PointMap

A Geospatial Mapping Tool

High Level Design Document
(Revised)

Document Drafted By:

Team Name	NightWatch
Team Leader	Julian Poyourow

Julian Poyourow	013466646
Chris Meyer	014520504
Luke Clements	014736434
Ryan Cole	013133768
Alfredo Vargas	014270722

November 11, 2018

Table Of Contents

Introduction **2**

Goals and Rationale for the New System 2

Project Purpose 2

System Goals and Objectives 2

Proposed System 3

Factors Influencing Technical Design **3**

Relevant Standards 3

Assumptions and Dependencies 3

Constraints 3

Proposed System **4**

High Level Architecture 4

Application Architecture 5

Feature Dependencies 7

Network Architecture 8

Technology Architecture 9

Platform 9

System Hosting 9

Connectivity Requirements 9

User Interface Designs 9

Security and Privacy Architecture 10

Authentication 10

Authorization 10

Encryption 10

Introduction

This document describes the proposed high level system architecture of PointMap. We will discuss the factors influencing the technical design and demonstrate the high level architecture of the proposed system, along with the expected requirements. This will be described at a high level to ensure clear communication and a clear understanding of the architecture that is proposed for the software development of the system.

Goals and Rationale for the New System

The goal of the proposed system of PointMap is to create a tool that fills a large gap within the geospatial market. Stakeholders will include, but will not be limited to, clients within the following sectors:

- Landscaping companies and contractors
- Nonprofit urban forestry services
- City forestry services, as well as city water and power services
- National parks and forest services

This proposed architecture is designed in a way that will benefit the possible stakeholders, by offering a system that is lightweight, scalable, and accessible. PointMap is a web application, described within, that will satisfy these goals. This software will be web-based. Frameworks and technologies used to develop this system will allow for future integration of new proposed features and will allow for the application to be scaled to fit the needs of new customers.

Project Purpose

This project will create a new system that allows organizations to better handle large quantities of geospatial data. We have spoken to professionals and representatives within several forestry industries and have determined there is a lack of lightweight geospatial mapping technology available for tracking large areas of individual points at low cost.

System Goals and Objectives

When dealing with a large number of work sites, it is challenging for an organization to keep track of the location, details, and service entries regarding each site.

PointMap will allow clients to track and store information about points of interest in a timely, efficient and easy to use manner. The application will allow entities non-repudiation of work associated with those points.

Proposed System

PointMap is a web application hosted by AWS on a Windows IIS server. The server will adhere to the RESTful principles and allow clients to connect statelessly over HTTPS.

Factors Influencing Technical Design

This section described the standards, assumptions and constraints that influence the technical design of the proposed system.

Relevant Standards

PointMap will follow the industry standards in geospatial mapping, web connectivity, and user data management. The application will accurately show GPS points down to a 4-meter radius with a 95% confidence interval. The application will only use modern encryption standards and proper key sizes when encrypting data in motion. Clients will only be able to connect to the application over port 443/HTTPS. Any connection over port 80/HTTP will be automatically upgraded to HTTPS.

Assumptions and Dependencies

The system will be dependent on AWS for web server hosting. We are additionally assuming clients will be using the latest, or one version prior of Chrome browser. The Map View utility in PointMap will depend on the Google Maps API. We are assuming clients will have JavaScript enabled within their corporate environment.

Constraints

- Application will only have browser support for the latest version of Chrome and one version prior
- Servers will only allow connection originating from North America
- REST architecture constraints
    - Every object and resource in the system must accessible through a unique identifier
    - Web services can only be accessed through HTTP methods (GET, PUT, DELETE, POST, HEAD, OPTIONS, etc)
    - The web server will communicate Statelessly with the client
- Due to our application being a Single Page Application, client-side JavaScript will handle the routing instead of the browser
- Ajax constraints
    - Cannot cancel page navigation

