**Software Design**

**Document**

**for**

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

**Version 1.0 approved**

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

**CSULA**

**3-5-2018**

Table of Contents................................................................................................................... <pg #>

Revision History..................................................................................................................... <pg #>

1. Introduction................................................................................................................ <pg #>
   1. Purpose........................................................................................................... <pg #>
   2. Document Conventions……………………................................................. <pg #>
   3. Intended Audience and Reading Suggestions................................................ <pg #>
   4. System Overview........................................................................................... <pg #>
2. Design Considerations............................................................................................... <pg #>
   1. Assumptions and dependencies..................................................................... <pg #>
   2. General Constraints........................................................................................<pg #>
   3. Goals and Guidelines.....................................................................................<pg #>
   4. Development Methods................................................................................... <pg #>
3. Architectural Strategies.............................................................................................. <pg #>
4. System Architecture................................................................................................... <pg #>
   1. ....................................................................................................................... <pg #>
   2. ....................................................................................................................... <pg #>
5. Policies and Tactics.................................................................................................... <pg #>
   1. Specific Products Used.................................................................................. <pg #>
   2. Requirements traceability............................................................................... <pg #>
   3. Testing the software....................................................................................... <pg #>
   4. Engineering trade-offs................................................................................... <pg #>
   5. Guidelines and conventions........................................................................... <pg #>
   6. Protocols........................................................................................................ <pg #>
   7. Maintaining the software............................................................................... <pg #>
   8. Interfaces........................................................................................................ <pg #>
   9. System's deliverables..................................................................................... <pg #>
   10. Abstraction..................................................................................................... <pg #>
6. Detailed System Design............................................................................................. <pg #>

6.x Name of Module........................................................................................... <pg #>

6.x.1 Responsibilities...................................................................................... <pg #>

6.x.2 Constraints............................................................................................. <pg #>

6.x.3 Composition........................................................................................... <pg #>

6.x.4 Uses/Interactions.................................................................................... <pg #>

6.x.5 Resources............................................................................................... <pg #>

6.x.6 Interface/Exports.................................................................................... <pg #>

1. Detailed Lower level Component Design

7.x Name of Class or File..................................................................................... <pg #>

7.x.1 Classification.......................................................................................... <pg #>

7.x.2 Processing Narrative(PSPEC)................................................................ <pg #>

7.x.3 Interface Description.............................................................................. <pg #>

7.x.4 Processing Detail.................................................................................... <pg #>

7.x.4.1 Design Class Hierarchy................................................................... <pg #>

7.x.4.2 Restrictions/Limitations.................................................................. <pg #>

7.x.4.3 Performance Issues........................................................................ <pg #>

7.x.4.4 Design Constraints......................................................................... <pg #>

7.x.4.5 Processing Detail For Each Operation........................................... <pg #>

1. User Interface
   1. Overview of User Interface.......................................................................... <pg #>
   2. Screen Frameworks or Images..................................................................... <pg #>
   3. User Interface Flow Model........................................................................... <pg #>
2. Database Design
3. Requirements Validation and Verification................................................................. <pg #>
4. Glossary..................................................................................................................... <pg #>
5. References................................................................................................................. <pg #>

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
| Steven Castro | 2-4-18 | Revised some wording across the entire document. | 1.05 |
| Steven Castro | 3-5-18 | Added additional detailed system info to section 1.4 | 1.1 |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

**1.1 Purpose**

This document outlines in detail the software architecture and design for the full name(IWCQCSLA) application. This document provides multiple views of the design of the application to better help understand the underlying architecture for this application. This document intends to be the sole source for reference while developing this application.

**1.2 Document Conventions**

For this document, all requirements are within the same priority unless *italicized*. *Italicized* modules are priority requirements.

**1.3 Intended Audience and Reading Suggestions**

The intended audience for this document is for anyone who is looking to understand the architecture of the application. This documents is being written so that the software developers can use this to develop the application by using this sole document as the resource for all requirements and how to implement the requirements.

**1.4 System Overview**

The core system languages that this application shall be using is PhP, Javascript, Apache, and PostGRE SQL on an Ubuntu OS.

The system will utilize HTML for displaying the basic info on the webpages with events to link to other modules. The additional modules being displayed should be called from a PHP file or through a Javascript function. The PHP should handle the SQL transactions to the server to protect from SQL injection. The Javascript can handle the visualization of data through a heatmap or PI chart.

Apache will handle the offline website development while the PostGRE SQL will be responsible for handling the data, server side.



