Software Design Specification

for

Red Team Health Inspection Software

Version 1.0 prepared: 11/12/2015

Prepared by Red Team

CS320

Revision History

Version	Date	Author	Change Description
1.0	11/13/15	Red Team	First draft
2.0	12/1/15	Red Team	Final draft

TABLE OF CONTENTS

1. Introduction

1.1. Purpose

The purpose of this document is to provide specific design details regarding each of the sub projects *printing, mapping* and *data entry*. The specifics will explain ambiguities such as what operating system will be used, what languages each portion will be written in and other similar technical details that need to be explained.

1.2. Scope

The intended audience for this document is for both developers and users trying to familiarize themselves with how each sub project will function, both from a developer standpoint and a user standpoint.

1.3. Definitions, Acronyms, and Abbreviations

Term/Acronym	Definition

2. References

Software Requirements Specification for Red Team Health Inspection Software Version 1.1

3. SYSTEM ARCHITECTURAL DESIGN

3.1. System Description

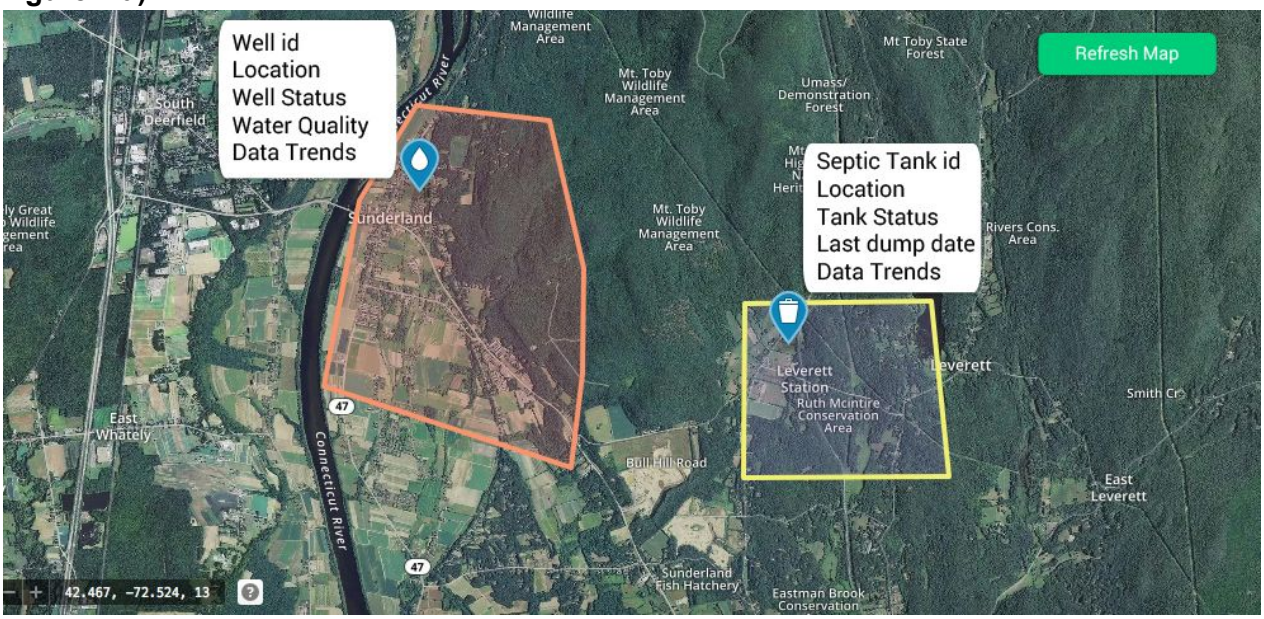
The complete system will be a desktop application written with node.js and express.js to run in a web browser, and a tablet application created for android devices. This document focuses on two components of the desktop application and one component of the tablet application - Mapping and Data Entry (desktop) and Printing (tablet).

3.1.1. Mapping

The figure below gives an idea of how the mapping portion of the application will function. The pins will provide pop-up dialogs that show off the information corresponding to the pin type selected when clicked. The highlighted areas

are the different lots.

Figure 1.a)



3.1.2. Printing

There will be a ‘Print Report’ button for completed reports. After the user clicks the ‘Print Report’ button, the application will present a formatted report to the user. One page of the report will be displayed on the screen, and the user can navigate the report using the ‘Previous’ and ‘Next’ buttons on the bottom of the screen. On the side of the screen, there will be a ‘Print Report’ button that prints the complete report, and a ‘Print Page’ button that prints the page the user is viewing.

After a print option is selected, a popup informs the user that the ‘Report is being printed’. The user is brought to the viewing page for the report they printed after the printing process is finished.

3.1.3. Data Entry

There will be three tabs: one for “Health Inspection”, “Wells” and “Septic System” respectively. When the user clicks on one of these tabs, the application will show the corresponding form for data entry.

If the location and other header information is previously existing, the user will be able to select it from a drop down list and the fields of the information of that location will be auto-filled. If it is a new location, the user can click the “Add New” button to enter the information of the new location or header information. On each form, there will be fields for information pertaining to the location, i.e. location ID, inspection reports, violations, etc. After finishing filling the report, the user can click the “Submit” button to save the data to database.

If the user needs to edit or delete report data, they will use the search component to find a particular inspection or report, click a link within the search results which directs the user to the data entry view, and then click the “Submit” button or “Delete” button at the bottom of the report after making changes.

3.2. System Architecture

3.2.1. Mapping and Data Entry

We have chosen a client/server architecture for the mapping project. The user will using a modern browser such as mozilla’s firefox or google chrome. The browser provides a suitable, and flexible platform to display the application’s user interface. A server application will be running on the user’s local machine and this local server application will be nodejs.

