# Web Interface For Commercial Real Estate Price Estimations

by Colton Proctor <cproctor@uwyo.edu>

## 1. Introduction

### 1.1. Problem Description

The real estate market is one of the largest industries in modern society. Commercial real estate makes up almost half of this industry yet there is little accessibility to this market. There are many tools available on the internet for assisting with residential real estate, however commercial is more of a mystery. The prices for whole apartment complexes, warehouses, and shopping centers are not given wide accessibility held only by licensed real estate professionals. The goal of this web interface is to deliver comparison estimations easily to anyone via the internet.

Instead of making this service into an app for a phone this functionality will be provided as a web service. This is in order maximize the products accessibility to any potential user. The framework used was the Django platform, this allowed for a lot of the html background to be done inexplicitly. It has been used in large scale web development for a large set of websites.

The intent of this document is to outline the different methods and systems that were used in order to deliver this web service. The document is meant for a coworker to become familiar with the different parts of the system and how they interact each other. It will accomplish that by giving an overview of the different parts and their interactions.

### 1.2. System Overview

The system is composed of two main parts namely, the web interface and the database. Use of this webpage is as follows. A user will come to the webpage and see boxes in which to input data. The data needed corresponds to that which is in the database, location of the property, type of property, size etc. You will enter in as much data as you have about the property in question and submit the form. Then the website retrieves the nearest neighbors to the data you input. It then adds the information that you input into the database.

## 2. Design Considerations

In this section pitfalls are discussed. These are the issues that were run into during the development and where I could expand functionality. They are largely related to the nature the online nature of the service, as networking comes with a breadth of security concerns. There are also concerns about scalability and form interactions with the database.

## 2.1. Assumptions and Dependencies

Since this service will be provided via a webpage it has little hardware and software requirements. Once the server is up and running anyone with an internet-enabled device will be able to access it. There is no requirement for the device to be a particular brand or run a specific browser since we are not using any applets or flash enabled scripts. However, scaling of the website depending on screens was a huge issue. Writing CSS that automatically scaled the webpage and still allowed access to the forms in a readable way was difficult.

A large problem that was encountered was the information users enter into the forms. They didn't always have all of the information required, so the number of features used was scaled down. The validation of these forms was also an issue since there were only certain values that were valid for each field. If all of the features weren't entered the accuracy of the return suffered. The ability for someone to flood the database with their own entries that skew that dataset is also a problem since each entry is added to the database to expand the set and does not get validated except by manual inspection.

The website currently uses an SQLite database which is not largely scalable. Upon deployment this will need to be switched to a more production ready database system.

## 2.2. General Constraints

The main limitation of the system is the database used for comparison evaluations. The size of the database and the complexity of each of its entries directly correlates to the accuracy of the returned results. There also needs to be a validation on the database in order to ensure its accuracy. Outliers or other erroneous data can drastically affect the resulting estimate depending on the algorithms used. Currently it is K-Nearest Neighbor which returns the most similar instance in the database. The service is limited by the available information and can be improved by gathering a larger database of more complex entries.

This is also the case with the data entered by the user. The more data they enter the more accurate an analysis they receive but the more annoying it is to fill out the forms. There is a fine balance between information that is readily available by all property owners and that which is needed to give an accurate result.

The forms are easy to use on any device since they are typing based instead of drop down. The issue with this is data validation and returning of errors. This had to take into account different screen sizes and methods of interaction, such as keyboard and mouse or touch screen. The results must also be checked since you give the user what each database columns name is there is danger of injection attacks.

### 2.3. Goals and Guidelines

The main focus is ease of use. The website needs to be accessible and easy to use for anyone on any device. This allows for the widest range of possible users to benefit. Secondarily the service needs to be built with a more scalable database with the intent of expanding the user base.

### 2.4. Development Methods

The system was designed modularly by using a main program with applications. The applications provided services while the main program serviced the url parsing and returning information to get requests.

This was how the Django framework works and is one of their core design methodologies. It emphasizes the design framework that I wanted to implement in my design phase since it uses modular design. Modular design is helpful since when one piece wasn't working I could work on that part separate from the rest of the website and leave that which was working alone.

## 3. Architectural Strategies

The Django framework allowed for the use of bootstrap as well. By separating out the different pages templates with a core html page they inherited from made specializing the page a more clean design to understand.

I used SQL queries to return the result from the database. In order to use the learning algorithms through scikit learn or numpy my understanding was that it required javascript. I wasn't sure how to leverage those packages through javascript either.

Django is beneficial to the web service as it extends its functionality. Buttons can be added to the webpage which will link back to new modules that can provide any desired effect. This is done through the addition of applications and links to those from the main page. It requires the adding of new forms, views, and urls in order to provide this functionality. This can extend the current base website in further ways.

The use of a SQL database is due to the database already existing in that format. Potentially with the use of datamining techniques a larger database can be gathered and merged with the existing database. I have experience with SQL queries and they have also been proven for their reliability through many applications.

## 4. System Architecture

The system is hierarchically two main sub-systems namely, the web interface and the database. The web interface is where all of the functionality actually resides, whereas the database is there solely to query. The server will deliver the web page initially and then receive the filled out forms from the user. It then queries the database for the relevant data that it needs in order to perform its analysis. The server then returns the result from its calculation to the end user.

The webpage itself exists solely for inputting information. The model is a large set of features that exist in the database that I received. The csv was loaded into the database through this model. It is shown to the user through a view. The view I used does not use all of the features that are contained in the model. Through the admin panel you can examine the entire database and each feature of all of the instances.

### 4.1. Database Manager

The database application is the main part of this website. It is what interacts with the form and returns the results. It first converts the form data into a SQL query and then returns the results back through its view. This is where a more complicated computation could be performed. It would also be a good idea to interpolate the other features from the inputted data in order to create a more robust dataset, rather than a sparsely filled one.

## 5. Network Handler

### 5.1. Interface/Exports

This is the main program that was created by Django. It allows for the fielding of IP requests to the website. It then returns views based off the url input. It contains an admin page and a main page which allow you to access or interact with the database. This is the bulk of the program that was completed, the rest of the work is behind the scenes in the application.

## 6. Conclusion

### 6.1. Final

The end result of the program is a web interface with an underlying database. The database needs to be changed in order to scale upon distribution. The results aren't returned consistently. I would also like for them to be more accurate and only a single result instead of a group of results. That forces the user to guess what their property value is based off the different similar properties instead of getting a result that leverages machine learning. The machine learning would be able to give a more accurate result instead of an extrapolation off of an interpretation. I would also like the website to be prettier and have the ability to have user profiles and a monetary system for more accurate of time intensive results.