**Municipality Permit ChatBot**

**Technical Design Document**

**For UMGC Summer SWEN 670**

**Version 2.0**

**23, July 2020**

**Prepared by UMGC ChatBot Team**

Rusty Baker

Nathaniel Muesing

Joshua Piersol

Matthew Slaymaker  
Sepribo Taylor-Harry  
Subhash Gandhi Vallala

Technical Design Document Approvals

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Approved by:  Dr. Mir Assadullah |  |  |
| Approved by:  "Project Manager" |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Name | Description of Change(s) |
| 1.0 | 11 June 2020 | Nathan Muesing | Initial Release |
| 1.1 | 2 July 2020 | Nathan Muesing | Incorporated Stakeholder feedback. |
| 1.2 | 23 July 2020 | Nathan Muesing, Rusty Baker | Minor modifications to document involving but not limited to the addition of revision history, title page information, and simplification of sentences throughout the document. |
| 2.0 | 05 August 2020 | Nathan Muesing, Rusty Baker | Update of human interface design to reflect changes in design. |
|  |  |  |  |

Contents

[INTRODUCTION 6](#_Toc47559270)

[Purpose 6](#_Toc47559271)

[Scope 6](#_Toc47559272)

[Overview 7](#_Toc47559273)

[Definitions and Acronyms 7](#_Toc47559274)

[SYSTEM OVERVIEW 8](#_Toc47559275)

[SYSTEM ARCHITECTURE 8](#_Toc47559276)

[Architectural Design 8](#_Toc47559277)

[Decomposition Description 9](#_Toc47559278)

[Exception Handling 10](#_Toc47559279)

[Design Rationale 11](#_Toc47559280)

[DATA DESIGN 11](#_Toc47559281)

[Data Description 11](#_Toc47559282)

[Data Dictionary 13](#_Toc47559283)

[COMPONENT DESIGN 15](#_Toc47559284)

[City Official Client Front End 15](#_Toc47559285)

[City Official Client Back End 16](#_Toc47559286)

[Chatbot Component 17](#_Toc47559287)

[HUMAN INTERFACE DESIGN 18](#_Toc47559288)

[Overview of User Interface 18](#_Toc47559289)

[Sign-In Page 18](#_Toc47559290)

[Main Landing Page 19](#_Toc47559291)

[Create New Zone Page 1 20](#_Toc47559292)

[Modify Zone List Page 21](#_Toc47559293)

[Modify Permit Page 22](#_Toc47559294)

[Modify Regulation Page 23](#_Toc47559295)

[Add Permit/Regulation Page 23](#_Toc47559296)

[Deletion Verification Popup 24](#_Toc47559297)

[REQUIREMENT MATRIX 25](#_Toc47559298)

Table of Figures

[Figure 1 – High level visualization of architecture 9](#_Toc46413354)

[Figure 2 -Entity relationship diagram for the MPC database 13](#_Toc46413355)

[Figure 3 Component process flow 16](#_Toc46413356)

[Figure 4 Chatbot dialog flow 17](#_Toc46413357)

[Figure 5 City official sign in page 18](#_Toc46413358)

[Figure 6 City official new sign in page 19](#_Toc46413359)

[Figure 7 Main page for city official client 20](#_Toc46413360)

[Figure 8 Create new zone page 1 21](#_Toc46413361)

[Figure 9 Create new zone page 2 22](#_Toc46413362)

[Figure 10 Modify zone list page 22](#_Toc46413363)

[Figure 11 Modify permits/regulations selection page 23](#_Toc46413364)

[Figure 12 Modify permit information page 24](#_Toc46413365)

[Figure 13 Modify regulation information page 24](#_Toc46413366)

[Figure 14 Add new permits/regulations 25](#_Toc46413367)

[Figure 15 Deletion verification popup 25](#_Toc46413368)

Table of Tables

[Table 1 - Definitions and Acronyms 7](#_Toc46413369)

[Table 2 - Data Dictionary 13](#_Toc46413370)

[Table 3 - Requirements Matrix 26](#_Toc46413371)

# INTRODUCTION

## Purpose

The purpose of this technical design document (TDD) is to describe the system design and architecture of the Municipality Permit Chatbot (MPC). The design and concepts of the MPC are provided so that the development team will have the necessary information to continue forward with an overview of expected performance and system components. Any reference to the application or system within this document refers to the MPC.

## Scope

The scope of this TDD is to provide an initial detailed design of the MPC system for use by the UMGC Chatbot team. This document can be used for verifying and validating the system with stakeholders. The system components covered in this document are as follows:

* City Official Client Front-End– Allows for configuration of the chatbot dialog flow
* City Official Client Back-End – Persistently stores dialog flow
* Chatbot – IBM Watson Assistant cloud instance

Testing characteristics are outside the scope of this document. The test environment and test cases will be covered in the chatbot test plan. Also, certain city zones may preclude specific permit types from applying. In these cases, it is up to city officials to ensure proper compliance and any checks to ensure this is out of scope for this project. In addition, it is also understood that different municipalities vary in the styling of their hosted websites. The styling of the Chatbot is out of the scope of this MPC system. The population and management of chatbot dialog flow, intents, and entities are in scope. This document will be revised based on stakeholder feedback and is subject to change.

