Software Requirements Specification: carstats.ca

Michael Zaghi
Student ID: 301350563
Course: CMPT 376W
Date: 2020-04-09

Table of Contents

List of Figures

Revision History

Name	Date	Reason For Change	Version
Michael Zaghi	2020-04-09	Creation of document.	v1.0.0

1 Introduction

1.1 Purpose

The purpose of this document is to provide system details of the *autotrader.ca* web application for its successful implementation and development. The main components of the document focus on describing the product along with its features and requirements. Technical information on system design, dependencies and workflow is also included to give the reader an operational understanding of system processes.

1.2 Document Conventions

References to other sections of the document are highlighted **bold** while references to features, information within the application or key terms are *emphasized*. Defenitions of key terms are available in **Appendix A: Gloassary**. When referring to an object on a page, for example a button, the name of the element will be wrapped in single quotation marks. For section 4 **System Features**, individual requirements (labeled REQ) inherit the priority of the feature they are under. Features are categorized by high, medium or low priority. High priority features are core features which are required for the basic functioning of the entire application. Medium priorty features are features that must be completed, but after the high priority features. Low priority features relate to aesthetics or feature improvements.

1.3 Intended Audience and Reference Documentation

The intended audience of this document includes software developers, product management, system operations, and quality assurance. Useful documentation specific to the project and for the development of the system can be found through the links and references below:

Web Framework: https://www.djangoproject.com/
Dashboarding: https://plot.ly/python/
DBMS: https://dev.mysql.com/doc/
Statistics: Probability and Statistics for Engineering and the Sciences (9th ed.) by Jay L. Devore. Publisher: Duxbury Press
Statistical Computing: https://pandas.pydata.org/docs/reference/index.html#api

1.4 Project Scope

Carstats.ca is a web application designed to help end users understand the pricing of new and used vehicles across Canada. The application is primarily targeted at the private consumer market and dealership market. The overall goal of the application is to allow users to make a more informed buying or selling decision that will save them time and money. This is accomplished using data science and statistical inference. For example, many owners have a rule of thumb about when they sell their vehicle. This is usually an arbitrary number like ‘4 years’. The real question that needs to be answered in this case is ‘*What is the*

optimal time to sell my car based on age, region, vehicle colour, and mileage?'. Similarly, a dealership would like to know the answer to '*What is the vehicle type in my region which has the smallest turnover and the greatest profit margin?* '. Answering these questions captures the idea behind and scope of the application.

2 Overall Description

2.1 Product Perspective

The *carstats.ca* web application is a standalone product that follows a standard multi-tier architecture. The application is data-driven, and will need to run batch jobs and data pre-processing along with the expected volume of HTTP requests. The main source of data comes crawling vehicle ads from websites like *carpages.ca*, *autotrader.ca*, and *kijiji.ca*.

The collected data contains features like vehicle make, year, model, type, colour, kilometres, province and city, among others. This data is then cleaned and pre-processed depending on the computational intensity required for the analytics. Upon request, processed data is then aggregated and rendered through the plotly-dash web application for graphs and visualizations. Plotly-dash is a seperate application which is integrated into the Django web framework.

2.2 Product Features

The *autostats.ca* product is composed of four key features which can be used to increase vehicle market insight and allow the user to make better buying or selling decisions. These are: (1) **Dashboards** (2) **Dealer Insight** (3) **Vehicle Valuation Tool** (4) **Vehicle Comparison Tool**.

2.2.1 Dashboards

Three dashboards are available which are separated by and contain three different types of statistical methods. These are: *Descriptive Analytics*, *Value Analytics*, and *Predictive Analytics*. The dashboards allow users to view and interact with real-time information and models. The intention behind the dashboards, besides providing interesting information, is to allow the user to perform their own analysis and decision making. Each dashboard is selected based on vehicle *make*, *model*, *year* and *type*.

- **Descriptive Analytics:** The descriptive analytics dashboard contains descriptive statistics that summarize the features of the data. This includes statistics like mean, median, mode, standard deviation (or variance), minimum and maximum for data like price, kilometres and colour. The data will be summarized by tables and graphing tools like histograms, line charts, boxplots and pie charts.
- **Value Analytics** The value analytics section is focused on providing measures of relative value for vehicles within a specific class. A vehicle is classified by its price range and type (*e.g.* sedan, convertible, van, *etc.*). This allows for relative comparison using

ratios like *price to kilometres* and *average physical depreciation*. Vehicles are given an in-class ranking for each metric and also a weighted-average overall ranking across all metrics. Vehicle value is also comparable across geographies like province and city. The visualization of these metrics is done through tables and histograms.

- **Predictive Analytics** The predictive analytics dashboard focuses on predicting the future price of a vehicle using historical data. It will include a range of techniques including linear or nonlinear regression, historical trends and machine learning models. Visualization of these models and relevant outputs, like predicted price, is done through tables and line charts.

2.2.2 Dealer Insight

The dealer insights feature gives vehicle dealerships a better idea about what vehicle inventory will maximize their profit margin on each unit sold. The feature provides information on vehicle volume, average turnover, and dealer premium, all within a specified geographic location. The data is summarized graphically so that the information can be digested and compared.

2.2.3 Vehicle Valuation Tool

The vehicle valuation tool is a standard feature that allow the user to enter relevant information about the vehicle they wish to sell, like kilometres, location, condition, colour and add-on features. The output is a probability distribution that gives a scale of estimated 'bad' to 'good' selling prices, along with the expected turnover for each price. The turnover describes the expected amount of time the seller would wait to sell the vehicle at the specified price.

2.2.4 Vehicle Comparison Tool

The vehicle comparison tool is a feature that allows the user to select up to six vehicles and compare their value analytics side-by-side. This feature is useful for the user who has several vehicle purchases in mind and would like to compare the relative value of each vehicle directly, as well as their aggregate scores.

2.3 User Classes and Characteristics

This section defines the three main user classes or userbase: (1) **casual user**, (2) **expert user** and (3) **dealer**. A user class is a grouping of potential users who are expected to use the system in a similar way and with similar intentions. Note that all types of users are expected to reside in Canada since all the data is collected from Canadian vehicle ads and prices are in Canadian dollars (CAD).

2.3.1 Casual User

Description: The casual user consists of the majority of the userbase. These users are typically not active because they are only in the market for a new or used vehicle once every

3-5 years. They are more attracted to the *Vehicle Valuation* and *Vehicle Comparison* tools because they provide a more 'yes' or 'no' answer to their buying or selling decision compared to the dashboards.

