The Bachelor Pad

HUAXIAO CHEN, AURNIK ISLAM, SITUO MENG, DEAGAN MONAHAN, AND SHUANGLI ZHOU, UNIVERSITY OF CALIFORNIA, SAN DIEGO

1. INTRODUCTION

Most college students do not own their own house while most colleges and universities do not offer students four year on-campus housing. Thus, every year, lots of college students are working hard just to find appropriate accommodations which are cheap, safe, fun, and near to campus. However, there is no integrated system to help students to do that: you have get traffic information, pricing information and surrounding information from three different websites.

Bachelor Pad will serve as a totally new and convenient way for students to find off-campus housing in just one meticulously designed system. We were able to integrate all the important factors regarding housing selection in our web app and change the boring and exhausting housing-choosing experience into a joyful journey.

In section two, we will further discuss the motivation behind the project. We will talk about why the current popular real-estate websites are not sophisticated enough to provide the users a wonderful experience and then go through how Bachelor Pad will help people save time and find perfect accommodations.

In section three, we will explore the spirit behind Bachelor Pad. We will go through how the idea was formed, and how the idea was realized by multiple iterations of wire-framing and prototyping

In section four, we will get to the technological system architecture of the application, which includes the MVC framework and backend use of data from different APIs. We will include the functionality and implementation details of our features regarding the architecture as well.

In section five, we will show how we successfully made use of all the HCI principles in our system. We will go from feature to feature, explaining why thing are made the way they are.

Finally, in section six, we will discuss how our system can be further adapted and developed to become a more sophisticated online real-estate application that fulfill the requirements of more users.

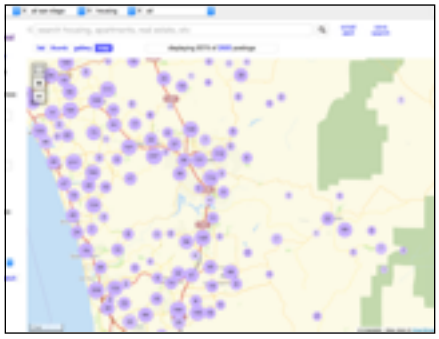
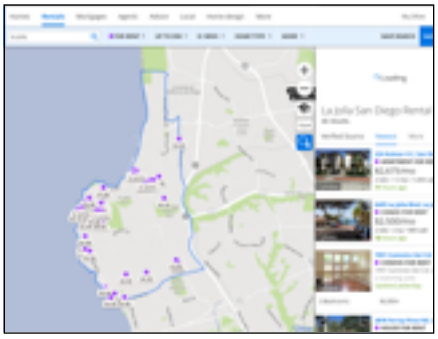
2. MOTIVATION AND BACKGROUND

2.1 History

With the development of the Internet, people stop satisfying with the traditional information searching platforms. Be more specifically, when people are trying to find a house to buy or rent, instead of reading advertisements on the newspaper or just hanging out on the street to find big signs saying “FOR SALE”, they turn on their computer and type in the keywords in search engines like Google(the left pic.) to easily get much more information much easily.

2.2 Current Solutions

There has been lost of brilliant ideas and projects done driven by the goal to eliminate the communication barrier between owners and buyers. The most famous ones include Craigslist(founded in 1995, the mid pic.), which includes all kinds of trading information. However, its housing system is hard satisfying. The system is hardly interactive and the searching system is based only on keyword — if the owner forgot to add the tag “near to UCSD” when he/she was posting the house, you can never find it under “near to UCSD” category. Not to mention “UCSD” is different from “University of California, San Diego” in this system. Another more professional and sophisticated system called Zillow(founded 2005, the right pic.) was really a huge progress. Although Zillow started even without a map, it started its collaboration with Google quickly and is basically the best real estate system in

terms of user experience.

2.3 Motivations

Although some systems, like Zillow, have done a great job showing relative information, we believe that there are still more things we should show to the users to help them quickly find their dream houses. Targeting the college students, we determined the most significant things we would like to show to the users: transportations, prices and surroundings. We want our users to quickly know about how convenient it is to get to a particular place(UCSD for now), how expensive is the house(or for some particular area) and what are the buildings near the housing(restaurants, grocery stores and clubs).