## Overview

Information on the system architecture, component design, database schema, and interfaces are detailed in this document. The sections covered in this Requirements from the software requirement specifications will be presented with their implementation for traceability. This document was composed in compliance with the recommended practices outlined in IEEE std 1016-1998 for software design descriptions.

## Definitions and Acronyms

Table 1 - Definitions and Acronyms

|  |  |
| --- | --- |
| **ChatBot** | A computer program designed to simulate conversation with human users. |
| **DevOps** | Development and Operations Team |
| **DFD** | Data flow Diagram |
| **ERD** | Entity Relationship Diagram |
| **GUI** | Graphical User Interface |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **MPC** | Municipality Permit Chatbot |
| **PaaS** | Platform as a Service - a cloud computing model in which a third-party provider supplies the software and hardware infrastructure required to develop, run, and manage an application. |
| **TDD** | Technical Design Document |
| **UMGC** | University of Maryland Global Campus |
| **UI** | User Interface |

# SYSTEM OVERVIEW

The MPC is a system that leverages IBM Watson Assistant for its machine learning capabilities to create a chatbot. Additionally, two interfaces allow city officials to manage available permit information and a chatbot interface for city residents. The city official client interface development is centered on Angular to create an easy to navigate HTML page. Any city can include an HTML snippet with the chatbot script element at the bottom of their web page(s) to add the Chatbot.

City-data is added to the chatbot dialog flow via the city official client. City-data consists of permits and regulations that are tied to zones. OpenLayers is used to allow zones to be created by drawing point to point outlines that can be references to geolocation data.

# SYSTEM ARCHITECTURE

## Architectural Design

The MPC system is made up of three main components. The city official client is an Angular service that is the UI to manipulate data persistently stored by a MongoDB backend. An instance of MongoDB is run locally for development purposes and allows future users of the MPC to migrate later if they wish to use their solution. It may be possible to rely on the Watson Assistant component to avoid needing this database component. However, if there is an issue with the IBM cloud instance, the system would not be able to recover without a persistent data storage solution.

The chatbot component is hosted within IBM's could environment. For the MPC the Node API will be utilized to update and manage the dialog of the Chatbot. See the following figure for the architectural design that covers the three main components.

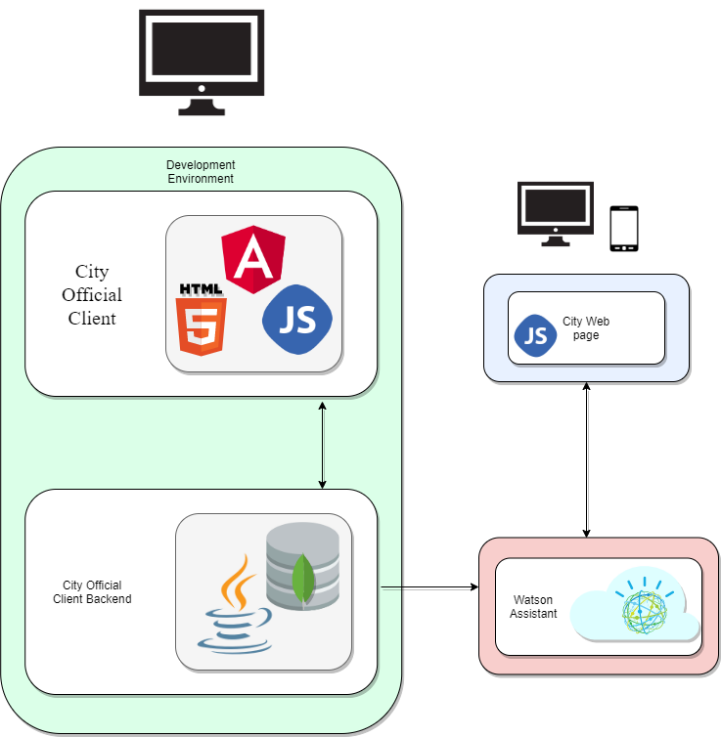


Figure 1 – High-level visualization of architecture

## Decomposition Description

The three MPC system components discussed in the architectural design section fulfill specific roles meeting the requirements levied on the system. The city official client supports the desired chatbot product, while the MPC backend provides redundancy and stability. Each of these components is detailed further below:

* City Official Client Front-End: Is an Angular based component that allows city officials to submit data to populate chatbot branches and dialog responses. Related city permits and regulation web pages that are hosted by the city are linked to zones. Zones can be created and tracked with geolocations utilizing OpenLayers. The OpenLayers API is utilized for both creating zone vectors and for retrieving geolocation data uploaded to OpenLayers. Zone vectors are stored as a series of latitude-longitude coordinate points.
* City Official Client Back-End: This component is made up Express, Node, and MongoDB. Utilizing MongoDB allows data to be stored in a JSON like structure which mirrors the format that Watson Assistant takes in for configuration changes. Data submitted will be lumped together based on zone, permit, and regulation types. This will help with database normalization and reduce the amount of information needed to be submitted by city officials for a functional chatbot.
* Chatbot: Is provided by a cloud instance of Watson Assistant. Functionality is determined based on intents, a dialog topic importance level, and entities. Intents are groupings of responses that are created based on defined categories. Entities are words/terms that prime Watson Assistant with context to respond with. Key words are supplied with permit and regulation types allowing for the machine learning capabilities of the Watson assistant to match the intent with the resident’s query. City residents enter an address which is transformed into a latitude-longitude coordinate point utilizing a webhook to a geocoder. This point is check against zone coordinates to make relevant permits and regulations available to residents.