Characteristics:

- They are the type of person who conducts significant research and due diligence before committing to a significant financial decision
- Have a basic understanding of statistics and have the ability to use and act on information
- Worried about the price depreciation of their vehicle over the medium to long term

Benefits:

- They can use the information to negotiate a better price for their vehicle
- They know the price depreciation of their vehicle over time and can factor this into their buying or selling decision

2.3.2 Expert User

Description: The expert user is someone who uses the system more frequently for the purpose of buying and then flipping the vehicle for a small profit. This may involve making small fixes to the vehicle before selling or simply using the vehicle for a short period of time before rotating to a new one. On average, this type of user is on the system once a week to check weekly price movements, number of ad openings and average prices across different areas of the Province.

Characteristics:

- Have a 'deal seeking' or 'bargain hunting' personality
- Lookup and make calls on multiple ad postings per week
- Stiff negotiators and are looking to maximize their profit in the least amount of time

Benefits:

- They can cross reference potential purchases with prices and average turnover
- They can make use of the bargaining power metric to decide if an ad price can be lowered
- Know how to filter their search based on average price in nearby cities and number of open ads for a vehicle

2.3.3 Dealer

Description: The vehicle dealership is mainly concerned with managing their inventory and purchasing vehicles which they know they can sell for a good price. When the dealership has an option to purchase a vehicle, the manager can make an informed decision on whether to purchase the vehicle at a given price. This type of user would use the system frequently,

every day to once per week, depending on the size of the dealership and the volume of purchases.

Characteristics:

- Professional
- Looking to expand business and grow
- Goal is to maximize profit by utilizing any information that is available

Benefits:

- They can gauge what their expected inventory is going to look like based on the average turnover metric
- All the information they need to make purchasing decisions is available through the *Dealer Insight* feature
- Decision making is based on quantitative data rather than rough estimates or guessing

2.4 Operating Environment

The *carstats.ca* application is designed to operate on the following systems and technology:

- **Web Framework:** Django 2.2
- **Frontend:** Bootstrap 4 and Jquery 3.4
- **Backend/Database:** MySQL 5.7
- **Web Server:** NGINX 1.17.0 on a remote Ubuntu 18.04/Linux machine

2.5 System Architecture

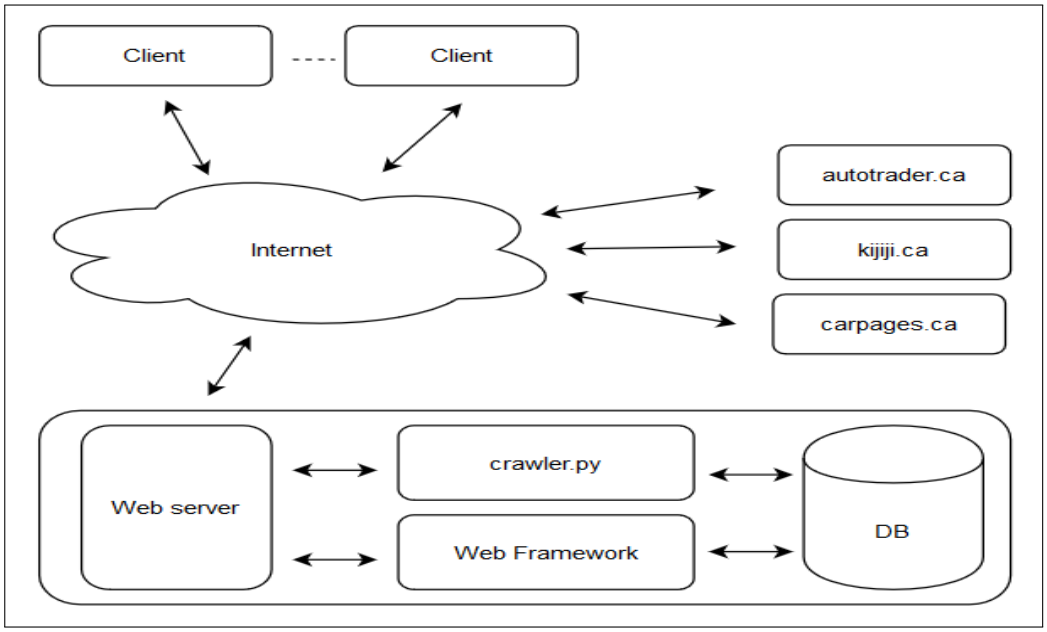

Figure 1: Diagram of system architecture for *carstats.ca*

The *Figure 1* diagram shows the main components of the web application which form a multi-tier client-server architecture. The web server (NGINX) processes incoming network requests over HTTP from the client. The web server talks to the Django web framework which handles the business logic of the request for the application. Since the NGINX web server does not natively speak Python, the WSGI protocol supported by Django is used to standardize communication. Depending on the type of request, the MySQL database may be altered. After the request has been handled, the web server will return a response back to the client.

2.5.1 Batch Processing

Several processes or 'batch jobs' need to be continuously run in order to properly maintain the application. These are scripts which are scheduled to run as cron jobs at various dates and times. The two most important types of jobs are *crawler.py* (see *Figure 1*) and the jobs that aggregate and pre-process raw data for modelling. *crawler.py* is a Python script that crawls the ad postings of *autotrader.ca*, *kijiji.ca*, and *carpages.ca* and extracts the relevent information from the file object that is returned.

This data is then inserted into the *main* table (see *Figure 2*). Data processing jobs extract data from *main*, manipulate the data using pandas dataframes and insert the processed dataframe into the *stats_total* table.

