Coupons Nao!

Detailed Design

COP4331_001, Spring 2011

Team Name: The A-Team
Modification history:

VERSION	DATE	WHO	COMMENT
v0.1	3/23	Taylor Kourim	Added in Sequence Diagrams
v0.0	3/23	Chris Nergard	Added Trace of Requirements

Team Members:

- Stephen Bryant - sbryant31@gmail.com
- Taylor Kourim - tkourim@knights.ucf.edu
- Daniel Kaplan - kaplan@knights.ucf.edu
- Isaiah Walker - senthose@gmail.com
- Chris Nergard - cnergard@gmail.com

Contents of this Document

Design Issues:

The first GUI prototype for our application was a tab-based prototype, using a Tab Manager. It moved slowly and was exceedingly difficult to implement, so we switched to a second GUI prototype. This was a minimalist button-based prototype. It was easy to implement and aesthetically pleasing.

We did not prototype the actual functionality of the application. Instead, we weighed the pros and cons of each design issue and based on our decisions will develop a prototype that will likely not change, barring any massive oversight on our part. The tradeoffs and design issues that we contemplated are in the following paragraphs.

We had to understand how to deal with locations. We realized we had to get the location from Android's Location class in the API. Android has a "getDistance" method which allows us to get the distance between multiple locations, which would have been trivial to implement. At first our idea was to retrieve all of the companies from the database and use "getDistance" to sort them by order of distance. Obviously, this led to a huge efficiency problem since we would waste a lot of time transferring locations between the app, PHP, and database. Our solution was to store latitudes and longitudes of the locations in the database and create a PHP script which would query by distance. This way, we could select the top few and transfer them to the app. All of the work would be done efficiently, server side, thus decreasing the total amount of locations that need to be transferred and increasing performance of the actual application.

As per Figure 9, an issue we have to deal with is transferring coupon objects to and from the database. The way we're doing this is through XML files for information and JPG files for Images. The information we decided to include in our prototype is ID, Name, Rank, Submitted (date), Expire (date), and Description. We have a PHP file that queries and returns it to the APP. Figure 5 outlines how server requests work.

From Figure 11 we can see that CouponList browses the database based on the companies and not on the coupons themselves. This is a design issue that we had to resolve—whether to organize the list by individual coupons or by the companies to which those coupons pertain. The trade-off is that the users are going to be more interested in an actual place or business as opposed to a coupon in which they have no context to apply it, but this implementation requires a higher level of abstraction and makes our methods more complicated. However, we think the increased difficulty is a better decision overall because it results in a better product for the user.

When users submit their own coupons, we need to find the image resource from their phone. The application publishes an Intent to the Gallery application (which ships with Android) or a similar application to find a picture and returns the URI of the picture they want to associate with the coupon. From there, we can get the explicit file name from the storage and send it to the database. This is opposed to opening a camera and taking a picture of the associated business, which we think could make it unreliable.

We are implementing our database with three tables, Company, Coupon, and Rating. They are all interlinked. This is opposed to our original idea of having one single table with company, coupon, and rating. This relationship is illustrated in Figure 1b.