## Exception Handling

For the MPC system, the three high-level components are coupled until a stable initialization has occurred to populate the chatbot dialog flow. Confidence is placed in city officials' knowledge of permit and zone compliance. As such, in depth checks are outside the scope of this project zoning data entered by city officials is taken as true and accurate. Incorrect data entered can be managed by the city official client and altered. However, incomplete entries for permits, regulations and zones will be rejected. These exceptions will notify the city official what fields are required.

Exception handling for the city resident chatbot will be done through dialog default cases. These will resolve to messages to residents asking for complete addresses or providing city official contact information.

A private alert component in the city official client is used to alert city officials when an exception occurs. This component also blocks users from trying to navigate to other webpages before they have signed in.

## Design Rationale

The goal of the MPC system is to allow a natural flow of responses to meet residents' permit information queries. Requirements levied on the system call for a design that is simple and intuitive for city officials to use while providing accurate information to residents. The MPC design is focused on creating a lightweight solution to manage a chatbot through IBM's Watson Assistant. The design of this system is limited by both the limitations on resources and time available for stakeholder interactions. The focus was centered on solutions that are both open source and utilized by a sizeable group of related industries.

The MPC architecture allows components to be readily hosted locally or moved to be remote. The default setup for MPC is a local environment hosting both the front-end city official client and backend. This default setup was chosen to simplify development. A remote setup can be specified by modifying the Angular environment files of the city official client. To connect to a remote MongoDB instance URLs within each Angular service must be changed. Meanwhile, the utilization of IBM's cloud environment for an instance of Watson Assistant allows a solution that can be tailored to meet the chatbot request volume. An array field is planned for use in the database for zone entities that reference permits and regulations. This was done as permits/regulations can be tied to multiple zones.

# DATA DESIGN

## Data Description

The MPC system data by default will be stored in a local MongoDB server instance. The data model consists of five entities. The entities and their attributes are explained below:

* User: This entity stores the details of a city official that is employed by a city. It includes the first name, last name, email, and password attributes.
* City: This entity stores the essential information and home webpage for that city.
* Zone: This entity stores the zone description and symbol. It references the city the zone falls in as well as the permits and regulations tied to it.
* Permit: This entity stores a description, URL, and permit application URL.
* Regulation: This entity stores a description and regulation URL.

All the entities, except for the association tables, will have a unique ID as the primary key. These primary keys will be autogenerated based on triggers upon successful additions. The two association tables will use two foreign keys as a composite primary key. This was necessary for both permits and regulations as they both can belong to multiple zones, while multiple zones can be linked to a single permit/regulation. See figure 2 below for the MPC entity relationship diagram.

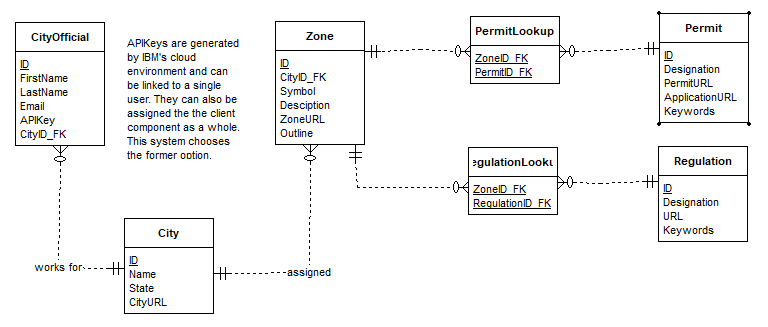


Figure 2 -Entity-relationship diagram for the MPC database

## Data Dictionary

Table 2 - Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity | Field | Type | NULL | Default |
| User | ID | Primary key: uuid | No | uuid |
|  | CityID\_FK | uuid | No | uuid |
|  | FirstName | varchar(35) | No |  |
|  | LastName | varchar(35) | No |  |
|  | Email | varchar(50) | No |  |
|  | APIKey | varchar(100) | No |  |
|  |  |  |  |  |
| City | ID | Primary key: uuid | No | uuid |
|  | Name | varchar(50) | No |  |
|  | State | varchar(2) | No |  |
|  | CityURL | varchar(1500) | No |  |
|  |  |  |  |  |
| Zone | ID | Primary key: uuid | No | uuid |
|  | CityID\_FK | Foreign Key: uuid | No | uuid |
|  | Symbol | varchar(5) | No |  |
|  | ZoneURL | varchar(1500) | No |  |
|  | Description | varchar(250) | No |  |
|  | Outline | Array(Double) | No |  |
|  |  |  |  |  |
| Permit | ID | Primary key: uuid | No | uuid |
|  | Code/Designation | varchar(50) | No |  |
|  | PermitURL | varchar(1500) | No |  |
|  | ApplicationURL | varchar(1500) | No |  |
|  | Keywords | varchar(250) |  |  |
|  |  |  |  |  |
| PermitLookup | (ZoneID\_FK, PermitID\_FK) | Foreign Key: Composite | No |  |
|  |  |  |  |  |
| Regulation | ID | Primary key: uuid | No | uuid |
|  | Code/Designation | Varchar(50) | No |  |
|  | RegulationURL | varchar(1500) | No |  |
|  | Keywords | varchar(250) |  |  |
|  |  |  |  |  |
| RegulationLookup | (ZoneID\_FK, RegulationID\_FK) | Foreign Key: Composite | No |  |