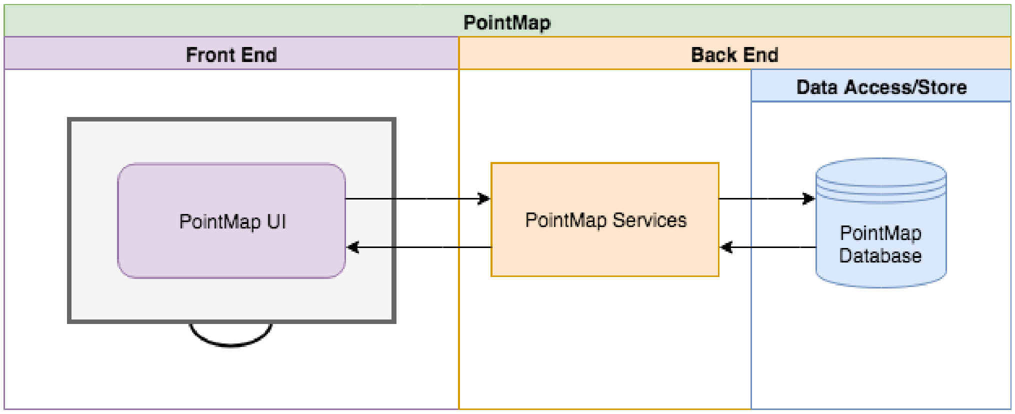
- Will not work if JS is disabled in the browser

Proposed System

The following diagrams will depict the system and the interactions of its features. All of these proposed architectures and designs are susceptible to changes throughout the development process and in accordance to satisfying the client's requirements.

High Level Architecture

This system will take on a Single Page Web-Application architecture. At a high level, this architecture will consist of two main abstractions: the front-end, and the back-end

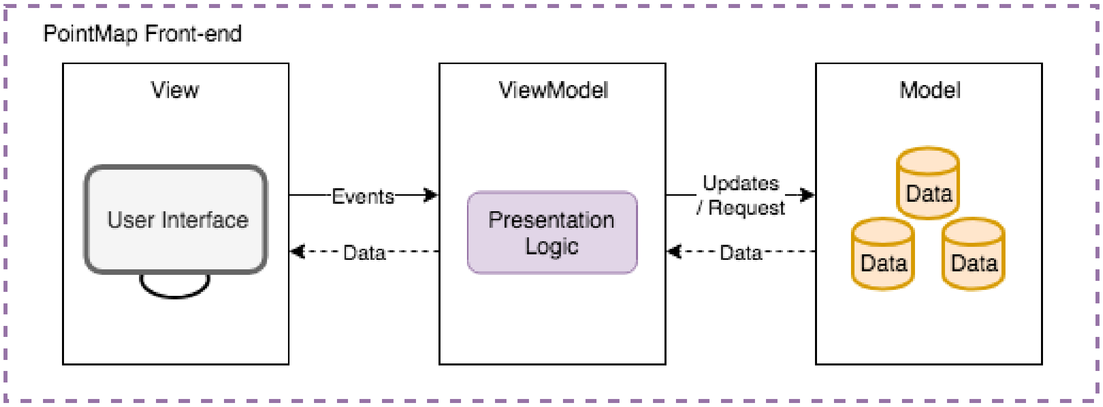


This architecture will influence the overall design and construction of the system. The front-end will consist of the application user interface and interactive components and will follow a MVVM architecture. The back-end will have a Service-oriented architecture which will consists of the application services which will handle user requests and interact with the database to allow for the flow of data between the UI and back-end. The database will consist of the data store and access capabilities of the application.

