**Software Design**

**Document**

**for**

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

**Version 1.3 approved**

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

**CSULA**

**4-18-2018**

Table of Contents................................................................................................................... <pg 2>

Revision History..................................................................................................................... <pg 4>

1. Introduction................................................................................................................ <pg 5>
   1. Purpose........................................................................................................... <pg 5>
   2. Document Conventions……………………................................................. <pg 5>
   3. Intended Audience and Reading Suggestions................................................ <pg 5>
   4. System Overview........................................................................................... <pg 5>
2. Design Considerations...........................................................................................................<pg 7>
   1. Assumptions and dependencies..................................................................... <pg 7>
   2. General Constraints........................................................................................<pg 7>
   3. Goals and Guidelines.....................................................................................<pg 7>
   4. Development Methods................................................................................... <pg 7>
3. Architectural Strategies..........................................................................................................<pg 8>
4. System Architecture................................................................................................... ……....<pg 9>
   1. ....................................................................................................................... <pg 9>
   2. ......................................................................................................................<pg 10>
5. Policies and Tactics..............................................................................................................<pg 12>
   1. Specific Products Used.................................................................................<pg 12>
   2. Requirements traceability.............................................................................<pg 12>
   3. Testing the software......................................................................................<pg 12>
6. Detailed System Design............................................................................................. ……..<pg 13>

6.1 Website Main…...........................................................................................<pg 14>

6.1.1 Responsibilities....................................................................................<pg 14>

6.1.2 Constraints...........................................................................................<pg 14>

6.1.3 Composition.........................................................................................<pg 14>

6.1.4 Uses/Interactions..................................................................................<pg 14>

6.1.5 Resources.............................................................................................<pg 14>

6.1.6 Interface/Exports..................................................................................<pg 14>

6.2 Dashboard...................................................................................................<pg 15>

6.2.1 Responsibilities....................................................................................<pg 15>

6.2.2 Constraints...........................................................................................<pg 15>

6.2.3 Composition.........................................................................................<pg 15>

6.2.4 Uses/Interactions..................................................................................<pg 15>

6.2.5 Resources.............................................................................................<pg 15>

6.2.6 Interface/Exports..................................................................................<pg 15>

1. Database Design  
    7.0 Database Design Description ...... ........................... ........................... ……..<pg16 >
2. User Interface
   1. Overview of User Interface.......................................................................<pg 17>
   2. Screen Frameworks or Images..................................................................<pg 17>
   3. User Interface Flow Model.......................................................................<pg 17>
3. Requirements Validation and Verification................................................................<pg 18>
4. Glossary....................................................................................................................<pg 20>
5. References................................................................................................................<pg 20>

**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 |
| Steven Castro | 4-11-18 | Changed wording and added new areas for the Dashboard. Added page numbers to the table of contents | 1.2 |
| Steven Castro | 4-18-18 | Added some class definitions to section 6. Updated Table of contents | 1.3 |

