|  |  |
| --- | --- |
| Technical Design Document  UMGC Fall 2020 | Benjamin Fetterman, Benjamin Murray, Hanim Danur, James Cornelius, Robert Lee  SWEN 670 |

Technical Design Document Approvals

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

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Name | Description of Change(s) |
| 1.0 | 9/29/2020 | Rob Lee | Initial Release |
| 2.0 | 10/12/20 | Benjamin Fetterman | Data Design Updated |
| 3.0 | 11/3/2020 | Robert Lee | Updates based on production version |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1.0 Introduction 5](#_Toc55320762)

[1.1. Purpose 5](#_Toc55320763)

[1.2. Scope 5](#_Toc55320764)

[1.3. Overview 5](#_Toc55320765)

[1.4. Reference Material 5](#_Toc55320766)

[1.5. Definitions and Acronyms 6](#_Toc55320767)

[2.0 System Overview 7](#_Toc55320768)

[3.0 System Architecture 7](#_Toc55320769)

[3.1. Architectural Design 7](#_Toc55320770)

[3.2. Decomposition Description 7](#_Toc55320771)

[3.3. Exception Handling 8](#_Toc55320772)

[3.4. Design Rationale 8](#_Toc55320773)

[4.0 Data Design 9](#_Toc55320774)

[4.1. Data Description 9](#_Toc55320775)

[4.2. Data Dictionary 10](#_Toc55320776)

[5.0 Component Design 13](#_Toc55320777)

[5.3 Chatbot Component 13](#_Toc55320778)

[6.0 Human Interface Design 14](#_Toc55320779)

[6.1. Overview of User Interface 14](#_Toc55320780)

[6.2. Chatbot Not Activated 15](#_Toc55320781)

[6.3. Chatbot Activated 15](#_Toc55320782)

[6.4. Basic Functionality 16](#_Toc55320783)

[7.0 Requirement Matrix 17](#_Toc55320784)

List of Tables

[Table 1 References 5](#_Toc55320785)

[Table 2 Definitions and Acronyms 6](#_Toc55320786)

[Table 3 Data Dictionary 10](#_Toc55320787)

[Table 4 REST Controller 13](#_Toc55320788)

[Table 5 Requirement Matrix 17](#_Toc55320789)

List of Figures

[Figure 1 Database Structure Diagram 10](#_Toc55320790)

[Figure 2 Chatbot Dialog Flow 14](#_Toc55320791)

[Figure 3 Chatbot Not Activated 15](#_Toc55320792)

[Figure 4 Initial Dialog Window 15](#_Toc55320793)

[Figure 5 City Chatbot Functionality 16](#_Toc55320794)

1. Introduction

Purpose

This document, formally known as the Technical Design Document (TDD), provides a detailed overview of the City of Pasadena’s City Chatbot system design and architecture that is used on the UMGC Chatbot application; here onto described as the system, agent, or application. The overall purpose of this document is to explain how the application will work and the architecture used to throughout the application.

### Scope

The scope of this application is defined as an application that takes preestablished city zoning map and associated codes and regulations and make them available for city residents to easily obtain information using a question and answer format chatbot. The chatbot for this application is built using the City of Pasadena California and a web platform chatbot as a test case. The design and implementation of the chatbot is to be done in such a way that any city can take the core architecture, provide their data set, and provide a chatbot like functionality to their residents.

Overview

This TDD is broken down into five major components of the application. These sections describe the overall approach taken to design and develop the Chatbot application and are broken down as follows:

* **System Overview**: The system overview section provides the general description of the Chatbot application, its functionality, the context for which it can be used, and various design choices made.
* **System Architecture**: The system architecture section provides a break down for the modular components within the chatbot application and covers the rationale for the specific breakdown.
* **Data Design**: The data design section describes the makeup and application of the database and its various tables throughout the application. This section additionally defines the structure of the database and the tables used.
* **Component Design**: The component design section describes how each component is used to complete the overall chatbot application.
* **Human Interface Design**: The human interface design section provides an overview of the operation and implementation of the user interface (UI) for the chatbot.