2.5.2 Database ERD

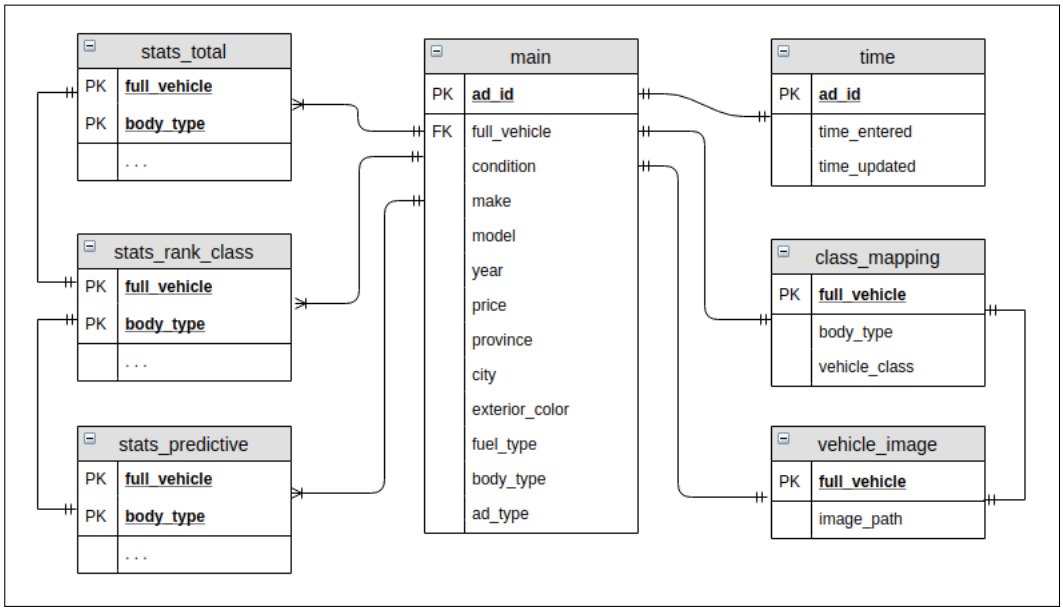

Figure 2: ERD (Entity Relationship Diagram) for the *carstats.ca* database

The *carstats.ca* database uses the MySQL RDBMS. *Figure 2* shows the ERD for the major tables inside the database. The table *main* contains the raw data processed by *crawler.py*. The PK *ad_id* is a unique identification created for each ad which is associated with a vehicle. The FK *full_vehicle* is a concatenation of the *make*, *model* and *year* fields. *main* has a one-to-one relationship with the tables *time*, *class_mapping* and *vehicle_image*. It has a one-to-many relationship with tables *stats_total*, *stats_rank_class* and *stats_predictive* since vehicle statistics are computed for multiple vehicle types (*e.g.* a 2001 Honda Civic Coupe compared to a 2001 Honda Civic Hatchback).

2.5.3 Database Table Descriptions

A brief description of the tables in the *carstats.ca* database and its contents:

- **main:** Contains the raw data scraped by *crawler.py*. The central table that has a relationship with all the other tables in the database.
- **class_mapping:** Maps a vehicle to a pre-computed class based on price range and type.
- **time:** Keeps a record of the first and last time an ad was scraped by *crawler.py*.
- **vehicle_image:** Each record references the file path to the image of each vehicle stored in the server's filesystem.
- **stats_total:** Contains pre-processed metrics that are used in the *Valuation Dashboard*, *Dealer Insight*, *Vehicle Valuation Tool* and *Vehicle Comparison Tool* features.
- **stats_rank_class:** Uses the metrics in table *stats_total* and the classification in *class_mapping* to rank each vehicle by metric and in aggregate.
- **stats_predictive:** Stores the results of model training for each vehicle.

2.6 Design and Implementation Constraints

2.6.1 Software Constraints

Batch processing needs to be run outside of peak hours, since these processes can significantly impact the performance of the application.

2.6.2 Hardware Constraints

The system implementation uses only one physical server, which means there is no way to load balance HTTP requests among multiple servers. The performance of the application will also be constrained by the CPU (number of cores) and disk space of the server.

2.7 Assumptions and Dependencies

The following list of Python libraries are the core requirements for the implementation of the system and application. It is assumed that all Python libraries are installed using Python 3.6 or greater.

- **django 2.2:** Used as the web framework of the system. Version 2.2 is Django's version 2 LTS.
  `pip install "django>=2.2,<3"`
- **plotly-dash:** The Dash application by Plotly is a graphing application built on top of Flask.
  `pip install dash==1.9.1`
- **django-plotly-dash:** Embeds the Dash application into a Django template.
  `pip install django_plotly_dash`
- **pandas:** Used for computing statistics and processing data.
  `pip install pandas`
- **beautifulsoup4:** HTML/XML parser for *crawler.py*
  `pip install beautifulsoup4`
- **scikit-learn:** Machine learning in Python used for the predictive modelling.
  `pip install -U scikit-learn`

3 External Interface Requirements

3.1 User Interfaces

3.1.1 Homepage

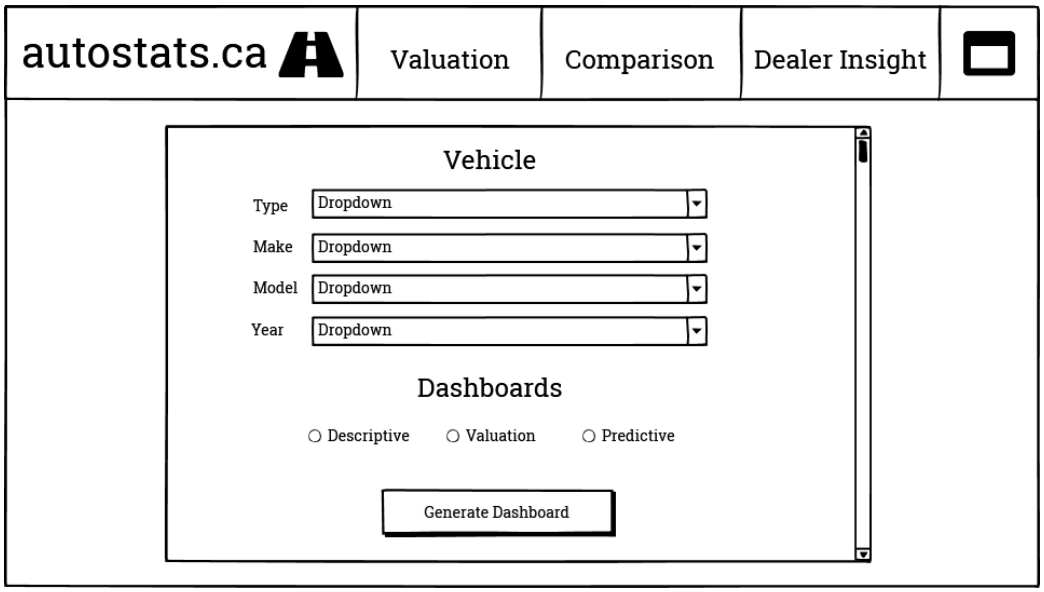

Figure 3: Homepage UI layout

The homepage contains the menu options for generating dashboards since this feature is the focal point of the application. Once *type*, *make*, *model* and *year* are selected from the dropdown elements, the user can select one to three dashboards to load (*Descriptive*, *Valuation* or *Predictive*). The top navigation bar includes the *carstats.ca* logo which links to the homepage, and contains labels which link to the *Valuation Tool*, *Comparison Tool* and *Dealer Insight* pages. The top right icon is a button that, if clicked, contains all previously saved dashboards.