3.2.2. Printing

The mobile application will be using a client/server architecture. The user will use the mobile application to send and retrieve data from the remote server. The printing module will be a subsystem of the mobile application.

The chosen architecture for the printing project is built on a client/server architecture. The client in this case will be the mobile printer, while the server will be the modules provided by the printing team.

3.3. Design Constraints

3.3.1. General constraints

The user for this project is not well versed in modern technology, so we will be required to use existing, or familiar systems to provide the requested functionality. For example, the user may not be familiar with installing a desktop application, but the user is familiar with using a web-browser to search for things.

3.3.2. Hardware constraints

The mobile application may not always have Internet access. The data on the mobile device might need to be stored temporarily on the device before being transferred to a local machine.

The user will be using a printer to print certain documents from the mobile application. We will need to make sure that the printer in question is compatible with our software.

3.3.3. Software constraints

Mapping & Data Entry

The application will be designed in javascript, and because of this the application is restricted to using a client/server architecture on the local machine. Javascript and node.js are cross-platform, and so there aren't any other design constraints implied.

Printing

The mobile application will be an Ionic application programmed using HTML and Javascript. We will be using Javascript libraries to create the PDF of the report and print the report.

4. Components description

4.1. Introduction

Mapping & Data Entry

The application will use a Model-View-Controller design pattern, where the user will be changing the state of the application running on his local machine and use the view provided in the browser to display the information/state of the application.

Printing

The mobile application will use a Model-View-Controller design pattern.

4.2. Decomposition description

The desktop application system can be divided into the local server application, and browser application. Modules and interfaces are described in Section 5 (External Interfaces) .

The local server application will manage connections to the database, and forward information to the browser application.

The browser application is dependent on the local server application to be running. The browser application will forward messages to the server that will initiate the requesting process.

Subsystem 1: Data Entry

Component 1 : Local Server application

This portion will co-operate with mapping and searching in order to provide proper data of locations and reports.

Subsystem 2: Mapping

Component 1 : Local Server application

This portion will be responsible to responding to requests from the browser application with well data, septic data and lot data. This portion will also be responsible with co-operating with data entry and searching in order to provide proper redirection to appropriate well reports and septic tank reports.

Component 2 : Browser application

This portion will be responsible for displaying the information (refer to **Figure 1.a**) to the user, providing redirection to the well report and septic tank report displays, and also responsible for forwarding messages to the local server application. These requests include :

- Data request messages

  (necessary for retrieving well data and septic tank data)

- Database connection attempt (When a user clicks ‘retry database’ button)

This portion will also be responsible for responding/handling messages received by the server application. These handles include :

- Data successfully retrieved from database.
- Database connection successful

The mobile application system can be divided into the server application and the printing application. The server application will provide the necessary data to the printing application, which will then use the data to produce a printed report.

Subsystem 1: Mobile Printing

Component 1: Local Server application

This portion will be responsible for serving the water well report, septic tank report and restaurant report to the printing module.

Component 2: Printing application

The printing module of mobile application will provide a service that will format the data saved on mobile drive into the official format of the form records. It will also be responsible for taking the reports on the tablet and print them out on a mobile printer.

5. External interfaces

5.1. Introduction

In this section, we describe the external interfaces we will be using for the application, including user interfaces and external system interfaces.

5.2. User interfaces

The desktop application will need to be compatible with web browsers used by the client. The android application will need to conform to UI capabilities provided by the android operating system environment.

5.3. External system interfaces

Mapping & Data Entry

Mapbox.js - an open-source based js library that allows dynamic map integration and data entry. This will the main map interface that we will be working with and the user will use in the final product.

Node.js - lightweight JavaScript runtime environment. The application will be build using node.js to ensure compatibility and reliability with multiple systems that the user might choose to use.

Express.js - fast web framework for Node.js that will allow our application be accessible from multiple different machines, regardless of the user's location.

Printing

fill-pdf - a Node.js module for filling PDFs. The mobile application will use fill-pdf to fill out PDFs of inspection forms with entered data. We are using this because, as a node module, it will integrate easily into the Ionic application's codebase, and it will provide a lot of the functionality we need.

cordova-print-pdf-plugin - a Javascript library that will allow us to print PDFs in a Cordova application. To print on the Ionic application, we need to use this library.

6. Detailed design

6.1. Introduction

This section provides the internal details of mapping and data entry in the desktop application, and printing in the tablet application. Descriptions for any components, sub-components, functions, and classes are provided with detailed behavior specifications for each.

