SDPlot by LifeFitness

HONGYU LI, COLLIN STYLES, HIEN HOANG, ELLIOT HUMPHREY, and ARDAVAN AMINI,
University of California, San Diego

Choosing where to live and work is an important and difficult decision requiring a person to consider a wide array of factors. Our application, SDPlot, aims to simplify this process by providing an easy use visualization tool that allows users to quickly compare neighborhoods in San Diego based on a set of risk factors such as rates of vehicular injuries, assaults, and homicides. We gather some basic data from the user and then show them visualization that display how likely they are to be affected by various risk factors based on their background (e.g., age, gender, etc.). Users can choose from a variety of plots and, from a single glance, can gather information on the quality of life of San Diego neighborhoods. We hope that our application will allow users to gather the data they need to make an informed decision on where to live and work.

Categories and Subject Descriptors: 1 [**Introduction**] Overview and description; 2 [**Motivation and Background**] Reasons for making this app; 3 [**Design**] Conceptual ideas and plans preceding development; 4 [**System Development**] Tools and technologies used, development timeline; 5 [**Gesture-Based Interaction**] Leap Motion design and implementation; 6 [**HCI Principles**] Usability choices and UI design; 7 [**Testing and Evaluation**] Testing methodology and process; 8 [**Collaboration**] Division of work, contributions from each member; 9 [**Conclusion and Future Work**] Final remarks, ideas for improvement

General Terms: Risk Factor Visualization

Additional Key Words and Phrases: NodeJS, visualization, San Diego, risk factors, leap motion, D3

ACM Reference Format:

Daniel Pineo, Colin Ware, and Sean Fogarty. 2010. Neural Modeling of Flow Rendering Effectiveness. 2, 3, Article 1 (May 2010), 20 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

SDPlot is an application that allows users to easily visualize trends across the San Diego area for risk factors such as vehicular injuries, assaults, and lung cancer. The application is designed such that a user can easily see which areas are more at risk for specified risk factors. The team also sought to make the application personal by visualizing the data based specifically on the input that users provide. The user can enter his or her age and gender to see if he or she is significantly more or less likely to be affected by various risk factors compared to other people from different backgrounds. It is the hope that the use of this application can guide users when making decisions such as where to live and work.

In this paper, the motivations behind this project and what drove its creation will be discussed. The initial ideas and prototypes will be presented, which will showcase the journey that led the group from its first concept to an finished web application. Also, the design process will be examined in depth, and what influenced the decisions throughout said process. In detail, the development process will be considered with all the tools, technologies, and methods used to implement them. By considering gesture-based interaction with the use of Leap Motion, many changes within the design and functionality of the site will be noted as well. This will all lead to the integration of human-computer interaction principles into the development process and how it affects the final product. Finally, testing will be discussed, which shines a light on functionality and a focus on minimalism that can help streamline the site to reach a finished product that is both useful and easy to use.

2. MOTIVATION AND BACKGROUND

2.1 Motivations

In day-to-day life, people do not realize how demographics relate to them. For example, when a news channel states that women are twice as likely to get assaulted in a certain area, they do not see the relationship between that and other locations. There is clearly a need to create something that will help people understand how different risks affect them in various locations based on them in a more personal sense (i.e. based on location, age, gender, and ethnicity). By creating SD Plot, there is an opportunity to bridge the gap between how people perceive risk and how it actually affects them. Perhaps, by using SD Plot, someone might find a much safer neighborhood to go out to for the day when considering factors like assault or car accidents. In a broader sense, it might be useful for lawmakers and citizens of various neighborhood or cities to know what kinds of issues affect their young, middle-aged, and older residents most. That way, more attention can be brought to important issues that otherwise might be put aside or deemed less important.

2.2 The Technical Problem

Although there is a lot of data pertaining to various injuries, health issues, and crime, they are never put together in a way that is easily accessible to people. Even when they are accessible, users cannot make heads or tails of the data they are looking at. Also, it is not an easy task to help people visualize so many different types of datasets (e.g. number of deaths, number of hospitalizations, number of occurrences, and so on). Most importantly, it is important to note that it may be difficult to help people visualize data for so many separate variables like age, race, and gender; many dealing with that much data would not even think to integrate that all with location data as well (in an easy to use user interface)! Interactivity is also vital in this sort of proposal because a user just sitting in front of a few graphs will not grasp the full importance of what is going on; however, adding some sort of interaction gives the user an ease-of-access to the data. Whereas SD Plot is put through a simple premise of , others may run into trouble thinking of ways to give the user direct control over the data in an effective way.

2.3 Previous Solutions

There is no direct competitor in terms of displaying and interacting with the data, but there are a few sites that display raw San Diego County such as the SD County web page; however, they only display raw data in a format that is hard for the average person to fully understand and make use of. Healthdata.org also tries to help people visualize data through the use of maps with color gradients, but there is no interactivity at all. For example, to see male vs. female, the user has to look at two separate graphs and look at small sections of it back and forth to distinguish any difference whereas SD Plot lets users switch immediately between the two (and since it is a single map, the changes are apparent every time a change is made).