3.1.2 Dashboards

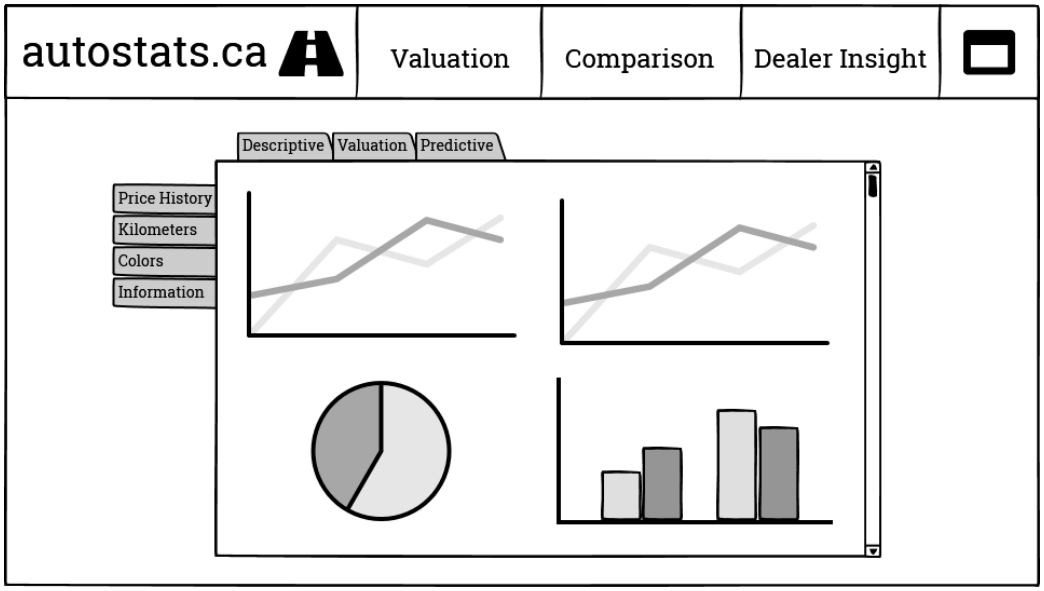

Figure 4: Dashboards UI layout

The dashboard page contains two main components. The side navigation bar allows the user to view related graphs and statistics for each dashboard type. The top navigation bar allows the user to load one of the dashboards that was selected from the homepage. For example, in *Figure 3*, the user opted to load all three dashboards from the homepage. The descriptive dashboard is currently selected, which is subdivided into three sections; *Price History*, *Kilometres*, and *Colours*. The *Information* button in the side navigation bar loads a new window or modal which contains help dialogue and definition of terms.

3.1.3 Valuation Tool

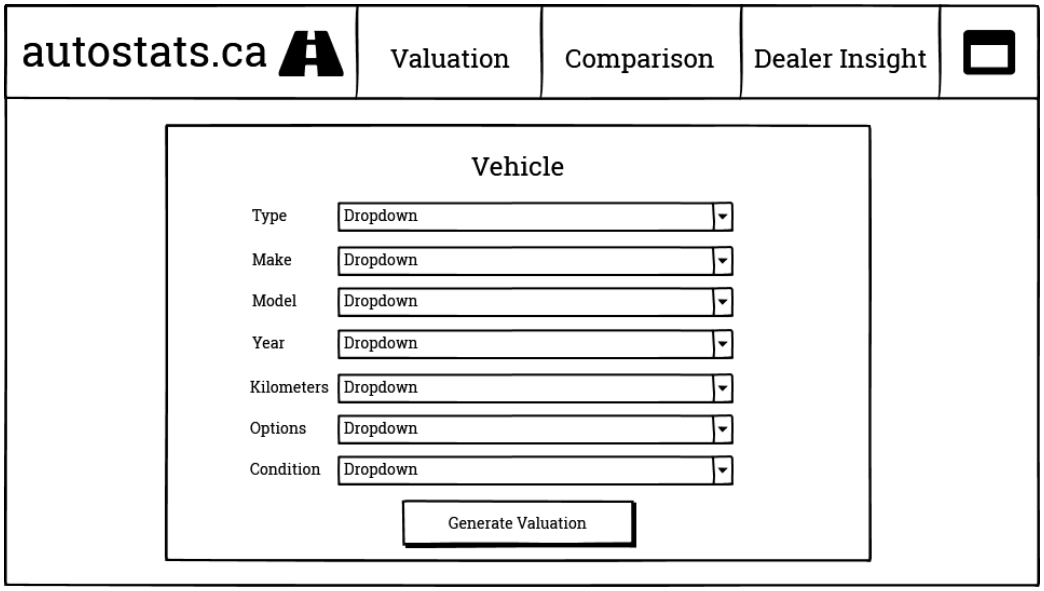

Figure 5: Valuation Tool menu options UI layout

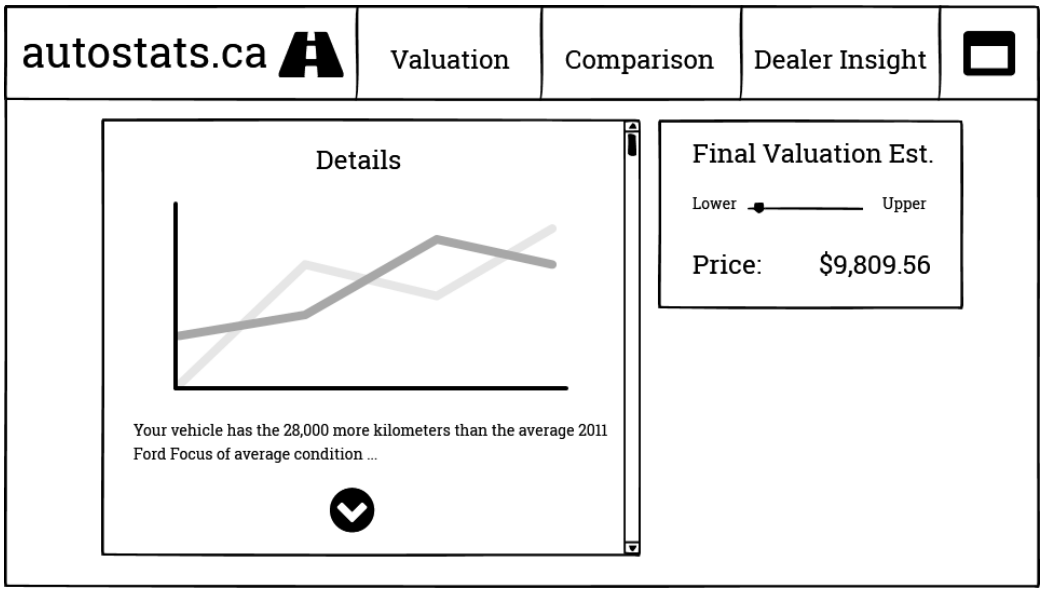

Figure 6: Valuation Tool UI layout and output

The valuation menu is a group of dropdown elements that allow the user to specify the details of the vehicle to be valued. Once the *Generate Valuation* button is clicked, a new valuation output page is loaded. In the details section, the page automatically scrolls down as each new element is loaded. The element contains a graph object with a description or explanation. Once complete, the *Final Valuation Estimate Box* is loaded which contains a slider that can be used to give the upper and lower bounds of the valuation.