6.2. Data Entry

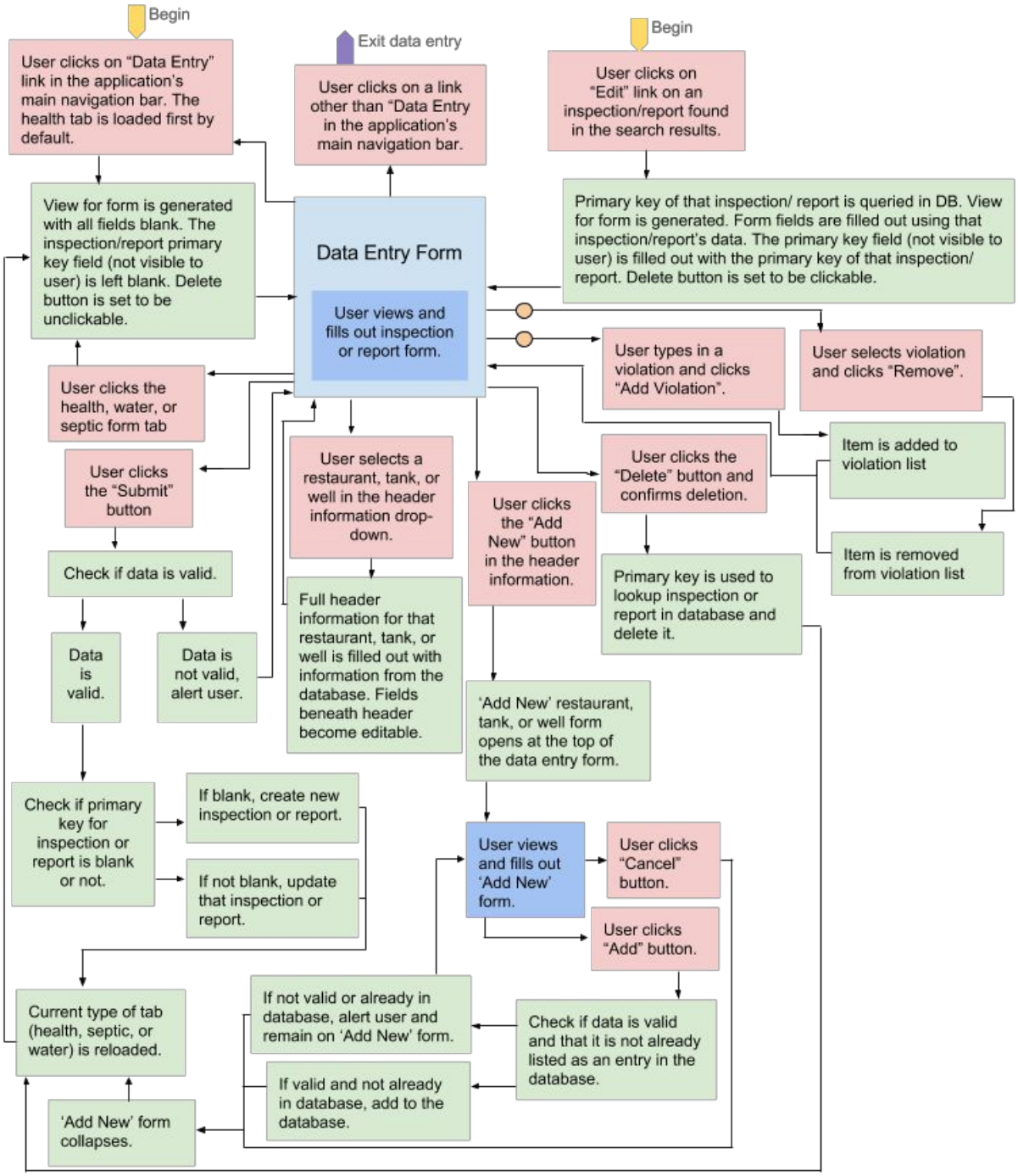
The data entry view will contain all forms for entering health inspections, septic tank reports, water well reports, and all forms for adding new restaurants, septic tanks, and wells to the database. It will be created using node.js and express.js.

The following is a diagram for the general behavior of the data entry form, and how the software will react to actions by the user. The key for the diagram is as follows:

- **Red** represents actions taken by the user
  - All boxes which begin with the word “User”
- **Green** represents actions performed by the software
  - All boxes aside from “Data Entry Form” which do not begin with the word “User”
- **Light Blue** represents an application view
  - The box labeled “Data Entry Form”
- **Dark Blue** represents user actions on the view which do not require extra software actions to be called
  - The two boxes which begin with “User views and fills out”
- **Orange** represents an action only accessible within the health form
  - Represented as circles on top of arrows
- **Yellow** represents entry points into the data entry form
  - Pentagons pointing down, “Begin”
- **Purple** represents exit points from the data entry form
  - Pentagon pointing up, “Exit Data Entry”
- **Arrows** represent the order of actions taken by both the user and the software

Notes:

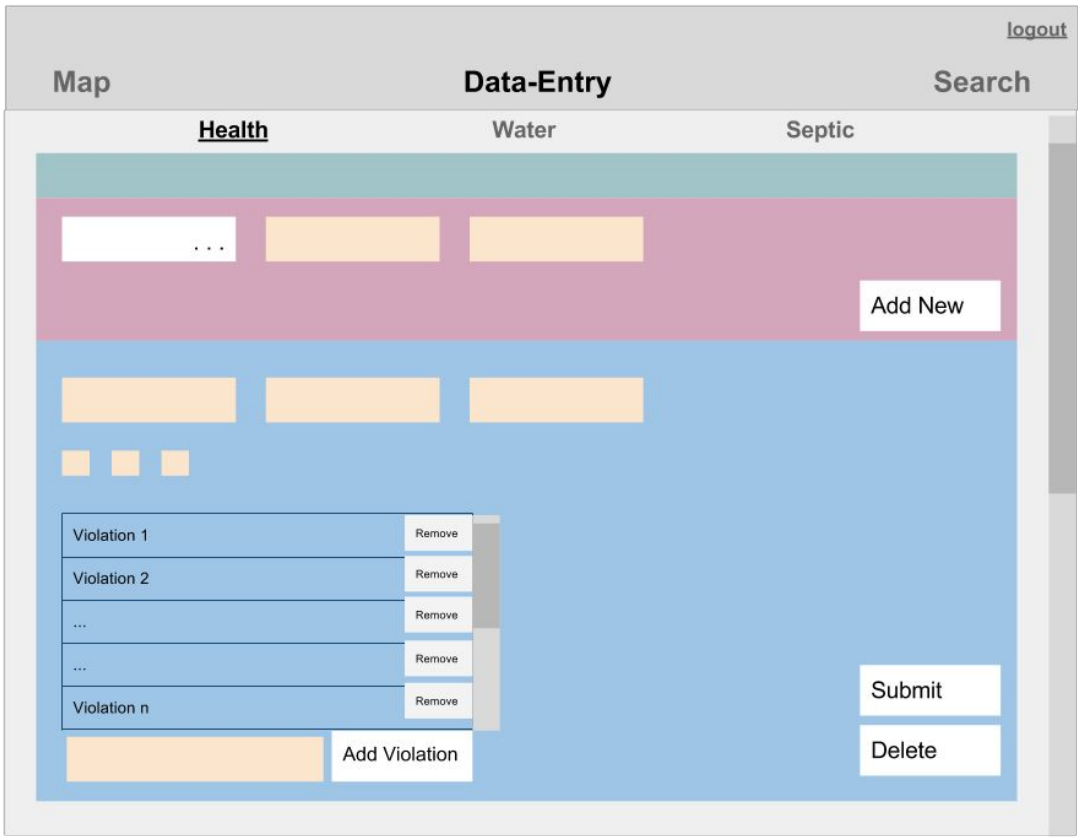
- While the behavior is the same for each of the three tabs, the specific fields in each type of form will vary.
- The “Add New” form above each header will differ depending on which form it is in. The health inspection form will have “Add New Restaurant”, the septic tank report will have “Add New Septic Tank” and the water well report will have “Add New Well”.