Reference Material

Table 1 References

|  |  |
| --- | --- |
| Title | Reference Location |
| Course Material | UMGC SWEN 670 Course Material |
| UMGC City Team 1 Software Design Document | UMGC One Drive SWEN 670 Spring 202 |
| UMGC City Team 1 Data Base | ec2-3-234-169-147.compute-1.amazonaws.com |
|  |  |

Definitions and Acronyms

Table 2 Definitions and Acronyms

|  |  |
| --- | --- |
| Acronym/Abbreviation | Definition/Reference |
| AI | Artificial Intelligence – An application that aims to mimic human intelligence. |
| API | Application Programming Interface |
| CH | Chatbot |
| CU | Conditional Use Permit |
| DevSecOps | Development, Security and Operations – Group of developers responsible for the deployment and security of an application. |
| ECUP | Expressive Use Permit |
| EPSP | East Pasadena Specific Plan |
| ETL | Extract, transfer, and load |
| FGSP | Fair Oaks/Orange Grove Specific Plan |
| GIS | Geographical Information System – System for working with geographical data. |
| HTTP | Hyper Text Transfer Protocol – A network protocol for specifying how servers and clients communicate with each other. |
| KML | Keyhole Markup Language – A markup language for visualizing geographical data. |
| LASP | Lincoln Avenue Specific Plan |
| MCUP | Minor Conditional Use Permit |
| MVC | Model-View-Controller |
| REST API | Representation State Transfer – An API for interacting with data. |
| SRS | Software Requirements Specification |
| UI | User Interface – The part of the application that users use to interact with the application. |
| URL | Uniform Resource Locator |

1. System Overview

The City Chatbot is an application that provides referential information zone and permit for the city of Pasadena, CA. It combines multiple technologies to execute a technical solution; an AI chatbot, an application layer, database, and webhook services. Together, these solutions coalesce into an application that processes both unstructured text and verbal inputs to fulfill end user requests.

1. System Architecture

This section provides a high-level overview of the system architecture. It identifies significant processes, design consideration, and justifications for use.

### Architectural Design

The system is hosted on a local environment. It comprises of 4 main components. The IBM Watson Chatbot agent, application layer, webhook services, and the database. The IBM Watson Chatbot agent is a virtual agent that processes natural language in text and audio format. The agent converts natural language to structured data for processing. The application layer resides within a Java Spring.io framework, and integrates with MapQuest Geocoding API, the agent, database, and webhooks. Webhook services handles calls to and from Watson and Spring.io, as well as calls to and from MapQuest and Spring.io. The system utilizes a MySQL database engine for data storage.

Decomposition Description

Each system component fulfills a processing requirement. The basic processing requirements to gather user input; process user input; extract, transfer, and load (ETL) processed data; and data storage. The following is a more detailed description of each component.

* IBM Watson Chatbot Agent - The agent categorizes text and audio input as intents. An intent is the end-user’s goal. It is a categorical concept allows development to translate and organize intents into entities. Each intent parameter has an entity type. Entity types dictates how data from the end user’s expression is extracted. Synonym use allows possible variations in user expressions to be categorized as a specific entity. Slots in a dialog node are optional parameters of information that the agent requires from the end-user. Slots within dialog nodes also behave as a quality control measure for gathering information. The default agent responses are static. Fulfillment is the response provided to the end user from the agent. This application will require custom fulfillment responses calling webhook services from the database.
* Application Layer - The application layer is a java spring.io framework designed to allow end users to input data into the agent dialog responses within the dialog branch nodes. It utilizes MapQuest Geocoding API to retrieve the latitude and longitude of the end-user’s street address. The application layer loops through each zone polygon, extending the class to add zone latitude and longitude coordinates with each iteration. Where a Polygon function validates the retrieved latitude and longitude to each data point within each extended polygon. With the correct matching zone, the application layer calls the webhook to push the response to the agent for fulfillment.
* Back-End – The data resides within a relational MySQL database. The MySQL is a standard relational database management system. The database integrates with the application layer receiving calls to progress the application to fulfilling the end user’s request.
* Webhook Services – Webhook services are handlers allowing the application layer to receive real-time information. Webhook services handles calls to and from Watson and Spring.io, as well as calls to and from MapQuest and Spring.io.