3. DESIGNING AND PROTOTYPING

3.1 Low fidelity prototype

We started designing the low fidelity prototype by thinking about all the main functions we need. We started by trying to figure out how to visualize that much data in just one map. And the biggest problem was that people might have different needs for the surroundings around an area, so we

came up with the idea to create two modes: the day and the night mode. At the same time, we believe that using sounds will be a great way to improve our user experience. We decided to have four buttons on the first view of the main page: About, Our team, Join us and Start. The Start button will jump right to the map and the map will have a switch button for sound on/off.

3.2 High fidelity prototype

For the high-fidelity prototype, We started using Sketch, an light-weight OS X application to render the whole webpage design. First we created a modern-looking front page, and then based on that design, we added more elements. And for the map, we added motion designs before the very first presentation, and the webpage demo is interactive, animated.

Also, instead of just putting those elements together, we care about details. The font Didot is carefully chosen to give the whole webpage a elegant feel and look, the alpha of all the elements is adjusted to be transparent, thus, those elements will not be intrusive and absurd. The back ground image is also deliberately chosen from hundreds of possible images. Since this image have a clear sky view, which is good for the typography to lay on, we used it as the main background.

On the high-fidelity prototype, we also created demonstrations of some key-features of our website. For example, the range view, which will show you the driving range in minutes (approximately) on click. In the presentation, the buttons are clickable to show the actual menu.

Also, on the sketch application, all of the layers are easy to separate and export. This made the whole web-making process easy and comfortable. Our website designer is able to create the webpage in very short time period.

In a nut shell, the high-fidelity prototype provides us a express way to create the actual webpage, and give us a direction to work on.

4. SYSTEM ARCHITECTURE

4.1 Node.js

Node.js is a platform built on Chrome's Javascript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices (cite). We chose to user the Node.js framework for our backend not only because this was the chosen design pattern throughout the course, but also for the ease of communication we achieved between the Javascript on the front end and the Javascript on the backend. Keeping the majority of our project written in Javascript allowed us to easily add modular code that was familiar from a developer standpoint whether you were writing serverside or clientside code.

4.2 AJAX

AJAX stands for Asynchronous JavaScript and XML, is the technique of exchanging data with a server, and updating parts of a web page - without reloading the whole page (cite). To reduce heavy computations on the client side of our application we used AJAX to query our remote Mongo database, access local Delphi files, and initialize all of our different 3rd party API calls to Zillow and Google. The implementation of this system allowed our application drastically reduce wait/loading times, and provide the user with a smooth, responsive experience.

4.3 MongoDB //TODO or erase

Rental price of houses in San Diego county is the key dataset to our team, because our website is designed with college students in San Diego in our mind. So since the Delphi

database doesn't provide us the data with we need, we have to turn to Zillow. However, we found two problems with getting rental price from Zillow API. First, although Zillow website itself hosts incredibly thorough information for every single documented house, there isn't much useful information in its API. Second, we can't actually get ahold of the property id which is required for calling Zillow API by specifying a certain region, for us, SanDiego. Since there is no other way around to call Zillow API, we had to resort to web crawling to secure Zillow Property ID first and, at the same time, rental related information. We used two Node module to achieve our goal. One is js-crawler.js , the other is cheerio.js. The former provides one API for crawling and returning HTML document while the second exposes APIs for manipulating DOM objects using JQuery on the server side. Equipped with these two modules, we are able to scrape Zillow website and get the raw textual information we want. Then we used regular expression to do some pattern matching in the text, constructed a MongoDB document schema and finally programmatically sent them into the remote MongoDB powered by mongolab ready for downstream usage in the front-end side. Because we are using the MVC model, our project is highly scalable in terms of data storage. We can provision more information when needed and store in the remote Mongo database server. The action of preparing information can be done in any way in any language that has a MongoDB driver. For any additional data-based functionalities, we only need to add corresponding query scripts to our app.js running on the server side, then using ajax calls to bind user-generated data request with the query.

4.4 Google Maps Javascript API (V3)

A very common idea in software development is avoid "Reinventing the wheel", so when it came to the design of our map we chose to stick to an option that allowed us vast customization features with incredible power and scalability. Google Maps Javascript API not only offers large amounts of global data to be easily added to your project, but google has gone above and beyond with this API when it comes to customization. Everything from

highway colors, map texts, to water colors and custom marker overlays, almost everything is personalizable to our application.

Using the customization of Google Map API, we were able to clearly distinguish our 2 different maps between “day” and “night” modes based on colors and intensities. On our custom “day” map, we were able to define saturations and hues in a way that helped the user focus on streets and housing areas, whereas in our “night” map we found a useful contrast between housing and nightlife.

