**Improve the WiFi Coverage and Quality in Cal State LA**

**V1.0 Report**

**Prepared by Steven Castro, Daniel Xu, Jorge Lima, Paul French, Albert Ting**

**Liasons: Jason Solis, Edmund Gean, Michael Lee**

**Advisor: Huiping Guo**

**CSULA**

**5-10-2018**

**Table of Contents**Introduction………………………………………………………………………………………………………………… ..(Pg.3)  
Related Works and Technologies……………………………………………………………………………………(Pg.4)  
System Architecture……………………………………………………………………………………………………….(Pg.5)  
Results and Conclusions…………………………………………………………………………………………………(Pg.10  
References…………………………………………………………………………………………………………………….(Pg.12)

**Introduction**

Wifi at CSULA is definitely an important thing for students and faculty alike. Whether it is used for entertainment, studying or work, Wifi is paramount to these things and without it we would be severely hindered in our everyday activities. The current Wifi infrastructure at CSULA is good, but however it is not without faults. There are some times where the Wifi is spotty and sometimes where the Wifi is unbearable to work with.

At CSULA, there was no current way to easily report these types of problems. The only solution that seemed plausible was to call up IT to fix the problem and if there was multiple calls the service would be slower due to a lack of data. To help alleviate this issue of a lack of data of where Wifi is spotty, we developed an enhancement to the CSULA GETmobile application on both Android and IOS. This enhancement was a new webpage for the users to access in order to report a WiFi outage. This application, in addition to the user self-reported data and comments, would also grab other statistics from the report and sort it into a database that could be viewed in many different ways for the purpose of improving the Wifi infrastructure over time through giving the ITS department a tool to get this data.

Because the main hindrance for improving the Wifi infrastructure is a lack of data, the most logical thing was to create an application to help get this data. However, to develop an application that would be used by the public, there was precautions that we had to take in order to protect our database and the application. Some of these steps were to keep the database and the webserver on different servers. In addition we had to also protect against any forms of SQL injection incase a malicious user wanted to drop all our backend database tables. In addition to this, we also decided to keep the application web-based instead of developing two native applications for two big reasons. These reasons were that we would be able to easily implement the webpage into the CSULA GETmobile app and that we would be able to not worry about updating and developing two versions of the same application. In addition to developing these webpages, we also needed to test to see how much simultaneous users we would be getting in case there was a time when we had over 100 users on the application at once.

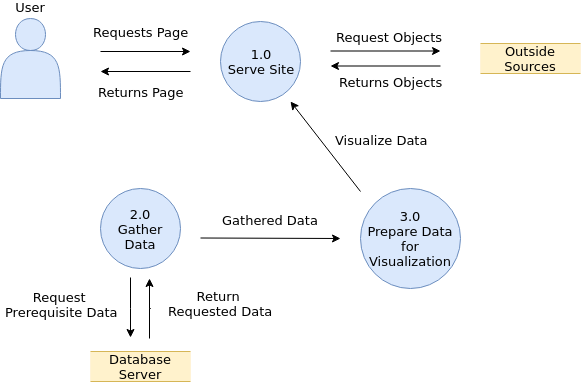
**Related Works and Technologies**

The Wi-Fi Reporting Application is created mostly with PHP and Javascript languages. PHP is used to push and pull data from our PostgreSQL database server. Javascript is used to pull data from our users’ devices and utilize our 3 main APIs: HTML5 Geolocation Services, Mapbox, and Highcharts. HTML5 Geolocation Services API allows our team to locate the user’s device when generating a report. Mapbox works together with the HTML5 Geolocation Services to visualize the GPS data, rendering the user’s location on a map, as well as rendering various metrics like report density and the real time Wi-Fi status of buildings on campus in our dashboard view. Highcharts is mainly utilized in the dashboard view of our project, helping ITS to visualize all the data from the reports in easy to understand charts and graphs. On the server side, we are using Apache as our web server, PostgreSQL for our database server, and Memcached to reduce load on our servers when there are users viewing the dashboard views.