Exception Handling

Exception handling allows the system to safely process data outside of system parameters, avoiding crashes, freezing, and other failure scenarios. The system’s exception handling will be performed through the dialogue flow and error handlers throughout the application. All errors will be processed as a general error and the agent will fulfill the user request by communicating a general error.

Design Rationale

The design rational is formed through utilizing solutions configured within the boundaries of project, technical, and functional requirements.

* IBM Watson Chatbot Agent – Due to the nature of the application’s development process. The AI chatbot agent must be free, and accessible by future development teams. The agent is not open source. This allows the application to integrate the IBM agent at no cost, and expands future operability and extensibility.
* Application Layer – The application layer utilizes a java Spring.io framework. Spring.io is a module, enterprise-grade distribution, and widely used framework for building modern applications. It was determined by the primary stakeholder to migrate the application to a java Spring.io framework for operability and extensibility.
* Back-End – MySQL is a common, open source, and widely used relational database engine. The MySQL database engine provides data security, on-demand scalability, round the clock uptime, and other features. It’s wide use and extensive features will avoid future conflicts within compatibility and extensibility.
* Webhook Services – Webhook services employs multiple APIs to handle call backs between the application layer and MapQuest, as well as the application layer and the agent.

1. Data Design

Data Description

The Municipality Permit Chatbot system data will be stored in a MySQL data base server instant provided by DevSecOps. The data model entails five entities, as shown below:

* City: This entity contains the information and webpage for the city. As the team inherited this data base it was decided to keep this artifact, it severs no current purpose.
* City\_User: This entity contains about a city user for a potential back end. As the team inherited this data base it was decided to keep this artifact, it severs no current purpose.
* Zone\_Land\_Use: This entity contains descriptions about the possible use of lands in zones
* Zone: This entity contains the information about the city zone, its description and symbol.
* Development\_Standards: This entity contains information about zones and their development regulations. It has the Foreign Key of Zone\_ID.
* Allowed\_Land\_Use: This entity contains information about permits. It has the Foreign Keys of Zone\_ID and Zone\_Land\_Use\_Id
* Authorities: This entity contains information about the authorities that can access the admin portal. As the team inherited this data base it was decided to keep this artifact, it severs no current purpose.
* Map: This entity contains the latitude and longitude information about the city zones deriving from the KML map. It has the Foreign Key of Polygon\_ID.
* Polygons: This entity breaks down the polygons and the zones derived from the KML map and the zones that they share

The data base is inherited from the SWEN 670 Spring 2020 Chatbot team. As such the team only added two entities, maps and polygons. These two new entities are derived from a KML map of the city of Pasadena. Otherwise the team decided to rely on the data and structure for said that the Spring 2020 chatbot team chose. While there are several artifacts the team has decided to keep them in case back end development has need of these entities.

