Coupons Nao!

High Level Design

COP4331_001, Spring 2011

Team Name: The A-Team
Modification history:

VERSION	DATE	WHO	COMMENT
v0.0	3/23/2011	Taylor Kourim	Uploaded pictures
V0.1	3/23/2011	Stephen Bryant	Updated Captions

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

High-level Architecture:

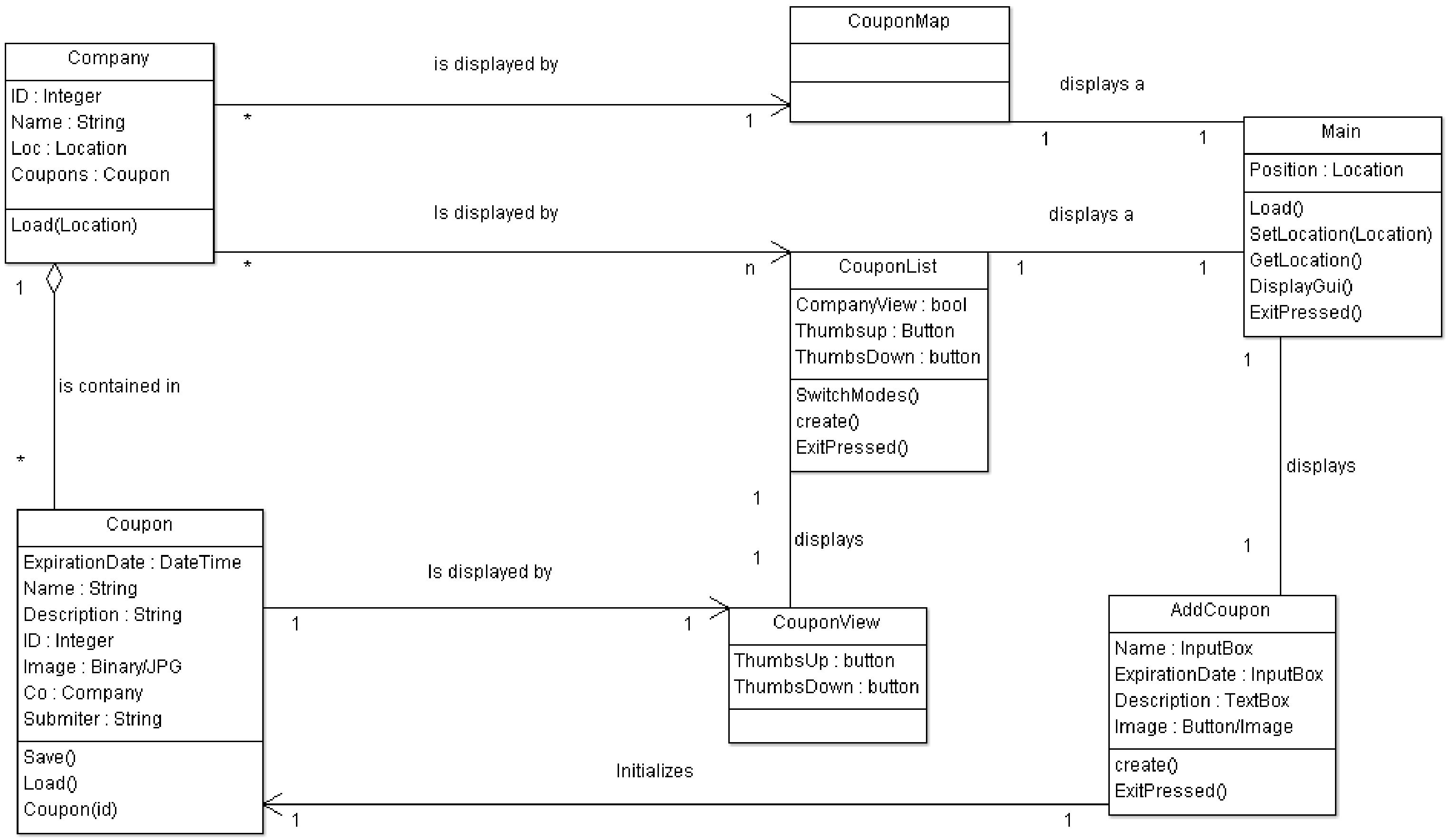

Figure 1a: *Class Diagram for the client app.*

Our app contains two "views" currently, CouponList and AddCoupon. CouponList is a list view of coupons which contains buttons for rating coupons. Our AddCoupon view is essentially a form to create coupons for submission to the database. CouponMap is a future possibility, in which we integrate with Google Maps, but we have currently foregone it. As you can see, companies are displayed by the coupon list or coupon map. Each company contains 1 or more coupons. When a company is clicked and a coupon is selected, the user is given a discount code or register code for the cashier to input if applicable, or is simply shown a screen with a picture of the coupon.