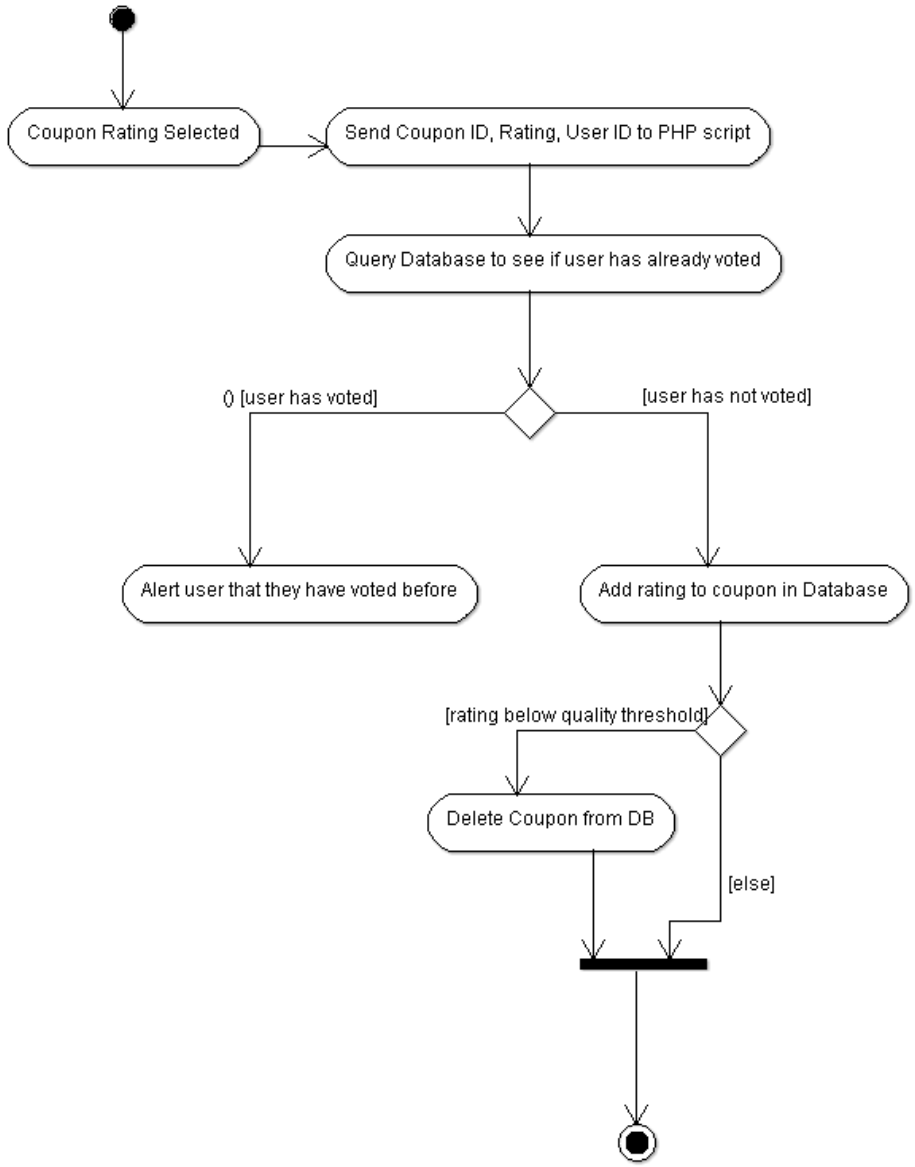

Figure 4: *Activity Diagram – Rate Coupons*

In order to rate coupons, we must interact with the server and make user verification decisions.