3.1.4 Dealer Insights

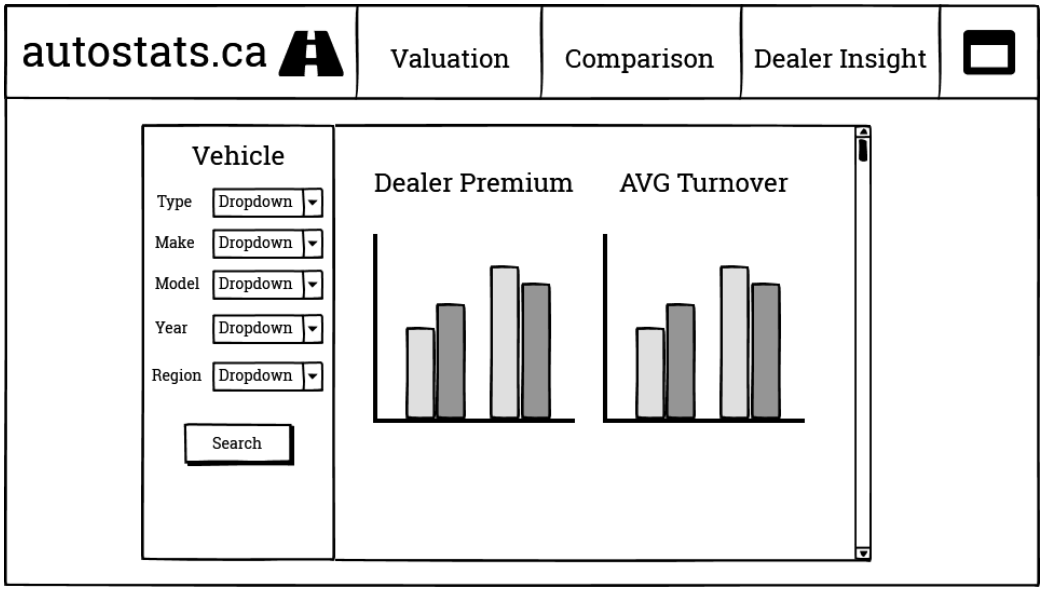

Figure 7: Dealer Insights UI layout

The *Dealer Insights* page is similar to a dashboard layout. It contains graphs that are of specific interest to vehicle dealers. *Figure 6* shows graphs for the dealer premium and average turnover. The left-hand form is used to generate the dashboard. At least one of the fields must be filled before searching.

3.1.5 Comparison Tool

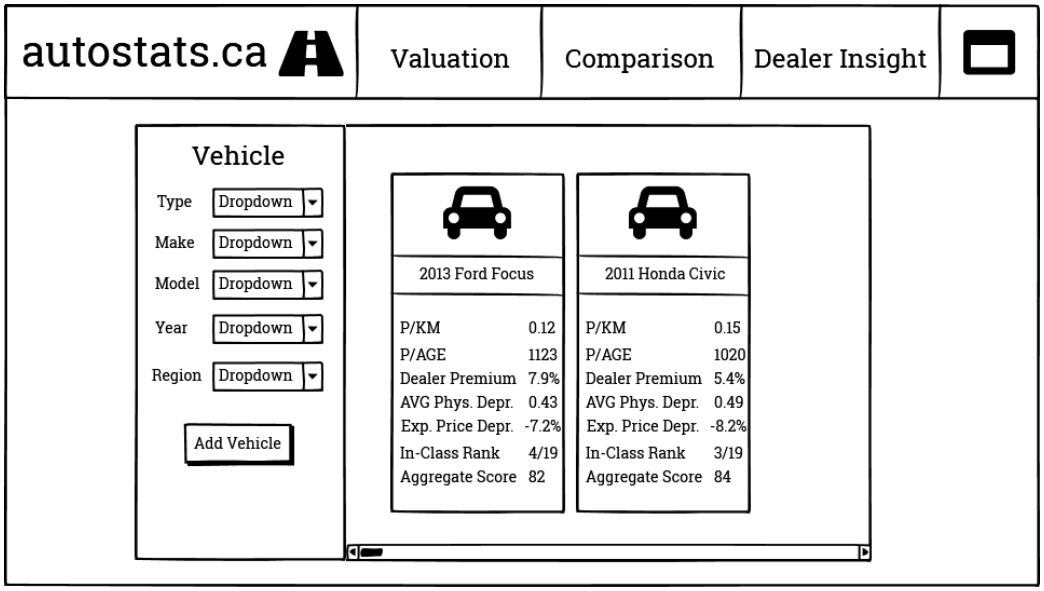

Figure 8: Comparison Tool UI layout

The *Vehicle Comparison* page is used to compare valuation metrics for vehicles side-by-side. Up to six vehicles can be added to the tool. The left-hand form is used to add vehicles where all dropdown fields are required to be filled in.

3.2 Hardware Interfaces

None

3.3 Software Interfaces

The three main software interfaces are for the Django web framework API, the plotly-dash graphing library API, and the MySQL DBMS interface. Links to the API documentation can be found in **Section 1.3** – *Intended Audience and Reference Documentation*.

3.4 Communications Interfaces

The UI must be fully supported on major browsers (Internet Explorer, Edge, Firefox, Safari, Chrome, Opera), including mobile browsers. Network server protocols are handled by NGINX.

4 System Features

4.1 Main Menu Form

4.1.1 Description and Priority

Description: The form located on the homepage which allows the user to load the dashboards. Includes four chained dropdown lists (*type*, *make*, *model* and *year*), three radio buttons (*Descriptive*, *Valuation* and *Predictive*) and a submit button. All elements except vehicle type are initially disabled.

Priority – High: This feature is essential, since it sends the request which allows the server to load the dashboards. The dashboards are the central feature of the entire application.

4.1.2 Stimulus/Response Sequences

Stimulus: user selects type from dropdown.
Response: make dropdown enabled and options filtered by type.

Stimulus: user selects make from dropdown.
Response: model dropdown enabled and options filtered by type and make.

Stimulus: user selects model from dropdown.
Response: year dropdown enabled and options filtered by type, make and model.

Stimulus: user selects year from dropdown.
Response: descriptive, valuation and predictive dashboards radio buttons enabled.

Stimulus: user selects descriptive, valuation or predictive radio button(s)
Response: 'generate dashboard' button enabled.

Stimulus: user clicks the 'generate dashboard' button.
Response: request sent to load the dashboard(s) for selected type, make, model and year.

4.1.3 Functional Requirements

REQ-1: Dropdown elements should be chained, which means that the subsequent dropdown is filtered by the value of the previous dropdown(s).
REQ-2: A changed dropdown value for a previously selected dropdown should void/disable all subsequent element(s).
REQ-3: All dropdown elements should have values before the dashboard options are enabled.
REQ-4: The *Generate Dashboard* button should only be enabled once all dropdown elements have values and at least one dashboard is selected.
REQ-5: If the database has a count less than 30 for the selection, an error message appears in a modal with the message; *not enough samples for this selection, please try a different vehicle*.