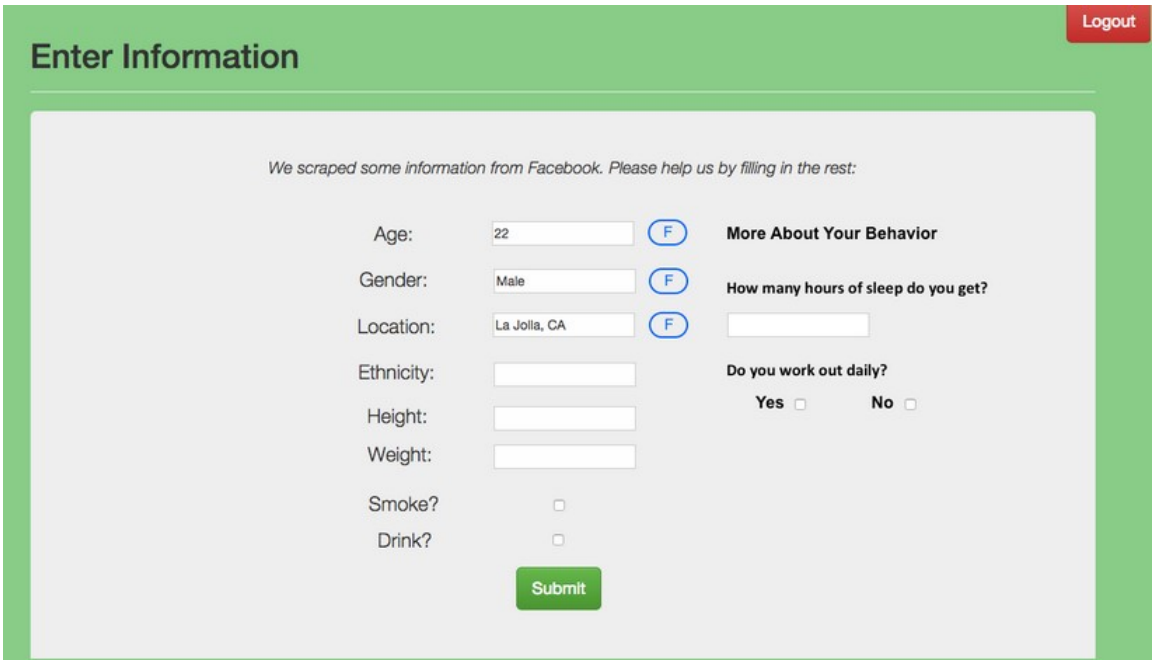
3. DESIGN

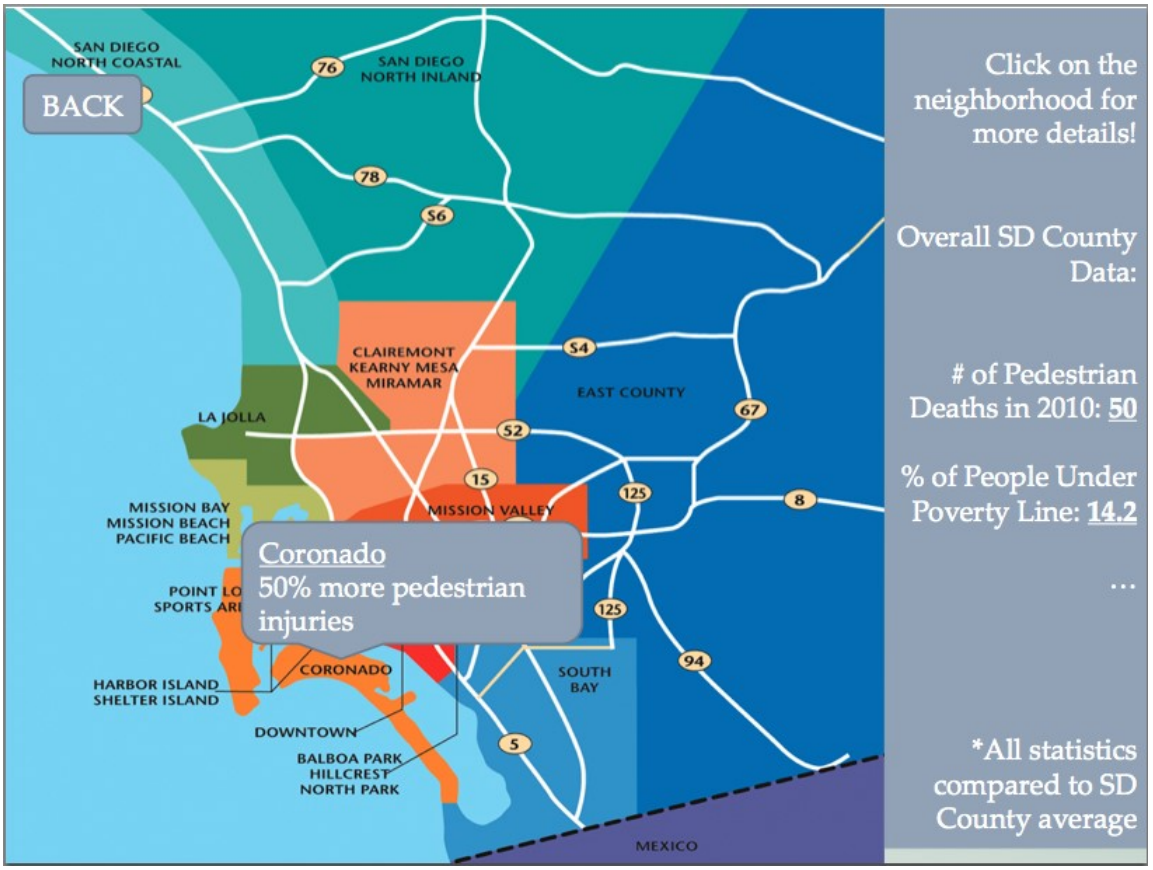
Our idea began with trying to think of a way to get user information regarding Body Mass Index, and visualize their risk of developing certain ailments based on this information. We thought about how weight could affect health negatively, and how we could encourage change or tips for users to improve their health. We also considered how behavior (like alcohol use and sleep) could affect their health. Ultimately, this was changed as there was no available BMI data to choose from, and in place, various datasets with a focus on age and gender were chosen. We were also interested in how data could differ across locations within San Diego County, and how people may be at different risks of ailments, disease or injury depending on their location, age and gender. We explored different ways of visualizing data

on a map, and having alternative visualizations to complement a map. This ended with using Google Maps, drawing a map of San Diego County on top, and having complementary visualizations in the form of d3 bar charts

3.1 Protoypes

Our prototypes were fairly quick and helpful. We would discuss ideas with each other, and either draw them out on paper or whiteboards. This gave us the flexibility in erasing and changing ideas within seconds on the page/whiteboard. By drawing them out, we could see the basic ideas we had and their feasibility, and helped us fine tune what we wanted. We would draw or write out what datasets we were interested in, what type of data we wanted from a user, and drawing the basic flow of our application. After deciding upon an idea for our app, we moved to wireframing. Both the asking of user data idea and the map visualization idea began out in the prototype phase before making it to the wireframing stages. Our wireframe prototypes helped us see how our paper/whiteboard prototypes would actually look as an application. We decided to wireframe the application flow, collection of user data (which changed to asking what data a user would like to see), and the map, providing us with potential UI content to incorporate into the final design.





3.2 Pros and Cons of Different Designs

Collection of User Data vs. Asking User to Choose Data. During development of our design, we changed from collection of user data to asking of data (reasons why are discussed later). Each of these has pros and cons. Collection of data from a user would allow us to create a personal experience and potential log changes of the data over time (e.g. how BMI changes for a user). The major drawback of this design is that it may not be entirely user friendly, as a user may feel uncomfortable providing this data in the interest of privacy. It also implies that a user can not easily change what they have submitted (as the visualization would be constrained using the data a user inputted). Asking of data to choose from also allows for a somewhat personal experience, but without violating a users privacy. A user can simply navigate between various selection fields (such as gender or age) without committing to their choices (as collection of data implies). This encourages a visualization that allows a user to change what they want to see. Minimal Map of San Diego County vs. Google Maps Visualization. During our design process, we actually implemented both a minimal svg map of San Diego County and a Google Maps Visualization. Having a minimal map can be beneficial because it only presents whatever data is attached with the map (as compared with having streets, freeways, and names everywhere on the map). However, a downside to this is that a user who is unfamiliar with San Diego County may have no bearings of what they are looking at. There is no context for a user to figure out what specific location they are looking at, other than a name that tells them that it is San Diego County. The Google

Maps visualization does have a potential issue of crowding the data we want to present to a user with the data Google Maps already have (like freeways, names, etc.). However, this issue is actually good because it provides locational context that allows users to see where exactly San Diego County is, which is especially helpful for users unfamiliar with the region.

3.3 How Design Idea Evolved Over Time

The initial idea began with considering how we could visualize data about anxiety data to a user. This quickly changed to wanting to visualize a users BMI and risk factors they may have of a result of their BMI. After brainstorming, we wanted to have additional user fields such as alcohol use, age, and gender that could give us further insights into risk factors for their health. This would be displayed as standard bar charts. After exploring possible data sets available, we shifted our idea to focus on various datasets that DELPHI provided. We did not want to be limited to a specific dataset, so we planned on having a visualization that would include and make use of many datasets. We also wondered how we could incorporate location into the visualization, and explored different possible map visualizations. We wanted a user to be able to see specific information for a location by clicking on a region of the map, and displaying more detailed information to complement this. For a substantial portion of the development, we were interested in how we could get a users information and use this to create a visualization specific to them. We wanted to provide a personal experience for our users, and considered the possibility of having users log their data over time to track this data. This would have allowed users to see their past history concerning their data, and encourage users to make changes to their lifestyle if appropriate. Additionally, had we gone this route, it would have been interesting to compare their data with other users, and and compare this with a dataset to see how people and data regarding people changes over time. Due to differences in datasets we ended up using, we decided to focus on how age, gender, and location affect the visualization of the data. We opted for users selecting a dataset, age, and gender from the first step of our app, and informing users of their risk of certain ailments, injury, or disease based on their location. We also decided on having a home screen that would capture a users interest, have user select a dataset, gender and age range and display a page that would include a map visualization, d3 bar graphs, and a dynamic navigation system to allow the user to change their dataset, gender and age to change the data being visualized.