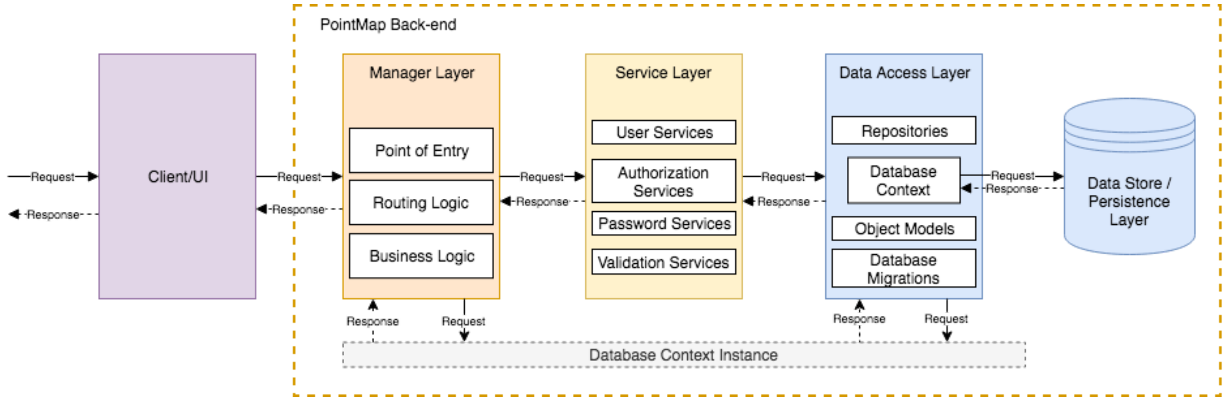
Application Architecture

The application architecture of the proposed system will follow and is directly influenced by the systems high level architecture described above.

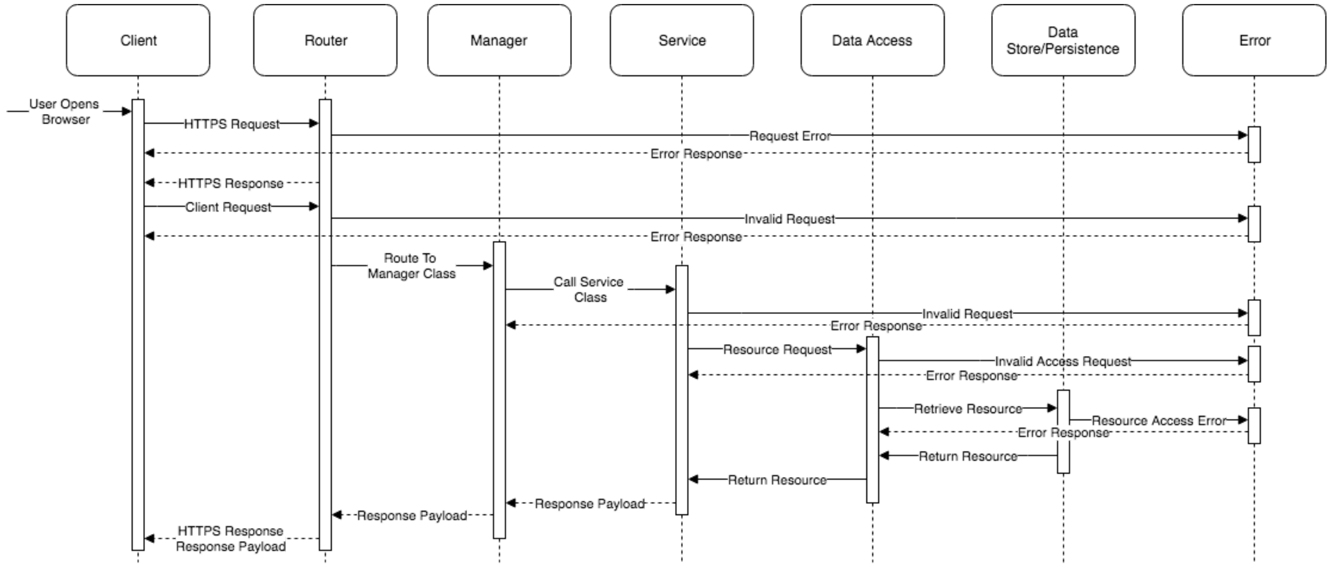
The MVVM (*Model, View, ViewModel*) architecture establishes a design pattern that will be utilized in the development of the application's front-end. The *View* layer displays the content needed to be show to the user, the user interface. The *Model* layer consists of the data objects that have been obtained/created by the application. And the *ViewModel* layer allows for the binding of data between the *View* and *Model* layers and is responsible for handling any request from the client. These layers interact to obtain the needed resources for the client and/or to perform and handle actions from the client via requests and responses sent to one another. This provides encapsulation of system components that are needed to properly manage and maintain the system.