Begin
User clicks on "Data Entry" link in the application's main navigation bar. The health tab is loaded first by default.
Exit data entry
User clicks on a link other than "Data Entry in the application's main navigation bar.
Begin
User clicks on "Edit" link on an inspection/report found in the search results.
View for form is generated with all fields blank. The inspection/report primary key field (not visible to user) is left blank. Delete button is set to be unclickable.
Data Entry Form
User views and fills out inspection or report form.
Primary key of that inspection/ report is queried in DB. View for form is generated. Form fields are filled out using that inspection/report's data. The primary key field (not visible to user) is filled out with the primary key of that inspection/ report. Delete button is set to be clickable.
User types in a violation and clicks "Add Violation".
User selects violation and clicks "Remove".
User clicks the health, water, or septic form tab
User clicks the "Submit" button
User selects a restaurant, tank, or well in the header information drop-down.
User clicks the "Add New" button in the header information.
User clicks the "Delete" button and confirms deletion.
Item is added to violation list
Item is removed from violation list
Check if data is valid.
Data is valid.
Data is not valid, alert user.
Full header information for that restaurant, tank, or well is filled out with information from the database. Fields beneath header become editable.
Primary key is used to lookup inspection or report in database and delete it.
'Add New' restaurant, tank, or well form opens at the top of the data entry form.
Check if primary key for inspection or report is blank or not.
If blank, create new inspection or report.
If not blank, update that inspection or report.
User views and fills out 'Add New' form.
User clicks "Cancel" button.
User clicks "Add" button.
Current type of tab (health, septic, or water) is reloaded.
If not valid or already in database, alert user and remain on 'Add New' form.
Check if data is valid and that it is not already listed as an entry in the database.
If valid and not already in database, add to the database.
'Add New' form collapses.

Basic Wireframe of Concept

The following wireframe represents the primary components of the data entry view with the "Add New (Restaurant/Well/Tank)" form collapsed. The Health tab is used as an example. The Water and Septic tabs will reflect this wireframe with the exception of the Add Violation button and list, which will not exist for those forms and is unique to the Health tab.