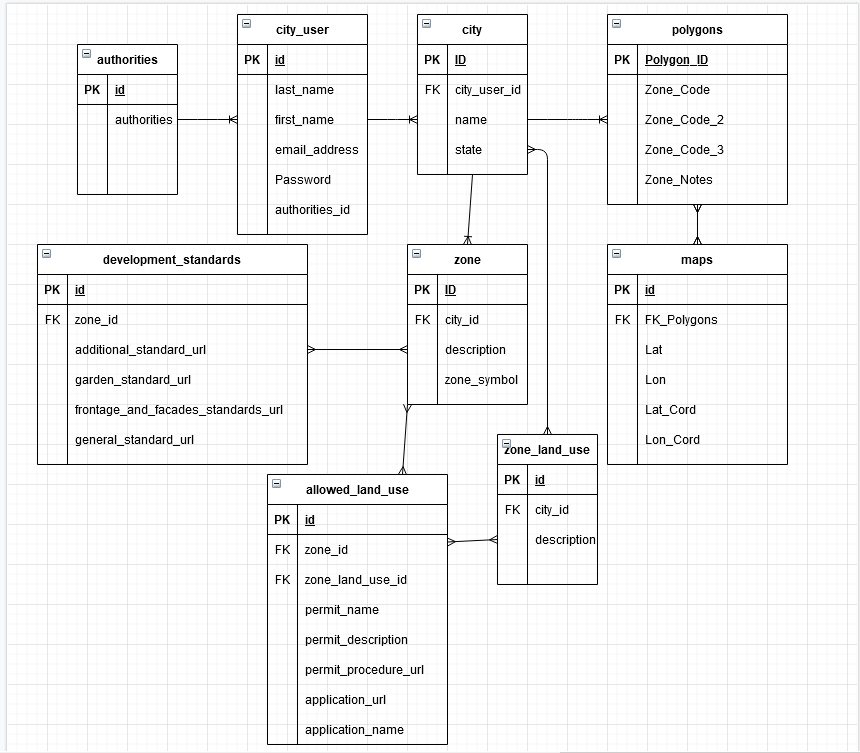


Figure 1 Database Structure Diagram

Data Dictionary

Table 3 Data Dictionary

| Entity | Field | Type | NULL | Default |
| --- | --- | --- | --- | --- |
| city\_user | id | Primary Key: Autogenerated | No | Autogenerated |
|  | last\_name | varchar(30) | No |  |
|  | first\_name | varchar(50) | No |  |
|  | email\_address | varchar(50) | No |  |
|  | password | varchar(100) | No |  |
|  | authorities\_id | Foreign Key: Autogenerated | No |  |
|  |  |  |  |  |
| city | id | Primary Key: Autogenerated | No | Autogenerated |
|  | city\_user\_id | Foreign Key: Autogenerated | No | Autogenerated |
|  | name | varchar(50) | No |  |
|  | state | varchar(50) | No |  |
|  |  |  |  |  |
| zone\_land\_use | id | Primary Key: Autogenerated | No | Autogenerated |
|  | city\_id | Foreign Key: Autogenerated | No | Autogenerated |
|  | description | varchar(1000) | No |  |
|  |  |  |  |  |
| zone | Id | Primary Key: Autogenerated | No | Autogenerated |
|  | city\_id | Foreign Key: Autogenerated | No | Autogenerated |
|  | description | varchar(1000) | No |  |
|  | zone\_symbol | varchar(5) | No |  |
|  |  |  |  |  |
| development\_standards | id | Primary Key: Autogenerated | No | Autogenerated |
|  | zone\_id | Foreign Key: Autogenerated | No | Autogenerated |
|  | additional\_standard\_url | varchar(2000) | No |  |
|  | garden\_standard\_url | varchar(2000) | No |  |
|  | frontage\_and\_facades\_standards\_url | varchar(2000) | No |  |
|  | general\_standard\_url | varchar(2000) | No |  |
|  |  |  |  |  |
| allowed\_land\_use | id | Primary Key: Autogenerated | No | Autogenerated |
|  | zone\_id | Foreign Key: Autogenerated | No | Autogenerated |
|  | zone\_land\_use\_id | Foreign Key: Autogenerated | No | Autogenerated |
|  | permit\_name | varchar(100) | No |  |
|  | permit\_description | varchar(1000) | No |  |
|  | permit\_procedure\_url | varchar(2000) | No |  |
|  | application\_url | varchar(2000) | No |  |
|  | application\_name | varchar(50) | No |  |
|  |  |  |  |  |
| authorities | id | Primary Key: Autogenerated | No | Autogenerated |
|  | authority | varchar(50) | No |  |
|  |  |  |  |  |
| maps | id | Primary Key: Autogenerated | No | Autogenerated |
|  | FK\_Polygons | Foreign Key: Autogenerated | No | Autogenerated |
|  | Lat | Double | No |  |
|  | Lon | Double | No |  |
|  | Lat\_Cord | Int | No |  |
|  | Lon\_Cord | Int | No |  |
|  |  |  |  |  |
| polygons | Polygon\_ID | Primary Key: Autogenerated | No | Autogenerated |
|  | Zone\_Code | varchar(100) | No |  |
|  | Zone\_Code\_2 | varchar(100) | Yes |  |
|  | Zone\_Code\_3 | varchar(100) | Yes |  |
|  | Zone\_Note | varchar(50) | No |  |