**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.

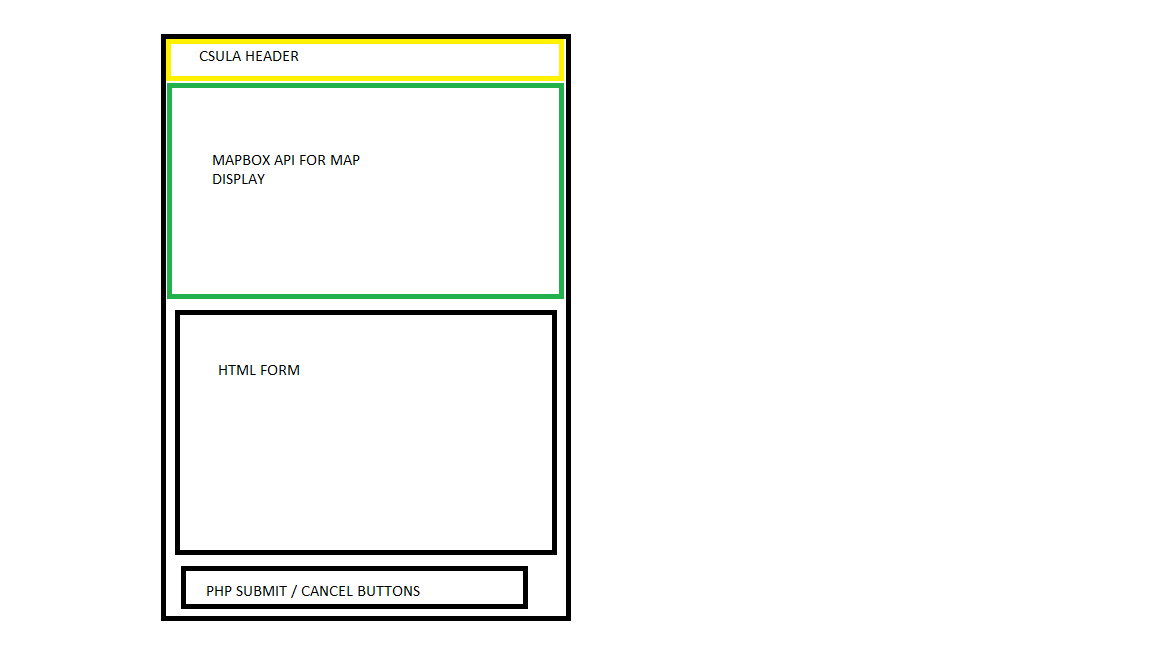


FIGURE 1.4-1: System Architecture Layout  
**2. Design Considerations**

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.

The Web Server should also be responsible for hosting the Dashboard webpage. This Dashboard webpage is used only for viewing statistical information based on reports and can show all the reports in the database through different views. This does not input any data to the database as it only displays information from it.

**4. System Architecture**

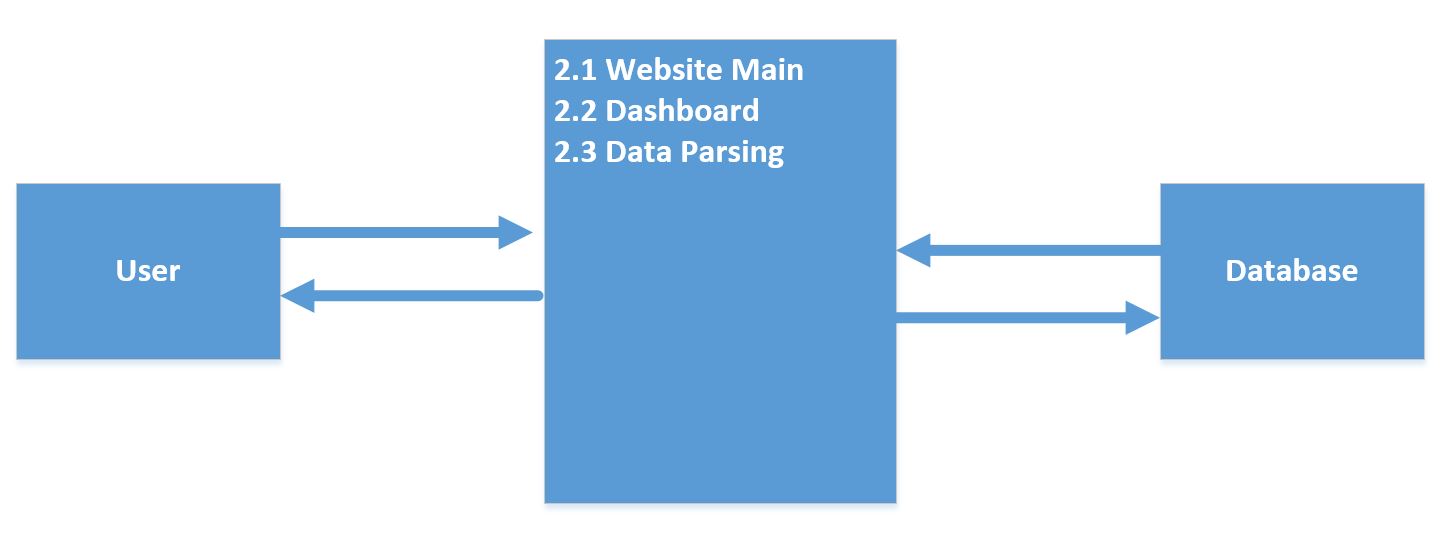


Figure 4.1 – DFD LEVEL 0 of IWCQCLA

The system shall be divided into two main parts. The main parts are Website Main and Dashboard. Data Parsing is a submodule that exists in each of the two main modules. The details of what submodule has the data parsing element is described in section 6.   
2.1 Website Main shall handle the main website output for the mobile site. This inputs data into the database through the form located in it.  
2.2 Dashboard handles the information that was entered into the database by the user in 2.1. This will display all the reports in different views through tables for all the data and charts.