In addition to Google’s map API, Google offers multiple other API additions to compliment different data sets within the map. Among these additional API’s are Google Fusion Tables and Layers, Live Traffic information, GeoLocations, Google Places API, Google Autocomplete searches just to name a few. The Google Map API quickly became the foundation of our application that provided us the strength and scalability to implement useful features and controls. The Google Map API comes equipped with a large array of event listeners, that proved to be invaluable not only for our standard mouse operations, but when we began to implement our Leap Motion Gestures. Adding functionality to allow the leap motion to not only control the map navigation but to also control the data calls through the map even listeners were invaluable to keeping our development times on schedule. The decision to utilize the Google Maps API has left us in the position to easily and quickly increment additional features in the future.

4.5 Google Fusion Tables/Layers

Google Fusion Tables is an experimental data visualization web application to gather, visualize, and share data tables (cite). You have the option to search through already defined sets of data that others have made, or you can do as we did, and create your own table with your own datasets and parameters. Combining the data from the Delphi database and the boundary regions provided by the Google Maps API we were able to quickly merge the

datasets into a useful format that allowed us to provide the user with what is called a Fusion Layer. Fusion Layers are a format that communicates easily with the map visualization and overlays the layer onto the map. This merging gives a quick and efficient way to display our Fusion Table inside the map. One major benefit from using Google Fusion Tables is the large amount of community datasets that are available online that could potentially provide our application with future data sets that can be easily added to our table or added as multiple Fusion Layers. A second benefit of utilizing the Google Fusion Tables is the ease of scalability. Our table could easily be expanded for more regions, more parameters, or larger times spans. Although we only barely touched the possibilities of the Fusion Tables, these tables would be strong factors in the future scalability of our application.

4.6 Google Reverse Geocoding API

Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers on a map, or position the map. Reverse geocoding is the process of converting geographic coordinates into a human-readable address (cite).

Among the many benefits we were allowed by using the Google Maps API, one important feature that surfaced was the opportunity to use Google's Reverse Geocoding system. Although very easy to implement, it has great potential to be used in a variety of powerful applications. Using Reverse Geocoding, we were able to allow the user to click anywhere on the map, and retrieve the "human readable" address, which we were then able to send to the Zillow API and retrieve housing stats not only about the house we clicked on, but, also comparable addresses in the area. Due to the limitations of the Zillow API this Google API was a very important factor to the success of our Zillow API integration.

Google Street View API

The Google Street View Image API lets you embed a static (non-interactive) Street View panorama or thumbnail into your web page, without the use of JavaScript. The viewport is defined with URL parameters sent through a standard HTTP request, and is returned as a static image (cite). Keeping in mind, the datasets of our application and HCI principles, the integration of the Google Street View API played an important role in meshing our data results from the map into a UI design that was helpful to the user. We used the Google Street View API as a visual accompaniment to the Zillow housing results to give the user a visual indication of what to expect when it came to the location of that property. Although this was a design addition that we added towards the end of our implementation, the use of the Google Street View API in the finished product adds useful design feature to quickly and effectively inform the user in a visual way.

4.7 Zillow API

The Zillow Housing API, was an important dataset provider in our application. In addition to the different Delphi datasets we used, we felt it was important to be able to provide users with fresh, up-to-date housing and entertainment information. For this additional data we chose to use the Zillow Housing API, along with the node-zillow module. Utilizing the Zillow API allowed us access to Zillow's vast database of housing information. While the implementation of node-zillow allowed us to keep our development in a familiar javascript format. Although the Zillow API allows access to most of Zillow’s information, it still came with its limitations that we had to workaround. One of the biggest issues we faced with the Zillow API was the lack of a multiple listing query. Zillow does not offer a query that can return multiple results based on a price range or number of bedrooms or any other parameter for that matter. So along with the previously mentioned Google Reverse GeoCoding and some Zillow web scraping, we were able to store Zillow data into our

database which allowed us to perform useful queries on the housing results that would benefit our application.

4.8 Features

4.8.1 Day/Night Mode

We decided early in development that our user experience could be enhanced by not only showing useful information regarding housing, traffic and entertainment. But as with many data sets, these comparisons are not always constant. So we decided to not only give users information regarding “day” statistics, but also to make them aware of data shifts through time periods beyond day time hours. We separate our data sets between “night” and “day” modes to show useful information based on these settings. So whether a user wants to analyze traffic and shopping information in certain neighborhoods, or if they are interested in local nightlife locations, all the data is separated by a simple mode change. Visually, the user is clearly aware of the relations between home locations and desired local establishments very quickly thanks to our innovative blending of technical and HCI integration. Day mode has a custom map with light colors that emphasize streets and roads and shopping. While the night mode changes in contrast to emphasize local routes to entertainment. These modes can be toggled with the map overlay icon on the bottom left of the map or with the custom gestures through Leap Motion.