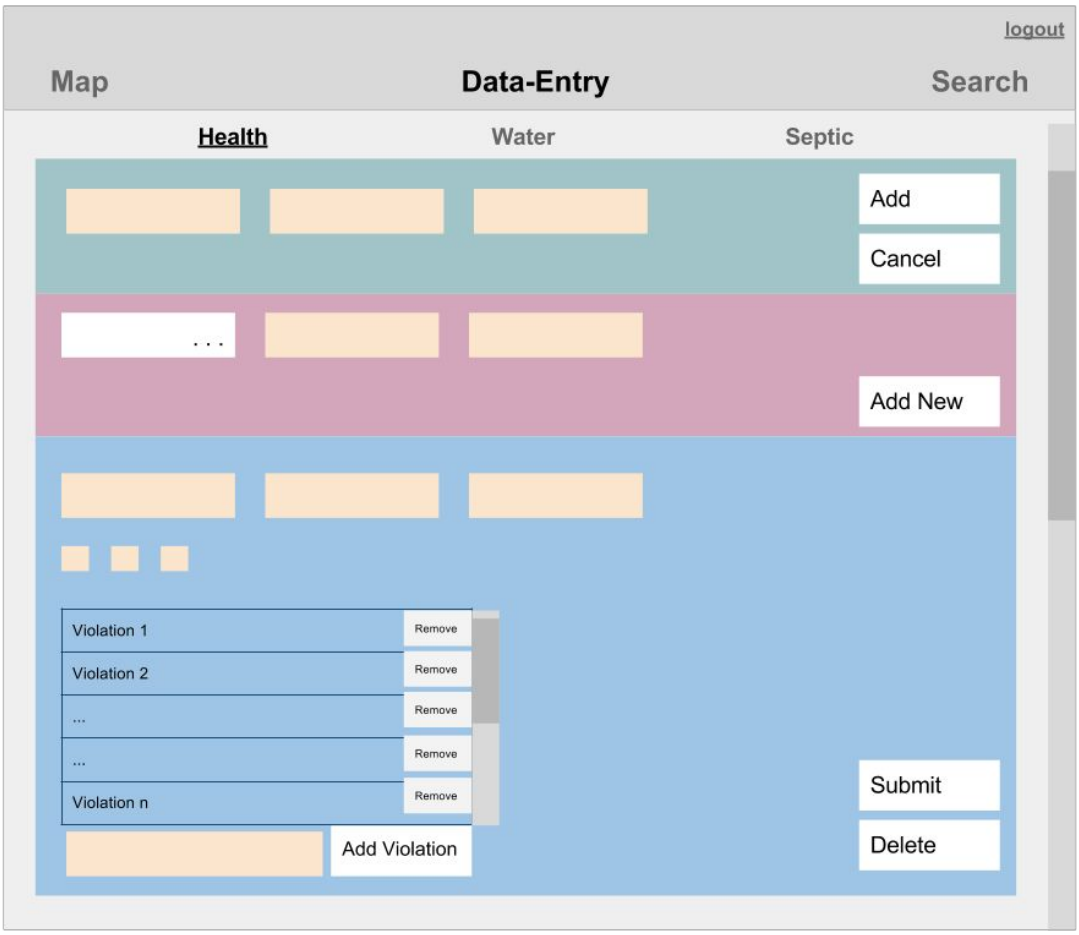
- Logout link ends the user's session
- Map, Data-Entry and Search links navigate between views in the application
- Health, Water, and Septic links display each type of form
- Green box at the top represents the collapsed "Add New" form
- Red bar in the middle represents the header information
- White box with ellipses represents the drop down for selection header information
- White box with "Add New" represents the button which will reveal the "Add New" form above in the green box (see Wireframe 2 below)
- Yellow boxes represent form fields. These will be specified in complete detail when the database scheme is completed. Only a select few are illustrated to indicate that that piece of the form will have fields
- White box with "Submit" will send the information to the database
- White box with "Delete" will send a request to remove an inspection or report from the database
- Black grid represents the list of violations
- White box with "Add Violation" beside yellow form field adds a code violation to the violation list
- Gray "Remove" button removes a violation from the violation list



Data Entry Wireframe 1

The following wireframe represents the primary components of the data entry view with the "Add New (Restaurant/Well/Tank)" form de-collapsed.

- Yellow boxes in green box represent fields which the user may fill out for a restaurant, well, or septic tank
- White box with "Add" adds a new restaurant, well, or septic tank to the database
- White box with "Cancel" collapses the green box and clears the "Add New" form fields, returning it to the state depicted in Wireframe 1



Data Entry Wireframe 2

Components

All functions mentioned are Javascript functions unless specified otherwise. The format for the application's data will be json unless specified otherwise.

Data Entry View

This will be created as an .ejs file with node.js and express.js. It will contain the health inspection form, water report form, and septic tank report form each as HTML forms. The add new restaurant, add new well, and add new septic tank forms will also exist in this file as HTML forms. All components mentioned regarding data entry will exist within this view of the application.

Tab Navigation Bar (Health, Water, Septic).

This navigation bar will be used to separate the forms into three groups based on health inspections and restaurants, water wells, and septic tanks. It will be created with an HTML list and styled with CSS to create the three separate viewable tabs.

An alternative design for this could have been creating separate views for each of the health, septic, and water well forms, however the tabular option was determined to be the best at this time as it keeps all data entry components contained within one module. This allows the client to easily navigate the application's different data entry forms in one place and helps the development process by not splitting the data entry forms across several parts of the application.

Health Tab & Components

The health tab is the default loading point of the data entry view. It will be labeled with the word "Health" and clicking it will reveal the boxes for Add New, Header Information, and the rest of the health inspection form.

- "Add New" box

This collapsible box will hold the Add New Restaurant Form, Add Button, and Cancel Button. By default it will be collapsed when the user clicks on the Health tab, and can only be opened by clicking the "Add New" button in the header information box (see below).

    - Add New Restaurant Form

      The Add New Restaurant Form is the first item under the health tab. It will be hidden from the user by default within the collapsed div element. This form will include all fields for adding a new restaurant to the Restaurant table of the database, as well as any relevant fields for adding a new location entry to the Location table of the database for that restaurant. The Add New Restaurant Form will be an HTML form.

        - Add Button

          The Add Button will exist at the end of the Add New Restaurant Form. Its functionality is to send the contents of the form as a json object to a method provided by the database, which attempts to add the data to the database.

        - Cancel Button

          The Cancel Button will exist at the end of the Add New Restaurant Form. Its functionality is to close the collapsible div which holds the Add New Restaurant Form and clear the form's fields.

- "Header Information" box

The Header Information box will be a div containing the Header Information Form, Drop Down list, and Add New Button. It will be located underneath the Add New box and above the health inspection form box.

- Header Information Form

  The Header Information Form will include the information regarding the restaurant, as well as any other fields reflected in the header of the physical paper health inspection form used by the client. It will include a drop down selection and an Add New button. All required fields in this form will need to be filled out before the Health Inspection form underneath will become editable.

  - Drop Down List

    This list will include all restaurants in the database for the user to choose from. Selecting an option will populate the fields in the header information form with the appropriate data.

  - Add New Button

    This button will open the Add New Restaurant Form above by de-collapsing the Add New box.

- "Restaurant Health Inspection" box

This div will exist directly underneath the header information box. It will contain the Restaurant Health Inspection Form, the Add Violation Button, Remove Violation Button, Violation List, Submit Button, and Delete Button.

- Restaurant Health Inspection Form

  This form will be an HTML form and will contain all fields (aside from header information) which need to be filled out for a health inspection.