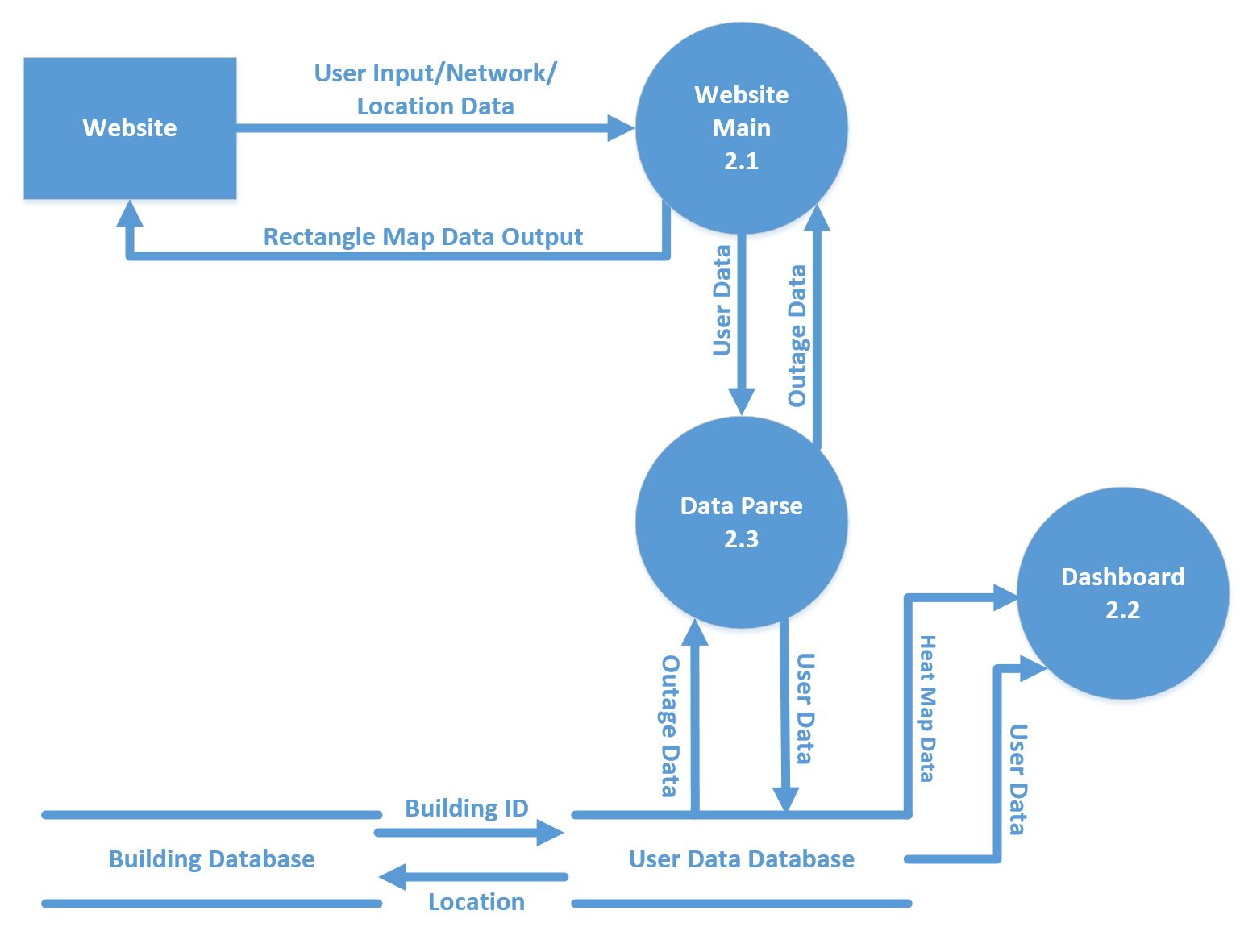


Figure 4.2 Level 1 DFD for IWCQCSLA

This application is split up into two main modules.