1. Component Design

**5.1 Service Layer**

The service layer of the application resides within the back-end Java and is designed with Model-View-Controller (MVC) like architecture. Model design architecture follows MVC design pattern to separate the application on layer from the user interface. Spring-MVC-framework provides the necessary annotation and structure required to create the Restful service. The service module is designed with a REST API controller class and seven data model classes.

**5.2 REST Controller**

The system’s CityZoneProjectContoller class is used to define which service entry point corresponds to a given HTTP URL, and how parameters are to be read from the HTTP request. The REST Controller is designed with the following API endpoints:

Table 4 REST Controller

| GET Method and Endpoint | Definition |
| --- | --- |
| GET/city-zoning-project-management/cities/{id}/zones | Retrieves all zones belonging to a city |
| GET/ city-zoning-project-management/city/{id}/zones/{id}/permits | Retrieves the permit regulations in a city zone |
| GET/ city-zoning-project-management/city/{id}/permits/{id}/ regulation | Retrieves the information about city Regulations. |
|  |  |

### 5.3 Chatbot Component

The figure below displays the component of the front-end client submitted via the front end to/ the IBM Watson chatbot and the MySQL Databa base assistance. The back end involves the controller that formatted data to be submitted through the Watson Assistant API. The Municipality permit chatbot component is an instance of IBM’s Watson Assistant. Its configuration is managed by the city official client. As the validity of methods planned to send data through the Watson Assistant API are confirmed this design is expected to be updated.