3.4 Timeline

Week 6:. Brainstorm ideas, what type of data to get, how data can be visualized, quick prototypes, setting up version control, and setting up basic application in Node.js (routing, basic UI, basic flow of site)

Week 7:. Continue changing ideas, work on quick prototypes and wireframing, getting data from DELPHI, getting GeoJSON data of San Diego County

Week 8:. Begin exploration of Leap Motion, implementation of web application, SDPlot decided as name

Week 9:. Continue implementation of web application, finalize design decisions

Week 10:. Finalize application

3.5 Final Design

Above is the final design before implementing our application.

4. SYSTEM DEVELOPMENT

4.1 Architecture

Our app will be implemented using Node.js as a backend. All datasets from Delphi will be stored as JSON files at the server side. Users will access our app using any standard web browser, although our app will primarily target users on desktops (as opposed to mobile users). In order to see the visualized dataset, the user will be asked to input some basic information. Whenever the user selects a target dataset, the server will load the selected dataset from the local database, and send it to the client. Our app will be hosted on Heroku. This makes organizing the application server and the database server much simpler. The application server will contain all of the codes and resources related to the UI.

4.2 Technology Used

4.2.1 *Technologies*

Node.js. is a platform built on Chrome's JavaScript runtime. Its fast, easy and scalable network applications. D3 for data visualization

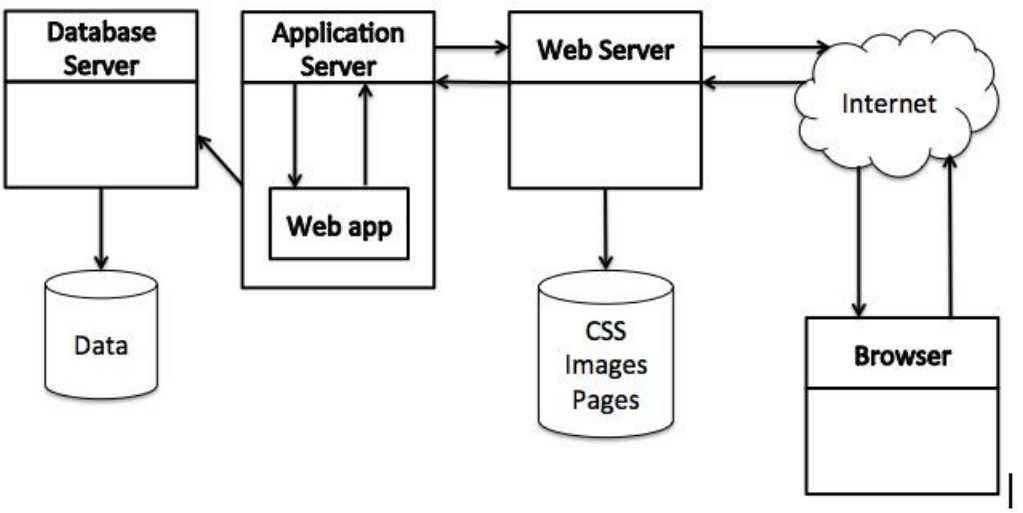
Google Maps API and GeoJSON. for Drawing San Diego Map on Google map.

Heroku. for deployment.

Bootstrap and CSS. for styling.

Leap Motion. for gesture control on Google map.

Postgresql. for accessing and managing datsets.



4.2.2 *APIs*

Google Maps API. One of our goals is to visualize the dataset by mapping them on to a map containing all regions of San Diego County. We used D3JS to draw the map at first; however, we later found that Google Map API fit our needs better. With Google Map API, we can enable the user to zoom in, zoom out, and move around on the map. Furthermore, we can interacted the Google Map API with Leap Motion so that the user will be able to do gesture control.

D3JS. We used D3JS to visualize the dataset we got from Delphi. Although we decided to use Google Map API to draw the map, D3JS is still very helpful for drawing the bar chart. With the bar chart using D3JS, the user will have a better understanding on how the data of a region is different from other regions. We also enable the interaction between the Google Map and the D3JS so that the user can more easily select and see the region that he is interested in.

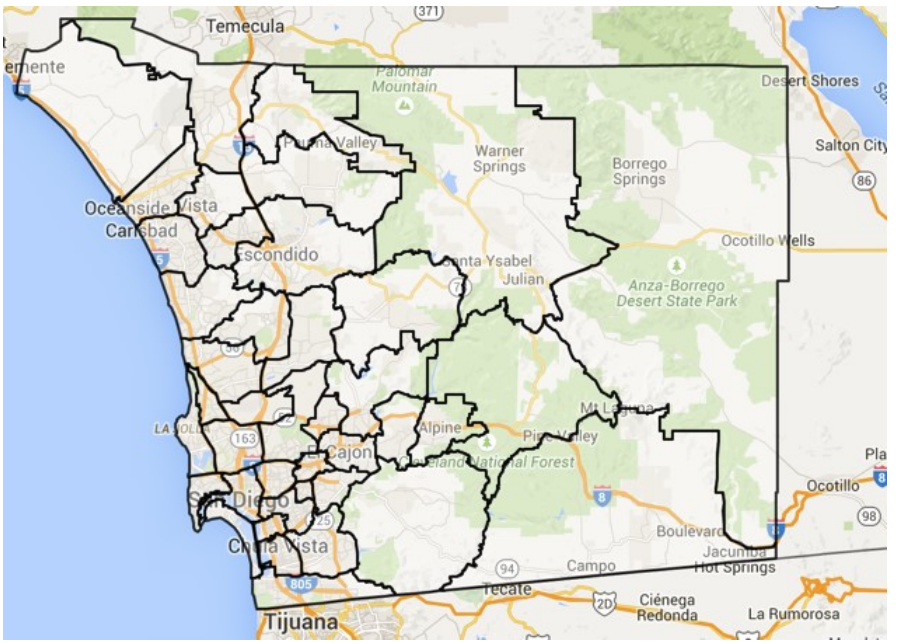
4.2.3 *Data Sources*

Delphi. We downloaded multiple datasets from Delphi using Postgresql, including: Vehicular Injuries (Alcohol Related), Vehicular Injuries (Total), Anxiety Disorder, Assaults, Asthma, Chronic Alcohol Related Disorder, Chronic Substance Related Disorder, Firearm Related Injuries, Homicide, Lung Cancer, and Self-Inflicted Injuries. Instead of picking one dataset, we let the user to decide which dataset he wants to see. We process all these datasets so that they are stored in similar format, and are fitted to the Google Map.