# COMPONENT DESIGN

## City Official Client Front End

The front end of the city official client is built using the Angular framework to provide city officials with an easy to navigate interface. OpenLayers API is used to allow city officials to dynamically draw point to point zone borders on an interactive map. This front-end component loosely follows a model view controller architecture. The difference is that data submitted is both persistently stored in the database and sent to modify the Chatbot. With various Angular elements are divided into components. The figure below displays the components of the front-end client.

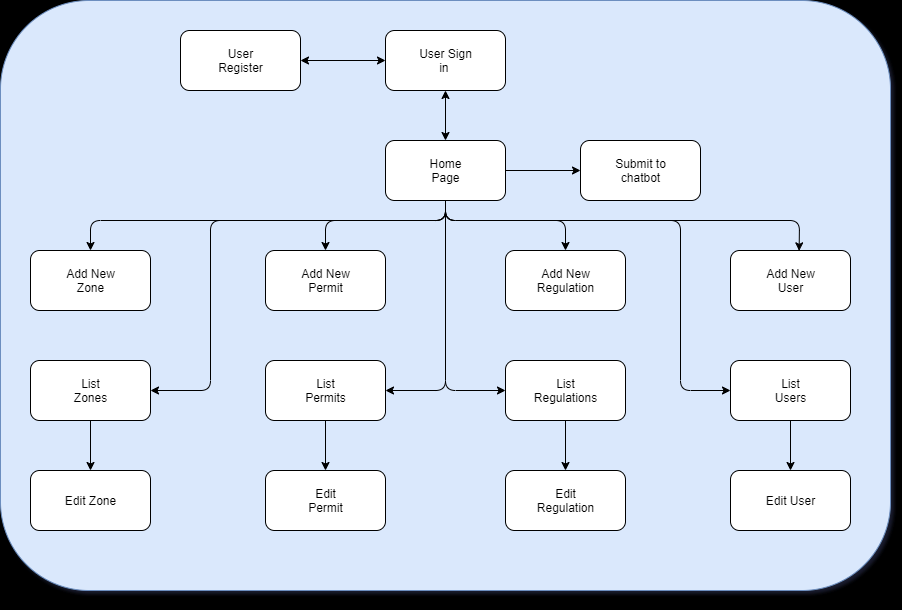


Figure 3 Component process flow

There are four main entities in this client: users, zones, permits, and regulations. The user (city official) components verify and retain the user's information. Editing permits and regulations are done through their own contexts. A zone will reference permit and regulation names. Some components are utilized in multiple cases using a context switch saving resources. This is done for add and edit components. A context switch is used to detect if the component is in add or edit mode.

## City Official Client Back End

The service layer of the UMGC City Application resides within the backend Java microservice and is designed with a Model-View-Controller (MVC) like architecture. Data submitted via the front-end is sent to both to the IBM Cloud and MongoDB instance utilizing Node.js. The backend involves the controller that formatted data to be submitted through the Watson Assistant API.

## Chatbot Component

The MPC chatbot component is an instance of IBM's Watson Assistant. The city official client manages its configuration. As the validity of methods planned to send data through the Watson Assistant API is confirmed, this design is expected to be updated.

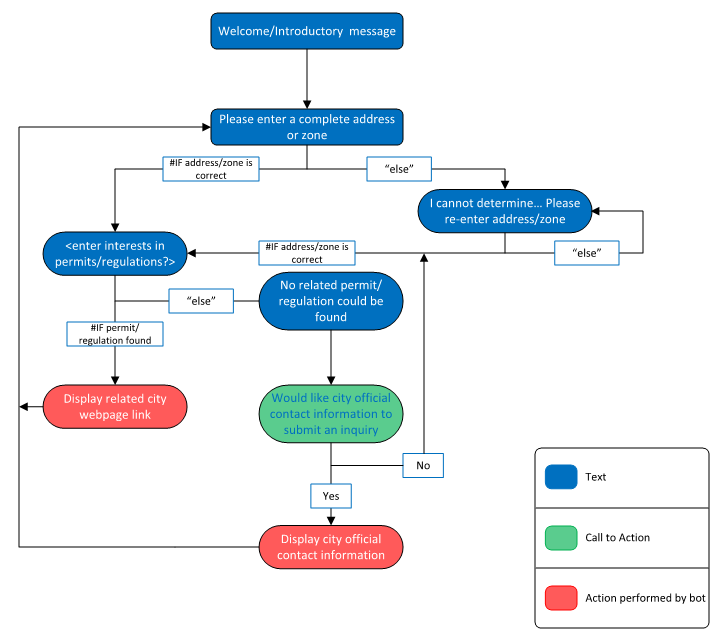


Figure 4 Chatbot dialog flow

# HUMAN INTERFACE DESIGN

## Overview of User Interface

## Sign-In Page

City officials will be greeted by a sign in page. If this is their first use of the MPC they must register a user account to progress.