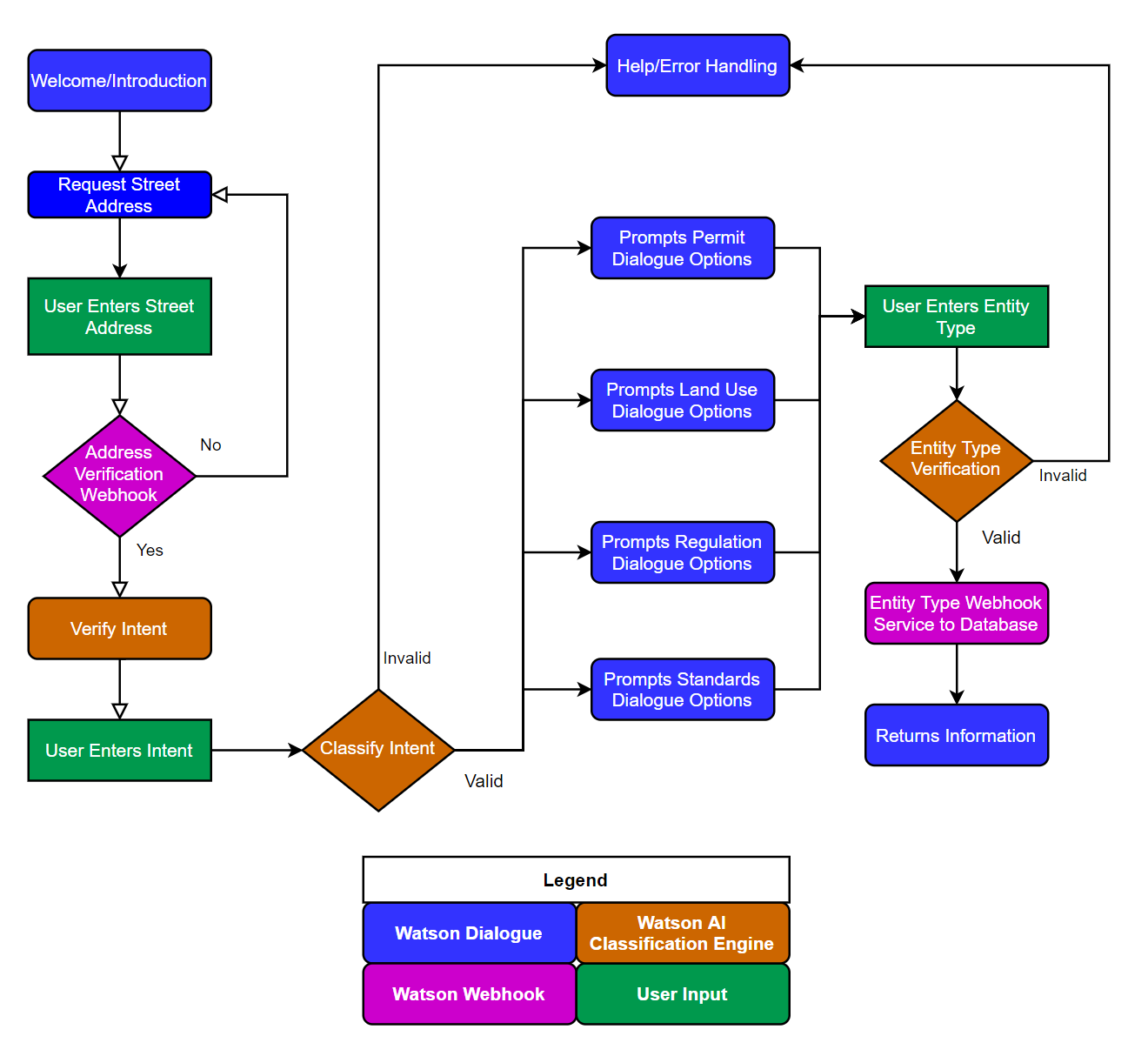


Figure 2 Chatbot Dialog Flow

1. Human Interface Design

Overview of User Interface

Below are a series of prototypes for the Municipal Permit Chatbot System. These prototypes represent the expected interactions that residents will have with the chatbot. The designs are not finalized and could change as development continues.

### Chatbot Not Activated

The chatbot is shown as a small circle on the bottom right of the webpage when it is not activated. When a resident clicks on the circle to activate the chatbot, a dialog window will open for the resident to communicate with the chatbot.



Figure 3 Chatbot Not Activated

Chatbot Activated

The chatbot displays a dialog window once activated. The chatbot greets the resident and asks them to first enter their address before asking any questions. The resident’s address is needed in order to provide the correct permit and regulation information for the resident. Residents can type their input at the bottom of the dialog window or click on the microphone icon to speak their input. Messages from the chatbot are indicated by the dark blue background. Residents can close the dialog window by either clicking the chatbot icon in the top left corner or by clicking outside of the dialog window.

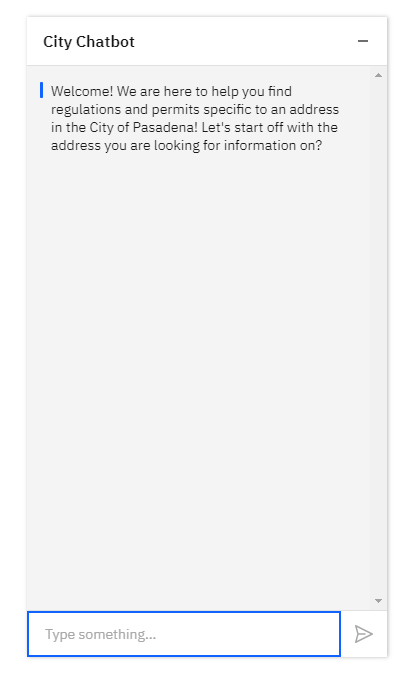
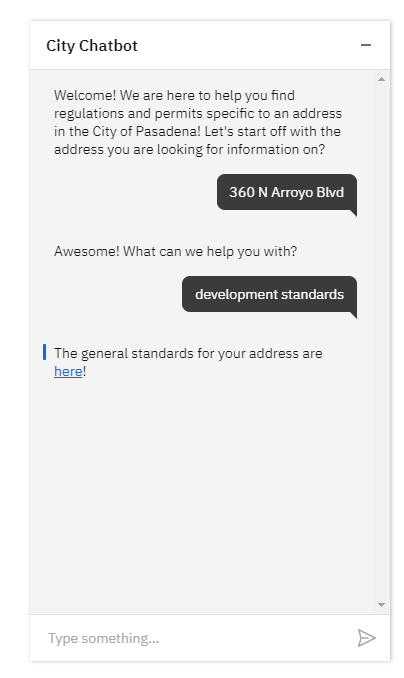