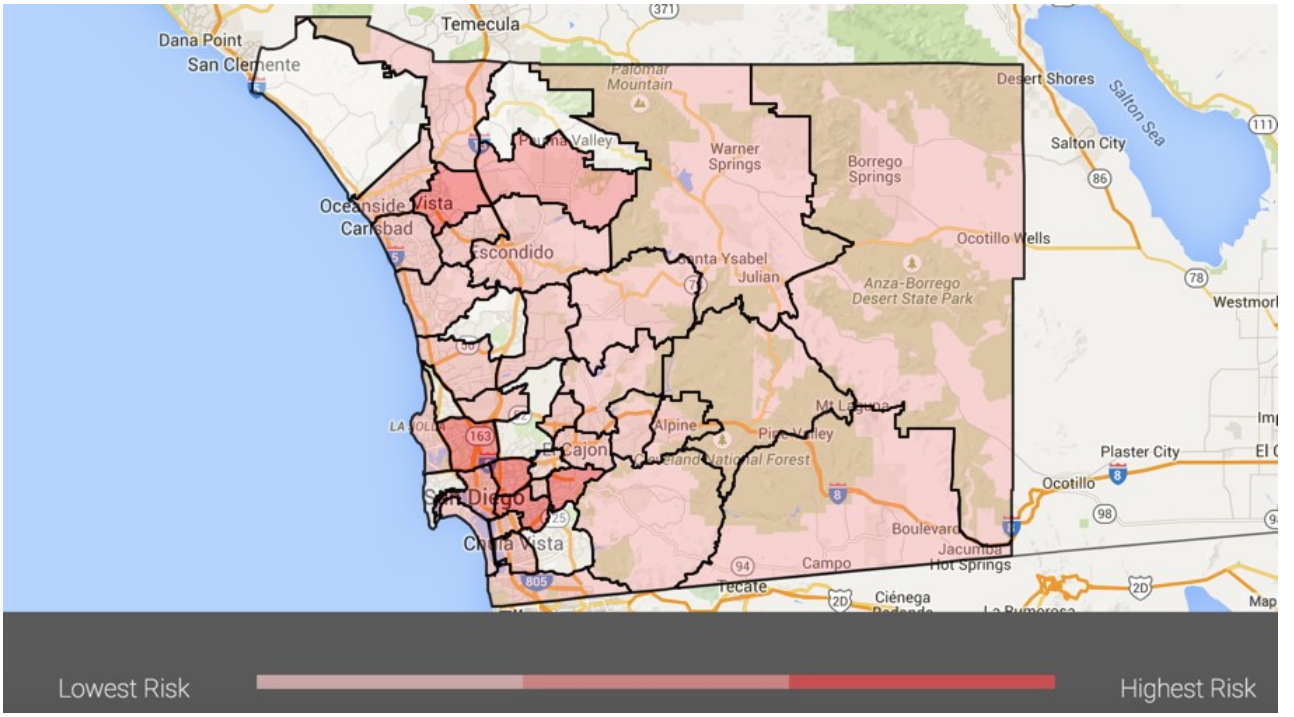
Subregional Areas from SANDAG. : In order to draw the map of San Diego County, we needed the boundary data of San Diego County. At beginning, we tried to use the boundary dataset from Zillow. However, it did not contain the whole area of San Diego County. Then we found the Subregional Areas dataset from SANDAG fit our need perfectly. With this GeoJSON dataset, we generate area polygons of every region in San Diego County on the Google Map.

4.3 Features

1. Enable the users to choose their neighbourhood in San Diego layout on top of Google map so that they can see the risks most near them.
a. We get the Subregional Areas dataset from SANDAG, make it as Geojson and draw the different areas of San Diego County shown on the map.
b. At first we thought about implementing two maps for the users to choose from. However, we thought that it might confuse users, so instead we just use one map which including the San Diego County on top of google map so user can get a more general idea about the areas and with Google Map API, so users can interact more with that by zooming and moving around with mouse or gesture control.

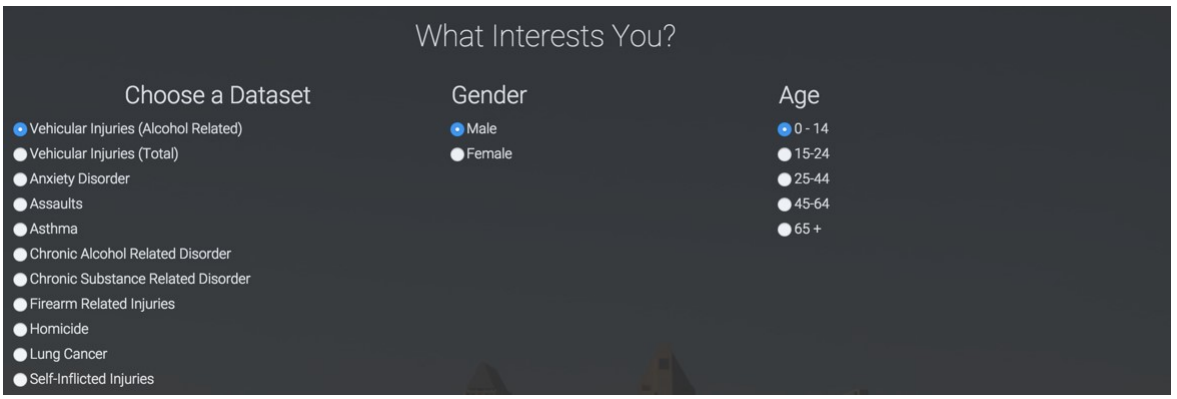


c. After we draw the different areas of San Diego County on the Google Map, we color these different areas with different colors based on the data of each area. The colors are from deep red to light red, representing the risk of this area. If an area is colored with a deeper color, it means the risk is higher in that area. With this feature, the user can get a better idea about the different risks in different areas.

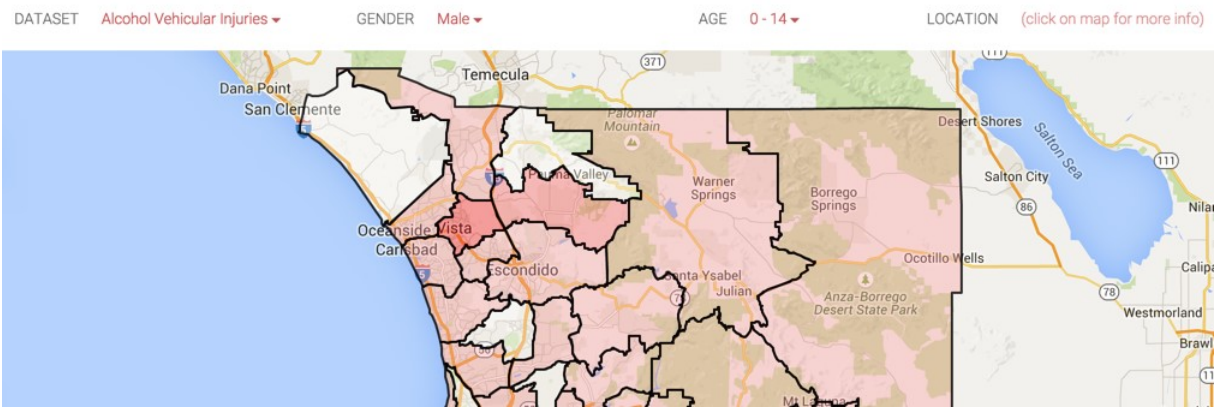


2. Enable the users to input their personal information, such as their age and gender, so that they can see the risks that are most relevant to them.