2.1 Website 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 Dashboard

This module is responsible for displaying all the data inserted into the database from 2.1 Website Main. This module deals with displaying the data in various ways. Some examples of this is to display the data in a chart and display it in table format with different ways to order the data such as by severity or time reported. This module shall be mostly implemented in PhP due to its features with data management. It shall have a heatmap to display reports to make the data from reports easier to read. This module does not input any new data into the database.

2.3 Data Parsing  
This third 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.   
This module exists within each of the Website Main Module and within the Dashboard module. It exists in those two modules through a subclass that parses the data out differently based on the module. The details of what subclass has it is detailed in section 6.

**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.

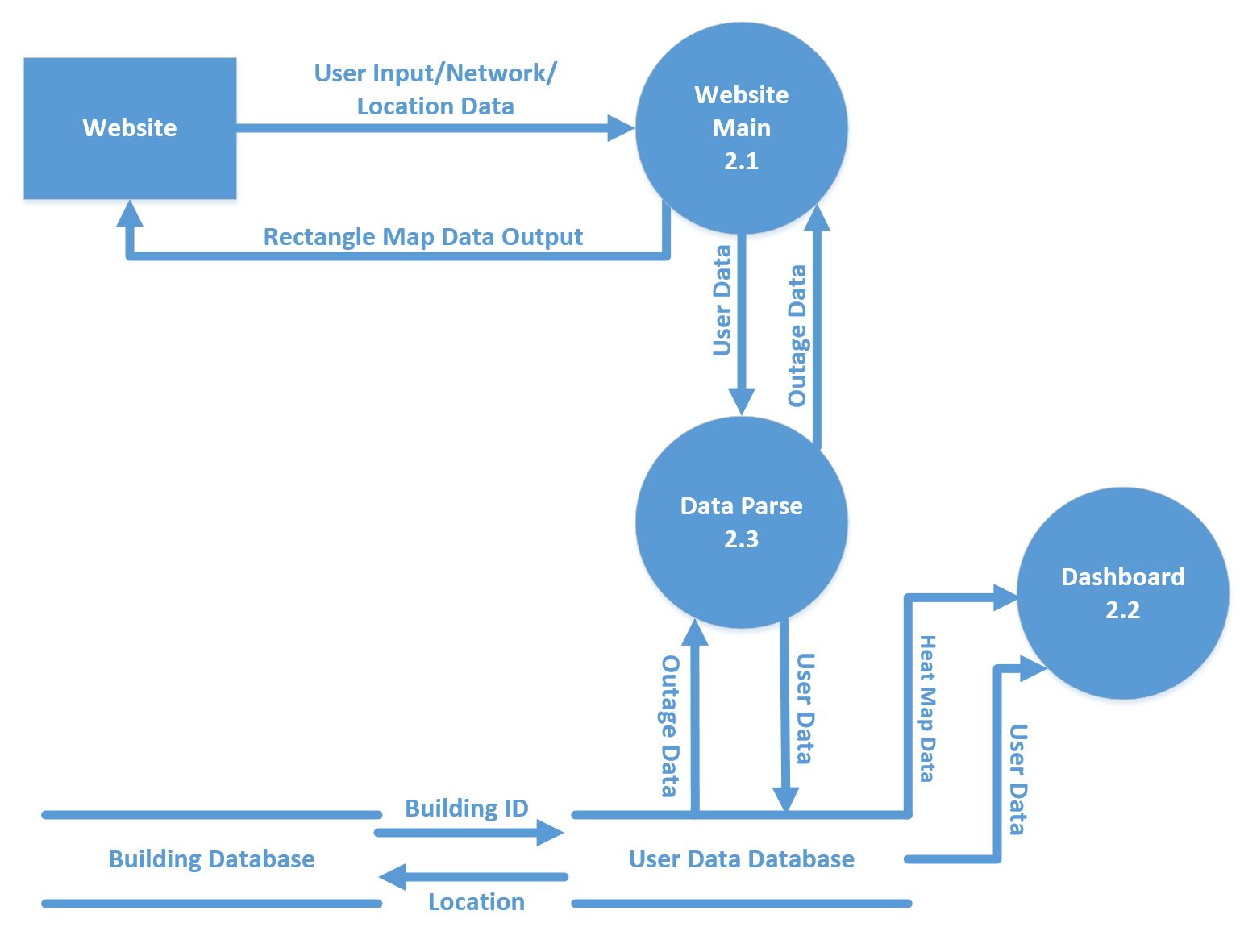
**5.3 Plans for testing the software**

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.