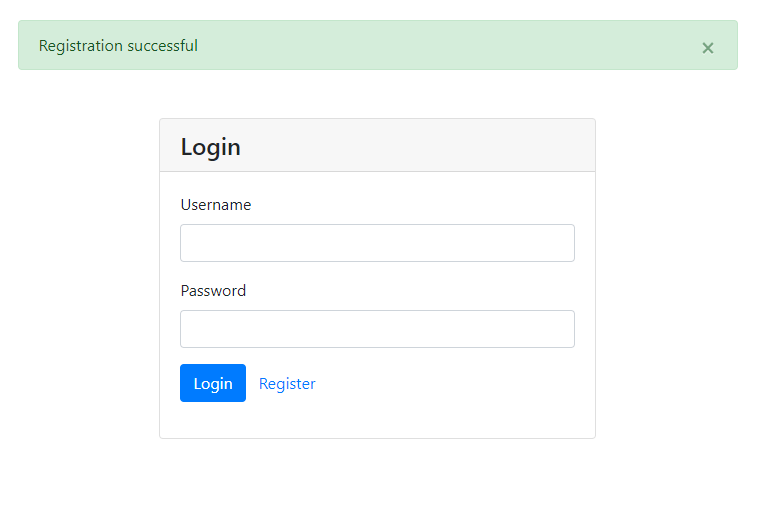


Figure 5 City official sign in page

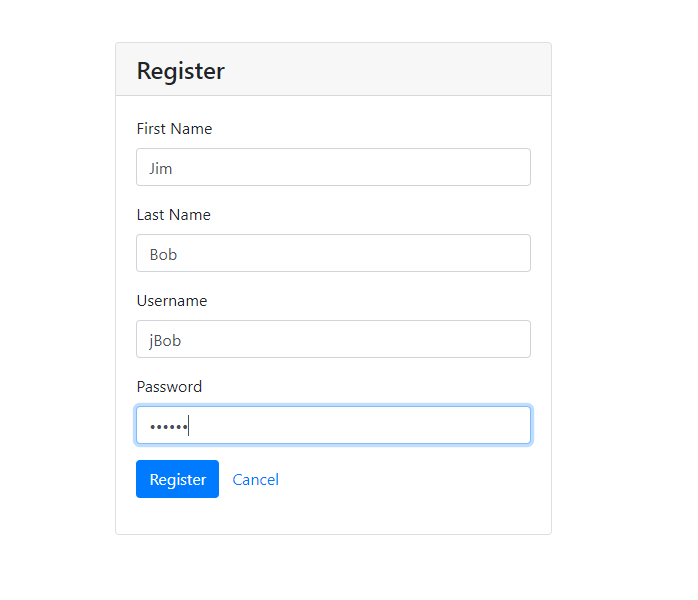


Figure 6 City official new sign in page

## Main Landing Page

The main page will allow city officials to access the multiple functions of the client. A new zone can be created that will ask for permit and regulation information associated with the new zone. Permits/regulations associated with a zone can be modified. A listing of each data type can be seen by navigating using the top navbar or view buttons. A submit to chatbot button triggers the current data to be pushed to the Watson Assistant.

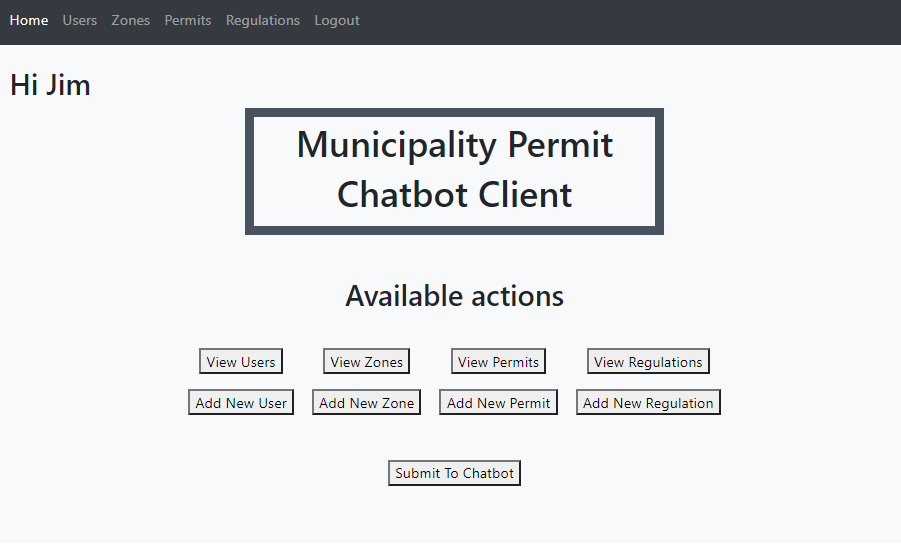


Figure 7 Main page for city official client

## Create New Zone Page 1

When selecting to create a new zone, the city official will be able to outline the zone border. This will be done by drawing a point to point outline. These areas are called vector layers. When a user draws multiple vector layers only the most recently drawn one is used to populate the zone’s coordinates upon submittal. The user can click the clear button to clear all drawn vectors from the map. The relevant zone website URL and Symbol are entered as well before continuing. An optional description can be entered as well. Current permits and regulations will populate and can be selected from checkboxes to be associated with the zone.