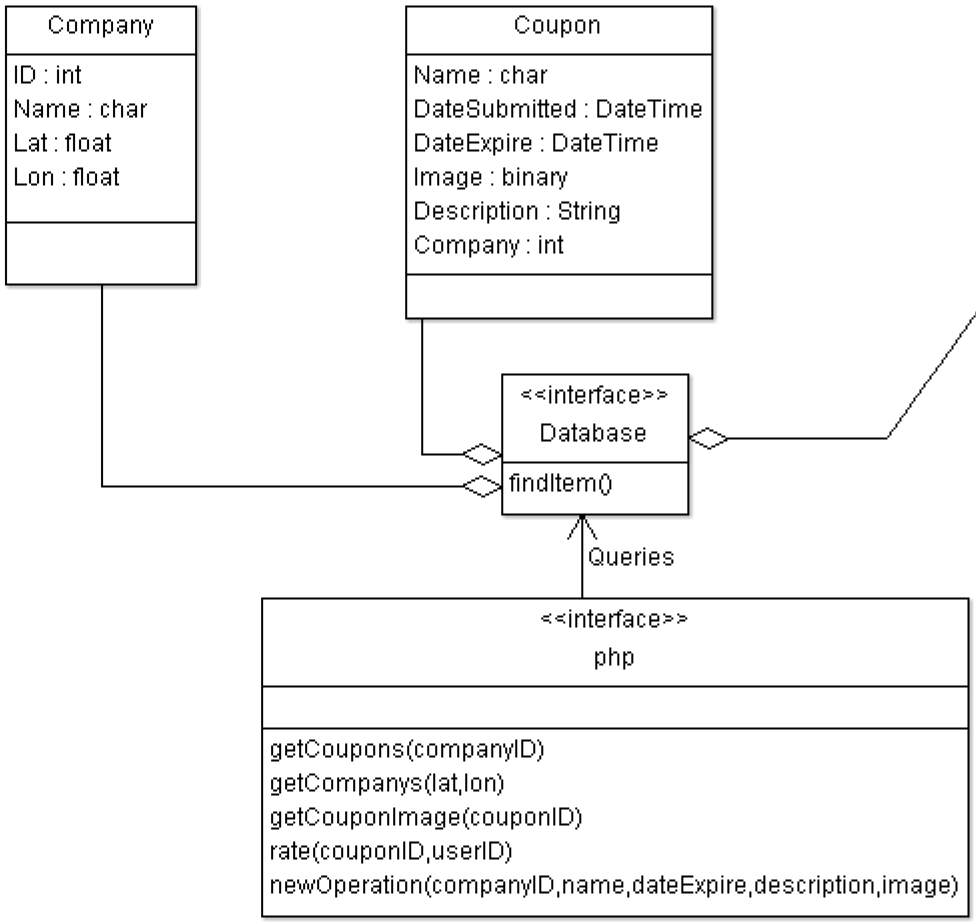

Figure 1b: *Class Diagram for the server*

The entries in our database are company, coupon, and rating, and we have PHP scripts in order to communicate between the server and the application.