Figure 4 Initial Dialog Window

Basic Functionality

The agent has common basic features of many chatbot applications. Below is a list of those features:

* Text Entry Box – Allows the user to enter text.
* Send Button – Allows the user to commit entered text. Alternatively, the “enter” button can be pressed when the text entry box is activated.
* Chatbot Response Bubbles – Agent responses based on user input.
* User Entry Bubbles – Text representations of the user’s entries.
* Minimize Button – Minimizes the agent’s window.
* Fulfillment of Requests – Often in the form of a hyperlink to the URL of the customer request.



**Minimize Button**

**Welcome Message**

**User Entry Bubble**

**Fulfillment**

**Text Input Box**

**Send Button**

Figure 5 City Chatbot Functionality

1. Requirement Matrix

Functional requirements are descriptions of what the system shall do. They describe the intended behavior of the system. Below are the functional requirements. These requirements and system’s functional design are described in more detail within the Software Requirements Specification (SRS).

Table 5 Requirement Matrix

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| CH-1 | Chatbot | The system shall have a chatbot. |
| CH-2 | Chatbot – Greeting | The chatbot shall have a greeting. This greeting shall direct the input of the chatbot user. |
| CH-3 | Chatbot – Allowed Land Uses | The chatbot shall provide zoning and permit links and information. Such as:   * Definition * Land use category * Variance * Entitlement process where applicable for Conditional Use permit (CU), Minor Conditional Use Permit (MCUP), and Expressive Use Permit (EUP) |
| CH-4 | Chatbot – Application Link | The system shall be able to search for common application permits and forms. Such as:   * Short term rental * Home occupation permit * Variance/minor variance, * Conditional Use and Minor Conditional Use * Accessory dwelling unit * Accessory structure * Zoning * Certificate of Appropriateness (CoA) * Tree removal * Reasonable Accommodation Permit |
| CH-5 | Chatbot – Retrieve Regulations | The system shall be able to retrieve regulations. Such as:   * Short term rental * Home Occupation permit * Occupation riding regulation * Occupation license regulations * Accessory structure * Zoning * Home occupation sanitation * Accessory dwelling units * Projects subjected to CoA * Sober living facility operations |
| CH-6 | Chatbot – Retrieve Standards | The system shall be able to retrieve standards for developing structures. Such as:   * LASP Development Standards * EPSP Development Standards * FGSP Development Standards * Waiver of development standards * Accessory dwelling units * Daycares * Lots |
| CH-7 | Chatbot – Retrieve Permits Applicable to Zoning Statutes | The system shall be able to retrieve permits allowable within a specific zone, based off of user input of the address information. |
| CH-8 | Chatbot – GIS maplayer integration | The chatbot shall be linked to the GIS maplayer sharing identified geolocation information. |
| CH-9 | Chatbot – Address Requirements | The chatbot requires the following information to retrieve zoning and permit information:   * Street Address * City * State * Zip code |
| CH-10 | Address Validation | The system shall validate the address information against existing zoning and permit information. |
| CH-12 | Chatbot – Other Information | The chatbot shall provide a URL to the website’s frequently asked questions page, if a user submits information not relevant to zones and permits. |
| CH-13 | Chatbot - Help | The chatbot shall display information in using the chatbot when the word “help” is the input. |