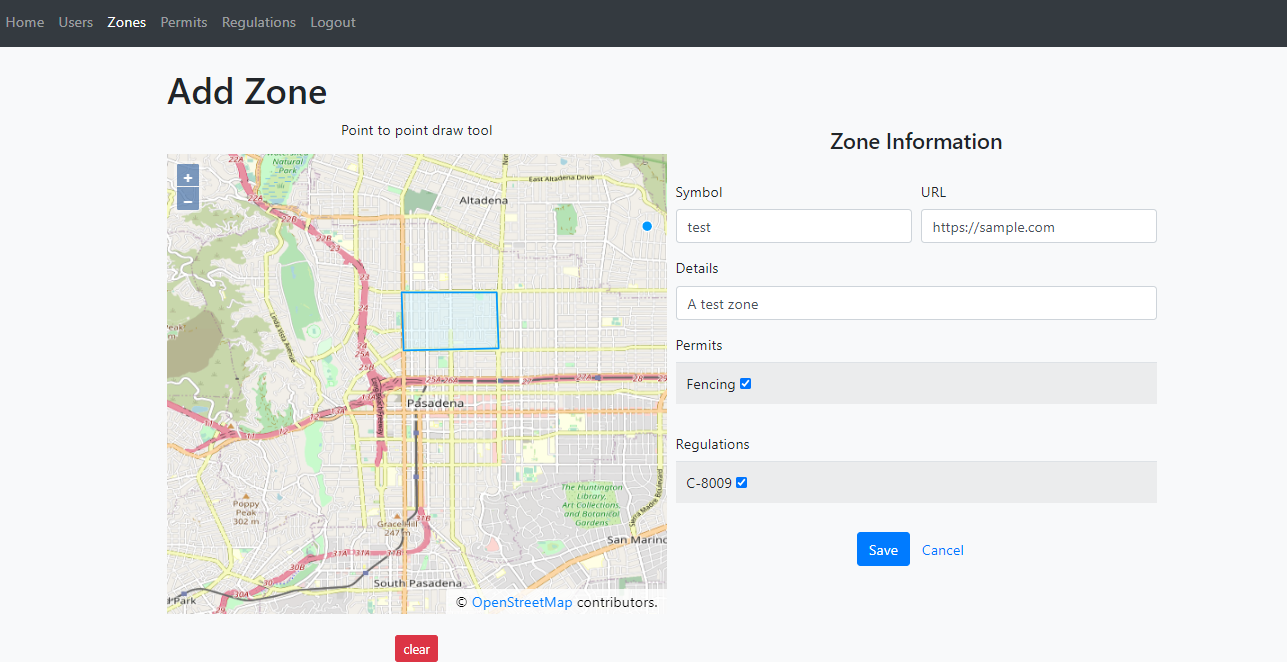


Figure 8 Create new zone page 1

## Zone Modify Page

Data of the zone populates the fields and can be modified. If the user wishes the stored coordinates of the zone are not modified, then they should not attempt to redraw the zone. Zone coordinates will be retained if no new zone is drawn. Current permits and regulations will populate and can be selected from checkboxes to be associated with the zone.

## 

Figure 9 Modify zone

## Zone List Page

To view/modify information associated with a zone a table with brief information is presented. This page does not allow changes to permits or regulations directly. If a permit or regulation name is edited or they are removed each zone must be edited to reflect those changes.