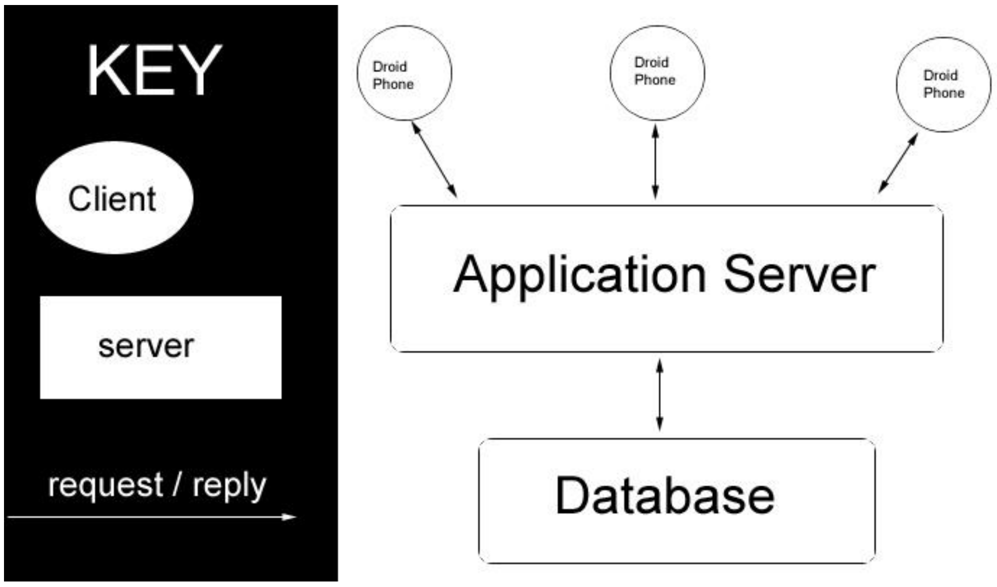

Figure 2: *Architectural Style: Client-Server*

As you can see, our system is connected via the "application server." Multiple clients connect to the server, which performs certain database functions and returns easily accessible information to the phone.

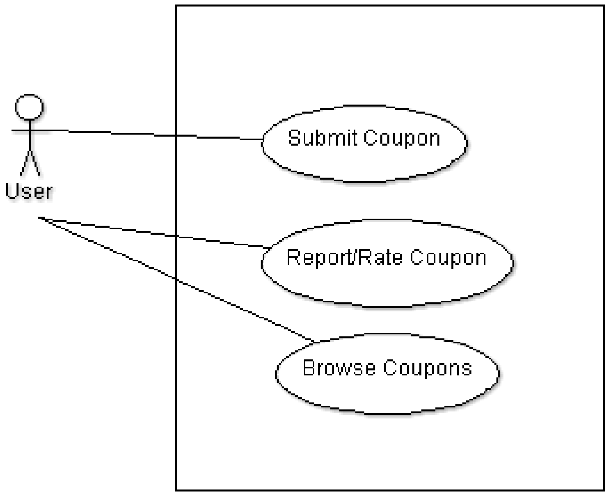

Figure 3: *Use Case Diagram*

From the user's point of view, there will only be three actions as of this time. The user can submit coupons, rate coupons up or down, and browse the list-view of coupons.

Design Issues:

When designing the system, our foremost concern was technical difficulty and simplicity. Since we are all inexperienced with the Android API, Google Maps, and Database Systems, we had to make a few proof-of-concept prototypes. As of the writing of this document, we have done two prototypes and revised our design. The "main" prototype was the tab-based GUI. The GUI worked out well, so we're continuing with the android platform. The Google Maps prototype never came through, so we decided to drop the concept and reintroduce it if we have time later. This changed our design from a "map-based" design to a "list based" coupon search, which will be easier to implement.

Maintainability was not much of an issue, since we can use a tool such as PHPMyAdmin to perform administrative functions server-side. Besides for that, minimal maintenance will be required in the final product. Also, since we have decided to forego Google maps integration, our performance is increased.

Testability is a difficulty with our design since only the GUI and PHP/Database functions can be tested easily as units. After that, interfacing will be the main issue. However, the Android AVD (emulator) that is built-in to Eclipse will help considerably when it comes to testing the look and basic functions of the applicable.

Looking through the many styles of architecture, we began with a pipe-and-filter style, but found that this didn't answer questions of the interconnectedness of the parts of our program. We found that the most important emphasis we have is transferring data between interfaces, as you can see in figure 2. We have our clients, our server script, and a database. This led us to adopt a client-server architectural style. The benefit of this is that we could switch from PHP to ASP on the server-side, or we could switch from Android to iPhone or to an interactive website as our "client." This makes our design very reusable and portable.

We had a few trade-offs to look at. The first one is the decision to work for android as opposed to a website. By working on Android we increase the development difficulty, but we gain important location-based features which are of core value to our program. The second trade-off was integrating with Google Maps. After some research, we found that Google Maps integration could possibly pose many more technical challenges than we can handle. Of course, we are losing "cutting-edge" appeal by switching to a "list-mode."