A Service-oriented architecture provides a system design that separates the applications components and services into encapsulated abstraction layers. The Manager Layer contains the system's point of entry, routing logic, and the business logic that is invoked to, at the best of its ability, satisfy the business requirements of the client. The Service Layer will handle the systems modules and single task services that will be used by the application's features. Some of these core services provided within this layer include: user services, authorization services, password services, and validation services. The Data Access Layer will establish the accessibility of data to the services and will include: service repositories, object models, database migrations, and the database context for data access. And the Data Store/Persistence Layer will store the systems data and will provide Create, Read, Update, and Delete (CRUD) functionality.

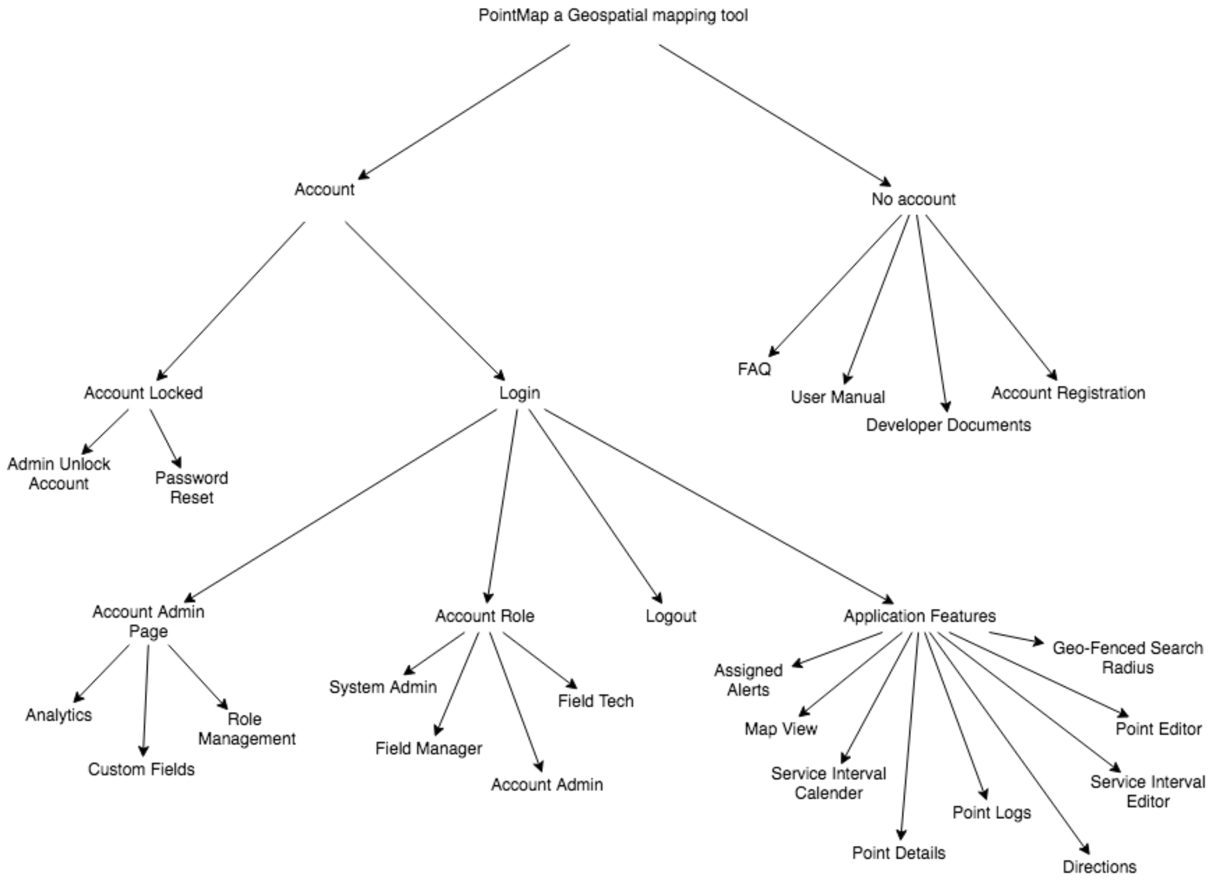


The interaction between all of these layers will provide the functionality and usability of the overall application. Every layer will communicate directly with its immediate layer via requests and responses. Any error that occurs within a layer, will be caught and and an appropriate response will be sent. Below is a high level diagram of how this communication occurs.