  - Add Violation Button

    This button will exist beside one of the health inspection form fields for violations. Clicking this button will append the violation typed into the field to a list of violations (see Violation List below).

  - Remove Violation Button

    This button will exist beside all violations in the violation list. Clicking it will remove the selected violation from that list.

  - Violation List

    This list will include all violations entered and added using the form field and Add Violation button for violations.

  - Submit Button

This button will submit the entire health inspection form to json format. This json will be sent to a function provided by the database to update or add a health inspection report to the database.

- Delete Button

This button will request confirmation from the user before continuing. If the user confirms to delete, the button will submit a primary key value to the database from a hidden field in the form. If this value is blank, nothing will be deleted. Otherwise the form for that key will be removed from the database.

Functions:

- populate_health_form function

This function takes json data and generates the health inspection form with the fields filled out based on that data. It also fills out the primary key field of the form (not editable by the user) with information from the json data to reference preexisting forms for data editing.

- generate_empty_health_form function

This function generates the form on the health inspection tab with all fields blank.

- validate_new_restaurant_data function

This function checks to see if all required fields are filled out and valid for adding a new restaurant and location before proceeding with submission.

- validate_health_inspection_data function

This function checks to see if all required fields are filled out and valid for a health inspection before proceeding with submission.

- delete_health_inspection function

This function checks to see if there is a primary key listed in the form, and if there is will call the delete function provided by the database.

- confirm_health_inspection_exists

This function uses the form's primary key to check if the pre existing form actually exists in the database.

- insert_health_inspection function

This function adds a new health inspection using functions provided by the database.

- update_health_inspection function

This function updates a pre existing health inspection using functions provided by the database.

Water Tab & Components

The water tab will be labeled with the word “Water” and clicking it will reveal the boxes for Add New, Header Information, and the rest of the water report form.

- “Add New” box

  This collapsible box will hold the Add New Well Form, Add Button, and Cancel Button. By default it will be collapsed when the user clicks on the Water tab, and can only be opened by clicking the “Add New” button in the header information box (see below).

    - Add New Well Form

      The Add New Well Form is the first item under the water tab. It will be hidden from the user by default within the collapsed div element. This form will include all fields for adding a new well to the Well table of the database, as well as any relevant fields for adding a new location entry to the Location table of the database for that well. The Add New Well Form will be an HTML form.

        - Add Button

          The Add Button will exist at the end of the Add New Well Form. Its functionality is to send the contents of the form as a json object to a method provided by the database, which attempts to add the data to the database.

        - Cancel Button

          The Cancel Button will exist at the end of the Add New Well Form. Its functionality is to close the collapsible div which holds the Add New Well Form and clear the form’s fields.

- “Header Information” box

  The Header Information box will be a div containing the Header Information Form, Drop Down list, and Add New Button. It will be located underneath the Add New box and above the water report form box.

- Header Information Form

The Header Information Form will include the information regarding the well, as well as any other fields reflected in the header of the water report currently used by the client. It will include a drop down selection and an Add New button. All required fields in this form will need to be filled out before the water report form underneath will become editable.

- Drop Down List

This list will include all wells in the database for the user to choose from. Selecting an option will populate the fields in the header information form with the appropriate data.

- Add New Button

This button will open the Add New Well Form above by de-collapsing the Add New box.

- "Water Report" box

This div will exist directly underneath the header information box. It will contain the Well Water Report Form, Submit Button, and Delete Button.

- Water Report Form

This form will be an HTML form and will contain all fields (aside from header information) which need to be filled out for a water report.

- Submit Button

This button will submit the entire water report form to json format. This json will be sent to a function provided by the database to update or add a water report to the database.

- Delete Button

This button will request confirmation from the user before continuing. If the user confirms to delete, the button will submit a primary key value to the database from a hidden field in the form. If this value is blank, nothing will

be deleted. Otherwise the form for that key will be removed from the database.

Functions:

- populate_water_form function

This function takes json data and generates the water report form with the fields filled out based on that data. It also fills out the primary key field of the form (not editable by the user) with information from the json data to reference preexisting forms for data editing.

- generate_empty_water_form function

This function generates the form on the water report tab with all fields blank.

- validate_new_well_data function

This function checks to see if all required fields are filled out and valid for adding a new well and location before proceeding with submission.

- validate_water_report_data function

This function checks to see if all required fields are filled out and valid for a water report before proceeding with submission.

- delete_water_report function

This function checks to see if there is a primary key listed in the form, and if there is will call the delete function provided by the database.

- confirm_water_report_exists function

This function uses the form's primary key to check if the pre existing form actually exists in the database.

- insert_water_report function

This function adds a new water report using functions provided by the database.

- update_water_report function

This function updates a pre existing water report using functions provided by the database.

Septic Tab & Components

The septic tab will be labeled with the word “Septic” and clicking it will reveal the boxes for Add New, Header Information, and the rest of the septic report form.

- “Add New” box

This collapsible box will hold the Add New Tank Form, Add Button, and Cancel Button. By default it will be collapsed when the user clicks on the Septic tab, and can only be opened by clicking the “Add New” button in the header information box (see below).

  - Add New Tank Form

The Add New Tank Form is the first item under the septic tab. It will be hidden from the user by default within the collapsed div element. This form will include all fields for adding a new tank to the Septic Tank table of the database, as well as any relevant fields for adding a new location entry to the Location table of the database for that tank. The Add New Tank Form will be an HTML form.

    - Add Button

The Add Button will exist at the end of the Add New Tank Form. Its functionality is to send the contents of the form as a json object to a method provided by the database, which attempts to add the data to the database.