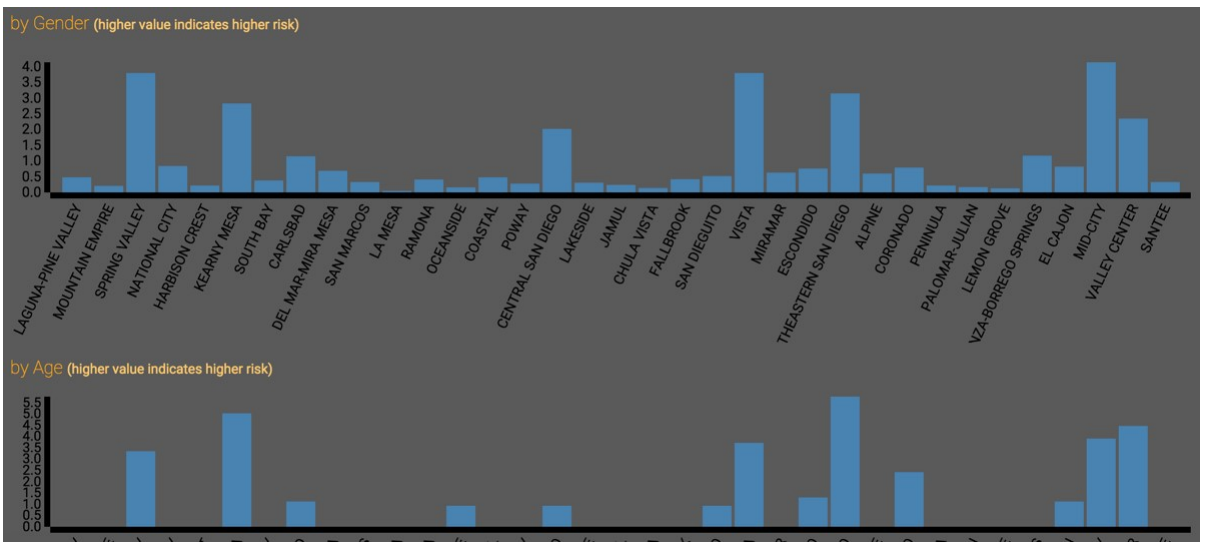
a. When the users go through the home page, we ask them to input their personal information. Then we display the most relevant dataset for them to see.



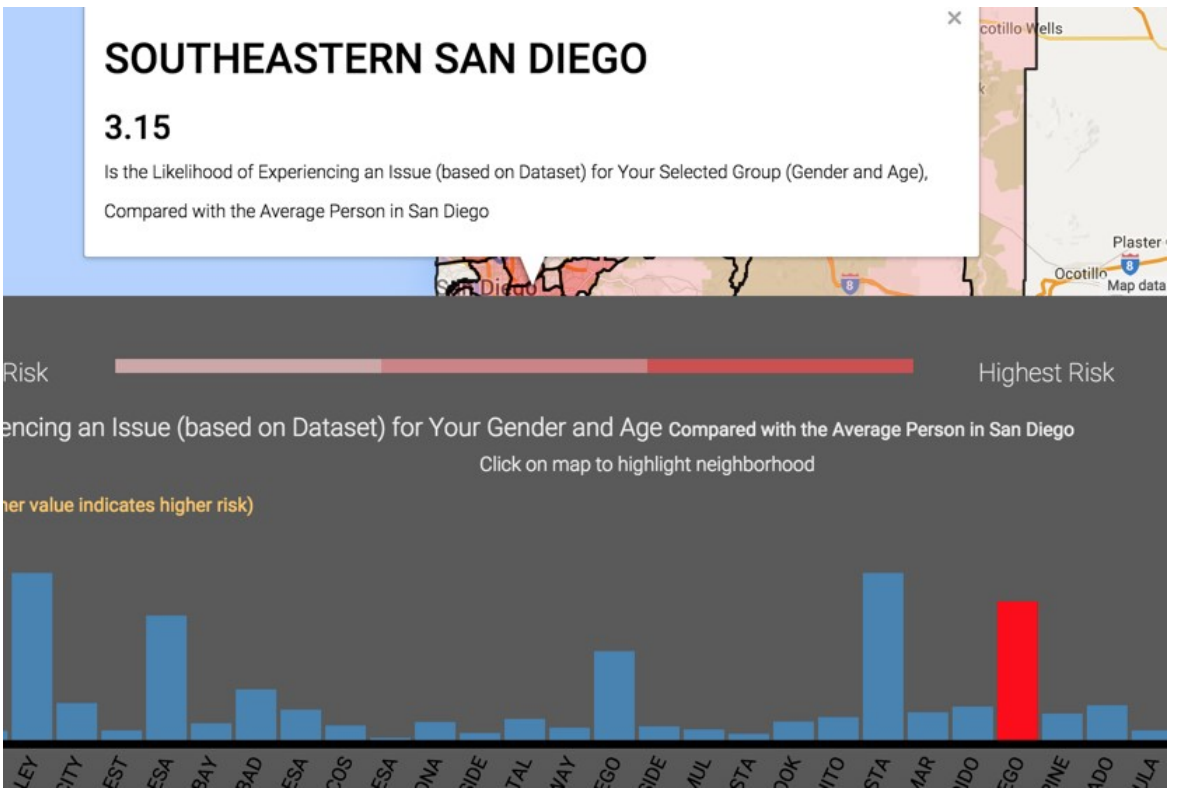
b. We also enable the users to interact with their input so that they can see how the risks they are facing may vary if their lifestyle is changed. While the users change their input, the web app will load the related dataset instantly.



3. Visualize data with D3JS and enable the users to interact between the map and the D3.
a. Using bar charts to visualize the specific dataset. Since the datasets are separated by gender and age, we display two bar charts for each of them. With this visualization, the user can get a better idea about the data.



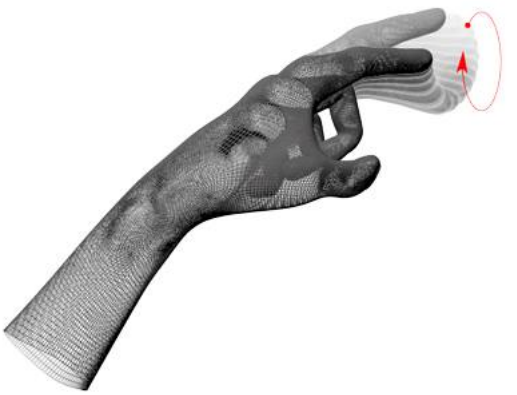
b. When the user clicks on a region on the Google Map, the selected region will be highlighted on the bar chart. In this way, it is much easier for the user to find the region he is interested in.



4. Using Leap Motion to enable the users to do gesture-based interactions.
We embedded Leap Motion to the Google Map API so that users can easily use their hands to control the map to zoom in and out, and move around on google map.

Circle - A single finger tracing a circle.
Swipe - A long, linear movement of a finger.
Screen Tap - A tapping movement by the finger as if tapping a vertical computer screen.
Hold hand: select the area or make a mark on that area

Draw a circle: zoom out the map



Screen tap: zoom in the map

Swipe up , down , left or right: move map to up , down left or right.

5. GESTURE-BASED INTERACTION

5.1 Introduction

In our modern world, gesture control is a feature that many people look for in an application. People always want a convenient way to control our devices wirelessly and gesture control is a potential solution to this problem. It can help users control applications without traditional interfaces such as keyboards and mice that can be limiting in some situations. As stated by Kurtenbach and Hulteen (1990): A gesture is a motion of the body that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on its way to hitting a key is neither observed nor significant. All that matters is which key was pressed. It is clear that gesturing contain a lot of potential information and the user can use gestures to control and give instructions to an application.

5.2 Appropriateness

Since the majority of the interaction and exploration of the data in our application takes place in a map visualization, we decided to limit the Leap Motion interaction to the map of San Diego County. Human gestures are very flexible, complex, and fast but with Leap Motion we have some limits concerning the rate of capturing all of those gestures. Therefore, the gestures implemented in our web app needed to be unique and simple to appeal to both natural human movement and the limits of Leap Motion.