FIGURE 1.4-1: System Architecture Layout  
**2. Design Considerations**

For this application, it needs to be ran on a mobile web browser. This is for an easy implementation into the CSULA GET mobile app.

**2.1 Assumptions and Dependencies**

Some of the dependencies required for this application are as follows:

* Application must be able to be ran on a mobile web browser
  + Said web browser must be ran on multiple Android/IOS devices.
* Application must be mobile friendly (such as the UI scaled down to see everything in a mobile view).
* Web-based server and Database server must be separate servers. Cannot be hosted on the same server.

**2.2 General Constraints**

Some of the General Constraints that this application must adhere to are as follows:

* Database and Website servers must be separate.
* Must be completely web based.
* Must be able to support up to 100 simultaneous users

**2.3 Goals and Guidelines**

* The end product should be easy to use.
* Must be available from within the CSULA GET mobile application.
* The application’s form must be easy to fill out with a majority of information automatically filled out.
* Software must be completed by the end of the Spring Semester 2018.

**2.4 Development Methods**

This project shall be developed using the Water Fall Development method. Requirements for this project shall be gathered first and then developed in two week periods.

The software side of development shall be in an MVC approach with clients gathering data from the separate databases and passing filtered data back into the respective databases.

**3. Architectural Strategies**

3.1 Database Server

The database server should serve as a hub for all the information regarding reports. This server should be responsible for handling all reports and should serve as a host to the data regarding location with form data. The database server must remain separate from the Web server to prevent a malicious user from directly interacting with the backend server.

3.2 Web Server

The Web Server shall serve as the main host for the application within the CSULA GET mobile application. This server shall be responsible for handling the web-client that the users will interact with. This server shall be running on PhP and shall also be responsible for sending validated forms back to the database server. This server should serve as a frontend to help filter malicious users that can potentially attack the backend database. This server shall remain separate from the Database Server as a security precaution.