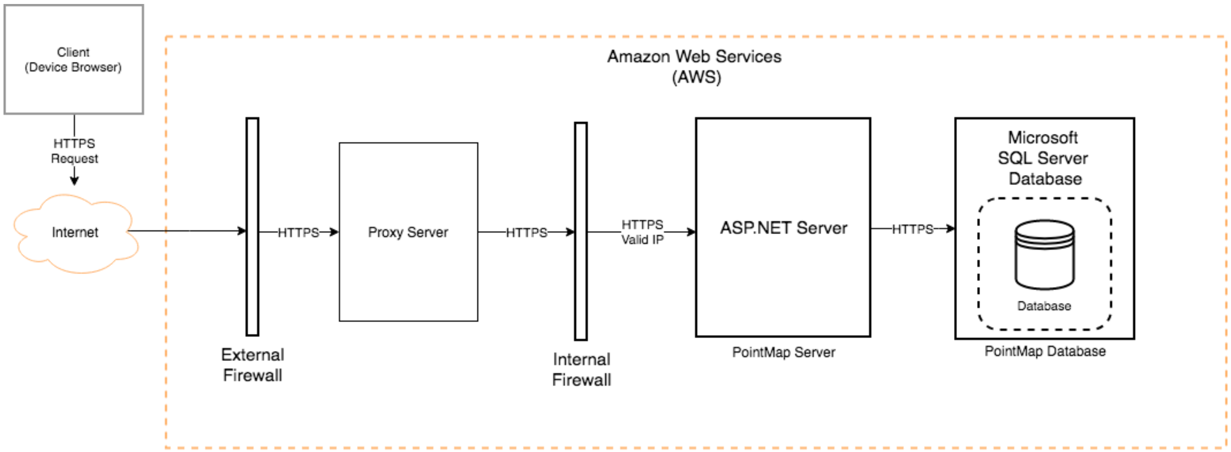
Feature Dependencies

The communication between layers provides the foundation of how the application features will operate and interact within the system. Below, are the core and application features of PointMap. Every child of a feature, as depicted in the tree below, are that features dependencies. This dependency tree shows the relationships between different features of the application.



Network Architecture

This section will demonstrate at a high level the network architecture of the proposed system of PointMap. The network architecture will rely on an internet connection and a web server to handle any request directly from the client.