    - Cancel Button

The Cancel Button will exist at the end of the Add New Tank Form. Its functionality is to close the collapsible div which holds the Add New Tank Form and clear the form’s fields.

- “Header Information” box

The Header Information box will be a div containing the Header Information Form, Drop Down list, and Add New Button. It will be located underneath the Add New box and above the water report form box.

- Header Information Form

The Header Information Form will include the information regarding the tank, as well as any other fields reflected in the header of the septic report currently used by the client. It will include a drop down selection and an Add New button. All required fields in this form will need to be filled out before the septic report form underneath will become editable.

- Drop Down List

This list will include all septic tanks in the database for the user to choose from. Selecting an option will populate the fields in the header information form with the appropriate data.

- Add New Button

This button will open the Add New Tank Form above by de-collapsing the Add New box.

- "Septic Report" box

This div will exist directly underneath the header information box. It will contain the Septic Report Form, Submit Button, and Delete Button.

- Septic Report Form

This form will be an HTML form and will contain all fields (aside from header information) which need to be filled out for a septic report.

- Submit Button

This button will submit the entire septic report form to json format. This json will be sent to a function provided by the database to update or add a septic report to the database.

- Delete Button

This button will request confirmation from the user before continuing. If the user confirms to delete, the button will submit a primary key value to the database from a hidden field in the form. If this value is blank, nothing will be deleted. Otherwise the form for that key will be removed from the database.

Functions:

- populate_septic_form function

This function takes json data and generates the septic report form with the fields filled out based on that data. It also fills out the primary key field of the form (not editable by the user) with information from the json data to reference preexisting forms for data editing.

- generate_empty_septic_form function

This function generates the form on the septic report tab with all fields blank.

- validate_new_tank_data function

This function checks to see if all required fields are filled out and valid for adding a new septic tank before proceeding with submission.

- validate_septic_report_data function

This function checks to see if all required fields are filled out and valid for a septic report before proceeding with submission.

- delete_septic_report function

This function checks to see if there is a primary key listed in the form, and if there is will call the delete function provided by the database.

- confirm_septic_report_exists function

This function uses the form’s primary key to check if the pre existing form actually exists in the database.

- insert_septic_report function

This function adds a new septic report using functions provided by the database.

- update_septic_report function

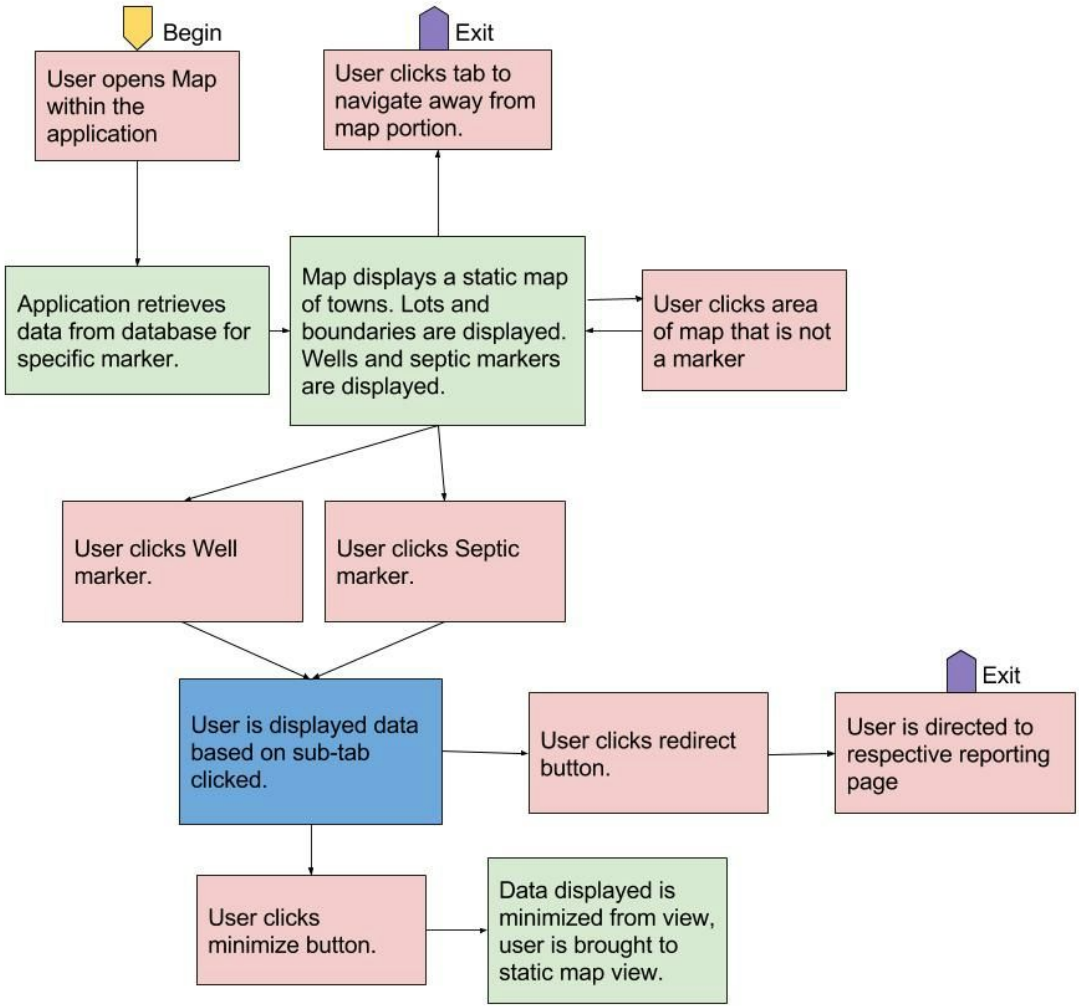
This function updates a pre existing septic report using functions provided by the database.

