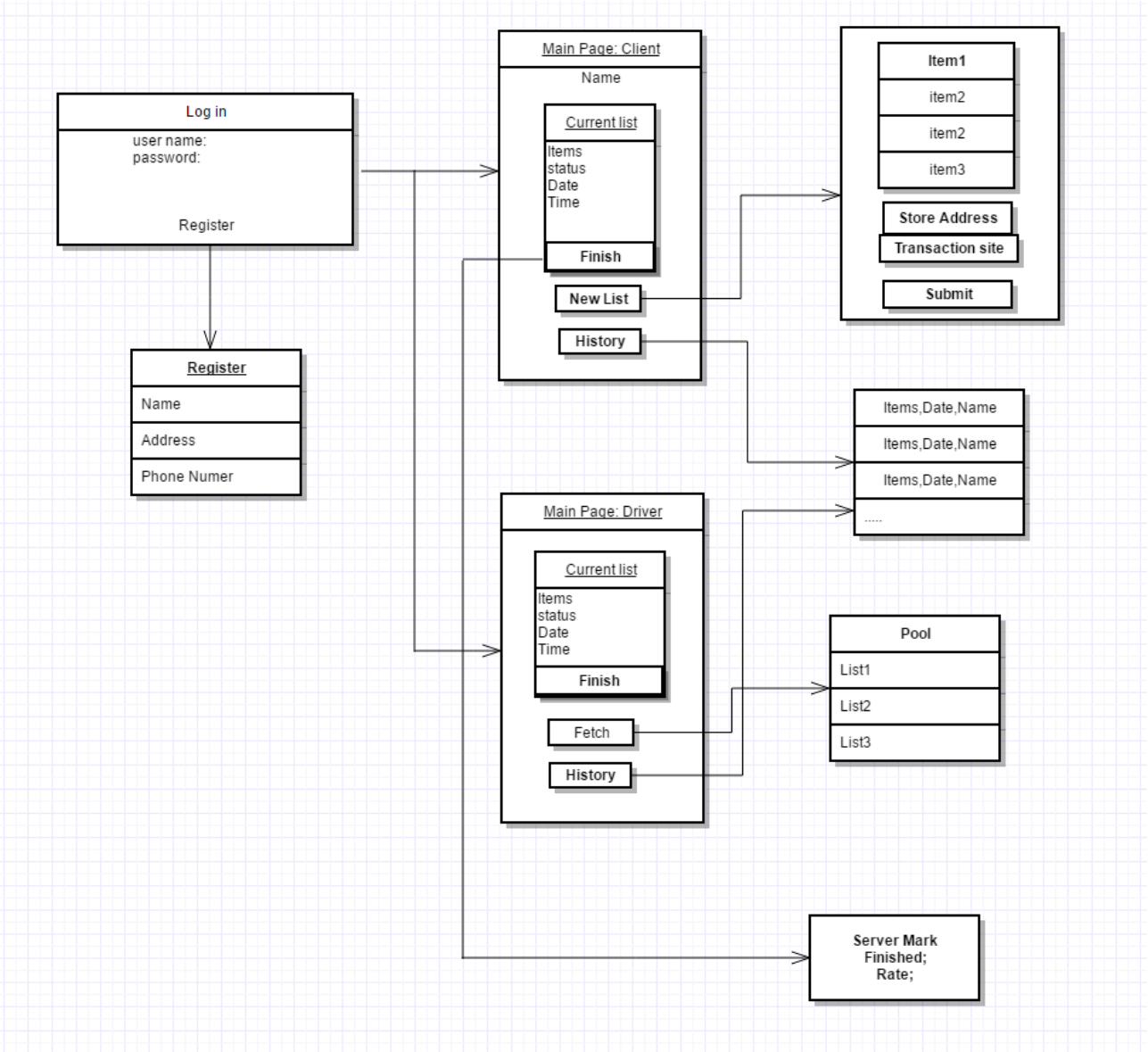


Figure 5.3

In figure 5.3 we show the parallel process when a client submits a list and a driver fetches it. In the beginning we can see that the driver has fetched the list of a client and on the right hand side we see what the clients sees when the driver fetched their list. At this moment, the client is unable to submit more lists and will wait until the deliver is either complete or canceled. When the driver presses the ‘confirm’ button, the driver needs to wait until the client confirms as well in order to have a mutual connection. The middle picture in the right hand side of figure 5.3 shows what the client sees when the driver presses the confirm button. A new button is available for the client from which the client can confirm that the groceries have arrived. By doing so, the client and driver get redirected to their main pages and process can start again.

5. User Interface Design



This GUI design will be implemented in html and php and will be built in the website itself. Current design for registration requires name, address and phone number to allow text-based conversation after the list has been chosen by a driver. For the first prototype, we will implement a very simple GUI to begin testing because we want to make sure the internal functions are correctly defined and implemented before we put a pleasant cover on top of our software.