**Final Report for Knabba Project**

**Team Name: Knabba Squad**

**Ashirwad Shrestha**

**Eric Buhr**

**Jed Brundidge**

**Business Problem:**

Potential renters confront problems finding their preferred rental property in a short period of time before someone else signs the lease. Often a listed property is already rented out and time and effort is wasted pursuing it. Property owners have a difficult time finding qualified tenants. This adds costs for upkeep or uncertainty with unreliable tenants. Knabba (a new business venture) plans to solve this issue by creating better connections between property owners and potential renters.

**Solution:**

To accomplish this goal, we have developed a match-making model similar to Thumbtack using a web application. This project involved development of a working prototype of this application. It is a rental connection that brings landlords of desirable properties and qualified tenants together.

**Overall Design:**

The overall design includes a front-end built from HTML and JS. It includes login and sign-up functions to route existing and new users. Foundation was used to help build and valid the forms. The user information is transferred through a PHP server side script to the database (AJAX calls with JSON and XML data formats are used as intermediaries or handoffs between the front and backend).

There are two different types of users: landlord and tenants. They are both held in the same table “kusers” in a MySQL database. Both users have profiles which they add details regarding properties or preferences for properties. The landlord can view tenant profiles using city search.

All relationships are initiated and based around an offer that is created by the landlord. The landlord sends the offer. The tenant then gets access to review the landlord profile. He or she then replies with an acceptance or no acceptance.

**Design Challenges:**

As for design challenges, number one was the database architecture. There was a lot of back and forth on this. At first, we were leaning toward two separate databases one for landlord and one for tenant. But we realized both users shared almost all of the fields in some way. The only difference was landlords had more access. So instead, it made more sense to use one table for both users.

Also, the multi-form design and validation was tricky . This is where Foundation was helpful. It was important to keep this aspect in order to achieve the mobile-friendly design. We all found working with PHP on the database back-end and hand-offs was difficult because of the syntax. Some IDE’s and tools missed some PHP syntax errors or noticed inconsequential issues including NetBeans among others.

The structure of the Google Maps function was challenging but purposeful choice. We did not want to dump all lat and lng points out to Google Maps API and top out on the query limit too quickly. So we put the radius search on the back-end in the SQL query as a consideration for the future. Adding the geometric equation along with multiple real\_escape\_strings made this part extremely difficult to code. (Attempting the search in JS or use of prepared statements would have been a better option.) We also had problems translating the XML output (which was the approach recommended by Google) to render on a map. JSON would have been a better choice there.

**Features (as presented to business team):**

Considering the time and resource limitations, we estimated that we could complete the initial features in red for the working prototype. As time permitted, we worked to add the additional features in blue (nice-to-have). We were able to complete all initial features, two nice-to-have’s (except for the preference-based search of tenants and email subscription) and three additional features.

**Done Working prototype (Initial features):**

X User account creation (landlord, tenant)

X User login

X Access profiles (landlord)

X Create/Send rental offer after seeing profile (map function)

X Respond to offer

X Each record has about 25 fields (details)

**If time allows, we will work to add these features to the prototype (nice-to-have):**

X Upload photos to landlord profile

Preference-based search of tenants

X Mobile-friendly format

email subscription

**Additional features we added:**

* FB Oauth
* Geocoding and map search function (not integrated)
* Optional tenant photo

**Compared to Original Plan (Why varied?)**

Obviously, getting an additional team member allowed us to tackle more features than we would have originally planned. But looking at all the essential and nice-to-have features combined, there was simply not enough time even with three team members to complete all items.

We identified two nice-to-haves that could be accomplished within the timeframe. Mobile-friendly was easy to achieve with Bootstrap. And the business team specifically asked for uploading photos as the first nice-to-have feature to complete. So once we had Jed on the team, we felt this was doable.

The additional features we pursued were probably influenced by having a business client involved. We felt that these additional features were much more impactful and exciting for the user vs. the other two nice-to-haves which seemed run of the mill. We were hoping to impress the business team with some of these additional features. Based on their immediate feedback, it sounds like we had accomplished that.

**Technologies Employed**

PHP, HTML5, CSS3, JS , JQuery, AJAX, MySQL, XML ,JSON, Bootstrap, Github, APIs: Facebook Oauth, Google Maps API, Foundation and Lightbox

**Challenges for the project**

* Working with Business Team
* Absorbing a new team member several weeks in
* Design of database
* Multiform design and validation
* PHP syntax
* SQL query/syntax
* XML with Google Maps

**If we had more time (which we might with a version 2):**

* Security
* Add Google and Twitter Oauth
* Overlays on map search (Circles)
* Revisit data architecture
* Consider Angular JS as alternative to jQuery
* Email notification of offer/reply
* Use JSON in more cases
* Version 3 --Consider mobile design (Android and iPhone)