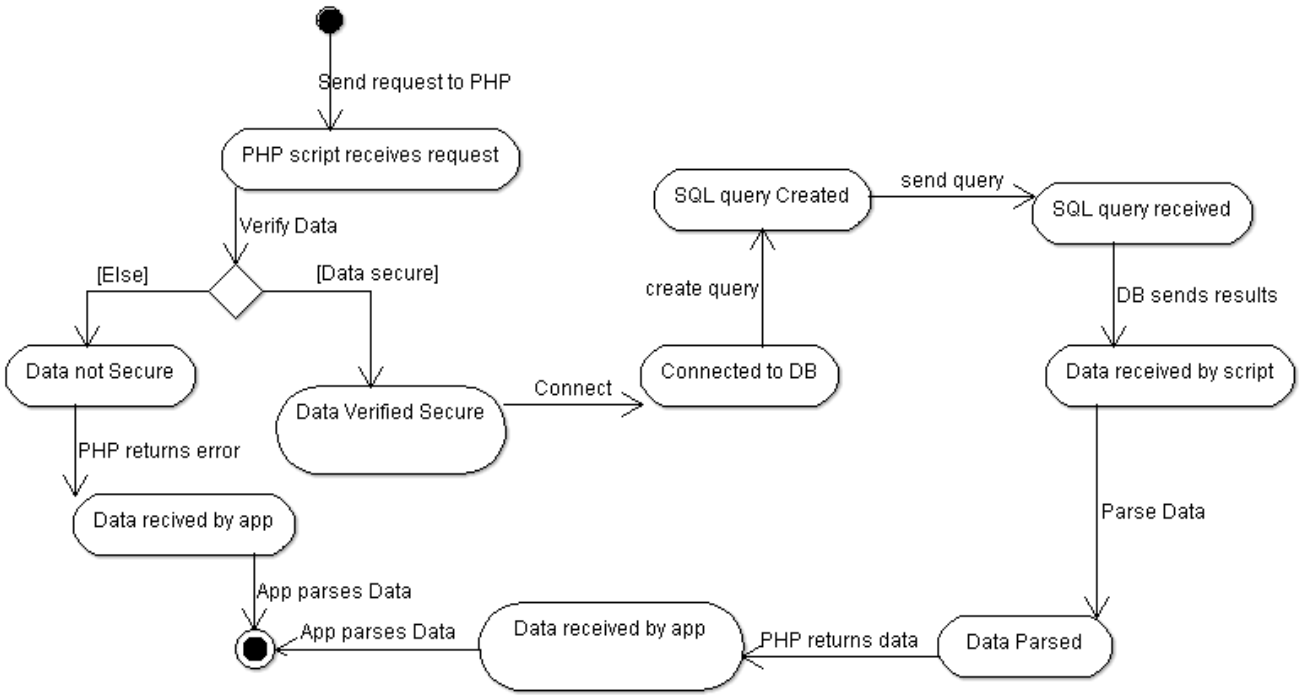

Figure 5: *Activity Diagram - Server Request*

When the Server receives a request from the application, it verifies the data is secure and then performs the query and executes the appropriate script, returning data to be parsed by the application.

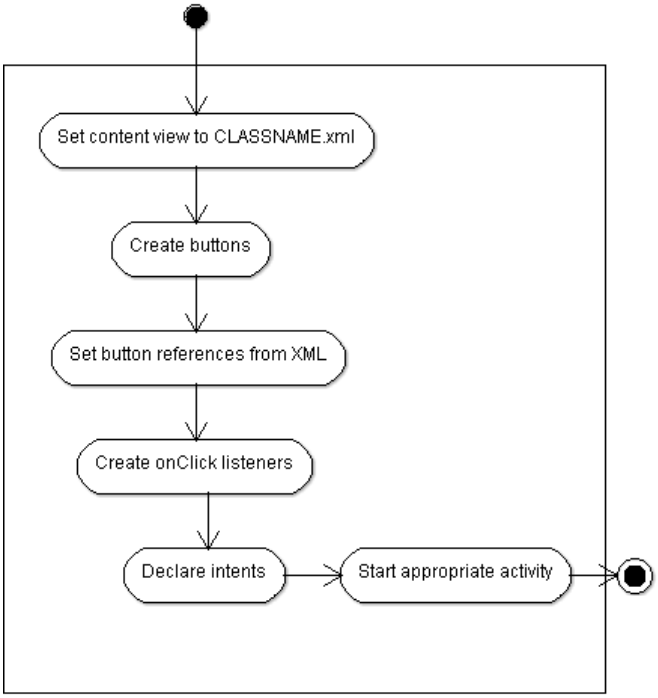

Figure 6: *Activity Diagram – Setup GUI*

This is the basic flow of GUI setup. We need to create a view, create buttons, set their listeners, and start the activity that corresponds with each button.