5.3b Testing for this application is done internally through a QA process to make sure the product works as intended.

5.3c Testing for simultaneous users is done through JMeter.

**6. Detailed System Design**

  
Level 1 DFD for IWCQCSLA   
  
This project is split up into two main modules: Website Main and Dashboard. Details of these are covered below. Data Parsing exists in both and is detailed where it exists in the Composition subcategory for this section.

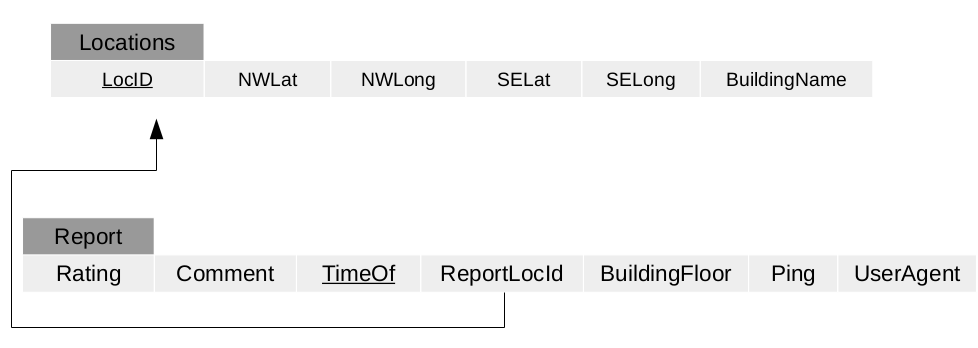
**6.1 Website Main**

**6.1.1 Responsibilities**This module is responsible for outputting the mobile website. It grabs the location of your phone GPS and displays that on the webpage in a map to show your current location for purposes of reporting. Along with displaying your current location, it also shows a rectangle around the pin generated to represent your location on the map drawn. The rectangle’s color will change depending on the average reported number on Wi-Fi quality for the area.   
The website outputted by this module also has a form on it to report a Wi-Fi outage.   
See 9.2 for an example of how the mobile website should look like.  **6.1.2 Constraints**The Website Main module is only meant to input new data into the database and to read the average reported number based on your location through GPS. Because of this, this module has no way to read the data entered into the database other than finding the average reported number. This module grabs data from two sources: GPS for Location and the average reported number in the area from the database.  **6.1.3 Composition**  
-report.php -main file for reporting website. Uses HTML for the website and imports different necessary outside dependency scripts for displaying maps and map layers.  
-report.css -Stylesheet for the Dashboard.  
-report-service.php - Helper getter file with methods that grabs existing data from backend server.  
-protomap.js - script to help map pinpoint and zoom into client geo location. It also finds building location according to the clients’ geo location and displays the wireless status at that location.  
-addreport.php - PhP file to pass the POST parameters on to our database server (prepared statements to resist any SQL injection).  
-empty.php, garbage.php - gets upload and download speeds using a web service and during that process it also gets the ping.  
-getIP.php - gets the IP address using a remote server.  
-speedtest\_worker.min.js - calls functions that call empty, garbage, and getIP to get the values for download speed, upload speed, ping and IP address and puts them in an array to be called by the javascript in our main page and parses it to the elements of the form.  
  
**6.1.4 Uses/Interactions**This module is a standalone module as it does not use methods from outside that are found elsewhere in this project. All methods pertaining to this module are listed in 6.3.3 Composition. **6.1.5 Resources**The Website Main module uses the following API: Mapbox API  
This project also uses separate servers for hosting the database and websites.  **6.1.6 Interface/Exports**See 6.1.3 for classes within this module. And 6.1.4 for Uses/Interactions.