4.2 Dashboard Loading Screen

4.2.1 Description and Priority

Description: After clicking the 'generate dashboard' button, a loading screen appears before the dashboard(s) are finished loading. This lets the user know that the system is working on the request. It may take several seconds (2s-30s) for the system to retrieve and process data from the backend to make it ready for the dashboard(s).

Priority – Medium: This feature is important but not critical. Visibility of system status is a good design heuristic to include because it keeps the user informed.

4.2.2 Stimulus/Response Sequences

Stimulus: user clicks the 'generate dashboard' button.
Response: a new page is loaded with a loading icon that shows the system status.

4.2.3 Functional Requirements

REQ-6: The loading icon should be an animated gif or similar which indicates that the system is working on the request.
REQ-7: While the system is working, there should be a 'Loading ...' message below the icon. If the system encounters an error when pre-processing data or loading the dashboard objects, there should be an 'Error' message below the icon.

4.3 Dashboard Selection

4.3.1 Description and Priority

Description: When the user generates their selected dashboards (*Descriptive*, *Value* or *Predictive*), they have the option to load any one dashboard view by selecting the available tabs at the top of the dashboard layout.

Priority – High: This feature is critical. Since the main menu form gives the user the option to generate multiple dashboards, they need to have the option to view all these selected dashboards through the dashboard layout.

4.3.2 Stimulus/Response Sequences

Stimulus: user clicks the *Descriptive* tab.
Response: descriptive dashboard left-hand navigation menu and objects load.

Stimulus: user clicks the *Valuation* tab.
Response: valuation dashboard left-hand navigation menu and objects load.

Stimulus: user clicks the *Predictive* tab.
Response: predictive dashboard left-hand navigation menu and objects load.

4.3.3 Functional Requirements

REQ-8: When the *Descriptive*, *Valuation* or *Predictive* tab in the dashboard layout is clicked, the corresponding left-hand navigation menu and graph objects for that dashboard should become available.

4.4 Descriptive Dashboard

4.4.1 Description and Priority

Description: The *Descriptive* dashboard contains summary statistics and plotly-dash graph objects which give an overview of a selected vehicle type, make, model and year. The dashboard is broken up into three components; *Descriptive Statistics*, *Price History*, *Kilometres*, and *Colours*.

Priority – High: This feature is critical. It is one of the core components of the dashboard layout.

4.4.2 Stimulus/Response Sequences

Stimulus: user clicks the *Descriptive Statistics* navigation button.
Response: descriptive statistics objects load inside the dashboard interface.

Stimulus: user clicks the *Price History* navigation button.
Response: descriptive statistics objects load inside the dashboard interface.

Stimulus: user clicks the *Kilometres* navigation button.
Response: kilometres objects load inside the dashboard interface.

Stimulus: user clicks the *Colours* navigation button.
Response: colours objects load inside the dashboard interface.

4.4.3 Functional Requirements

REQ-9: The *Descriptive Statistics* section should include a summary statistics table for Price (CAD) and Kilometres composed of Mean, SD, Variance, Median, Minimum and Maximum.

REQ-10: The *Descriptive Statistics* section should include a posting details table for open and closed vehicle postings composed of Total, Oldest, Newest, Mean Volume, Mean Turnover (Days) and Dealer (%).

REQ-11: When the *Descriptive Statistics* button is clicked on the left-hand navigation menu, REQ-9 and REQ-10 should be loaded inside the dashboard interface.

REQ-12: The *Price History* section should include a line graph titled 'Price and Volume Timeseries' with 'Price (CAD)' on the left y-axis, 'Volume (Number of Vehicle Postings)' on the right y-axis, and 'Date' on the x-axis. The graph has two series, one for Price (CAD) and the other for Volume.

REQ-13: When the *Price History* button is clicked on the left-hand navigation menu, REQ-12 should be loaded inside the dashboard interface.

REQ-14: The *Kilometres* section should include a histogram titled 'Kilometres' with 'Vehicles' on the y-axis and 'Kilometres' on the x-axis.

REQ-15: The *Kilometres* section should include a scatterplot titled 'Kilometres to Price' with 'Price (CAD)' on the y-axis and 'Kilometres' on the x-axis. It should also include a linear regression line.

REQ-16: When the *Kilometres* button is clicked on the left-hand navigation menu, REQ-14 and REQ-15 should be loaded inside the dashboard interface.

REQ-17: The *Colours* section should include a pie graph titled 'Exterior Colour (%)' which shows the proportion of vehicle colours.

REQ-18: The *Colours* section should include a pie graph titled 'Exterior Colour AVG Price (CAD)' which shows the average price per colour.

REQ-19: When the *Colours* button is clicked on the left-hand navigation menu, REQ-17 and REQ-18 should be loaded inside the dashboard interface.

REQ-20: There should be a 'Help' button on the left-hand navigation menu that opens a new window and contains definitions for Mean, SD, Variance, Median, Minimum, Maximum, Turnover and Volume when the descriptive dashboard is active.

4.5 Valuation Dashboard

4.5.1 Description and Priority

Description: The *Valuation* dashboard contains metrics and plotly-dash graph objects which give a relative comparison of value between vehicles of different makes and models. The dashboard is broken up into three components: *Value Statistics*, *Buying Power* and *Value by Location*.

Priority – High: This feature is critical. It is one of the core components of the dashboard layout.

4.5.2 Stimulus/Response Sequences

Stimulus: user clicks the *Value Statistics* navigation button.
Response: value statistics objects load inside the dashboard interface.

Stimulus: user clicks the *Buying Power* navigation button.
Response: buying power objects load inside the dashboard interface.

Stimulus: user clicks the *Value by Location* navigation button.
Response: value by location objects load inside the dashboard interface.

4.5.3 Functional Requirements

REQ-21: The *Value Statistics* section should include a table titled 'Value Ratios' which includes Mean, Minimum, Maximum, SD, In-Class Rank and Total Rank for each metric P/KM, P/AGE, Dealer Premium and AVG Physical Depreciation. Below the table there are four buttons: P/KM, P/AGE, Dealer Premium and AVG Physical Depreciation. When clicked, they toggle a histogram which shows the In-Class Vehicles on the x-axis and the respective metric on the y-axis .

REQ-22: The *Value Statistics* section should include a table titled 'Depreciation' which includes 1 Week, 1 Month, 3 Month, 6 Month, 1 Year and 3 Year for each of Percent (percentage change), Dollar (CAD) (dollar change), In-Class Rank and Total Rank. Below the table, there are six buttons: 1 Week Price Depreciation, 1 Month Price Depreciation, 3 Month Price Depreciation, 6 Month Price Depreciation, 1 Year Price Depreciation and 3 Year Price Depreciation. When clicked, they toggle a line chart with Date on the x-axis and the respective Price Depreciation on the y-axis.

REQ-23: When the *Value Statistics* button is clicked on the left-hand navigation menu, REQ-21 and REQ-22 should be loaded inside the dashboard interface.