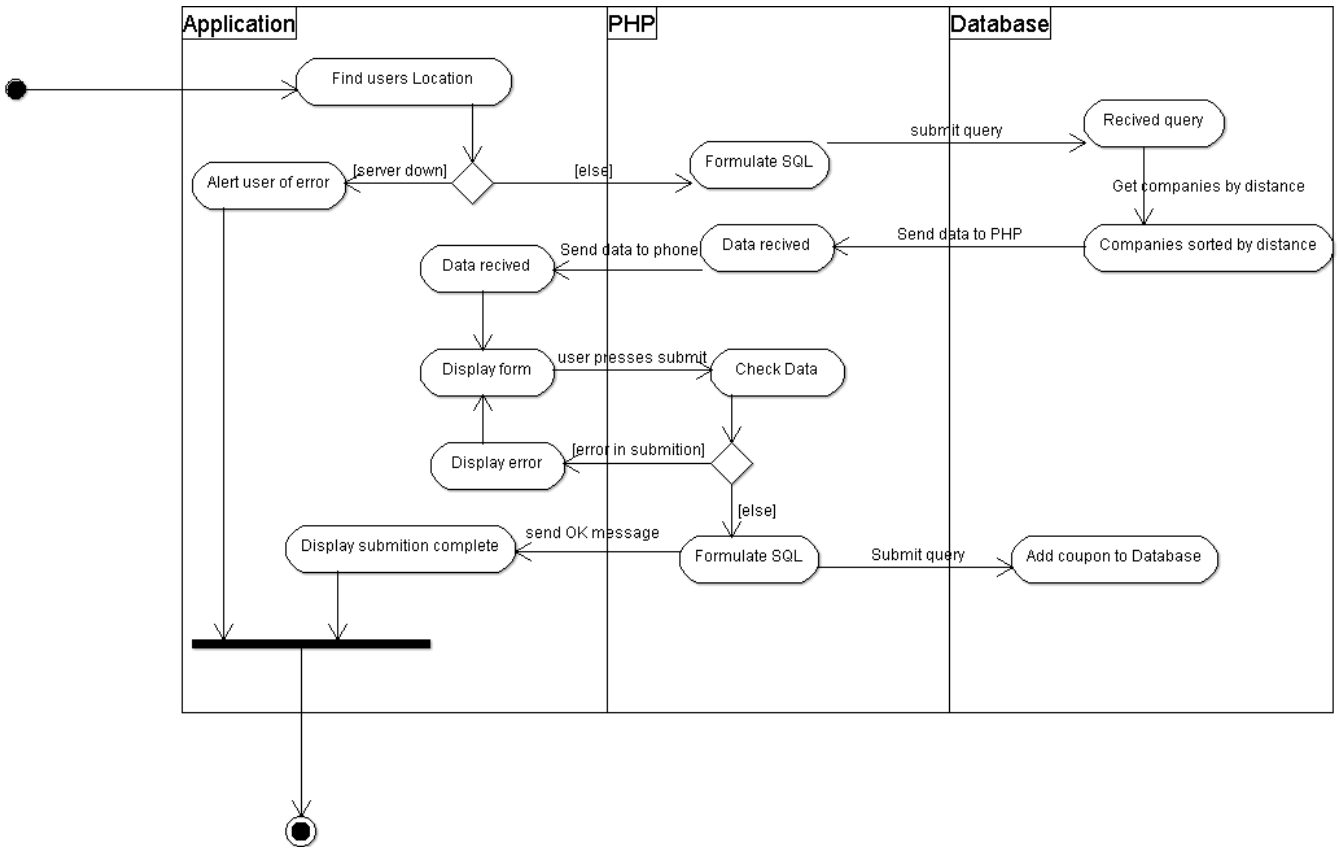

Figure 7: *Activity Diagram – Submit Coupon*

Submitting a coupon will involve going back and forth between the application, PHP, and Database. First, we must find a company from the database that is close, then get the user to submit the data and submit it, all while error checking.