4.8.2 Price overlay

The price overlay is generated from a Google Fusion Table and informs the user as to which region (by zip code) is more expensive based on the Delphi dataset of median housing cost by square foot. Performing a click on each of the regions will show a dialog window that displays the Zip code of that region, the median price per square foot of that region, and the

city name of that region. This feature is easily toggleable from the maps navigation bar and can be seen in both day and night mode.

4.8.3 Shopping markers

The shopping center markers are distinguished by a red (seductive rat?) custom icon image. These images are animated onto the map based on the Google Place API call results that are generated based on the latitude and longitude of the center of the map and the zoom of the map. These icons are toggleable from both day and night mode of the map. These Google Place calls are of the type, “grocery_or_supermarket”.

4.8.4 Entertainment markers

The entertainment markers are distinguished by a blue custom icon image. These images are animated onto the map based on the Google Place API call results that are generated based on the latitude and longitude of the center of the map and the zoom of the map. These icons are toggleable from both day and night mode of the map. These Google Place calls are of the types, “bar”, “night_club”.

4.8.5 Search Form

The user is able to perform housing searches based on a minimum and maximum rental cost. The min and max forms are located on the map navigation bar and can be initialized only after both the forms have been filled with valid numbers and the user clicks the “Search” button, which is also located on the navigation bar in the map. This search calls an AJAX function from the client side which initializes a database query on the server side that communicates with our remote MongoDB. The query results in either the housing data being applied to the map or an error alert in on the upper edge of the page.

4.8.6 Traffic Layer

A useful dataset that was incorporated into the map was the ability to see live traffic alerts via the "Traffic" toggle in the map navigation bar. The convenience of this layer allows the user to get an idea of what kind of traffic to expect at the current time of day around the region they are considering.

4.8.7 Sound Experience

The innovative sound feature that is applied to the navigation of our map is achieved through the use of the node module, Howler.js (cite) and the Google Places API. While the Howler module handles the manipulation of the audio loops and volume fading, the Google Places API initializes the howler effects by calling a Places query based on the location of the map. After the Google Places call has returned, it notifies the Howler module as to how many results have returned, which allows the Howler module to fluctuate the volume of the sounds as needed. The type of sounds that play are an ambient public drone sound during day mode and a club thumping bass line during night mode. Sound effects can be toggled from the upper navigation drawer of the site.

5. GESTURE-BASED INTERACTIONS

5.1 Appropriateness

Since we are using a map, it felt natural to have gestural input to control the zoom and panning on the map. We wanted to make the map feel like the user was interacting with a physical paper map, except with dynamically loaded information based on the user's current map center. The Leap Motion can be a little finicky so we didn't want the gestural input to become a nuisance rather than a benefit when users used our application, so we made a lot of checks to make sure that the gestural interaction was smooth for the user and did not make any changes when the user accidentally moved their hands. In the end, we created an experience that worked well with our application because the user can easily

navigate the map and change between our day and night modes without having to touch their computer at all.

5.2 Taxonomy

The main gesture for navigating the map involves the user turning their hand into a fist, and then moving their hand in any direction. This is supposed to symbolize the user grabbing the map and dragging around the point that they grabbed so that they can see more of the map. Additionally, grabbing and pushing out is like pushing the map away so it zooms out, and grabbing and pulling in is like pulling the map closer so it zooms in. The gesture for night mode is all fingers down except for the thumb and pinky, forming the shape of a drink or symbolizing partying which is very fitting because our night mode highlights the nightlife in San Diego. The symbol for day mode is the pinky, ring, and middle finger extended to form an “OK” sign because this is a hand sign people are familiar with and it correlates well with our more easygoing day mode.