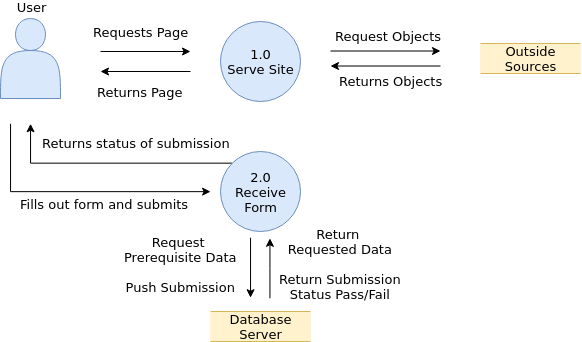
**System Architecture**

System Architecture

DFD Level 0 for Dashboard



DFD Level 0 for Mobile View



Mobile View

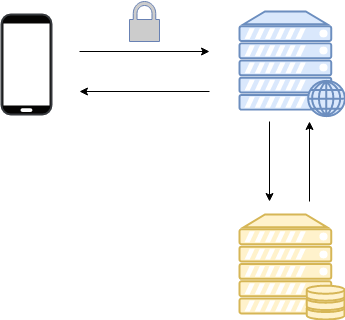
Straight-forward serving of the webapp, user requests report page, web server serves the PHP file. once the user has the page it then will try to get geolocation with the relevant HTML 5 APIs. If it succeeds then the map provided by Mapbox’s API will then call a JavaScript function that calls on an API function that zooms in on provided coordinates, and we then draw objects if applicable (marker to signify location, average rating if found on those coordinates signified by draw a rectangle with the appropriate color.) Then the user will fill out the report, and our hidden form fields will gather additional information and pass them on to a PHP file (addreport.php) that is setup with prepared statements filled out ahead of time to insert the row to our Reports table.

Those prepared statements will only continue if the report’s geolocation falls in our desired range, if it doesn’t it will return a failure status to the user and redirect them back. if it continues, it will add the grid coordinate to the report row, and then check if it falls under any building coordinate range, and if it does filling out that column in the report, if not then giving it a default ‘Outdoor’ value. Then it will execute the final insert, giving the status back to the user whether it succeeded or failed and return them back to our landing page.

Dashboard

The dash is a one way relationship, in that it won’t get any data from the user, it will just gather all the data we’ve already set it up to do. All the queries have been made ahead of time (~10 in total) and the PHP helper file (dashboard-service.php) will query them (either again from the database server or from the cache we’ve set up,) and the rest of our dashboard files (dashboard.php, dashboard.js) will do their appropriate job to help the user visualize the data. This means setting up the rows to html tables styled for better viewing, a map that helps pinpoint the reports in a manner of ways, tabs to help categorize the data, and other functions.