6.3. MAPPING

Mapping will display all of the wells, septic tanks and lot boundaries that are in the database. The user will be able to click on desired markers that represent either a well or a septic system and view associated data with that specific marker

The following is a diagram for the general behavior of the mapping form, and how the software will react to actions by the user. The key for the diagram is as follows:

- **Red** represents actions taken by the user
  - All boxes which begin with the word “User”
- **Green** represents actions performed by the software
- **Light Blue** represents an application view
  - The box labeled “Mapping View”
- **Dark Blue** represents a pop-up menu
  - “Septic Tank Pop-up”
  - “Well Pop-up”
- **Yellow** represents entry points from the mapping form
  - “Enter Mapping”
  - “Mapping Redirect”
- **Purple** represents exit points from the mapping form
- **Arrows** represent the order of actions taken by both the user and the software



Notes:

- This key differs from the key presented in section 6.2

Components

- All functions mentioned are Javascript functions unless specified otherwise.

Septic

- load_septics_on_map function
- update_septic_tank function
- examine_septic_tank function
- minimize_septic_report function

Water

- load_wells_on_map function
- update_well function
- examine_well function
- minimize_well_report function

Lots

- load_lots function

General

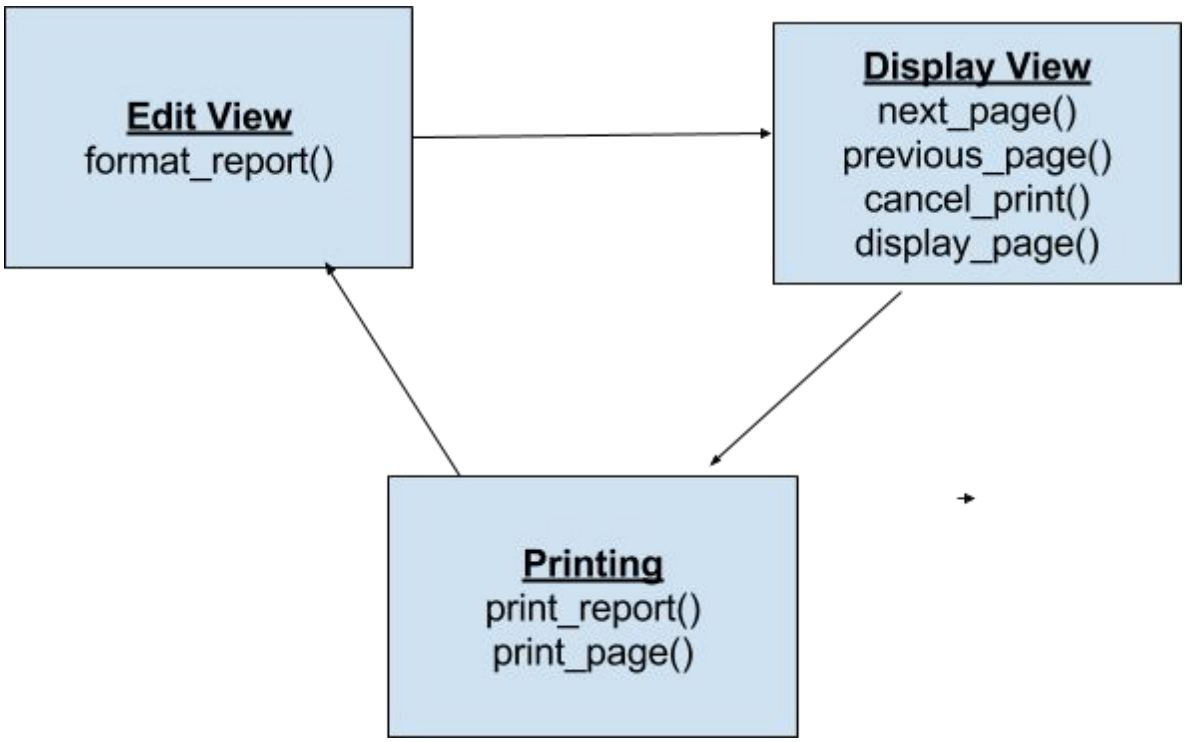
- delete_map_item function
- establish_connection function

- retry_connect_to_database function
- zoom_in function
- zoom_out function
- traverse_map function
- set_map_coordinates function

6.4. PRINTING

Reference 3.1.2 for the anticipated behavior of the printing module.

The printing project in the tablet application will either print an entire report or a selected page in the report. The "Print Report" button on the Edit View will display a page of the formed report using the Display View. Then the user will select "Report Button" to print all pages or "Print Page" to print the specific page only. These buttons will call functions organized into the Printing class.



Components

Edit View

Buttons:

- Print Button

Functions:

- format_report

Display View

Buttons:

- Next Button
- Previous Button
- Cancel Button
- Print Report Button
- Print Page Button

Functions:

- next_page
- previous_page
- cancel_print
- display_page

Printing

Functions:

- print_report
- print_page

7. Annexes

7.1. Traceability

Requirement	Functional Design	System Design	Tests
The user create new reports	Data Entry	Page 10-22	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20
The user edits report data	Data Entry	Page 10-22	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20
The user deletes report data	Data Entry	Page 10-22	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20
The user can print a report	Printing	Page 24	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20
The user can print a specific page of a report	Printing	Page 24	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20

Mark locations	Mapping	Page 23	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20
Display map	Mapping	Page 23	Test Plan: Page 7, 10, 12, 14, 15, 16, 19, 20