**4. System Architecture**

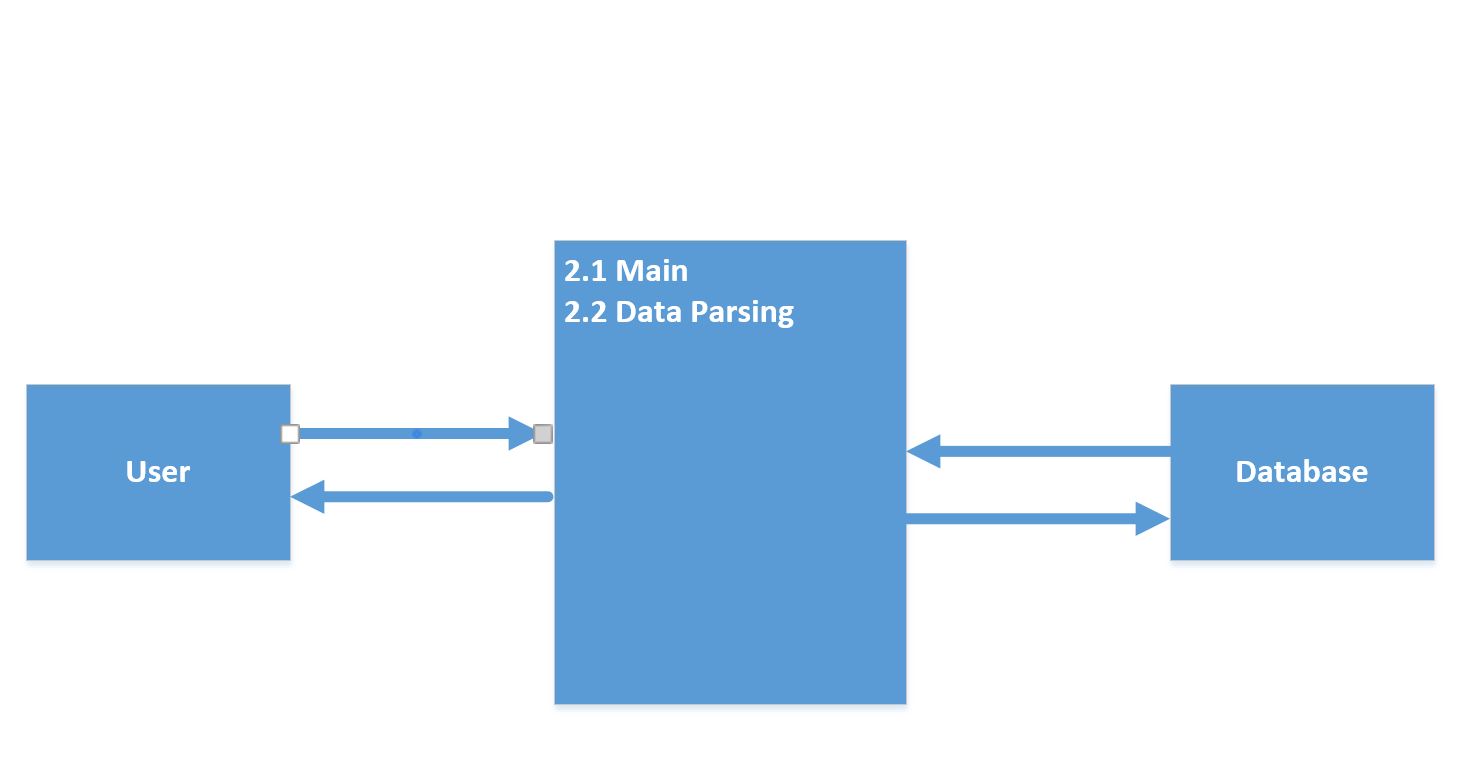
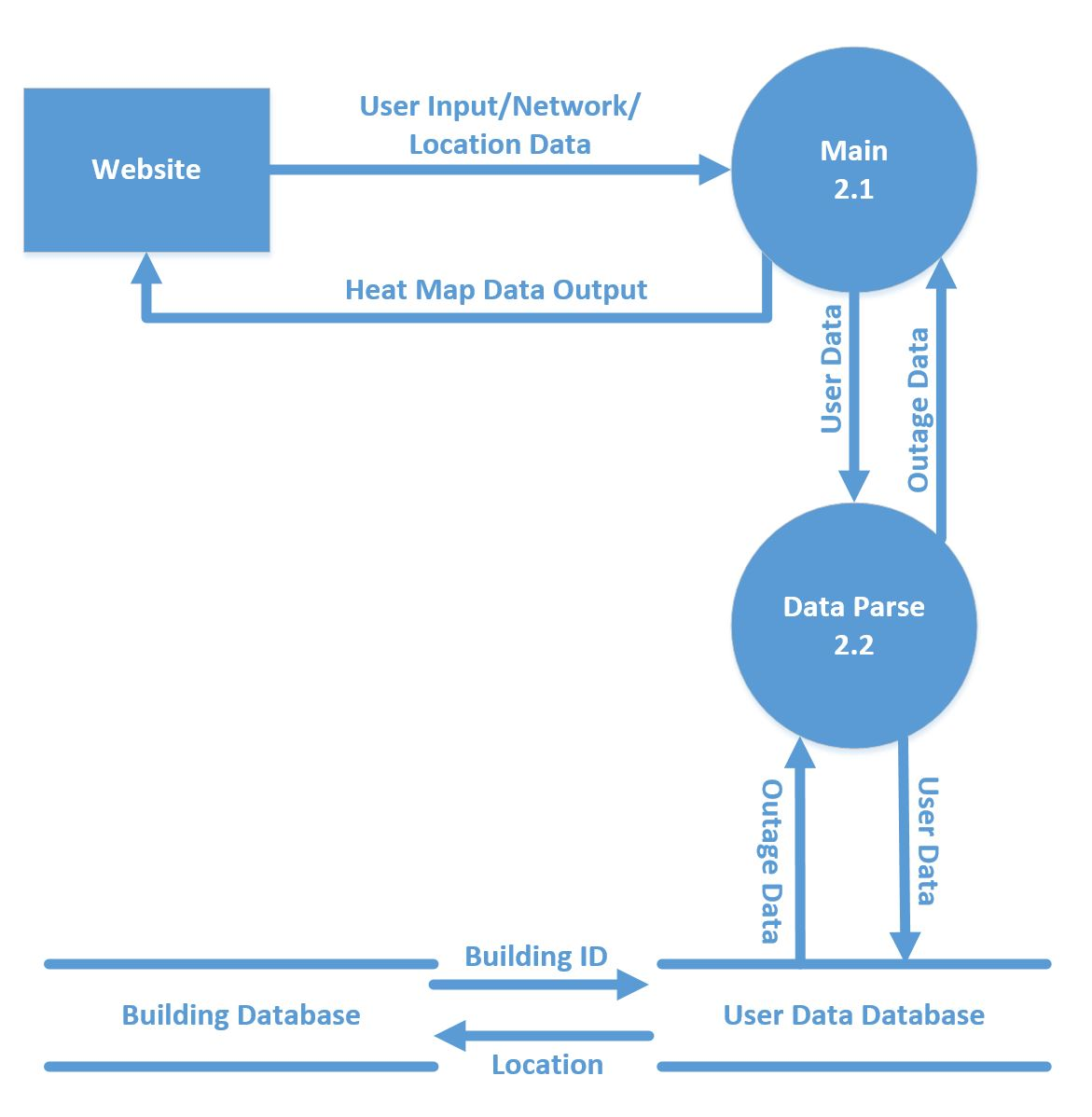