5.3 Gestural Input and Tasks

Gesture	Task
close fist	begin panning/zoom
open fist	end panning/zoom
move fist left	pan right
move fist right	pan left
move fist up	pan down
move fist down	pan up
move fist towards screen	zoom out
move fist away from screen	zoom in
thumb and pinky extended	go to night mode
middle, ring, and pinky fingers extended	go to day mode

5.4 Device Selection

We originally wanted to use the Kinect hardware, but decided not to after finding very little JavaScript documentation for it. After this we found a new device that is worn like a bracelet that can track the position of the and and rotation, however we decided not to use this either because we wanted to track the user's fingers as well as hand position. So, we decided to use the Leap Motion because it was simple to set up, and tracked everything we needed it to. It's not quite as powerful as the Kinect because the area that it tracks above the device is not very wide so the user needs to stay within that range to avoid error, but if that is done then the device works well for our needs.

5.5 Testing and Evaluation

When we first got the Leap Motion working, we had some friends test the application (at the time we just had simple panning across the map). The problem at the time was that the user would often have to move their hands around a lot to try to figure out where the device would detect them, and when it did it often scrolled too much or in the wrong direction. To fix this we added feedback on the page of the application to show where on the screen the user's hand currently was, how strongly the it was currently being tracked, and whether or not it registered the user grabbing the map. Then we had people test this along with our new zoom features. What we saw was that most of the time the user did not want to zoom but the map would zoom large amounts. We first tried to fix this by just requiring a large amount of movement in order to trigger zooming, but even then there were times where the map would zoom when the user didn't want to zoom. We fixed this by waiting for the user to finish their movement before zooming, because Google Maps has predefined zoom levels and so zooming instantaneously is very distracting. There would be times when the user would accidentally move their hand forward and then back to where they originally were before releasing the grab, and so the map would zoom in and then out instead of just calculating the total change in distance moved at the end like it does now.

The Leap gestures are taken care of in our index.handlebars file at Line 1484. We are using the screenPosition plugin to convert the user's hand position to a position on our map to see where they are grabbing. Line 1496 describes the panning that takes place if we register that the user is trying to grab the map, and then Line 1511 shows what happens when the user is not grabbing anything. Within this,

Line 1514 shows what happens if we see that the user is not grabbing anything but the user was just grabbing something, and so we are registering this as the user letting go of the map. At this point we check to see if the user was trying to zoom in/out and handle that within the map. Line 1526 checks for the gesture to change to night mode, and then right after that Line 1529 checks for the gesture to change to day mode. Within all of our checks we also take into account the confidence level of the Leap Motion so that we are not making changes when the device is not entirely sure about what it is detecting.

6. HCI PRINCIPLES

Generally, the usage of the HCI principles has been a very important part to our project since the day the project started. We believe what we provide is not only a website or a product, but a service. We designed all the functions not only based on what we want to show, but more on what we want the users to perceive. Our main view of the app is a map. We chose to use Google maps API because we believe that Google maps is a top notch navigation tool in terms of user experience — it is simple, clear and informative.

6.1 Visibility

We believe experience is more than what you see. Other than a carefully designed beautiful page, we have sound to inform the users of the surroundings of a particular area — the louder the sound, the more grocery stores/clubs there are in that area. And to distinguish the two different types of buildings, we used different sounds as well — crowded voices for the day mode and club music for the night mode, so that the users can intuitively get a basic sense of the surrounding in that area.

6.2 Simplicity

Simplicity is always a very important factor affecting the performance of a system. We tried to design the system as simple as we can in terms of using. We only have four interactive buttons in the

first page — About, Our team, Join us and Start, with Start at the center of the screen. Every button is clearly self-explained and the Start button will guide the user directly to the map.

6.3 Consistency

Being consistent is also extremely important in HCI design. We wanted the user to gain a very consistent experience during their exploration of the site — they will feel no obstacles and they can easily recognize our site even without reading the titles. We had Leon designed all the original icons and buttons to confirm with the main theme and we made sure that every interactive button is always at the same position.

7. TESTING AND EVALUATION

7.1 General

We started test every functionality at a very early stage of the project. Our testing was not only about if the function was working or not, but also useful or not. We deleted some of the not-so-useful functions like age demographics and focused on the important functions. After everyone has done their part of work separately, we started to merge and solidify the parts together. We have ran into problems, for example, we spent a long time trying to figure out why we were not able to parse the data to the map. And the Leap Motion was not always “giving” the map the correct instructions. However, after prolonged testing and debugging, we were able to fix most of the problems.