Naturalness: Our application uses simple actions such as clicking on and moving the map around. As such a simple set of actions exists, all that is needed from a gesture interpreting device is recognition of simple movements. Leap Motion serves this purpose, and as such is appropriate for our app. We believe our system for gesture control is natural, as the user is given only seven different choices of gestures: hand up, hand down, hand left, hand right, circle with a pointing finger, pinching with a finger, and making a grabbing motion. Because of this limit of seven gestures, the user should have no trouble of remembering how to interact with the application as its within the limit of a humans working memory (which is about seven chunks of information). The four hand movement gestures should translate easily to the user because they mirror expected mental models: moving hand left would cause the user to expect the map to move left, which is how our app handles this gesture input; etc. The finger gestures are also extensions of common mental models; grabbing at something indicates a persons interest to highlight the receiving object of the point, and our app maps this to clicking on the map.

Adaptability: The gesture detection is adaptable to the human in the sense that any user with the ability to use their hands can interact with our app. Users should have no trouble learning the six gestures, as they are extensions of natural human movements and respective mental models (see previous section). Limitations come in the form of users who may have motor impairments concerning hand movements. Our gesture detection also allows users who may not have a mouse to interact with the visualization.

Coordination: Leap Motion provides a relatively quick response to user input. Its not perfect, but its good enough so that a user of our application can perform a gesture and react to the new perceptive stimulus received (the map updating in some way, depending on the gesture). There are a few issues in that occasionally a gesture causes a big change in the map (e.g. moving hand left sometimes cause map to move pretty far to the left), but these are issues that would be fine-tuned in the future.

5.3 Taxonomy

Since our web application uses Google Maps, all possible actions that can be used are moving map around the area (up, down, left and right), zoom in and out of a specific area, and select desired area. Based on those actions and gestures, we can come up with following implementation:

To move the map around the area, the natural and easy way is to let user swipe up, down, left and right. In this way, the map will follow the gesture that user makes; ”swipe to left” translates to ”map moves to the left”, ”swipe to right” translates to ”map moves to the right”, ”swipe upward” translates to ”map moves up”, ”swipe downward” translates to ”map moves downward”.

For zooming into an area, as user can pinch with their fingers. For zooming out, users can create a circle with their finger. These gestures coincide with natural intentions of a user (a user might pinch on something to draw attention to it (zoom it), and circling mimics a user wanting to reel-out or undo (zoom out)).

To select an area, users can grab at a region. Its similar to a mouse click and unique comparing to the other gestures.

5.4 Gesture Table

Move hand upwards	Map pan vertically up
Move hand downwards	Map pan vertically down
Swipe to left	Map pan horizontally left
Swipe to right	Map pan horizontally right
Pinch forward	Zoom in the map
Make a circle	Zoom out the map
Grab	Select area on the map

5.5 Device (Leap Motion)

The following are a few reasons why we decided to use the Leap Motion device over other interaction devices:

—Cool way in interaction with fun: Leap Motion give a modern interaction with computer in our web application. This interaction gives user a great to interact with web app comparing to our original way, a physical mouse, which is boring. This interaction catch users attention and user enjoy using interaction with our application.

—A convenient way in wireless interaction: Leap Motion gives user the great way to interact with application remotely with gestures. This is very convenient in application such as presentation and medical field where user can visually see do surgery in simulation with their hands.

—Affordable device and Easiness: The Leap Motion give us the library, so its easy to implement in our web application. Also, the device is compact and user can easily buy it for less than $100.

Beside those good reasons, there are some limitations in leap motion:

—Instability: Because human gesture controls are complex and fast, so the device will not catch all of the gestures. This system will be unresponsive or unstable if users gestures are so fast or complicated in interaction.

—Tiresome: When using leap motion, the users have to hold hands in the air and if its a long interaction, itll makes user feel weary.

5.6 Testing & Evaluation

Before giving to user our web app to test we did try to fix as many errors as we possibly could to give user a great impression as the first time when they use it. However, our implementation in our web app was not as good as we expected. Because of time limit, I have to give acceptable version to my friend to see his feedback.

When the first time i showed to my friend how to use gesture control to control our webapp, he was like wow, he thought about future thing like in the hollywood movie. When i let him test our webapp and with Leap Motion, he was in awe. This was same feeling that i had when first time i saw this device.

when i showed him how he could use leap motion to control google map, he really liked it. He said that this could be used in some places like bus stop, so people can use it to check traffic, bus stop etc. while waiting for bus. This was really good idea actually, i never thought about it. This idea is realistic, helpful and convenient to user.

Also, there was some negative feedback which i think that was true. He said that this device was not stable (it was because our implementation was not good and users gesture was fast and flexible) and he felt sort of weary after keeping hands up in the air for awhile (We feel same thing when debugging it for awhile).

To implement leap motion in our web app, we have to use its library for node js and install npm. So all of that library located in public/leapmotion folder and all of codes are used placed in file googlemaps.js from line 518 - 619. Inside of that code, we have each function responsible for each gesture base:

—Draw a circle : zoom out the map (line 524 -531)

—Screen tap : zoom in the map (line 537 -543)

—Swipe up, down, left, right : move the map to left, right , down and up (line 545 -593)

—hold hand: select the region (line 596 -643)

6. HUMAN-COMPUTER INTERACTION PRINCIPLES

6.0.1 *General UI Implementation.* The primary philosophy that we adopted while developing our application was KISS: Keep it Simple, Stupid. While our application deals with a lot of data and the user has many possible choices when deciding which data to visualize, we wanted to make this process as simple as possible so that there was little room for error or confusion. Perhaps one of the most apparent design choices we made as a result of adopting this philosophy was to make the user interface as clean as possible. We went for a minimalist design and actively tested to ensure that only the most important and absolutely necessary elements were displayed. We believed that keeping the interface minimal makes the app very easy to use.

Our application has only two pages so the user will not ever feel lost when trying to navigate the site, as is often the case with modern websites. Furthermore, these two pages were created such that, even though they have the power to display a lot of data, all of the potential is organized into an easy to use navigation bar. This has the very desirable effect of making all of the functionality easy to both find and use.

6.0.2 *HCI Principles.* We keep the status of our system visible to users at all times, to prevent any confusion. When the user first enters the site, they land at a welcome page, which asks the user if they want to get started with the application. After hitting a ”Get Started” button, the page scrolls down to a form consisting of radio buttons. This form tells the user to pick what interests them, and clearly labels the different options (Datasets, Gender, Age). After submitting, whatever parameters the user picked are preloaded into a navbar that informs the user which dataset they picked, and what

gender and age they decided on. When a user changes these fields, the system updates the names of the selection to inform the user of the new parameters they have chosen. This navbar is fixed on the top of the page so that when a user scrolls, the system will keep them informed of what data they are viewing. It also serves a purpose of letting the user know what the last location on the map the clicked on which is helpful when a specific bar on d3 visualization is highlighted (related to location).

The user is granted freedom of control in our application. From start to finish, a user can pick whatever data and parameters they are intersecting in exploring. The application never forces a decision without the user's input. The user is also free to navigate the visualization as they please.

Our application has been designed carefully to prevent any errors from happening. One specific example of this is on the home page. In the bottom of the page where the application asks for a user's input, we decided to use radio button inputs. This tells the user that in each category to choose from (dataset, gender, and age), he or she can only select one value. Compare this to using checkboxes, which would allow mutiple selections. By restricting the user to one choice per category, we ensure our system won't receive any invalid input. Additionally, the form is preloaded with some selections just to make sure the user doesn't submit a blank form, which would break our app. Another example of this is towards the end of our application development, we discovered a bug that if a user directly goes to the googlemaps url (which they shouldn't be able to do because it requires data be posted to a server first), the app would render "cannot GET /googlemaps". To prevent users from replicating this error, in our system routing, we simply reroute a user attempting to directly access the googlemaps url to the hompage. This ensures a user must select some data before they can view the visualization.