REQ-24: The *Buying Power* section should include a histogram titled 'Buying Power' which has Vehicle Price on the x-axis and Expected Listing Decrease (CAD) on the y-axis.

REQ-25: When the *Buying Power* button is clicked on the left-hand navigation menu, REQ-24 should be loaded inside the dashboard interface.

REQ-26: The *Value by Location* section should include a boxplot titled 'Boxplot Prices by Province' with Price(CAD) on the y-axis and Provinces on the x-axis.

REQ-27: The *Value by Location* section should include a histogram titled 'Average Price - Cities in (Province)' with dropdown element that allows the user to select a Province. Price (CAD) is on the y-axis and City is on the x-axis. City lists the top 12 cities in the selected province that have the most vehicle postings.

REQ-28: When the *Value by Location* button is clicked on the left-hand navigation menu, REQ-26 and REQ-27 should be loaded inside the dashboard interface.

REQ-29: There should be a 'Help' button on the left-hand navigation menu that opens a new window and contains definitions for P/KM, P/AGE, Dealer Premium, AVG Physical Depreciation and In-Class rank when the value dashboard is active.

4.6 Predicative Dashboard

4.6.1 Description and Priority

Description: The *Predictive* dashboard uses plotly-dash graph objects to display the results from forward looking price prediction models. The expected future price of the vehicle gives the user a way to optimize their buying or selling decisions. The dashboard is broken up into two components; *Historical Comparison* and *Time Series Forecasting*.

Priority – High: This feature is critical. It is one of the core components of the dashboard layout.

4.6.2 Stimulus/Response Sequences

Stimulus: user clicks the *Historical Comparison* navigation button.
Response: historical comparison objects load inside the dashboard interface.

Stimulus: user clicks the *Time Series Forecasting* navigation button.
Response: time series forecasting objects load inside the dashboard interface.

4.6.3 Functional Requirements

REQ-30: The *Historical Comparison* section should include a line graph titled 'Previous Year Price' with Price (CAD) on the y-axis and Date on the x-axis. The model works as follows; given a vehicle, it's one year forward looking price change is simply the price change that has already occurred for the same one year old vehicle. Similarly, the two year forward looking price change is based on the one year old for the first year and the two year old for the second year. It is a piece-wise defined function. Although this backwards method is not accurate for predicting price, users may want to know the price trend of previous years for the same model.

REQ-31: When the *Historical Comparison* button is clicked on the left-hand navigation menu, REQ-30 should be loaded inside the dashboard interface.

REQ-32: The *Time Series Forecasting* section should include a line graph titled 'Predicted Price' with Price (CAD) on the y-axis and Date on the x-axis. It is required that the model with the lowest error and highest accuracy be used. As a result, more statistical linear regression models like Ridge Regression and Lasso or machine learning methods like RNN or XGBoost may be used through the scikit-learn library. In either case, the model will be trained on features like Kilometres, Colour, Listing type (private or dealer), and possibly metrics like Turnover and Average Physical Deprecation. The output of the training is the label which is price. The graph must also have extra functionality that indicates to the user time ranges that are good for selling and buying, as well as the optimal buy and sell time.

REQ-33: When the *Time Series Forecasting* button is clicked on the left-hand navigation menu, REQ-32 should be loaded inside the dashboard interface.

REQ-55: There should be a 'Help' button on the left-hand navigation menu that opens a new window and contains a description of the methodologies and models used for the historical comparison and time series forecasting.

4.7 Saving Dashboard Views

4.7.1 Description and Priority

Description: Each user can save up to five sets of dashboards for a selected vehicle through the available 'Save' button on that vehicle interface. This action, caches the page on the server for quick access and stores reference cookies on the user's browser. The user can access saved dashboards through the top-right menu option in the navigation bar.

Priority – Medium: This feature is not necessary for the application to be useful, but it does benefit the overall user experience.

4.7.2 Stimulus/Response Sequences

Stimulus: the user navigates to the dashboard and clicks the 'Save' button.
Response: the 'Dashboards' button in the top-right of the navigation menu highlights.

Stimulus: the user clicks the 'Dashboards' button.
Response: a dropdown menu appears.

Stimulus: the user clicks on a saved dashboard.
Response: the dashboard interface loads for that vehicle and its corresponding dashboards.

4.7.3 Functional Requirements

REQ-34: When the 'Dashboards' button in the top navigation bar is clicked, a dropdown appears which lists all previously saved dashboards. Each link is labeled by *type*, *make*, *model*, *year* and *dashboard(s)*.

REQ-35: The 'Dashboards' dropdown should contain a 'CLEAR' button, which removes all previously saved dashboards and their references.

REQ-36: Each individually saved dashboard should be able to be removed from the 'Dashboards' list and have its corresponding reference removed.
REQ-37: When a saved dashboard is selected, the cached page should be served and the dashboard interface should be loaded.
REQ-38: When the 'Save' button in the dashboard interface is clicked, the 'Dashboards' button should highlight and a link to the saved dashboard should be available in the dashboards dropdown menu.

4.8 Navigation Bar

4.8.1 Description and Priority

Description: The top navigation bar contains links to the valuation tool, comparison tool, and dealer insight page. It also contains the *carstats.ca* name and logo. The requirements for the 'Save' button, which is also in the top navigation bar, are covered in the *Saving Dashboard Views* (section 4.7).

Priority – High: This feature is necessary so that the user can reach core functionalities and features.

4.8.2 Stimulus/Response Sequences

Stimulus: the user clicks the logo or *carstats.ca* text in the top navigation bar.
Response: the user is directed to the homepage.

Stimulus: the user clicks the 'Valuation' button in the top navigation bar.
Response: the user is directed to the valuation tool.

Stimulus: the user clicks the 'Comparison' button in the top navigation bar.
Response: the user is directed to the comparison tool.

Stimulus: the user clicks the 'Dealer Insight' button in the top navigation bar.
Response: the user is directed to the dealer insight page.

4.8.3 Functional Requirements

REQ-39: There should be a top navigation bar that contains the buttons 'Valuation', 'Comparison' and 'Dealer Insight' which link to the pages for the valuation tool, comparison tool and dealer insight page respectively.
REQ-40: The top left of the navigation bar should include the *autostats.ca* logo with the title *autostats.ca* which links to the homepage.

4.9 Comparison Tool

4.9.1 Description and Priority

Description: The comparison tool is used for easy side-by-side comparison of vehicles and their valuation metrics.

Priority – Medium: This feature should be implemented after the dashboards. Many of the metrics and statistics pre-computed for the valuation dashboard are reused for the comparison tool.

4.9.2 Stimulus/Response Sequences

Stimulus: the user fills out the left-hand form for their vehicle and clicks ‘Add Vehicle’.
Response: the vehicle metrics and score appear in an element next to the form.