7.2 Data

Rental price of houses in San Diego county is the key dataset to our team, because our website is designed with college students in San Diego in our mind. So since the Delphi database doesn’t provide us the data with we need, we have to turn to Zillow. However, we found two problems with getting rental price from Zillow API. First, although Zillow website itself hosts incredibly thorough information for every single documented house, there isn’t much useful information in its API. Second, we can’t actually get ahold of the property id which is required for calling Zillow API by specifying a

certain region, for us, SanDiego. Since there is no other way around to call Zillow API, we had to resort to web crawling to secure Zillow Property ID first and, at the same time, rental related information. We used two Node module to achieve our goal. One is js-crawler.js , the other is cheerio.js. The former provides one API for crawling and returning HTML document while the second exposes APIs for manipulating DOM objects using JQuery on the server side. Equipped with these two modules, we are able to scrape Zillow website and get the raw textual information we want. Then we used regular expression to do some pattern matching in the text, constructed a MongoDB document schema and finally programmatically sent them into the remote MongoDB powered by mongolab ready for downstream usage in the front-end side. Because we are using the MVC model, our project is highly scalable in terms of data storage. We can provision more information when needed and store in the remote Mongo database server. The action of preparing information can be done in any way in any language that has a MongoDB driver. For any additional data-based functionalities, we only need to add corresponding query scripts to our app.js running on the server side, then using ajax calls to bind user-generated data request with the query.

7.3 Leap Motion

While the Leap Motion developer documentation is very helpful, it was very easy to get lost when initially getting the device to work with our application. All of our gesture-based functions had to be built custom to work with the features we wanted to control with our hands. For example, the device is very finicky in dim lighting so we had to mess around a lot with the “confidence” levels, which show how accurate the device is currently detecting hands. On top of this, we had to test how accurate we wanted the device to be because of human error. The device detects the smallest motions and initially it interpreted those as gestures that the user was not intentionally trying to make. To fix this problem with our zoom function, we had an initial check to see if the distance the hand moved was significant, and if it was then we created a smaller value corresponding to the distance moved to set the new zoom of the map (because Google Maps has preset zoom levels).

8. COLLABORATION

Within team The Bachelors, the majority of designing and styling was done by Leon(Shuangli) and Josh(Situo), back-end database was done by Mike(Huaxiao), front-end map was done by Deagan, and the Leap Motion was done by Aurnik.

Josh(Situo) Meng is a CogSci-HCI major. he had experience designing some simple systems and managing small teams. He added input in the function design and finished the low fidelity prototyping with Balsamiq Mockups after the team brainstorming. He also did notes taking to keep track of everyone's working progress. He is also in charge for result testing, documentation and presentation.

Leon(Shuangli) Zhou is a Computer Science major, however, he has designed his own iOS app before and has plenty of experience working with colors and stylings. He was in charge for most of the high fidelity prototyping using photoshop and Sketch, as well as front-end CSS/HTML.

Aurnik Islam is a Computer Science major. He was helping with the map and was in charge of the Leap Motion integration. Not unlike Deagan, Aurnik is a great programmer. He first came up with the innovative idea to split the data into night mode and day mode, which later became one of our best features, he then successfully integrated the Leap Motion with the whole system.

Deagan Monahan is a Computer Science major. He had experience developing Android app and was responsible for the development of the main map through the whole project. He was a smart and sophisticated programmer as well as a quick learner. He successfully integrated the surrounding buildings from Google Places API and traffic information from the Google Maps API into our web app.

Mike(Huaxiao) Chen is a Cognitive Science major and a data lover. He was responsible for most of the back-end data crawling and organizing, then parsing those data to Deagan and Aurnik. He was successful in getting data from the Zillow API and stored them into MongoDB.

9. CONCLUSION AND FUTURE POSSIBILITIES

9.1 Frequently visited places

The biggest limit for our system is that right now, it works only for UCSD. So the first improvement can be that giving the user a way to choose their frequently visited locations, and possibly more than one. We will then craw transportation information from google maps and analyze the recommended accommodations.

9.2 The better range

The time-to-school range we have right now is a static one. After the frequently visited places function has been added, we will definitely need to come up with a real time range. We will choose multiple points on the map and use the estimated time back from google maps to generate the new range, which will be more interactive and accurate.

9.3 The more powerful gesture

Although we have already done a great job integrating the Leap Motion into our web app, there are still more we can do. We will start by coming up with new features for more gestures, such as moving not only in the map, but the whole site. Simultaneously, we will explore more APIs of more gesture control devices such as Kinect and Myo. We believe that new ways of controlling will be a powerful way to boost our user experience.

9.4 Show more information

We have been working on lots of different information from Zillow as well. We will add a button to show the comments of a specific real-estate and probably the contact information of the house owner.