Figure 4.1 – DFD LEVEL 0 of IWCQCLA

The system shall be divided into two main parts. 2.1 shall handle the main application including the heatmap and form. It shall essentially be the application’s frontend where the users will interact with. 2.2 Data parsing should handle all the communication between the application and the database. This should parse data received from the database and should parse out data sent from the Main to the database.



Level 2 DFD for IWCQCSLA

This application is split up into two main modules.

2.1 Main   
This module is responsible for the frontend website. This shall be mostly implemented in PhP due to its features with data management. The frontend website should have a heatmap for areas with many reported outages from within the app. This also should include a view of your connection’s details including ping, download, and upload speeds. This web page should ask the user for their location. The second main part of the main is the ‘Report Outage’ form. This form should be responsible for generating data to be sent out to the database. This form should be partially automatically filled out if the user agrees to send over their location data. The form portions that should be automatically filled out shall be building they are currently in, and by default even if they do not share location it should fill out their connection details. The details to be filled in should be ping, upload speed, download speed. If the user does not give location, the user must fill out building location. Additional details such as floor and room shall also be available for reporting options on the form.

2.2 Data Parsing  
This second main module shall be responsible for all the backend data management. This data parsing module should be responsible for grabbing data from the database and using it to help generate the heatmap from module 2.1 Main. This also shall be responsible for sending data to the database. Its purpose is be to parse out malicious or invalid data to the database. This module is the sole way to communicate with the database to avoid confusion of different ways of communicating with the database server. This is also meant to help solve some security problems of malicious users abusing the 2.1 Main module’s web page for interaction with the database.

**5. Policies and Tactics**

**5.1 Choice of which specific products used**

Ubuntu LTS 16.04  
PostgreSQL 9.5  
PHP 7  
Apache 2.4.18

**5.2 Plans for ensuring requirements traceability**  
Requirements traceability shall be conducted through observation of the separation of server and source code being available to view and use by developers and those authorized to view this application’s details.

**5.3 Plans for testing the software**

5.3a Testing of this software shall be conducted during development and during a QA phase in which the product shall be undergoing many tests. The beta and alpha versions of this application shall be tested internally. Test cases shall be created at a future date when the application is functioning and performing according to its requirements as stated here and in the Business Requirements.

5.3b Testing for the amount of simultaneous users that this application can support shall be determined at a future date when the amount of users supported is detailed in the business requirements. \*\*\*\*

**6. Detailed System Design**

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD’s will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

**6.x Name of Component (Module)**

**6.x.1 Responsibilities**

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

**6.x.2 Constraints**

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

**6.x.3 Composition**

A description of the use and meaning of the subcomponents that are a part of this component.

**6.x.4 Uses/Interactions**

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass’s, and metaclasses.

**6.x.5 Resources**

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

**6.x.6 Interface/Exports**

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

**7. Detailed Lower level Component Design**

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

**7.x Name of Class or File**

**7.x.1 Classification**

The kind of component, such as a subsystem, class, package, function, file, etc.

**7.x.2 Processing Narrative (PSPEC)**

A process specification (PSPEC) can be used to specify the processing details

**7.x.3 Interface Description**

**7.x.4 Processing Detail**

**7.x.4.1 Design Class Hierarchy**

Class inheritance: parent or child classes.