Stimulus: the user fills out the form again with another vehicle and clicks ‘Add Vehicle’.
Response: the vehicle metrics and score are added next to the previous one.

Stimulus: the user clicks the ‘x’ button on the first vehicle.
Response: the vehicle is removed from the view.

4.9.3 Functional Requirements

REQ-41: There should be a form on the left-hand side of the layout that includes drop-down elements for type, make, model, year and region with a button ‘Add Vehicle’ which sends the request. Similar to REQ-1, type, make, model and year should be chained.
REQ-42: Each vehicle that is added should include the metrics P/KM, P/AGE, Dealer Premium, AVG Price Depreciation, Expected Price Depreciation, In-Class Rank and Aggregate Score.
REQ-43: The element containing the vehicle metrics should have an image of the vehicle and a heading with the name of the vehicle.
REQ-44: The element should be draggable, meaning that for two elements next to each other, click and dragging one into the other’s position will swap their positions.
REQ-45: There should be an ’X’ (close) button on each element. When clicked, it should remove the element. If an element is between two other elements and it is closed, one of the two neighbouring elements should take its place.
REQ-46: No more than five vehicle elements may be displayed at the same time.

4.10 Valuation Tool

4.10.1 Description and Priority

Description: The valuation tool determines fair price on a vehicle given factors like type, make, model, year, options, region and condition. As each factor is computed, the system

adds a plotly-dash graph and description explaining its impact on the total valuation.

Priority – Medium: This feature should be implemented after the dashboards. Many of the statistics computed for the descriptive dashboard are reused for the valuation tool.

4.10.2 Stimulus/Response Sequences

Stimulus: the user fills out the vehicle valuation form and click 'Generate Valuation'.
Response: As the valuation is being computed graph objects for each factor are displayed, after which the final valuation range is displayed.

4.10.3 Functional Requirements

REQ-47: When the user enters the valuation tool page, there should be a form with drop-down elements for type, make, model, year, kilometres, options, region, condition and a button named 'Generate Valuation' which submits the request. Similar to REQ-1, type, make, model and year should be chained.

REQ-48: Each factor in the valuation should have a corresponding plotly-dash graph which displays percentile followed by a brief description of how the factor contributed to the valuation. As each factor is loaded, the interface slides to the display for the next factor.

REQ-49: Once all factors and their objects have loaded, there should be up and down arrow buttons that move between each factor.

REQ-50: Once the final valuation is complete, there should be an element displayed which shows the final valuation estimate. The valuation should be a range which changes based on the value of a slider within the element.

4.11 Dealer Insights

4.11.1 Description and Priority

Description: The dealer insights page contains metrics, analytics and visualizations specific to the needs of vehicle dealerships.

Priority – Medium: This feature should be implemented after the dashboards. Many of the statistics computed for the valuation and predictive dashboards are reused in the dealer insights page.

4.11.2 Stimulus/Response Sequences

Stimulus: the user fills out the vehicle form and clicks 'Search'.
Response: The dealer insights page loads containing metrics and analytics.

4.11.3 Functional Requirements

REQ-51: When the user enters the dealer insights page, there should be a form with drop-down elements for type, make, model, year, region and a button named ‘Search’ which submits the request.
REQ-52: There should be plotly-dash graph objects that display average turnover and dealer premiums at the type, make, model or region levels.
REQ-53: There should be a plotly-dash graph object which shows price sensitivity with respect to time for vehicles that are inventory.
REQ-54: There should be plotly-dash graph objects that display predictive analytics for vehicle turnover, volume and price.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

The longest average load time for dashboards should be no longer than 30 seconds. This means that statistical computations that involve dataframes being loaded in and manipulated should be kept to a minimum, and any metrics which can be pre-processed and stored in the database should be. This is especially true for models which need to be trained, since their accuracy largely depends on the amount of data and the training time. Also, table locking in the database should be avoided since requests from multiple users may be blocked due to *crawler.py* inserting into the table. Using InnoDB over the MySIAM storage engine will avoid this issue.

5.2 Safety Requirements

It should be made clear that *carstats.ca* is not liable for any poor purchasing or selling decisions made by its userbase. Moreover, the accuracy of the data is not guaranteed and that any of the predications made by the models using that data are not 100 percent accurate.

5.3 Security Requirements

Since there is no user login and authentication currently planned, no private user data will be stored on the system. This means that *carstats.ca* is not impacted by external regulation and requirements on its data. Security controls dealing with the web server and web framework should be handled by their built in features like SSL/HTTPS, SQL injection and clickjacking protection.

5.4 Software Quality Attributes

Interfaces for computed metrics and statistics should be written in an object-oriented way to maximize their reusability and flexibility. This is important because the same metrics are

reused, manipulated and/or aggregated throughout different features of the application. Unit testing on each metric is required in order to validate its accuracy.

5.5 Business Rules

autostats.ca operates under the principle that all dashboards and features are freely available to its userbase. Any user can access the same information.

Appendix A: Glossary

Terms	Definition
System	
WSGI	*Web Server Gateway Interface.* Interface for web servers to forward the requests to the Django web framework.
cron	A time-based job scheduler for Unix-like operating systems such as Linux.
ERD	*Entity Relationship Diagram*. A structural diagram used for database design.
LTS	*Long-term support*.
RDBMS	*Relational Database Management System*. DB stands for database.
PK	*Primary Key* of a database table.
FK	*Foreign Key* of a database table.
API	*Application Programming Interface*
UI	*User Interface*
MyISAM	A storage engine employed by MySQL databases with a locking implementation that locks tables.
InnoDB	A storage engine employed by MySQL databases with a locking implementation that locks particular rows.
Statistics	
Variance	Measures the amount of variation or dispersion in the sample data.
SD	*Standard Deviation* is a measure of the amount of variation or dispersion of a set of the sample data using the same unit of measurement.
Mean	The average of the sample data.
Median	The value separating the higher half from the lower half of the sample data.
Minimum	The smallest number in the sample data.
Maximum	The largest number in the sample data.
CAD	*Canadian Dollars*. Every statistic that uses price is in Canadian Dollars.
Metrics	
P/KM	$P/KM = \frac{\text{Price}}{\text{Kilometres}}$
P/AGE	$P/AGE = \frac{\text{Price}}{\text{Year}_{\text{current}} - \text{Year}_{\text{start}}}$
Dealer Prem.	$DealerPremium = \frac{\text{Dealer Price}}{\text{Private Price}}$
Volume	$AVG\ Volume = \frac{1}{n} \sum_{i=0}^{n} \text{New Postings}_i$
Turnover	$AVG\ Turnover = \frac{1}{n} \sum_{i=0}^{n} (\text{Date Posting Removed - Date Posting Added})_i$
Phys. Depr.	$AVG\ PhysicalDepreciation = \frac{\text{Year}_{\text{current}} - \text{Year}_{\text{start}}}{\text{Expected Useful Life}}$