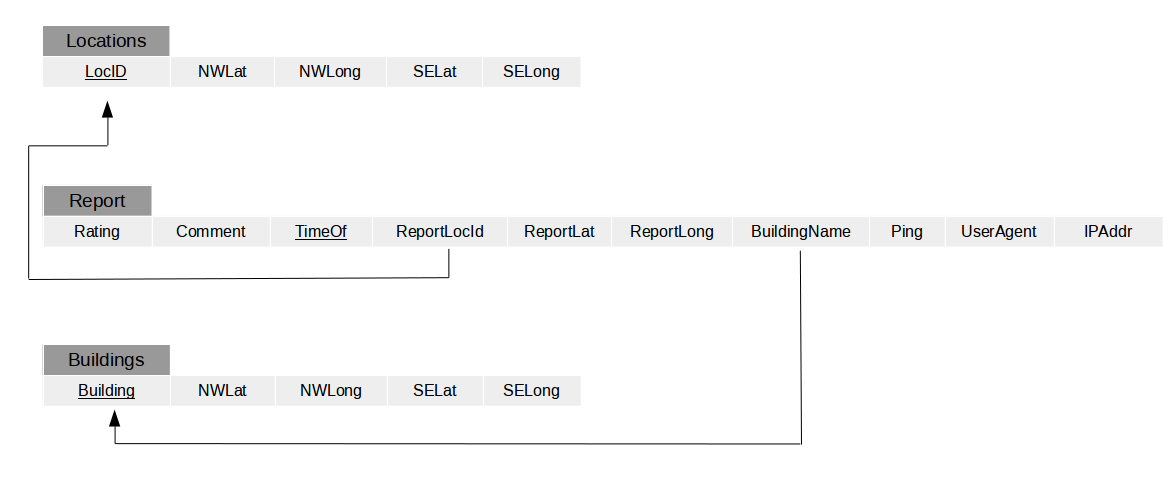
Simplified relationship between servers



Software

Our setup is two servers, one acting as the database server solely doing the job of managing and hosting the database and nothing else. It runs Ubuntu Linux 16.04, along with PostgreSQL acting as our management system. It has the appropriate ports open so we can communicate between servers. The other server is our web server running Ubuntu Linux 16.04 as well, with Apache2 to serve the pages to users, with Memcached to cache any rows we can to help alleviate load on the database server. The pages include PHP code, so we also have that package installed along with libraries to be able to link up with the PostgreSQL and Memcached. It also has the appropriate ports open and an HTTPS cert setup to allow for secure serving of the website to users.

Database Schema



Database

Three tables with Report housing individual reports. Locations and Buildings filled with hard-coded coordinates of our grid and building layers respectively. Whenever there is an insert of a report, we first check if it falls inside the Locations outer layer. If it does we continue to the second check if it falls under a building, if so we set the BuildingName accordingly. This categorizes our main data point meant to help WiFi around campus. We then visualize these categorizes in many ways to the user (heatmap, average per building, etc)

Layers VisualizedRed - Location Layer, uniform segments, meant to check if report falls in range

Blue - Building Layer, non-uniform segments, meant to check if report falls inside any building on campus

**Results and Conclusions**

Improve the Wi-Fi Coverage and Quality in Cal State LA was successfully realized into a mobile web application. This web application, with the help of the ITS department, is currently being implemented into the getMobile application for testing. In general, we came up with a good web-based solution for what the ITS department wanted to achieve. The application shows a mapbox map zoomed into the location of the user. We have implemented rectangles that change color based on the average reported Wi-Fi quality. The rectangle for the user’s location is shown on the map underneath which is a simple form for the user to select what they think of the quality of the Wi-Fi is for that location on a scale from 0 – 5. The user also has the option to submit a comment with his or her report. While these two form inputs are the only options shown to the user when the user submits the form, he or she will also be sending his or her ping, latitude, longitude, private IP, and information about his or her device. The application is a mixture of PHP, JavaScript, and HTML. The database we have is implemented with PostgreSQL. We have also created a dashboard that allows the ITS department to see many of the statistics from the reports that Cal State LA students will be submitting. The dashboard allows the ITS department to see graphs that show the operating systems of the submitted reports, a pie chart of the ratings reported, as well as several different map views to help locate the reported outages to quickly take care of them. This dashboard was successfully implemented using PHP.

The current server that our application runs on, however, is not ideal for the use of the application. This will become a nonissue when the ITS department moves our code and database to their own servers. The problems with the server include the inability to access the application and dashboard unless you are connected to the Cal State LA campus Wi-Fi, which would impede the ability for students to submit a report that the Wi-Fi for an area is completely down since they would not be able to login to the campus Wi-Fi in that situation. Another problem is the server stability. We often had trouble connecting to the server to update code or add files even if we were running the Cal State LA VPN.

In the future, we had several more plans for the application and dashboard that we were unable to implement during the project. We would like to be able to use the private IP from the reports to more accurately pinpoint the user’s location based on a database of the access points that the ITS department provided to us. This would not only allow us to see about which access point the user is reporting, but also the exact room and floor of a building in which the user is located. While the current server will not allow users outside of the campus Wi-Fi to access our application, when ITS changes the servers, we would like a pop-up notification to tell a user who is attempting to report a Wi-Fi outage from a location that is out of the bounds of the Cal State LA campus that, while they can still submit a report, his or her report will be discarded. We would also like for the map that the user sees on the web application to include nearby report rectangles. This way the user can find an area near them where the Wi-Fi is of a better quality. We believe that in time, it will not make sense for the rectangles to take into consideration every report that was made since the launch of our application. To remedy this, we would like to limit the rectangles to only consider the last 50 reports, or the last 24 hours of reports, whichever is most recent. This will give the user a better picture of how the Wi-Fi is now. Lastly, we want to implement a secure backend application which allows the ITS staff to login and clear the bad reports of a location that they have recently fixed.

**References**

Apache

<https://httpd.apache.org/>

Git

<https://git-scm.com/>

Highcharts

https://www.highcharts.com/

mapbox

https://www.mapbox.com/

PostgreSQL

<https://www.postgresql.org/>

Ubuntu

<https://www.ubuntu.com/>