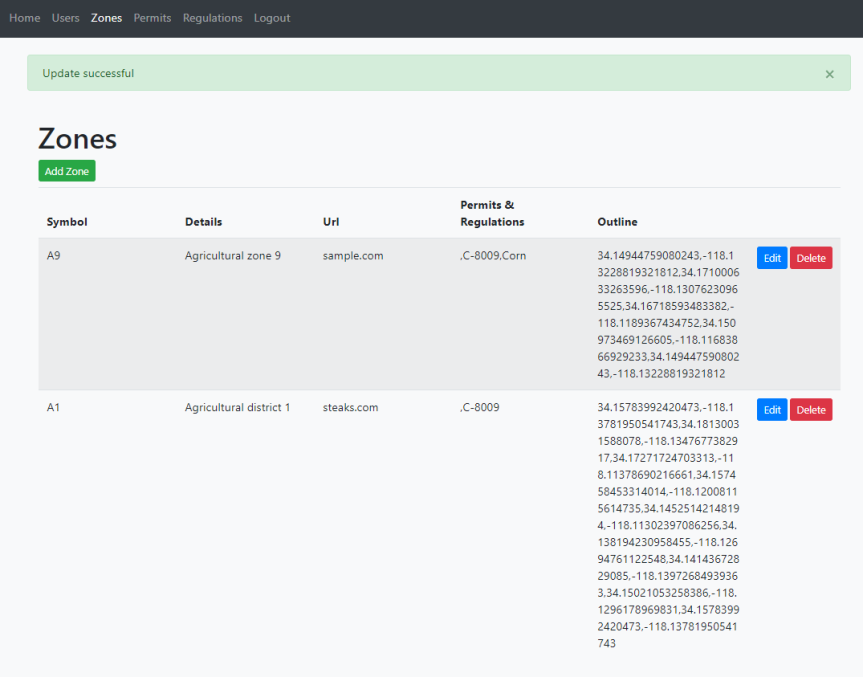


Figure 10 Modify zone list page

## Modify Permit Page

This page allows for modifications to permit information. If the cancel button is pressed no information will be saved. Keywords entered will be parsed and used for creating chatbot intent clues for dialog context.

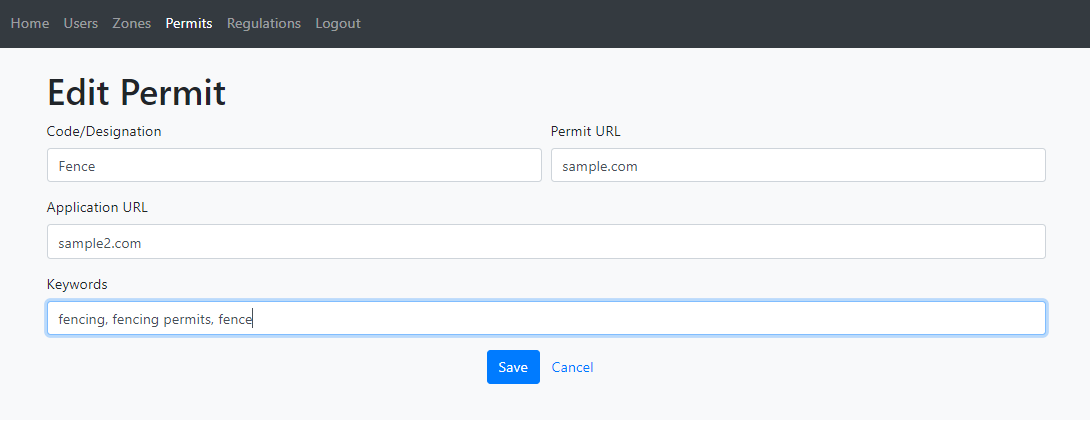


Figure 11 Modify permit information page

## Modify Regulation Page

This page allows for modifications to regulation information. If the cancel button is pressed no information will be saved. Keywords entered will be parsed and used for creating chatbot intent clues for dialog context.

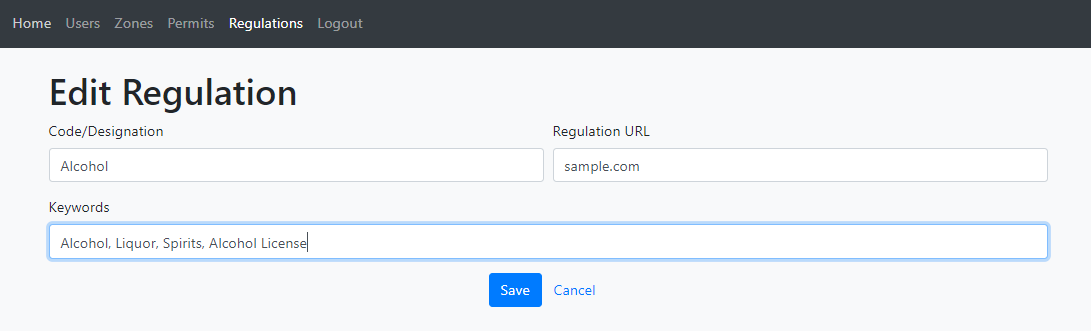


Figure 12 Modify regulation information page

## Add Permit Page

The following page allows new permits to be added. All fields are required to add a new permit.

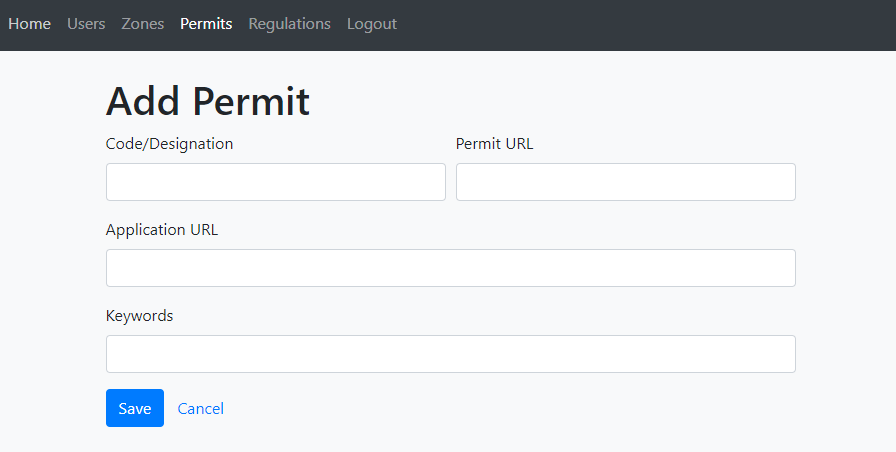


Figure 13 Add new permit

## Add Regulation Page

The following page allows new regulations to be added. All fields are required to add a new regulation.