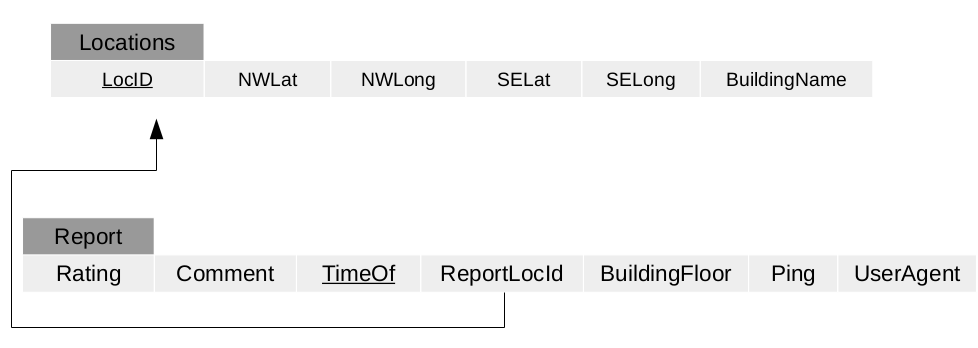
**7.x.4.2 Restrictions/Limitations**

**7.x.4.3 Performance Issues**

**7.x.4.4 Design Constraints**

**7.x.4.5 Processing Detail For Each Operation**

**8. Database Design**



The database shall have two main tables. The tables shall be Locations and Report. This is to match the report table with the Locations table for helping to generate a heatmap through the 2.1 Main module. This shall in turn be used to see how many reports were used in which ‘Locations’ for the heatmap. The ReportLocId shall be used as a key to access the ‘Locations’ table by matching with the LocID table.

**9. User Interface**

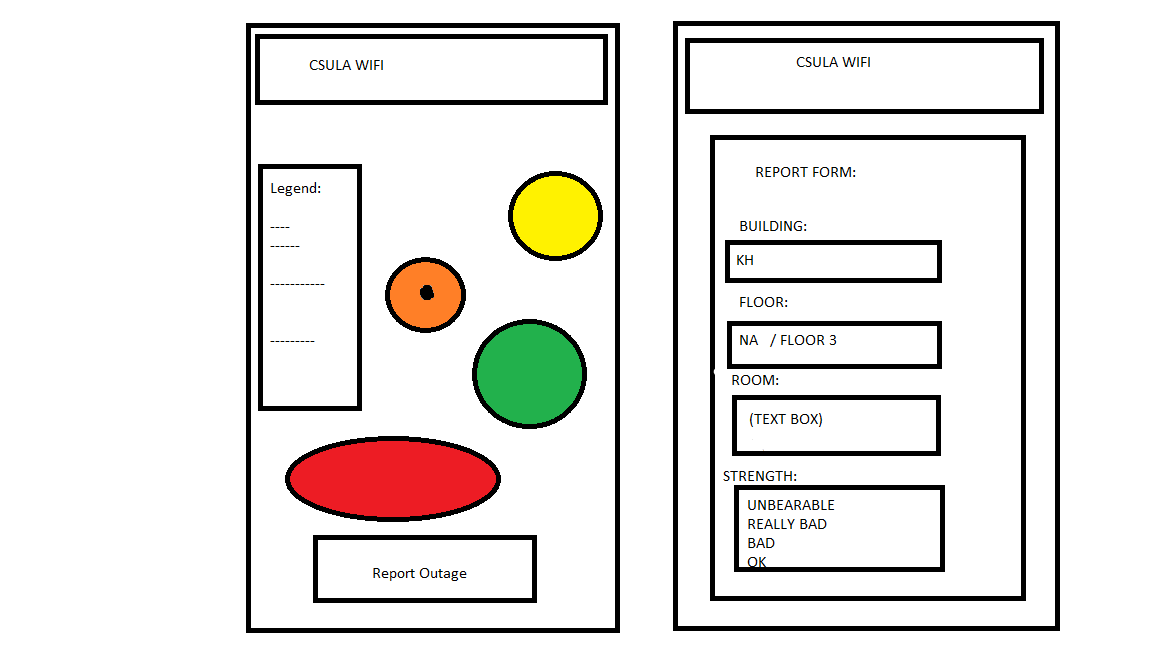
**9.1 Overview of User Interface**

The user shall have to have the CSULA GET mobile application downloaded. The user shall then have to navigate to this application’s web link through the front page of the application.

The user shall see a heatmap for reported outages in the front page of the application. The user shall be given a page that contains all the connection details that can be gathered through a browser such as ping, download speed and upload speed.

The user can then look towards the bottom of the mobile web page to see a ‘Report Outage’ button. This button shall bring up a form that should allow the user to fill in with details for reporting the Wi-Fi outage. The form shall have dropdown boxes that shall contain building, floor, room number. The form shall also have the user’s ping and other data details automatically filled out to help the user submit a form easily with most information automatically filled out.