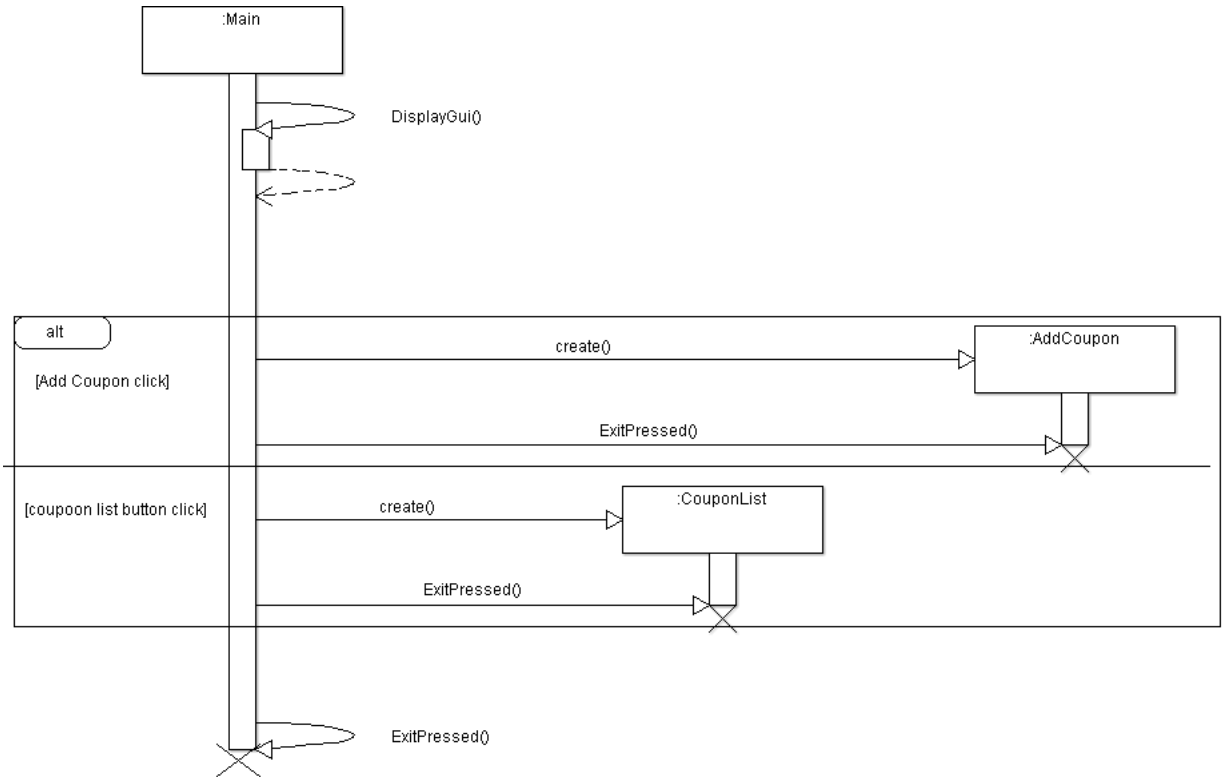

Figure 8: *Sequence Diagram – Main Activity*

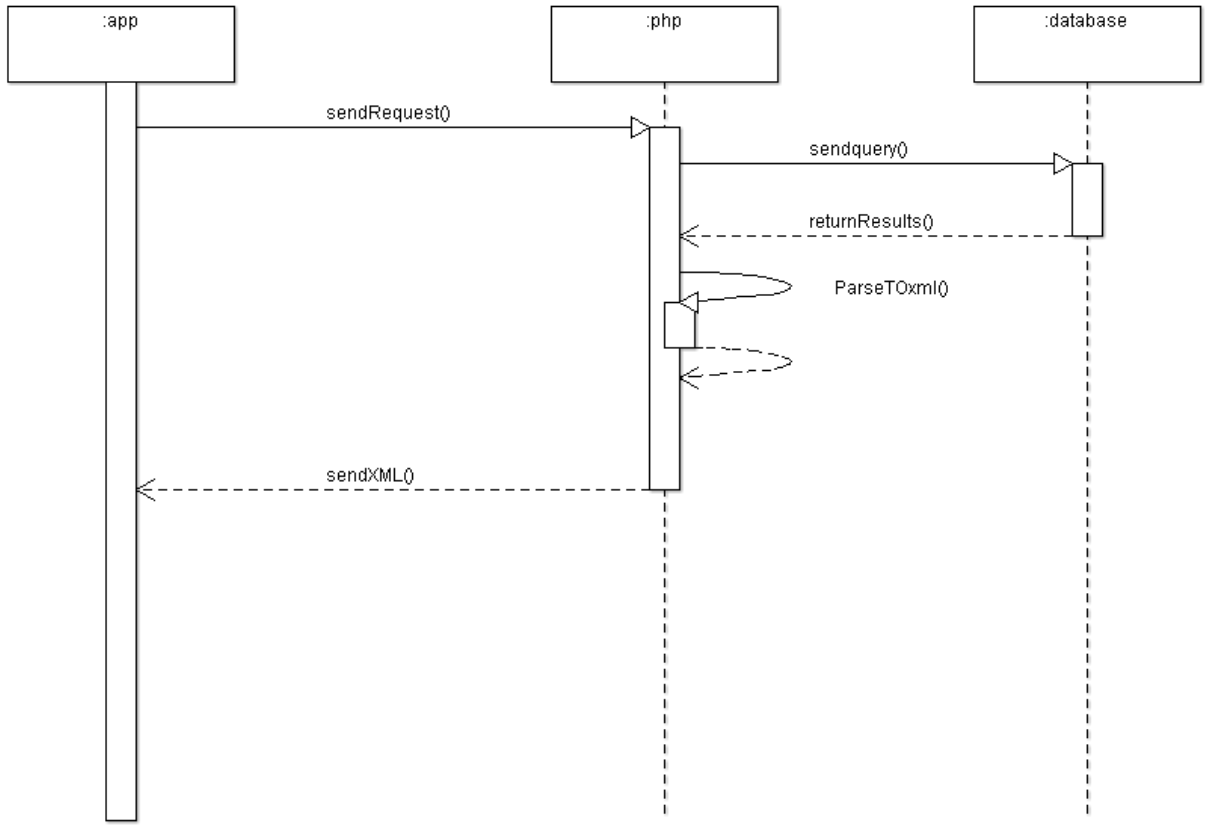

Figure 9: *Sequence Diagram – Server Request*

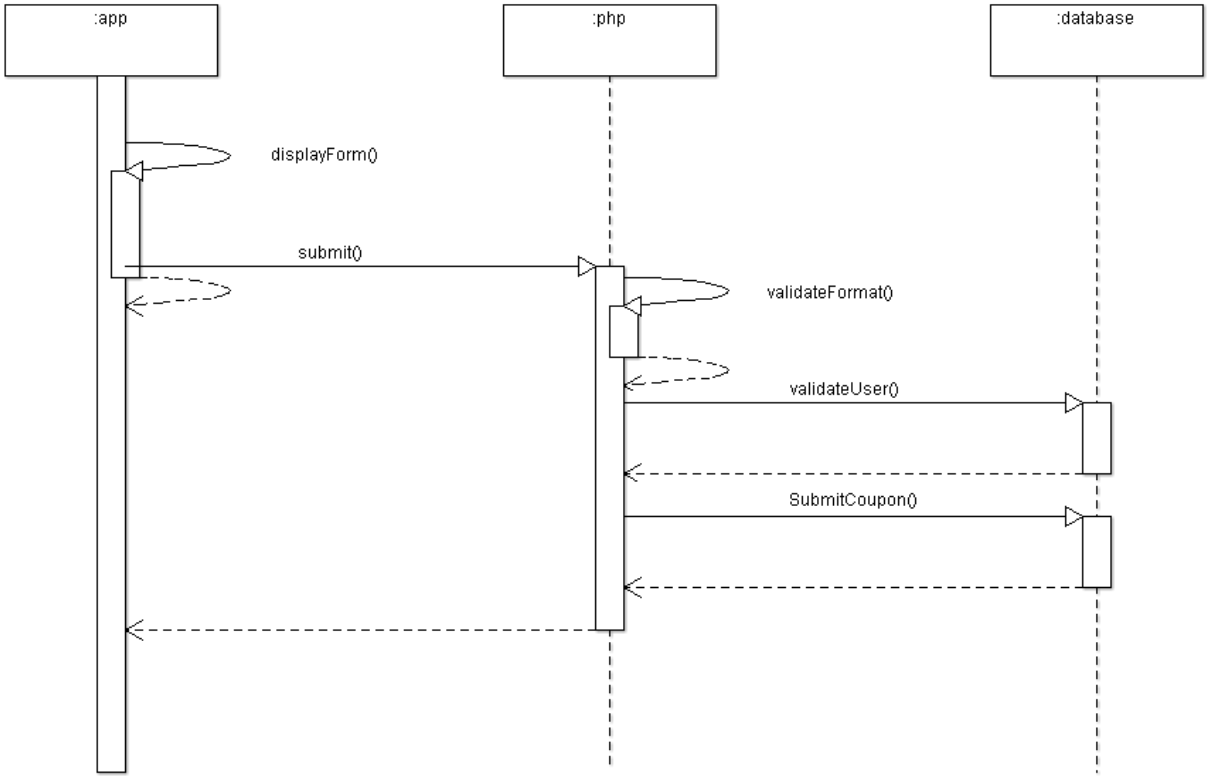

Figure 10: *Sequence Diagram – Add Coupon*

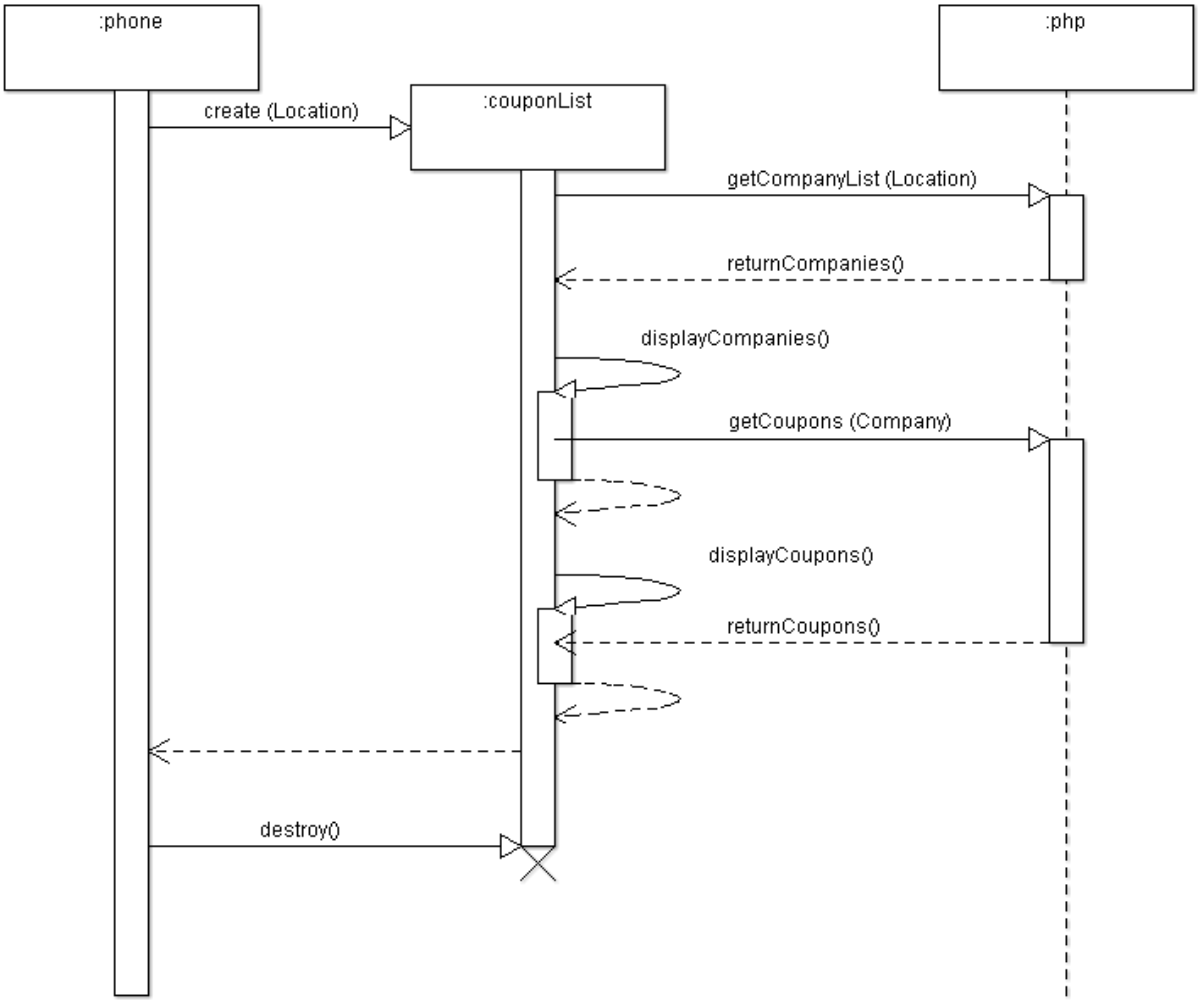

Figure 11: *Sequence Diagram – CouponList*

Trace of Requirements to Design:

Requirement	**SRS Number**	**Figure Location**
Functional	1. Special character check 2. Communication failure 3. Store location 4. Get rating 5. Get button click 6. Get coupon info	- Figure 5,7,10 - Figure 5 - Figure 1,7 - Figure 1a,1b,3,7, 9, 11 - Figure 6 - Figure 1,7,8
Interface	1. Database interfacing 2. Google maps interfacing	- Figure 2,5, 9, 10, 11 - Figure 1a
Physical Environment	1. Android phone 2. Internet connection 3. GPS capability	- Figure 2 - Figure 6 - Figure 1a, 6

User and Human Factors	1. Little prior knowledge 2. Button click 3. Special character check	- N/A - Figure 6 - Figure 5,7,10
Data	1. Distance calculation 2. Coupon object format 3. User id	- Figure 7 - Figure 1a, 1b - Comes from phone directly
Resource	1. Skilled person to build, use, maintain 2. Physical space: sever 3. Schedule 4. Hardware/Software tools	- N/A - N/A - See: Concept of Operations - See: Project Management Plan
Security	1. Queried data check 2. Activities logged 3. Server back up	- Figure 5,7,10 - Taken care of inherently by Android - N/A
Quality Assurance	1. Temporary storage 2. Reconnection prompt 3. Server statues script 4. User bug reports	- Handled through Android - Handled through Android - N/A - Handled through the Android Marketplace