SDPlot also has considered the use of recognition rather than recall for users. Between the home page and the visualization page, there is a page reload. To make sure the user didn't forget what they chose, and would thus be confused about what they were looking at, we update the navbar on the visualization page to reflect whatever input they chose from the home page. This eliminates any user confusion about what data they are looking at. Additionally, we do not require the user to recall if they decide to change what data they are viewing; the navbar updates to reflect and user input changes.

During design and development, we made sure to use language that would be familiar with users. Throughout the application, the word risk appears. The home page and visualization page both explain what we mean by risk (that risk is an approximation of how likely a certain person of gender and age is to experience an issue (such as ailment or disease) based on the dataset chosen and their location. We made sure to eliminate any technical words that may have appeared on the UI for our own understanding during development.

6.0.3 *Features and their Respective HCI Principles.* In the Features subsection of the System Architecture section, we described various features of our application.

1. Enable the users to choose their neighbourhood in San Diego laid out on top of the Google map so that they can see the risks most near them. SDPlot informs the user of what neighbourhood they click on via various means to keep the user informed of the system status. When a user clicks on a neighborhood, a pop-up on the map shows the neighbourhood name along with the computed risk factor for the area. This pop occus where they clicked so there's no confusion. The navbar also updates with this information, so that when a user scrolls down the page (and thus the pop-up is closed), they will know what neighbourhood they clicked on. To be even more confident that the user is aware of the neighborhood they clicked on, the bar that corresponds with the clicked-neighborhood is highlighted. The map also indicates risk via a color palette of red. Deeper red means a higher risk for the area, and lighter red means less risk. Based on last minute feedback we incorporated a legend explaining this so that a user is not questioning what shade of red means what.

2. Enable the users to input their personal information, such as their age and gender, so that they can see the risks that are most relevant to them. We wanted our app to allow user choices and user freedom when experiencing a visualization. We did not want our app to only focus on one dataset, with one particular population because we realized the diversity of users that exist. Because of this realization, by giving the users a choice in what data they can visualize, we believe users will be encouraged to explore data that is relevant to their own person (such as their gender and age) as well as other groups.

3. Visualize the data with D3JS and enable the users to interact between the map and the D3. By giving the users an opportunity to interact with our data, users will feel excited that there actions can affect the visualization. The alternative to this is to display a static image of a visualization, but this does not encourage user exploration. It is even possible that a static image of a visualization would cause frustration, for example a user rapidly clicking on the image because they expect some sort of interaction. By formatting a lot of data as a D3JS bar chart, we give clear visual distinctions between the neighborhoods and their data, so a user can process the information effectively and visually. An alternative would be to simply display a table of this information, but this is not very user accessible as it requires more thought and processing on the user's end.

4. Using Leap Motion to enable the users to do gesture-based interactions. One idea behind the incorporation of a gesture-based component of the application was to allow users to be flexible in how they explored the data. The app allows for the traditional keyboard and mouse interaction, and with Leap Motion a user has an alternative means to interact with the data. Also, this gesture-based component takes advantage of a human's natural movements and mental models. The gestures allowed are simple: move hand left, up, right and down, and grab, pinch, and circle with the fingers. These are all movements that are not foreign to typical hand and finger movements. The mental models associated with these chosen gestures map efficiently to updates within the system. For example, when a user moves their hand left in general, they usually mean to go left, or that there is something of importance on the left of their person. This mental model corresponds to our application in that when a user moves their hand to the left, the system updates by scrolling the map visualization to the left.

7. TESTING

Our testing was performed with a style similar to that of the Agile software development philosophy. While we did not use Agile for actual development, we followed its general principles throughout the duration of the testing of our application. Specifically, we used an iterative model in which we implemented small, atomic features individually and tested them. In contrast to an approach such as Waterfall, where everything is planned well in advance and then implemented in one long stretch, we would implement a feature and test it immediately. This allowed us to quickly determine which parts of our system were working and which needed work. It also made it apparent when a new feature would break pre-existing functionality.

This approach proved useful as our application has a relatively small core which we later scaled up. The basic functionality is the same for all data sets. Only the results that are shown on screen (i.e., the visualizations) are different. Because of this, we decided it would be best to get the basic features working for a single data set. We went through multiple cycle of adding new features to the core and testing them. After the results of the testing were satisfactory, we would scale the app up by adding new databases or new user filters (e.g., age, gender, race, etc.) and then test on the larger, more feature-rich version of the application. By testing early on, we were able to avoid headaches later and this approach allowed us to expedite both the development and testing phases of production.

Most of our testing was concentrated on ensuring that the features that we outlined in the planning phase were functioning properly. We tested to ensure that whatever data the user specified, the proper visualizations would be displayed. This involved checking that our server passed, received, and handled POST data correctly. Furthermore, we tested that the user could change the parameters at any point and that doing so would both update the visualization in real-time and that the new data would always be correct. The bulk of our testing was directed at the visualizations themselves as they make up the core functionality of our application. A lot of testing was necessary to make sure that the map was working properly, especially that it was showing the correct boundaries for neighborhoods and that the colors displayed matched the data. Also important was the testing of the interactivity of the map, such as the user's ability to click on an area and to have the corresponding data in the bar chart be highlighted for easy locating. Finally, we also extensively tested the application's usability. We constantly tried to reduce clutter by reflecting on which aspects of the UI were truly necessary and which could be removed. This allowed us to reduce the number of distractions for the user, making it easier for them to get started and see the data they wanted.

One shortcoming of our rapid, Agile approach was that we did not have the time in our short schedule to have the application tested by people outside of the development team. We recognized the importance of having users unfamiliar with our application attempt to use it. It would have been incredibly useful to have feedback from users, whether that be explicit feedback gained from asking them what they liked and did not like or less obvious feedback gained from observing the user and seeing where they had difficulty navigating the application. If we were to continue development, one of the first items to accomplish moving forward would be to have a large group of people test the application so we can gather data on what we need to improve in future versions.

8. COLLABORATION

Collin was responsible for a number of back end tasks. He primarily worked with the data acquired from the Delphi database. He worked with PostgreSQL to obtain the data and used Python to process the data. The raw data was riddled with issues such as data being reported as "<5" and the use of multiple symbols to report data that was essentially null. Therefore, Collin spent a good deal of time reading in data and applying filters to correctly deal with these issues. Furthermore, our map used different divisions than the data provided by Delphi. For example, there are two separate locations titled "Coastal" and "Coronado" with different data but both of those areas refer to the area "Coronado" in our map application. Collin worked to ensure that the data was properly mapped between these two systems so that the map visualization would be accurate. Finally, he worked with the front end developers to ensure that the data he was providing was formatted in such a way that it could be easily handled by the application itself.

Hongyu was working on the data visualization and data loading: implement the the map using Google Map API; draw the polygons for different regions on the map with the GeoJSON; load the dataset to the map and color different regions with different colors based on the dataset; revise the D3JS bar chart so that it shows the region name on the x-axis; enable the interaction between the Google Map and the D3; implement the function of loading different dataset dynamically when the user changes his input.

Elliot: He worked on the front end of our webpage. He designed our web app, made sure it look nice, modern, and easy to users based on HCI principles. He also collaborated with Hongyu to create a San Diego Map Geojson to draw San Diego map on top of google map.