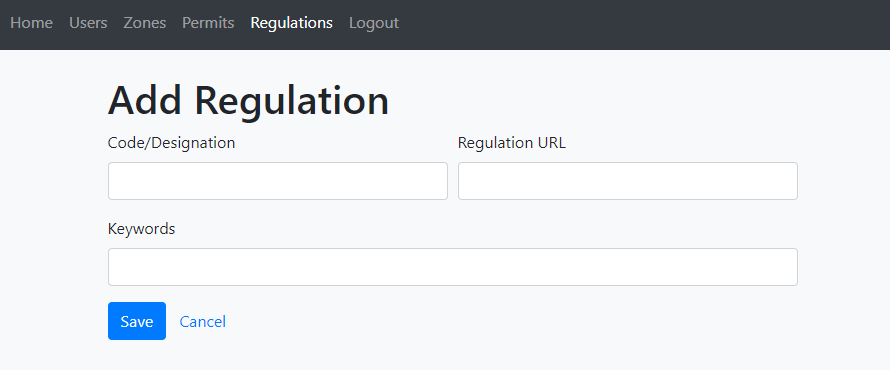


Figure 14 Add new regulations

## Deletion Verification Popup

This popup confirms any deletion with the object that will be deleted shown.

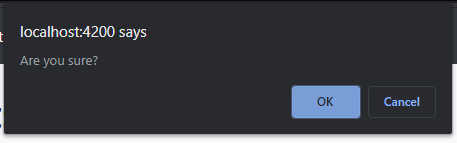


Figure 15 Deletion verification popup

# REQUIREMENT MATRIX

The following table is the requirement matric for the MPC system. This table serves as a cross reference between requirements and related sections. As this document is a living document related sections will be updated when altered.

Table 3 - Requirements Matrix

| **ID** | **Requirement ID** | **Requirement Description** | **System Design Component** |
| --- | --- | --- | --- |
| **1** | REQ 1.1 | Users shall be given a chatbot message box to submit queries. | Chatbot |
| **2** | REQ 1.2 | Users shall be able to submit a query in the message box by pressing enter or selecting a submit button. | Chatbot |
| **3** | REQ 1.3 | Users shall be prompted for a zone or address. | Chatbot |
| **4** | REQ 1.4 | The address provided with be validated by the tool. | Chatbot |
| **5** | REQ 1.5 | A user's entered zone or address shall be used for subsequent permit and regulation queries. | Chatbot |
| **6** | REQ 1.6 | The Chatbot shall provide an appropriate response to the user's queries related to zoning, city regulations, and permits. | Chatbot |
| **7** | REQ 1.7 | The city frequently asked questions page URL shall be displayed when users submit a query that is not relevant to zoning, city regulations, or permits. | Chatbot |
| **8** | REQ 1.8 | The city contact page URL shall be displayed when users submit multiple queries that are not relevant to zoning, city regulations, or permits. | Chatbot |
| **9** | REQ 1.9 | The Chatbot shall offer different zoning, city regulation, and permit informational responses that scale based on input from the city official client. | Chatbot |
| **10** | REQ 1.10 | The Chatbot shall be able to be added to a municipality's website. | Chatbot |
| **11** | REQ 1.11 | The Chatbot shall display help information on its use when the user asks for help in the message box or clicks a help icon. | Chatbot |
| **12** | REQ 1.12 | The Chatbot shall ask for an address when the user submits a query asking for their zone. | Chatbot |
| **13** | REQ 2.1 | The city official client shall allow city officials to add and modify chatbot responses and their corresponding questions. | City Official Client |
| **14** | REQ 2.2 | The city official client shall allow city officials to define zoning areas visually on a presented map. | City Official Client |
| **15** | REQ 2.3 | Defined zoning areas shall be persistently stored. | City Official Client |
| **16** | REQ 2.4 | The city official client shall allow officials to link regulation URLs to regulation responses to user queries. | City Official Client |
| **17** | REQ 2.5 | The city official client shall allow officials to link permit URLs to permit responses to user intents. The user intents will be interpreted from the user queries. | City Official Client |
| **18** | REQ 2.6 | The city official client shall allow officials to associate permits and regulations to a zoning area(s). | City Official Client |
| **19** | REQ 2.7 | The city official client shall be able to connect to a running IBM Watson assistant when an internet connection is possible. | City Official Client |
| **20** | REQ 2.8 | The city official client shall be able to be maintained/packaged in such a way that it can be pulled from a storage solution onto a workstation. | City Official Client |
| **21** | REQ 2.9 | The city official client shall run on Windows 8 and 10 operating systems. | City Official Client |