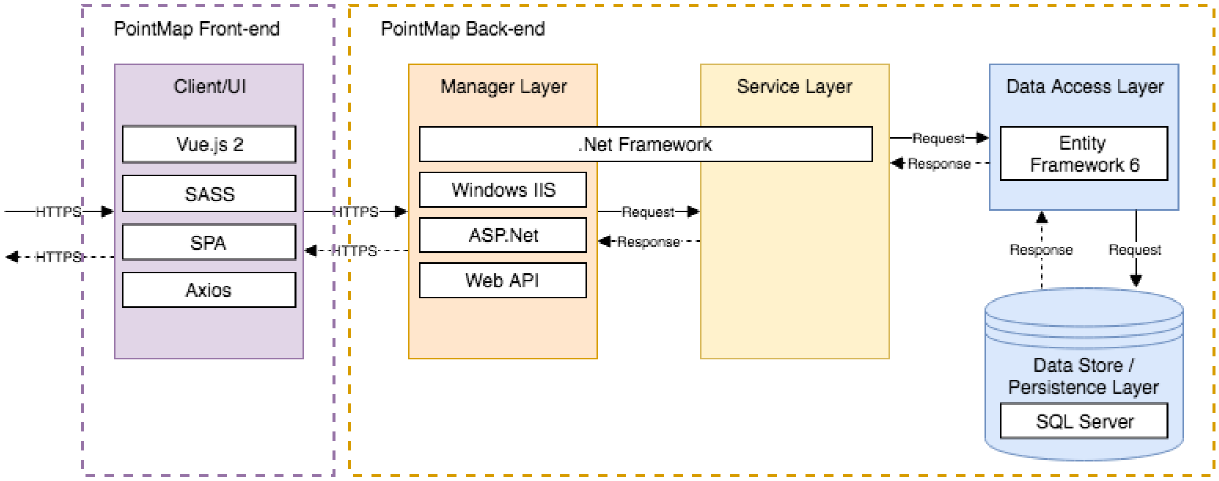


** based on the high level design by Vatank Vong*

The client (any device compatible with a modern browser) will depend on an internet connection to submit any request to the web server. Once a request is received by the web server, this request will first greet the firewall, which according to predefined business rules, will accept or deny certain IP's. The request will then be handled by the applications server which will communicate with the database accordingly to fulfill the systems business logic. This architecture allows for communication through HTTPS between the client and web server, allowing the client to be devices that support web browsers who communicate through HTTPS.

Technology Architecture

The technology architecture will provide a high level overview of the technology that will be utilized when connecting to PointMap



Platform

The application will be running on a Windows IIS server in a Windows Operating System. The IIS will be version 10.

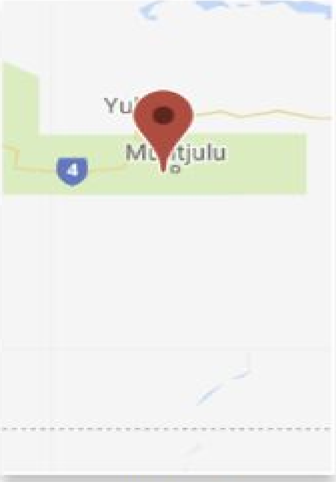
System Hosting

The system will be hosted by Amazon Web Services. The system will run within a Windows IIS environment.

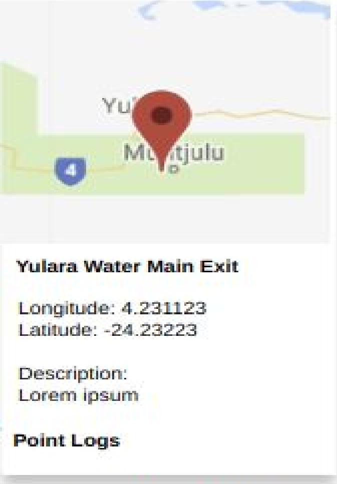
Connectivity Requirements

All connecting clients must connect over HTTP Strict Transport Security, any clients connecting over HTTP will be forced to upgrade to HTTPS. Connections will be limited geographically; only connections originating in North America will be permissible by the IIS server.

User Interface Designs



Map Page



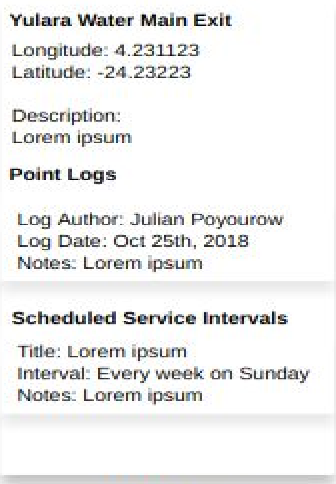
Yulara Water Main Exit

Longitude: 4.231123
Latitude: -24.23223

Description:
Lorem ipsum

Point Logs

Point Details Overlay



Yulara Water Main Exit

Longitude: 4.231123
Latitude: -24.23223

Description:
Lorem ipsum

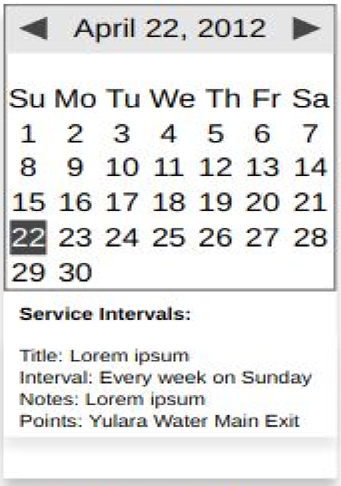
Point Logs

Log Author: Julian Poyourow
Log Date: Oct 25th, 2018
Notes: Lorem ipsum

Scheduled Service Intervals

Title: Lorem ipsum
Interval: Every week on Sunday
Notes: Lorem ipsum

Point Details Full



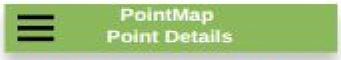
◀ April 22, 2012 ▶

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

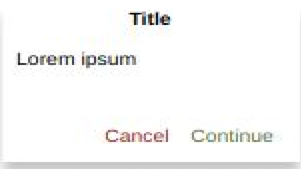
Service Intervals:

Title: Lorem ipsum
Interval: Every week on Sunday
Notes: Lorem ipsum
Points: Yulara Water Main Exit

Service Interval Calendar



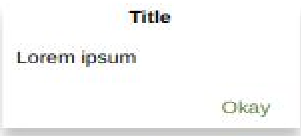
Header (Exists at the top of every page)



Title

Lorem ipsum

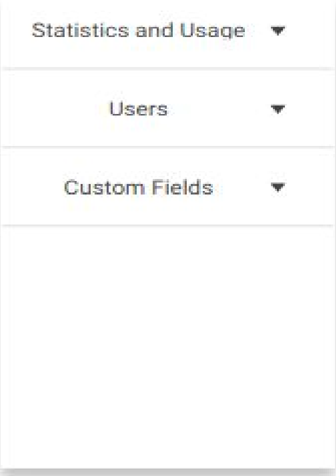
Alert with Confirmation



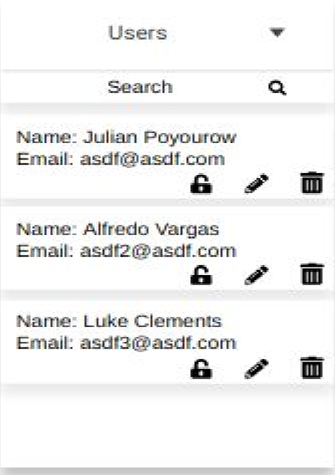
Title

Lorem ipsum

Alert



Root/Admin Dashboard

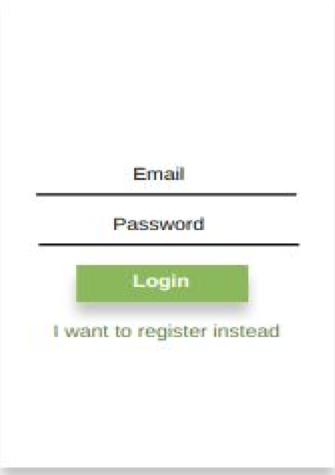


Name: Julian Poyourow
Email: asdf@asdf.com

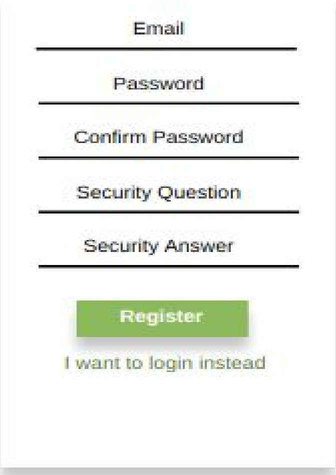
Name: Alfredo Vargas
Email: asdf2@asdf.com

Name: Luke Clements
Email: asdf3@asdf.com

Root/Admin Dashboard - Users



Login



Register

Security and Privacy Architecture

The following sections describe the encryption, authentication and authorization that our system will be utilizing.

Authentication

Initial connection from client to server will prompt a handshake protocol which will verify the identity of the server and client. Clients will only be allowed to access the application through an account. After login clients will be given a session token that will expire if the client has been idle for 30 minutes. Accessing the site with an expired token will require the client to renew their session token.

Authorization

User activity will be regulated using role-based access controls. The user role will provide access to certain features and functionality of our application and restrict features they should not have access to use. The application will follow the principle of least privilege.

Encryption

Any connection with the server will be over HTTPS, ensuring all data in motion from server to client will be encrypted. All user passwords in the database will be stored as hashed and salted values.