Hien Hoang suggested ideas for the project, created prototype, visualized what web page should be looked like and the flow of the web app. Also, he worked on gesture control and implemented it on google map and colaborate with Elliot and Chris to finish front end design and back-end to change

data dynamically. He planned schedule meeting, works for team members, and the timeline for the team to make sure the project was completed on time.

Ardavan: He worked on data analysis. He decided which data we should use, how to analyze it, and how it should be displayed to users. Also, he worked on the wireframe, planning the ideas behind the app, changing the prototypes, and ensuring that we have a great and simple design UI.

9. CONCLUSION AND FUTURE WORK

9.1 Conclusion

To close, we believe that we have created an application that provides a very useful service to those looking to explore trends related to various risk factors in the San Diego area. We began with a radically different idea of helping users track their health over time and ended up making something completely different. Nevertheless, we believe our product is both informative and easy to use. We've come a long way over the past several weeks and have faced many challenges but in the end we were able to come up with a website that provides a useful solution to a problem that many users may need help solving.

Throughout the development of SDPlot, our members have learned new skills, techniques, and knowledge. Prototyping has convinced some of us that design does not have to be final from the start; in fact, it is often better to have no idea of the final design to keep the ideas flowing and creativity brewing. Wireframing has provided a new platform of sharing design ideas with others, in a method that encourages feedback from other members on a project. Heuristics and HCI Principles have guided our members in creating a useful and approachable application for our users.

We believe SDPlot will be useful for a variety of users. Our friendly UI allows casual users to explore the data to their heart's content. By using the app, people have the opportunity to see how everything relates to each other. Our powerful backend provides flexibility of adding new datasets into the visualization by simply adding a few lines of code. The detail of the visualization encourages professionals to gather new insights from the data presented. This could be used in many applications from local government to even medical analytics.

9.2 Future Work

There is still a great deal of work that could potentially be done to improve our application. We only had a few weeks to completely plan, design, implement, and test our application. We feel that, with more time, we could flesh out our ideas and expand the functionality. While the constraints of the data available via Delphi mean that our original idea of tracking user progress over time is likely still too lofty, we think that Delphi still contains the potential for additional functionality. Specifically, we could add more data set from Delphi to allow users to explore more trends than are currently offered. Delphi has many more data sets that we considered adding but ultimately decided against due to time constraints. We specifically designed our application to handle Delphi data as generally as possible. Our system allows to easily plug in new data, provided it is formatted similarly. Unfortunately, not all of the data available from Delphi is formatted exactly the same. However, we have already developed techniques for dealing with data inconsistencies so acquiring and formatting new data shouldn't be prohibitively difficult.

Another area for improvement is user testing. Again due to time constraints, we were unable to have our application tested by users outside of our development team. Our familiarity with the system blinds us from seeing the issues with the interface. Therefore we believe it is imperative that we gather data from other users, especially those who aren't as technologically savvy as those of us who developed

the application. We would like to observe people using our application to see where they have difficulty. This will allow us to better focus our efforts in making the user interface simple and easy to use.

On that note, refining the user interface is a continuous effort that we would like to continue. In a sense, making a good user interface is not a goal that is achieved but rather a philosophy that one has to adopt and maintain throughout development. The requirements for what constitutes an ”easy to use” interface are constantly changing as users become familiar with new trends. As the environment that web applications live in rapidly changes, we need to constantly update our application to meet new requirements. Again, giving our application to users and noting their behavior will go a long way towards ensuring that our application is as intuitive as possible.

10. BIBLIOGRAPHY

Nielsen, Jakob. ”10 Heuristics for User Interface Design: Article by Jakob Nielsen.” Nielsen Norman Group. N.p., 1 Jan. 1995. Web. 7 June 2015.

Karam, Maria and schraefel, m. c. (2005) A Taxonomy of Gestures in Human Computer Interactions. Bill Buxton, Gesture-based Interactions (Ch 14), 2011

Designing the User Interface: Strategies for Effective Human-Computer Interaction (5th Edition) by Shneiderman, Ben, Plaisant, Catherine, Cohen, Maxine, Jacobs (2009)

Design for Information: An Introduction to the Histories, Theories, and Best Practices Behind Effective Information Visualizations by Meirelles, Isabel (2013) Paperback

Designing with the Mind in Mind, Second Edition: Simple Guide to Understanding User Interface Design Guidelines 2nd (second) by Johnson, Jeff (2014)

Natural User Interfaces Are Not Natural. Interactions - Norman - 2010

Received June 2015; revised June 2015; accepted June 2015

Online Appendix to: SDPlot by LifeFitness

HONGYU LI, COLLIN STYLES, HIEN HOANG, ELLIOT HUMPHREY, and ARDAVAN AMINI,
University of California, San Diego

A. EXECUTIVE SUMMARY

A.1 Overview

SDPlot is an application that visualizes the rates for various risk factors across the San Diego area. It is designed to allow people to quickly see which areas are more or less likely for a particular risk factor. Furthermore, it uses the user's age and gender to show data for that particular subset of people. In this way, the experience is made more personal. We hope that such an application can aid people when they are making important life decision such as where to live and work.

A.2 Design

Our design has evolved over the development of our application. We began with interest in BMI and collection of a user's data, and shifted to a San Diego County Map visualization of risk factors based on age, gender and location. Several design decisions were made after considering the pros and cons of each. From prototyping to final design, we explored countless ideas and believe we have reach an excellent final design.

A.3 Implementation

The application is built using Node.js as the back end and is hosted on Heroku. We used PostgreSQL to acquire data from the Delphi databases but, because the data is static and does not vary on a per-user basis, we store it locally so that we don't have to request it every time a new user connects. We made heavy use of JavaScript for the front end, specifically using the D3 library to create the bar chart visualizations that present the data. We also used the Google Maps API to display a map of the San Diego area and to overlay a neighborhood division layer. This top layer uses colors to show which areas have higher rates for a particular risk factor. A convenient navigation bar is provided that allows users to quickly and easily switch between data sets or to see data for people of different backgrounds.

A.4 Key Features

The application's main feature is the map which shows data in an easily digestible format. Users can select which data set they want to view and it will be displayed on the map using a color scheme that makes it easy to see which areas are more at risk for whichever risk factor the user chose. In addition to the map, the application displays bar charts so that the user can view the data in a more traditional format. The bar charts also allow the user to get raw data such as how many more times likely a particular area is to have, say, assaults, than the average. We also provide the user with the ability to specify their age and gender to see data that is more personalized to them. And if they desire to see data for people of different backgrounds, they can easily switch between the data sets in the same way that they switch between risk factors.

DOI : http://dx.doi.org/10.1145/0000000.0000000

A.5 Evaluation

We mainly evaluated our application iteratively, meaning that we would implement a single feature and then test that feature extensively to make sure it worked properly before moving on to implement another feature. We adopted this approach from the popular Agile software development methodology. This process allowed to easily spot correctness errors. It also helped us with usability as we could easily see when a new feature modified the site in a way that made it difficult to traverse.

A.6 Future Directions

Going forward, there is still a lot of room for the application to improve. One easy way to expand the app would be to increase the number of available data sets. The application is designed to handle the data from Delphi as generally as possible so inserting new data sets is fairly painless. In the same vein, we would like to add other user filters such as race and socio-economic class. Another important area for improvement is usability. This is a constantly changing requirement so we would like to test the app on a wide user base to see which areas of the site are the hardest to use. This will allow us to more efficiently dedicate effort to the areas of the site that need the most work. Finally, we would also like to integrate our application with Facebook. We could use data from Facebook to compare how a user's neighborhood compares with those of his or her friends.