**9.2 Screen Frameworks or Images**



9.2-1 Concept Framework for the application, showcasing the UI

**9.3 User Interface Flow Model**

To navigate from the heatmap and details page (left side of figure 9.2-1) the user shall have to click on the ‘Report Outage’ button. This should allow the user to access the form for reporting a Wi-Fi outage.

**10. Requirements Validation and Verification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements Related to Web Server (WS) Module (4.1)** | |  |  |
| Requirement No. | Requirement Description | Module that fulfills Requirement: | Testing Method |
| 4.1-1 | WS shall be running PHP. | 2.1 | DEV |
| 4.1-2 | WS shall support JavaScript. | 2.1 | DEV |
| 4.1-3 | WS shall support HTTPS. | 2.1 | DEV |
| 4.1-4 | WS shall request the user’s location data. | 2.1 | DEV/QA |
| 4.1-5 | WS shall use Google Maps APIs. | 2.1 | DEV/QA |
| 4.1-6 | WS shall store and manage data by retrieving and updating tables in the database server. | 2.1 | DEV |
| 4.1-7 | WS shall direct “Report Outage” button input to the “Reporting Outage” view. | 2.1 | QA |
| 4.1-8 | WS URL for the web page shall be accessible through the CSULA GETmobile app. | 2.1 | QA |
| 4.1-9 | WS generated webpages shall have a dropdown form menu that contains the ‘Report Outage’ Submodule. | 2.1 | QA |
| 4.1-10 | WS shall be running on a server separate from the database server. | 2.1 | DEV |
| 4.1-11 | WS heatmap shall be based off existing data from the Database server. | 2.1 | DEV |
| 4.1-12 | WS’s generated webpage shall be optimized to fit on mobile devices. | 2.1 | QA |
| 4.1-13 | WS’s generated webpage shall show statistics on download speed, ping, and other network information. | 2.1 | QA |
|  | **SUBMODULE 4.1.1 OUTAGE REPORT FORM (ORF)** |  |  |
| 4.1.1-1 | ORF shall be accessed through a button within the webpage generate from the WS module. | 2.1 | QA |
| 4.1.1-2 | ORF shall have a form with dropdown boxes. | 2.1 | QA |
| 4.1.1-3 | ORF’s dropdown fields shall be filled with information regarding location from within CSULA. | 2.1 | QA |
| 4.1.1-4 | ORF’s dropdown fields shall be partially filled in depending if the user has their location services on. The partially filled in forms shall be but not limited to: the building they are in, download speeds, ping. | 2.1 | QA |
| 4.1.1-5 | ORF’s shall send the form’s contents to the database server. | 2.1 | QA |
| 4.1.1-6 | ORF shall use prebuilt commands to update the SQL server. | 2.1 | QA |
| 4.1.1-7 | ORF shall have a text box for additional comments. | 2.1 | QA |
| 4.1.1-8 | ORF shall protect against SQL injection by parsing out input command. | 2.1 | QA |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements Related to Database Server (DS) Module (4.2)** | |  |  |
| Requirement No. | Requirement Description | Module that fulfills Requirement | Testing Method |
| 4.2-1 | DS shall be running PostgreSQL. | 2.2 | DEV |
| 4.2-2 | DS shall be running on a Linux based operating system. | 2.2 | DEV |
| 4.2-3 | DS shall be on a separate server from the WS. | 2.2 | DEV |
| 4.2-4 | DS shall have protection from SQL injection from the WS. | 2.2,2.1 | DEV/QA |
| 4.2-5 | DS shall store and send data from the SQL tables. | 2.2 | DEV |
| 4.2-6 | DS shall have a SQL table setup with locations from within CSULA as the primary key. | 2.2 | DEV |
| 4.2-7 | DS can only be updated using the WS’s Report Outage form module. | 2.1 | DEV/QA |
| 4.2-8 | DS shall be able to support multiple users submitting forms to the SQL database simultaneously. | 2.1, 2.2 | DEV/QA |
| 4.2-9 | DS shall be able to send data to WS for generating a heatmap. | 2.1 | DEV |
| 4.2-10 | DS shall only interact with the webserver by sending data using SQL commands. | 2.1 | DEV |

**11. Glossary**

IWCQCSLA – Improve Wifi Coverage and Quality at California State University Los Angeles ( title of project)

QA – Quality Assurance

DEV – development

**12. References**

(LINK TO THE BUSINESS REQUIREMENTS DOCUMENT HERE)