**6.2 Dashboard**

**6.2.1 Responsibilities**The dashboard is responsible for displaying all the reported data from the database. This grabs the data through multiple SQL transactions depending on what tab is selected. This dashboard provides different ways to view the data easily and shows charts based on attributes of a report such as the level of severity for the report.  **6.2.2 Constraints**Since this is only for viewing the reports, this dashboard cannot manipulate the database as it is a website that only reads data and outputs it. **6.2.3 Composition**-Dashboard.css – Stylesheet for the Dashboard  
-Dashboard.js – Javascript functions to draw the maps and report points onto the browser  
-Dashboard.php – main file for the website. Uses HTML for the website and imports different necessary outside dependency scrips for displaying maps, charts, etc.  
-Dashboard-Service.php – calls the queries needed from the database and pushes them onto dashboard.php **6.2.4 Uses/Interactions**This module is a standalone module as it does not use methods from outside that are found elsewhere in this project. All methods pertaining to this module are listed in 6.3.3 Composition.  **6.2.5 Resources**The dashboard uses the following APIs: Mapbox api, Jquery api, high charts api.  
This project also uses separate servers for hosting the database and websites.  **6.2.6 Interface/Exports**See 6.2.3 for classes within this module. And 6.2.4 for Uses/Interactions.

**7. 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.

**8. User Interface**

**8.1 Overview of User Interface**

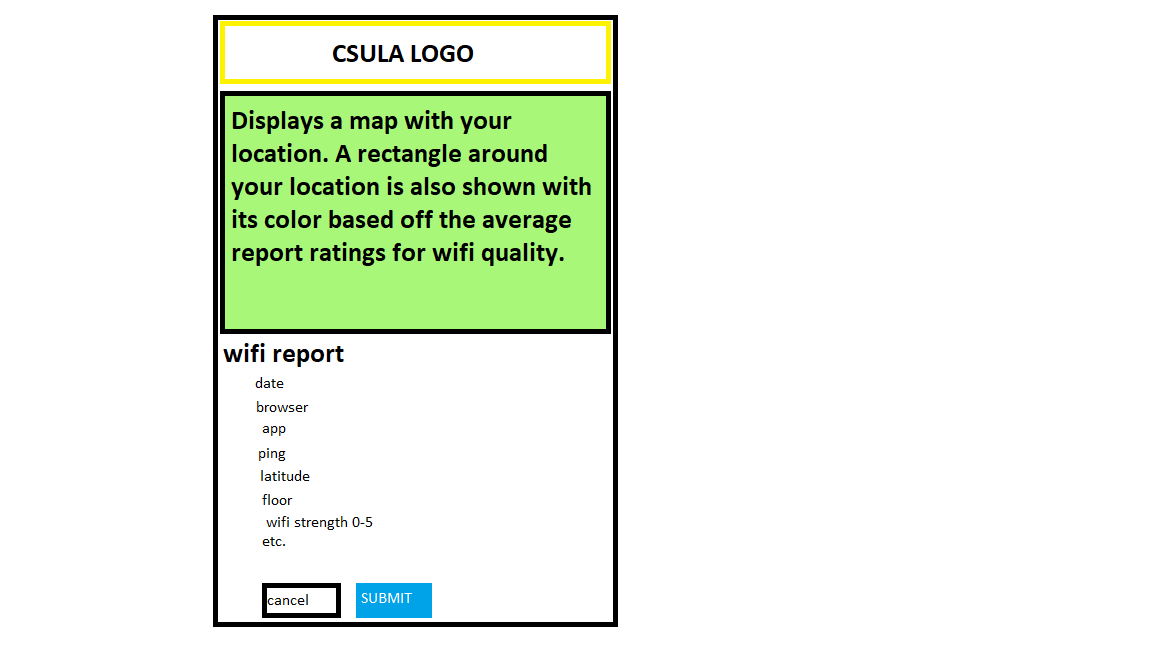
The user shall have to have the CSULA GET mobile application downloaded. The user shall then have to navigate to this application’s web link through the front page of the application.

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 report an outage through the field at the bottom of the page.

The form 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.

**8.2 Screen Frameworks or Images**



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

**8.3 User Interface Flow Model**

The user will start at the top of the page seeing details for the current area through reports and GPS location. From the top the user will work their way down to fill out a form to report an outage. The user can submit the form by pushing Submit or cancelling through the Cancel button.

**9. 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)** |  |  |
| 4.1.1-2 | 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 |

**10. Glossary**

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

QA – Quality Assurance

DEV – development

**11. References**

Refer to the Business Requirements Documentation